Zbornik radova 12 (20)

LOGIC IN COMPUTER SCIENCE

Matematički institut SANU

Zbornik radova 12 (20)

LOGIC IN COMPUTER SCIENCE

Editor:
Zoran Ognjanović

Matematički institut SANU

Издавач: Математички институт САНУ, Београд, Кнеза Михаила 36
серија: Зборник радова, књига 12 (20)
За издавача: Богољуб Станковић, главни уредник серије
Уредник свеске: Зоран Огњановић, као гост
Технички уредник: Драган Благојевић
Штампа: "Академска издања", Земун
Штампање завршено фебруара 2009.

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004 : 510.6(082)
LOGIC in Computer Science / editor Zoran Ognjanović. - Beograd : Matematički institut SANU, 2009 (Zemun : Akademska izdanja). - 215 str. : graf. prikazi ; 24 cm. - (Zbornik radova / Matematički institut SANU ; knj. 12 (20))

Tiraž 350. - Str 3: Preface / Zoran Ognjanović. - Bibliografija uz svaki rad.

ISBN 978-86-80593-40-1
1. Ognjanović, Zoran, 1964- [уредник] [аутор додатног текста]
a) Рачунарство - Математичка логика - Зборници
COBISS.SR-ID 156220940

PREFACE

This special issue of *Zbornik Radova* intends to make several new topics accessible to professional mathematicians and doctoral students. It also points out and helps readers to understand the corresponding research challenges and future research directions in some applications of mathematical logic, which can be seen as the calculus of computation, to foundations of theoretical computer science.

Since the sixties, when systematic work in mathematical logic started in Belgrade, our researchers have been working and publishing important papers in many standard areas of logic: model theory, proof theory, set theory, recursion theory, non-standard analysis, non-classical logics etc. That work attracted a significant number of collaborators from all universities in Serbia, and former Yugoslavia. Today, it might be said that theoretical achievements of what we can informally call "Belgrade school in mathematical logic" have been widely recognized.

On the other hand, during the last twenty years, as the usefulness of mathematical logic in computer science and artificial intelligence became more and more obvious, our logicians, organized in several scientific projects supported by Serbian Ministry of Science, have been establishing intensive interactions with those fields. Currently, along this line of research two five years projects "Representations of logical structures and their application in computer science" and "Models, Languages, Types, and Processes in Computing" are under realization (see http://www.mi.sanu.ac.yu/projects/projects.htm for more information). This issue collects four articles of members of those projects.

The paper by K. Došen and Z. Petrić is a survey of results about coherence for categories with finite products and coproducts. The investigated categories formalize equality of proofs in classical and intuitionistic conjunctive-disjunctive logic without distribution of conjunction over disjunction. Z. Ognjanović, M. Rašković and Z. Marković present probabilistic logics as a formalism for representing and reasoning with uncertain knowledge. The paper contains the axiomatizations of a number of logics, proofs of the completeness theorems, and discusses their decidability. The paper by M. Mosurović, T. Stojanović, and A. Kaplarević-Mališić gives an overview of basic description logics as well as some related original results concerning temporal extensions of Description Logics. S. Ghilezan and S. Likavec summarise their work in the field of computational interpretation of intuitionistic and classical logic in lambda calculus and its extensions.

I, as a guest editor, am grateful to many people who helped me, in particular to the members of the editorial board, referees, and of course the authors of the papers. However, I would especially like to mention professor *Slaviša B. Prešić* (1933–2008) who is widely credited as one of the most important Serbian logician, the founder of the Seminar for Mathematical Logic in Mathematical Institute of the Serbian Academy of Sciences and Arts, supervisor of many PhD and MSc theses, as a pioneer of applications of theoretical results from mathematical logic in computer science, etc. As a member of the editorial board, he strongly influenced work on writing this special issue, and I would like to thank him for that.

Zoran Ognjanović

Contents

Kosta Došen and Zoran Petrić

BICARTESIAN COHERENCE REVISITED

Abstract. A survey is given of results about coherence for categories with finite products and coproducts. For these results, which were published previously by the authors in several places, some formulations and proofs are here corrected, and matters are updated. The categories investigated in this paper formalize equality of proofs in classical and intuitionistic conjunctive-disjunctive logic without distribution of conjunction over disjunction.

Mathematics Subject Classification (2000): 18A30, 18A15, 03G30, 03G10, 03F05, 03F07, 03B20

Keywords: bicartesian categories, categories with finite products and coproducts, coherence, categorial proof theory, decidability of equality of arrows, conjunction and disjunction, decidability of equality of deductions, Post completeness

CONTENTS

1. Coherence

Categorists call *coherence* what logicians would probably call *completeness*. This is, roughly speaking, the question whether we have assumed for a particular brand of categories all the equations between arrows we should have assumed. Completeness need not be understood here as completeness with respect to models. We may have also a syntactical notion of completeness—something like the Post completeness of the classical propositional calculus—but often some sort of model-theoretical completeness is implicit in coherence questions. Matters are made more complicated by the fact that categorists do not like to talk about syntax, and do not perceive the problem as being one of finding a match between syntax and semantics. They do not talk of formal systems, axioms and models.

Moreover, questions that logicians would consider to be questions of *decidability*, which is of course not the same as completeness, are involved in what categorists call coherence. A coherence problem often involves the question of deciding whether two terms designate the same arrow, i.e. whether a diagram of arrows commutes. Coherence is understood mostly as solving this problem, which we call the *commuting problem*, in [22] (see p. 117, which mentions [20] and [21] as the origin of this understanding). The commuting problem seems to be involved also in the understanding of coherence of [17, Section 10].

Completeness and decidability, though distinct, are not foreign to each other. A completeness proof with respect to a manageable model may provide, more or less immediately, tools to solve decision problems. For example, the completeness proof

for the classical propositional calculus with respect to the two-element Boolean algebra provides immediately a decision procedure for theoremhood.

The simplest coherence questions are those where it is intended that all arrows of the same type should be equal, i.e. where the category envisaged is a preorder. The oldest coherence problem is of that kind. This problem has to do with monoidal categories, and was solved by Mac Lane in [23]. The monoidal category freely generated by a set of objects is a preorder. So Mac Lane could claim that showing coherence is showing that "all diagrams commute".

In cases where coherence amounts to showing preorder, i.e. showing that from a given set of equations, assumed as axioms, we can derive all equations (provided the equated terms are of the same type), from a logical point of view we have to do with *axiomatizability*. We want to show that a decidable set of axioms (and we wish this set to be as simple as possible, preferably given by a finite number of axiom schemata) delivers all the intended equations. If preorder is intended, then all equations are intended. Axiomatizability is in general connected with logical questions of completeness, and a standard logical notion of completeness is completeness of a set of axioms. Where all diagrams should commute, coherence does not seem to be a question of model-theoretical completeness, but even in such cases it may be conceived that the model involved is a discrete category.

Categorists are interested in axiomatizations that permit extensions. These extensions are in a new language, with new axioms, and such extensions of the axioms of monoidal categories need not yield preorders any more. Categorists are also interested, when they look for axiomatizations, in finding the combinatorial building blocks of the matter. The axioms are such building blocks, as in knot theory the Reidemeister moves are the combinatorial building blocks of knot and link equivalence (see [3, Chapter 1], or any other textbook in knot theory).

In Mac Lane's second coherence result of [23], which has to do with symmetric monoidal categories, it is not intended that all equations between arrows of the same type should hold. What Mac Lane does can be described in logical terms in the following manner. On the one hand, he has an axiomatization, and, on the other hand, he has a model category where arrows are permutations; then he shows that his axiomatization is complete with respect to this model. It is no wonder that his coherence problem reduces to the completeness problem for the usual axiomatization of symmetric groups.

Algebraists do not speak of axiomatizations, but of *presentations by generators and relations*. The axiomatizations we envisage are purely equational axiomatizations, as in algebraic varieties. Such were the axiomatizations of [23]. Categories are algebras with partial operations, and we are interested in the equational theories of these algebras.

In Mac Lane's coherence results for monoidal and symmetric monoidal categories one has to deal only with natural isomorphisms. However, in the coherence result for symmetric monoidal closed categories of [19] there are already natural and dinatural transformations that are not isomorphisms.

A natural transformation is tied to a relation between the argument-places of the functor in the source and the argument-places of the functor in the target. This

relation corresponds to a relation between occurrences of letters in formulae, and in composing natural transformations we compose these relations. With dinatural transformations the matter is more complicated, and composition poses particular problems (see [24]). In this paper we deal with natural transformations. Our general notion of coherence does not, however, presuppose naturality and dinaturality.

Our notion of a coherence result is one that covers Mac Lane's and Kelly's coherence results mentioned above, but it is more general. We call coherence a result that tells us that there is a faithful functor G from a category $\mathcal{S}$ freely generated in a certain class of categories to a "manageable" category $\mathcal{M}$. This calls for some explanation.

It is desirable, though perhaps not absolutely necessary, that the functor G be *structure-preserving*, which means that it preserves structure at least up to isomorphism. In all coherence results we will consider here, the functor G will preserve structure strictly, i.e. "on the nose". The categories $\mathcal{S}$ and $\mathcal{M}$ will be in the same class of categories, and G will be obtained by extending in a unique way a map from the generators of $\mathcal{S}$ into $\mathcal{M}$.

The category $\mathcal{M}$ is *manageable* when equations of arrows, i.e. commuting diagrams of arrows, are easier to consider in it than in $\mathcal{S}$. The best is if the commuting problem is obviously decidable in $\mathcal{M}$, while it was not obvious that it is such in $\mathcal{S}$.

With our approach to coherence we are oriented towards solving the commuting problem. This should be stressed because other authors may give a more prominent place to other problems. We have used on purpose the not very precise term "manageable" for the category $\mathcal{M}$ to leave room for modifications of our notion of coherence, which would be oriented towards solving another problem than the commuting problem.

In this paper, the manageable category $\mathcal{M}$ will be the category *Rel* with arrows being relations between occurrences of letters in formulae. In [14] and elsewhere we have taken *Rel* to be the category of relations between finite ordinals, which is not essentially different from what we do in this paper. The previous category *Rel* is the skeleton of the new one. We have mentioned above the connection between *Rel* and natural transformations. The commuting problem in *Rel* is obviously decidable.

The freely generated category $\mathcal{S}$ will be the bicartesian category, i.e. category with all finite products and coproducts, freely generated by a set of objects, or a related category of that kind. The generating set of objects may be conceived as a discrete category. In our understanding of coherence, replacing this discrete generating category by an arbitrary category would prevent us to solve coherence—simply because the commuting problem in the arbitrary generating category may be undecidable. Far from having more general, stronger, results if the generating category is arbitrary, we may end up by having no result at all.

The categories $\mathcal{S}$ in this paper are built ultimately out of *syntactic* material, as logical systems are built. Categorists are not inclined to formulate their coherence results in the way we do—in particular, they do not deal often with syntactically built categories. If, however, more involved and more abstract formulations of coherence that may be found in the literature (for early references on this matter

see [18]) have practical consequences for solving the commuting problem, our way of formulating coherence has these consequences as well.

That there is a faithful structure-preserving functor G from the syntactical category $\mathcal{S}$ to the manageable category $\mathcal{M}$ means that for all arrows f and g of $\mathcal{S}$ with the same source and the same target we have

$$f = g \text{ in } \mathcal{S} \text{ iff } Gf = Gg \text{ in } \mathcal{M}.$$

The direction from left to right in this equivalence is contained in the functoriality of G, while the direction from right to left is faithfulness proper.

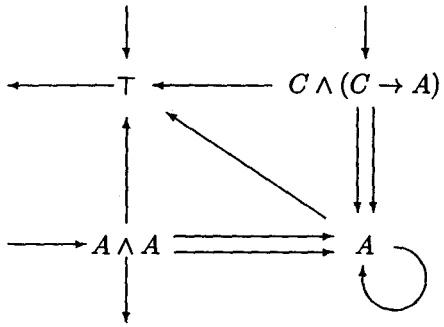
If $\mathcal{S}$ is conceived as a syntactical system, while $\mathcal{M}$ is a model, the faithfulness equivalence we have just stated is like a completeness result in logic. The left-to-right direction, i.e. functoriality, is soundness, while the right-to-left direction, i.e. faithfulness, is completeness proper.

If G happens to be one-one on objects, then we obtain that $\mathcal{S}$ is isomorphic to a subcategory of $\mathcal{M}$—namely, its image under G in $\mathcal{M}$. We will have such a situation in this paper, where G will be identity on objects.

In this paper we will separate coherence results involving terminal objects and initial objects from those not involving them. These objects cause difficulties, and the statements and proofs of the coherence results gain by having these difficulties kept apart.

2. Coherence and proof theory

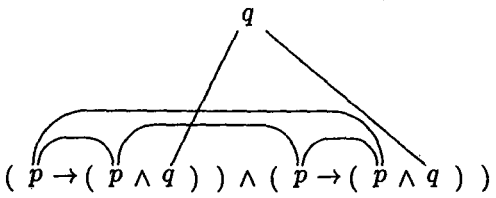
If one envisages a deductive system as a graph whose nodes are formulae:



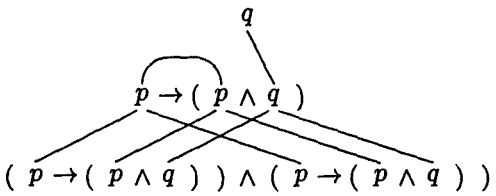
and whose arrows are derivations from the sources understood as premises to the targets understood as conclusions, then equality of derivations usually transforms this deductive system into a category of a particular brand. This category has a structure induced by the connectives of the deductive system. Although equality of derivation is dictated by logical concerns, usually the categories we end up with are of a kind that categorists have already introduced for their own reasons. The prime example here is given by the deductive system for the conjunction-implication fragment of intuitionistic propositional logic. After derivations in this deductive system are equated according to ideas about normalization of derivations that stem from Gentzen, one obtains the cartesian closed category $\mathcal{K}$ freely generated by a set of propositional letters (see [22] for the notion of cartesian closed category).

Equality of proofs in intuitionistic logic has not led up to now to a coherence result—a coherence theorem is not forthcoming for cartesian closed categories. If we take that the model category $\mathcal{M}$ is a category whose arrows are graphs like the graphs of [19], then we do not have a faithful functor G from the free cartesian closed category $\mathcal{K}$ to $\mathcal{M}$. We will now explain why G is not even a functor.

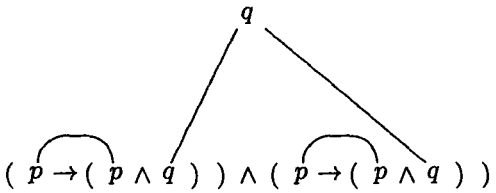
If $\eta_{p,q}$ is the canonical arrow from q to $p \to (p \wedge q)$, where $A \to B$ and $A \wedge B$ stand for B^A and $A \times B$ respectively, while $\hat{w}_A$ is the diagonal arrow from A to $A \wedge A$, then $G(\hat{w}_{p\to(p\wedge q)} \circ \eta_{p,q})$:



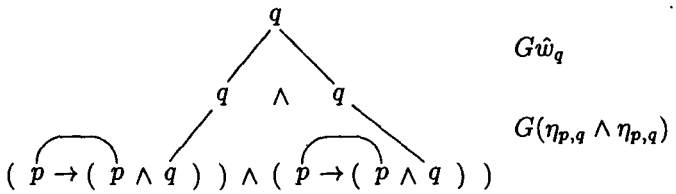
which is obtained from



is different from $G((\eta_{p,q} \wedge \eta_{p,q}) \circ \hat{w}_q)$:

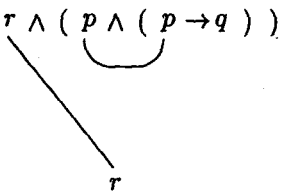


which is obtained from

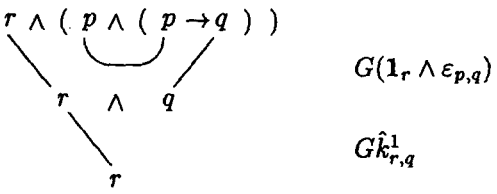


So, if $\hat{w}$ is a natural transformation, then G is not a functor. The naturality of $\hat{w}$, and other arrows of that kind, tied to structural rules ($\hat{w}$ is tied to contraction, and $\hat{k}^1$ below to thinning), is desirable because it corresponds to the permuting of these rules in a cut-elimination or normalization procedure.

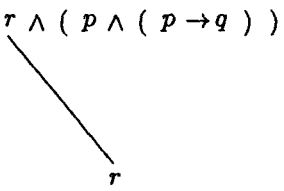
Dually, if $\varepsilon_{p,q}$ is the canonical arrow from $p \wedge (p \to q)$ to q, and $\hat{k}^1_{A,B}$ is the first projection from $A \wedge B$ to A, then $G(\hat{k}^1_{r,q} \circ (\mathbf{1}_r \wedge \varepsilon_{p,q}))$:



which is obtained from



is different from $G\hat{k}^1_{r,p\wedge(p\to q)}$:



So, if $\hat{k}^1$ is a natural transformation, then G is not a functor. The faithfulness of G fails because of a counterexample in [27], involving a natural number object in *Set* and the successor function. This does not exclude that with a more sophisticated model category $\mathcal{M}$ we might still be able to obtain coherence for cartesian closed categories (for an attempt along these lines see [25]).

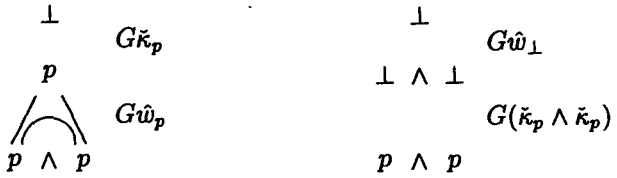
Equality of proofs in classical logic may, however, lead to coherence with respect to model categories that catch up to a point the idea of *generality* of proofs. Such is in particular the category *Rel* mentioned in the preceding section, whose arrows are relations between occurrences of propositional letters in the premises and conclusions. The idea that generality of proofs may serve as a criterion for identity of proofs stems from Lambek's pioneering papers in categorial proof theory of the late 1960s (see [22] for references). This criterion says, roughly, that two derivations represent the same proof when their generalizations with respect to diversification of variables (without changing the rules of inference) produce derivations with the same source and target, up to a renaming of variables.

Although coherence with respect to *Rel* is related to generality, it is not exactly that. The question is should $G\hat{w}_p$ be the relation in the left one or in the right one of the following two diagrams:

The second option, induced by dealing with equivalence relations, or by connecting all letters that must remain the same in generalizing proofs (see [12] and [13]), would lead to abolishing the naturality of $\hat{w}$. For example, in the following instance of the naturality equation for $\hat{w}$:

$$\hat{w}_p \circ \check{\kappa}_p = (\check{\kappa}_p \wedge \check{\kappa}_p) \circ \hat{w}_\bot$$

for $\check{\kappa}_p$ being the unique arrow from the initial object $\bot$ to p, we do not have that $G(\hat{w}_p \circ \check{\kappa}_p)$ is equal to $G((\check{\kappa}_p \wedge \check{\kappa}_p) \circ \hat{w}_\bot)$:



We obtain similarly that κ cannot be natural.

It is shown in [14] that coherence with respect to the model category *Rel* could justify plausibly equality of derivations in various systems of propositional logic, including classical propositional logic. The goal of that book was to explore the limits of coherence with respect to the model category *Rel*. This does not exclude that other coherence results may involve other model categories, and, in particular, with a model category different from *Rel*, classical propositional logic may induce a different notion of Boolean category than the one introduced in Chapter 14 of [14]. That notion of Boolean category was not motivated *a priori*, but was dictated by coherence with respect to *Rel*. The definition of that notion was however not given via coherence, but via an equational axiomatization. We take such definitions as being proper axiomatic definitions.

We could easily define nonaxiomatically a notion of Boolean category with respect to graphs of the Kelly–Mac Lane kind (see [19]). Equality of graphs would dictate what arrows are equal. In this notion, conjunction would not be a product, because the diagonal arrows and the projections would not make natural transformations (see above), and, analogously, disjunction would not be a coproduct (cf. [14, Section 14.3]) The resulting notion of Boolean category would not be trivial—the freely generated categories of that kind would not be preorders—, but its nonaxiomatic definition would be trivial. There might exist a nontrivial equational axiomatic definition of this notion. Finding such a definition is an open problem.

We are looking for nontrivial axiomatic definitions because such definitions give information about the combinatorial building blocks of our notions, as Reidemeister moves give information about the combinatorial building blocks of knot equivalence. Our axiomatic equational definition of Boolean category in [14] is of the nontrivial, combinatorially informative, kind. Coherence of these Boolean categories with respect to *Rel* is a theorem, whose proof in [14] requires considerable effort.

Another analogous example is provided by the notion of monoidal category, which was introduced in a not entirely axiomatic way, via coherence, by Bénabou in [2], and in the axiomatic way, such as we favour, by Mac Lane in [23]. For Bénabou, coherence is built into the definition, and for Mac Lane it is a theorem. One could analogously define the theorems of classical propositional logic as being the tautologies (this is done, for example, in [4, Sections 1.2–3]), in which case completeness would not be a theorem, but would be built into the definition.

In this paper we prove coherence for categories that formalize equality of proofs in classical and intuitionistic conjunctive-disjunctive logic without distribution of conjunction over disjunction. This fragment of logic also covers the additive connectives of linear and other substructural logics (where distribution anyway should not be assumed). When to this fragment we add the true and absurd propositional constants matters become more complicated, and we do not how to prove unrestricted coherence in all cases.

3. Lattice categories

In the remaining sections of this paper we deal with coherence with respect to *Rel* for categories with a double cartesian structure, i.e. with finite products and finite coproducts. We take this as a categorification of the notion of lattice. As before, we distinguish cases with and without special objects, which are here the empty product and the empty coproduct, i.e. the terminal and initial objects. Categories with all finite products and coproducts, including the empty ones, are usually called *bicartesian* categories (see [22]). Categories with all nonempty finite products and coproducts are called *lattice* categories in [14]. The results presented here are adapted from [9], [11], the revised version of [10] and [14, Chapter 9].

We pay particular attention to questions of maximality, i.e. to the impossibility of extending our axioms without collapse into preorder, and hence triviality. This maximality is a kind of syntactical completeness. (The sections on maximality improve upon results reported in [9], [11] and [10], and are taken over from [14, Chapter 9].)

Our techniques are partly based on a composition elimination for conjunctive logic, related to normalization in natural deduction, and on a simple composition elimination for conjunctive-disjunctive logic, implicit in Gentzen's cut elimination.

We define now the category $\mathbf{L}$ built out of syntactic material. The objects of the category $\mathbf{L}$ are the formulae of the propositional language $\mathcal{L}$, generated out of a set of infinitely many propositional letters, for which we use $p, q, r, \ldots$, sometimes with indices, with the binary connectives $\wedge$ and $\vee$, for which we use ξ. For formulae we use $A, B, C, \ldots$, sometimes with indices.

To define the arrows of $\mathbf{L}$, we define first inductively a set of expressions called the *arrow terms* of $\mathbf{L}$. Every arrow term will have a *type*, which is an ordered pair of formulae of $\mathcal{L}_\wedge$. We write $f : A \vdash B$ when the arrow term f is of type (A, B). Here A is the *source*, and B the *target* of f. For arrow terms we use $f, g, h, \ldots$, sometimes with indices. Intuitively, the arrow term f is the code of a derivation

of the conclusion B from the premise A (which explains why we write $\vdash$ instead of $\rightarrow$).

For all formulae A, B and C of $\mathcal{L}$ the following *primitive arrow terms*:

$$\mathbf{1}_A : A \vdash A,$$

$$\hat{w}_A : A \vdash A \wedge A, \qquad \check{w}_A : A \vee A \vdash A,$$

$$\hat{k}^i_{A_1,A_2} : A_1 \wedge A_2 \vdash A_i, \quad \check{k}^i_{A_1,A_2} : A_i \vdash A_1 \vee A_2, \quad \text{for } i \in \{1,2\},$$

are arrow terms. (Intuitively, these are the axioms of our logic with the codes of their trivial derivations.)

Next we have the following inductive clauses:

if $f : A \vdash B$ and $g : B \vdash C$ are arrow terms,
then $(g \circ f) : A \vdash C$ is an arrow term;
if $f_1 : A_1 \vdash B_1$ and $f_2 : A_2 \vdash B_2$ are arrow terms,
then $(f_1 \,\xi\, f_2) : A_1 \,\xi\, A_2 \vdash B_1 \,\xi\, B_2$ is an arrow term.

(Intuitively, the operations on arrow terms $\circ$ and ξ are codes of the rules of inference of our logic.) This defines the arrow terms of **L**. As we do usually with formulae, we will omit the outermost parentheses of arrow terms.

We stipulate first that all the instances of $f = f$ and of the following equations are equations of **L**:

categorial equations:

$$(cat\ 1) \qquad f \circ \mathbf{1}_A = \mathbf{1}_B \circ f = f : A \vdash B,$$

$$(cat\ 2) \qquad h \circ (g \circ f) = (h \circ g) \circ f,$$

bifunctorial equations:

$$(\xi\ 1) \qquad \mathbf{1}_A \,\xi\, \mathbf{1}_B = \mathbf{1}_{A \xi B}$$

$$(\xi\ 2) \qquad (g_1 \circ f_1) \,\xi\, (g_2 \circ f_2) = (g_1 \,\xi\, g_2) \circ (f_1 \,\xi\, f_2),$$

naturality equations: for $f : A \vdash B$ and $f_i : A_i \vdash B_i$, where $i \in \{1,2\}$,

$$(\hat{w}\ nat) \qquad (f \wedge f) \circ \hat{w}_A = \hat{w}_B \circ f,$$

$$(\check{w}\ nat) \qquad f \circ \check{w}_A = \check{w}_B \circ (f \vee f),$$

$$(\hat{k}^i\ nat) \qquad f_i \circ \hat{k}^i_{A_1,A_2} = \hat{k}^i_{B_1,B_2} \circ (f_1 \wedge f_2),$$

$$(\check{k}^i\ nat) \qquad (f_1 \vee f_2) \circ \check{k}^i_{A_1,A_2} = \check{k}^i_{B_1,B_2} \circ f_i,$$

triangular equations: for $i \in \{1,2\}$,

$$(\hat{w}\hat{k}) \qquad \hat{k}^i_{A,A} \circ \hat{w}_A = \mathbf{1}_A,$$

$$(\check{w}\check{k}) \qquad \check{w}_A \circ \check{k}^i_{A,A} = \mathbf{1}_A,$$

$$(\hat{w}\hat{k}\hat{k}) \qquad (\hat{k}^1_{A,B} \wedge \hat{k}^2_{A,B}) \circ \hat{w}_{A \wedge B} = \mathbf{1}_{A \wedge B},$$

$$(\check{w}\check{k}\check{k}) \qquad \check{w}_{A \vee B} \circ (\check{k}^1_{A,B} \vee \check{k}^2_{A,B}) = \mathbf{1}_{A \vee B}.$$

This concludes the list of axiomatic equations stipulated for **L**. To define all the equations of **L** it remains only to say that the set of these equations is closed under

symmetry and transitivity of equality and under the rules

$$(\circ\ cong)\quad \frac{f = f' \qquad g = g'}{g \circ f = g' \circ f'} \qquad\qquad (\xi\ cong)\quad \frac{f_1 = f_1' \qquad f_2 = f_2'}{f_1\ \xi\ f_2 = f_1'\ \xi\ f_2'}$$

On the arrow terms of $\mathbf{L}$ we impose the equations of $\mathbf{L}$. This means that an arrow of $\mathbf{L}$ is an equivalence class of arrow terms of $\mathbf{L}$ defined with respect to the smallest equivalence relation such that the equations of $\mathbf{L}$ are satisfied (see [14, Section 2.3], for details).

The kind of category for which $\mathbf{L}$ is the one freely generated out of the set of propositional letters (which may be understood as a discrete category) we call *lattice* category (see [14, Section 9.4], for a precise definition). Usually, such categories would be called categories with finite nonempty products and coproducts. The objects of a lattice category that is a partial order make a lattice.

4. The functor G

The objects of the category *Rel* are the objects of $\mathbf{L}$, i.e. the formulae of $\mathcal{L}$. An arrow $R\colon A \vdash B$ of *Rel* is a set of ordered pairs (x, y) such that x is an occurrence of a propositional letter in the formula A and y is an occurrence of a propositional letter in the formula B; in other words, arrows are binary relations between the sets of occurrences of propositional letters in formulae. We write either $(x, y) \in R$ or xRy, as usual. In this category, $\mathbf{1}_A\colon A \vdash A$ is the identity relation, i.e. identity function, that assigns to every occurrence of a propositional letter in A that same occurrence. In $\mathcal{L}$ there are no formulae in which no propositional letter occurs, but where we have such formulae (as in the language $\mathcal{L}_{\top,\bot}$ considered later in this paper), the empty set of ordered pairs corresponds to $\mathbf{1}_A\colon A \vdash A$ if no propositional letter occurs in A. The empty relation is the identity relation on the empty set.

For $R_1 : A \vdash B$ and $R_2 : B \vdash C$, the set of ordered pairs of the composition $R_2 \circ R_1 : A \vdash C$ is $\{(x, y) \mid \exists z(xR_1z \text{ and } zR_2y)\}$. Let $x_j(A)$ be the j-th occurrence of a propositional letter in A counting from the left, and let $|A|$ be the number of occurrences of propositional letters in A (so $1 \leqslant j \leqslant |A|$). For $R_i\colon A_i \vdash B_i$, with $i \in \{1, 2\}$, the set of ordered pairs of $R_1\ \xi\ R_2\colon A_1\ \xi\ A_2 \vdash B_1\ \xi\ B_2$, for $\xi \in \{\wedge, \vee\}$, is the disjoint union of the following two sets:

$$\{(x_j(A_1\ \xi\ A_2), x_k(B_1\ \xi\ B_2)) \mid (x_j(A_1), x_k(B_1)) \in R_1\},$$

$$\{(x_{j+|A_1|}(A_1\ \xi\ A_2), x_{k+|B_1|}(B_1\ \xi\ B_2)) \mid (x_j(A_2), x_k(B_2)) \in R_2\}.$$

With the operation on objects that corresponds to the binary connective ξ, this operation ξ on arrows gives a biendofunctor in *Rel*.

In *Rel* we have the relations $G\hat{w}_A : A \vdash A \wedge A$, $G\check{w}_A : A \vee A \vdash A$, $G\hat{k}^i_{A_1,A_2} : A_1 \wedge A_2 \vdash A_i$, and $G\check{k}^i_{A_1,A_2} : A_i \vdash A_1 \vee A_2$, for $i \in \{1, 2\}$, whose sets of ordered pairs are defined as follows:

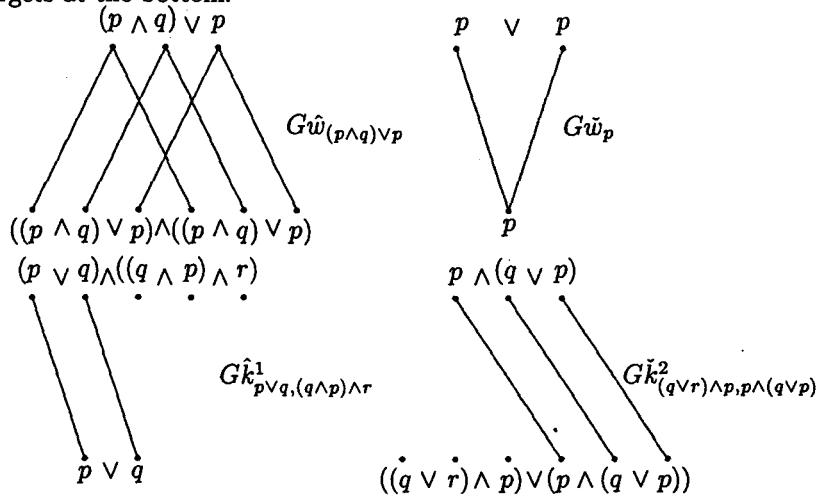
$(x_j(A), x_k(A \wedge A)) \in G\hat{w}_A$ iff $(x_k(A \vee A), x_j(A)) \in G\check{w}_A$ iff $j \equiv k \pmod{|A|}$;

$(x_j(A_1 \wedge A_2), x_k(A_1)) \in G\hat{k}^1_{A_1,A_2}$ iff $(x_k(A_1), x_j(A_1 \vee A_2)) \in G\check{k}^1_{A_1,A_2}$ iff $j = k$;

$(x_j(A_1 \wedge A_2), x_k(A_2)) \in G\hat{k}^2_{A_1,A_2}$ iff $(x_k(A_2), x_j(A_1 \vee A_2)) \in G\check{k}^2_{A_1,A_2}$ iff

$$j = k + |A_1|.$$

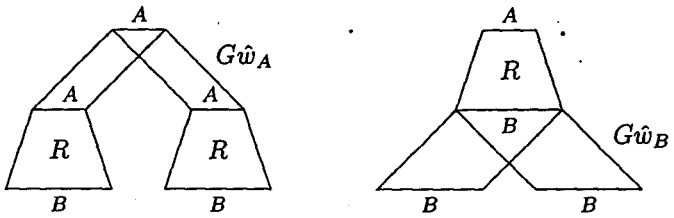
It is not difficult to check that all these arrows of *Rel* give rise to natural transformations. This is clear from the graphical representation of relations in *Rel*. Here are a few examples of such graphical representations, with sources written at the top and targets at the bottom:



For $R\colon A \vdash B$, the naturality equation

$$(R \wedge R) \circ G\hat{w}_A = G\hat{w}_B \circ R,$$

which corresponds to the equation ($\tilde{w}$ *nat*) of the preceding section, and which we take as an example, is justified in the following manner via graphs:



We can now define a functor G from the category $\mathbf{L}$ to the category *Rel*. On objects we have that GA is A. We have defined G above on the primitive arrow terms of $\mathbf{L}$, and we have

$$G(f \xi g) = Gf \xi Gg,$$
$$G(g \circ f) = Gg \circ Gf.$$

To ascertain that this defines a functor from $\mathbf{L}$ to *Rel*, it remains to check that if $f = g$ in $\mathbf{L}$, then $Gf = Gg$ in *Rel*, which we do by induction on the length of the derivation of $f = g$ in $\mathbf{L}$.

It is easy to check by induction that if for $f\colon A \vdash B$ we have $(x_j(A), x_k(B)) \in Gf$, then $x_j(A)$ and $x_k(B)$ are occurrences of the same propositional letter.

Our first task in this paper is to show that the functor G from $\mathbf{L}$ to *Rel* is faithful. We call this result *Lattice Coherence*, and we say that $\mathbf{L}$ is *coherent*. Since G is identity on objects, this means that $\mathbf{L}$ is isomorphic to a subcategory of *Rel*.

It is clear that if $\mathbf{L}$ is coherent in the sense just specified, then it is decidable whether arrow terms of $\mathbf{L}$ are equal in $\mathbf{L}$. In logical terms, one would say that the coherence of $\mathbf{L}$ implies the decidability of the equational system used to define $\mathbf{L}$. This is because equality of arrows is clearly decidable in *Rel*. So coherence here implies a solution to the commuting problem.

5. Coherence for lattice categories

We define by induction a set of terms for the arrows of $\mathbf{L}$ that we call *Gentzen terms*. The identity arrow terms $\mathbf{1}_A$ are Gentzen terms, and we assume that Gentzen terms are closed under the following operations on arrow terms, besides the operation $\circ$, where $=_{dn}$ is read "denotes":

$$\frac{f_1 : C \vdash A_1 \qquad f_2 : C \vdash A_2}{\langle f_1, f_2\rangle =_{dn} (f_1 \wedge f_2) \circ \hat{w}_C : C \vdash A_1 \wedge A_2}$$

$$\frac{g_i : A_i \vdash C}{\hat{K}^i_{A_{3-i}} g_i =_{dn} g_i \circ \hat{k}^i_{A_1,A_2} : A_1 \wedge A_2 \vdash C}$$

$$\frac{g_1 : A_1 \vdash C \qquad g_2 : A_2 \vdash C}{[g_1, g_2] =_{dn} \check{w}_C \circ (g_1 \vee g_2) : A_1 \vee A_2 \vdash C}$$

$$\frac{f_i : C \vdash A_i}{\check{K}^i_{A_{3-i}} f_i =_{dn} \check{k}^i_{A_1,A_2} \circ f_i : C \vdash A_1 \vee A_2}$$

It is easy to verify that the following equations hold for Gentzen terms (these equations can serve for an alternative formulation of $\mathbf{L}$):

$(\hat{K}1)$ $g \circ \hat{K}^i_A f = \hat{K}^i_A (g \circ f)$, $(\check{K}1)$ $\check{K}^i_A g \circ f = \check{K}^i_A (g \circ f)$,

$(\hat{K}2)$ $\hat{K}^i_A g \circ \langle f_1, f_2\rangle = g \circ f_i$, $(\check{K}2)$ $[g_1, g_2] \circ \check{K}^i_A f = g_i \circ f$,

$(\hat{K}3)$ $\langle g_1, g_2\rangle \circ f = \langle g_1 \circ f, g_2 \circ f\rangle$, $(\check{K}3)$ $g \circ [f_1, f_2] = [g \circ f_1, g \circ f_2]$,

$(\hat{K}4)$ $\mathbf{1}_{A\wedge B} = \langle \hat{K}^1_B \mathbf{1}_A, \hat{K}^2_A \mathbf{1}_B\rangle$, $(\check{K}4)$ $\mathbf{1}_{A\vee B} = [\check{K}^1_B \mathbf{1}_A, \check{K}^2_A \mathbf{1}_B]$,

$(\hat{K}5)$ $\hat{K}^i_D \langle f_1, f_2\rangle = \langle \hat{K}^i_D f_1, \hat{K}^i_D f_2\rangle$, $(\check{K}5)$ $\check{K}^i_D [g_1, g_2] = [\check{K}^i_D g_1, \check{K}^i_D g_2]$,

$(\hat{K}\check{K})$ $\hat{K}^i_C \check{K}^j_D h = \check{K}^j_D \hat{K}^i_C h$,

with appropriate types assigned to f, g, f_i and g_i.

It is very easy to show that for every arrow term of $\mathbf{L}$ there is a Gentzen term denoting the same arrow. We can prove the following theorem for $\mathbf{L}$.

Composition Elimination. *For every arrow term h there is a composition-free Gentzen term h' such that $h = h'$.*

Proof. We find first a Gentzen term denoting the same arrow as h. Take a subterm $g \circ f$ of this Gentzen term such that both f and g are composition-free. We call such a subterm a *topmost cut*. We show that $g \circ f$ is equal either to a composition-free Gentzen term or to a Gentzen term all of whose compositions occur in topmost cuts of strictly smaller length than the length of $g \circ f$. The possibility of eliminating composition in topmost cuts, and hence every composition, follows by induction on the length of topmost cuts.

The cases where f or g are $\mathbf{1}_A$ are taken care of by (*cat* 1); the cases where f is $\hat{K}^i_A f'$ are taken care of by $(\hat{K}1)$; and the case where g is $\langle g_1, g_2 \rangle$ is taken care of by $(\hat{K}3)$.

We have next cases dual to the last two, where g is $\check{K}^i_A g'$, which is taken care of by $(\check{K}1)$, and where f is $[f_1, f_2]$, which is taken care of by $(\check{K}3)$. In the remaining cases, if f is $\langle f_1, f_2 \rangle$, then g is either of a form already covered by cases above, or g is $\hat{K}^i_A g'$, and we apply $(\hat{K}2)$. Finally, if f is $\check{K}^i_A f'$, then g is either of a form already covered by cases above, or g is $[g_1, g_2]$, and we apply $(\check{K}2)$. ⊣

Note that we use only the equations $(\hat{K}1)$–$(\hat{K}3)$ and $(\check{K}1)$–$(\check{K}3)$ in this proof (which is taken over from [11], Section 3). We can then prove the following lemma for **L**.

Invertibility Lemma for $\wedge$. *Let $f\colon A_1 \wedge A_2 \vdash B$ be a Gentzen term. If for every $(x, y) \in Gf$ we have that x is in A_1, then f is equal to a Gentzen term of the form $\hat{K}^1_{A_2} f'$, and if for every $(x, y) \in Gf$ we have that x is in A_2, then f is equal to a Gentzen term of the form $\hat{K}^2_{A_1} f'$.*

Proof. By Composition Elimination for **L**, we can assume that f is composition-free, and then we proceed by induction on the length of the target B (or on the length of f). If B is a letter, then f must be equal in **L** to an arrow term of the form $\hat{K}^i_{A_{3-i}} f'$. The condition on Gf dictates whether i here is 1 or 2.

If B is $B_1 \wedge B_2$ and f is not of the form $\hat{K}^i_{A_{3-i}} f'$, then f must be of the form $\langle f_1, f_2 \rangle$ (the condition on Gf precludes that f be an identity arrow term). We apply the induction hypothesis to $f_1\colon A_1 \wedge A_2 \vdash B_1$ and $f_2\colon A_1 \wedge A_2 \vdash B_2$, and use the equation $(\hat{K}5)$.

If B is $B_1 \vee B_2$ and f is not of the form $\hat{K}^i_{A_{3-i}} f'$, then f must be of the form $\check{K}^j_{B_{3-j}} g$, for $j \in \{1, 2\}$. We apply the induction hypothesis to $g\colon A_1 \wedge A_2 \vdash B_i$, and use the following instance of the equation $(\hat{K}\check{K})$:

$$\check{K}^j_{B_{3-j}} \hat{K}^i_{A_{3-i}} g' = \hat{K}^i_{A_{3-i}} \check{K}^j_{B_{3-j}} g'. \qquad \dashv$$

We have a dual Invertibility Lemma for $\vee$. We can then prove the following result of [11, Section 4].

Lattice Coherence. *The functor G from* **L** *to* Rel *is faithful.*

Proof. Suppose $f, g\colon A \vdash B$ are arrow terms of **L** and $Gf = Gg$. We proceed by induction on the sum of the lengths of A and B to show that $f = g$ in **L**. If A and B are both letters, then we conclude by Composition Elimination for **L** that

an arrow term of $\mathbf{L}$ of the type $A \vdash B$ exists iff A and B are the same letter p, and we must have $f = g = \mathbf{1}_p$ in $\mathbf{L}$. Note that we do not need here the assumption $Gf = Gg$.

If B is $B_1 \wedge B_2$, then for $i \in \{1,2\}$ we have that $\hat{k}^i_{B_1,B_2} \circ f$ and $\hat{k}^i_{B_1,B_2} \circ g$ are of type $A \vdash B_i$. We also have

$$G(\hat{k}^i_{B_1,B_2} \circ f) = G\hat{k}^i_{B_1,B_2} \circ Gf = G\hat{k}^i_{B_1,B_2} \circ Gg = G(\hat{k}^i_{B_1,B_2} \circ g),$$

whence, by the induction hypothesis, we have $\hat{k}^i_{B_1,B_2} \circ f = \hat{k}^i_{B_1,B_2} \circ g$ in $\mathbf{L}$. Then we infer

$$\langle \hat{k}^1_{B_1,B_2} \circ f, \hat{k}^2_{B_1,B_2} \circ f \rangle = \langle \hat{k}^1_{B_1,B_2} \circ g, \hat{k}^2_{B_1,B_2} \circ g \rangle,$$

from which $f = g$ follows with the help of the equations $(\hat{K}3)$ and $(\hat{K}4)$. We proceed analogously if A is $A_1 \vee A_2$.

Suppose now that A is $A_1 \wedge A_2$ or a letter, and B is $B_1 \vee B_2$ or a letter, but A and B are not both letters. Then by Composition Elimination for $\mathbf{L}$ we have that f is equal in $\mathbf{L}$ to an arrow term of $\mathbf{L}$ that is either of the form $f' \circ \hat{k}^i_{A_1,A_2}$ or of the form $\check{k}^i_{B_1,B_2} \circ f'$. Suppose $f = f' \circ \hat{k}^1_{A_1,A_2}$. Then for every $(x,y) \in Gf$ we have $x \in GA_1$.

By the Invertibility Lemma for $\wedge$, it follows that g is equal in $\mathbf{L}$ to an arrow term of the form $g' \circ \hat{k}^1_{A_1,A_2}$. From $Gf = Gg$ we can infer easily that $Gf' = Gg'$, and so by the induction hypothesis $f' = g'$, and hence $f = g$.

We reason analogously when $f = f' \circ \hat{k}^2_{A_1,A_2}$. If $f = \check{k}^i_{B_1,B_2} \circ f'$, then again we reason analogously, applying the Invertibility Lemma for $\vee$. $\dashv$

This proof of Lattice Coherence is simpler than a proof given in [11]. In the course of that previous proof one has also coherence results for two auxiliary categories related to $\mathbf{L}$. We will need these categories later, but we do not need these coherence results. For the sake of completeness, however, we record them here too.

Let $\hat{\mathbf{L}}_\vee$ be the category defined as $\mathbf{L}$ with the difference that the primitive arrow terms $\check{w}$ and $\check{k}^i$ are excluded, as well as the equations involving them. The Gentzen formulation of $\hat{\mathbf{L}}_\vee$ is obtained by taking the operation $\vee$ on arrow terms instead of the operations $[\,,\,]$ and $\check{K}^i$.

The category $\check{\mathbf{L}}_\wedge$ is isomorphic to $\hat{\mathbf{L}}^{op}_\vee$. In $\check{\mathbf{L}}_\wedge$, the $\wedge$ and $\vee$ of $\hat{\mathbf{L}}_\vee$ are interchanged.

One can easily prove Composition Elimination for $\hat{\mathbf{L}}_\vee$ (and hence also for $\check{\mathbf{L}}_\wedge$) by abbreviating the proof of Composition Elimination for $\mathbf{L}$ above. For $\hat{\mathbf{L}}_\vee$ we do not have the cases where f is $[f_1, f_2]$ or $\check{K}^i_A f'$, but f can be $f_1 \vee f_2$. Then, if g is not of a form already covered by the proof above, it must be $g_1 \vee g_2$, and we apply the bifunctorial equation $(\vee 2)$.

A composition-free arrow term of $\hat{\mathbf{L}}_\vee$ may be reduced to a unique normal form, which can then be used to demonstrate coherence for $\hat{\mathbf{L}}_\vee$, i.e. the fact that the functor G from $\hat{\mathbf{L}}_\vee$ to *Rel* is faithful (see [11, Section 4]).

6. Coherence for sesquicartesian categories

We define now the category $\mathbf{L}_{\top,\bot}$, whose definition extends the definition of $\mathbf{L}$ with the terminal object $\top$ and the initial object $\bot$, i.e. nullary product and coproduct. The objects of this category are the formulae of the propositional language $\mathcal{L}_{\top,\bot}$, generated out of a set of infinitely many propositional letters with the binary connectives $\wedge$ and $\vee$, and the nullary connectives, i.e. propositional constants, $\top$ and $\bot$.

The arrow terms of $\mathbf{L}_{\top,\bot}$ are defined as the arrow terms of $\mathbf{L}$ save that for every object A we have the additional primitive arrow terms

$$\hat{\kappa}_A : A \vdash \top, \qquad \check{\kappa}_A : \bot \vdash A,$$

and for all arrow terms $f : A \vdash \top$ and $g : \bot \vdash A$ we have the additional axiomatic equations

$$(\hat{\kappa}) \quad \hat{\kappa}_A = f, \qquad (\check{\kappa}) \quad \check{\kappa}_A = g,$$

$$(\hat{k}\bot) \quad \hat{k}^1_{\bot,\bot} = \hat{k}^2_{\bot,\bot}, \qquad (\check{k}\top) \quad \check{k}^1_{\top,\top} = \check{k}^2_{\top,\top}.$$

It is easy to see that with the help of the last two equations we obtain that the pairs

$$\hat{k}^1_{\bot,\bot} = \hat{k}^2_{\bot,\bot} : \bot \wedge \bot \vdash \bot \quad \text{and} \quad \check{\kappa}_{\bot\wedge\bot} = \hat{w}_\bot : \bot \vdash \bot \wedge \bot,$$

$$\check{k}^1_{\top,\top} = \check{k}^2_{\top,\top} : \top \vdash \top \vee \top \quad \text{and} \quad \hat{\kappa}_{\top\vee\top} = \check{w}_\top : \top \vee \top \vdash \top$$

are inverses of each other. This shows that every letterless formula of $\mathcal{L}_{\top,\bot}$ is isomorphic in $\mathbf{L}_{\top,\bot}$ either to $\top$ or to $\bot$.

The kind of category for which $\mathbf{L}_{\top,\bot}$ is the one freely generated out of the set of propositional letters we call *dicartesian* category. The objects of a dicartesian category that is a partial order make a lattice with top and bottom.

By omitting the equations $(\hat{k}\bot)$ and $(\check{k}\top)$ in the definition of $\mathbf{L}_{\top,\bot}$ we would obtain the *bicartesian* category freely generated by the set of propositional letters (cf. [22, Section I.8]). Dicartesian categories were considered under the name *coherent bicartesian* categories in the printed version of [10].

We previously believed wrongly that we have proved coherence for dicartesian, alias coherent bicartesian, categories. Lemma 5.1 of the printed version of [10] is however not correct. We prove here only a restricted coherence result for dicartesian categories. A study of equality of arrows in bicartesian categories may be found in [5].

Suppose that in the definition of $\mathbf{L}_{\top,\bot}$ we omit one of $\top$ and $\bot$ from the language, and we omit all the arrow terms and equations involving the omitted nullary connective. When we omit $\top$, we obtain the category $\mathbf{L}_\bot$, and when we omit $\bot$, we obtain the category $\mathbf{L}_\top$. It is clear that $\mathbf{L}_\bot$ is isomorphic to $\mathbf{L}_\top^{op}$. In the printed version of [10] the categories for which $\mathbf{L}_\bot$ is the one freely generated by the set of propositional letters were called *coherent sesquicartesian* categories. We call them now just *sesquicartesian* categories.

The category *Set*, whose objects are sets and whose arrows are functions, with cartesian product $\times$ as $\wedge$, disjoint union $+$ as $\vee$, a singleton set $\{*\}$ as $\top$ and the empty set $\emptyset$ as $\bot$, is a bicartesian category, but not a dicartesian category. It is,

however, a sesquicartesian category in the $\mathbf{L}_\bot$ sense, but not in the $\mathbf{L}_\top$ sense. This is because in *Set* we have that $\emptyset \times \emptyset$ is equal to $\emptyset$, but $\{*\} + \{*\}$ is not isomorphic to $\{*\}$.

To define the functor G from $\mathbf{L}_{\top,\bot}$ to *Rel* we assume that the objects of *Rel* are the formulae of $\mathcal{L}_{\top,\bot}$. Everything else in the definition of *Rel* remains unchanged; in particular, the arrows are sets of ordered pairs of occurrences of propositional *letters* (no propositional constant is involved). In the definition of the functor G we stipulate that for $G\hat{\kappa}_A$ and $G\check{\kappa}_A$ we have the empty set of ordered pairs. This serves also for the definition of the functors G from $\mathbf{L}_\bot$ and $\mathbf{L}_\top$ to *Rel*.

We can establish unrestricted coherence for sesquicartesian categories, with a proof taken over from the revised version of [10], which we will present below. (This proof differs from the proof in the printed version of [10], which relied also on Lemma 5.1, and is not correct.) It is obtained by enlarging the proof of Lattice Coherence.

The Gentzen formulation of $\mathbf{L}_{\top,\bot}$ is obtained like that of $\mathbf{L}$ save that we have in addition the primitive Gentzen terms $\hat{\kappa}_A : A \vdash \top$ and $\check{\kappa}_A : \bot \vdash A$. For Gentzen terms we have as additional equations, besides $(\hat{\kappa})$ and $(\check{\kappa})$, the following equations:

$$(\hat{K}\bot) \qquad \hat{K}^1_\bot \mathbf{1}_\bot = \hat{K}^2_\bot \mathbf{1}_\bot,$$

$$(\check{K}\top) \qquad \check{K}^1_\top \mathbf{1}_\top = \check{K}^2_\top \mathbf{1}_\top,$$

which amount to $(\hat{k}\bot)$ and $(\check{k}\top)$.

We can prove Composition Elimination for $\mathbf{L}_{\top,\bot}$ by enlarging the proof for $\mathbf{L}$. We have as new cases first those where f is $\check{\kappa}_A$ or g is $\hat{\kappa}_A$, which are taken care of by the equations $(\check{\kappa})$ and $(\hat{\kappa})$. The following case remains. If f is $\hat{\kappa}_A$, then g is of a form covered by cases already dealt with. Note that we do not need the equations $(\hat{K}\bot)$ and $(\check{K}\top)$ for this proof (so that we have also Composition Elimination for the free bicartesian category).

Let the category $\hat{\mathbf{L}}_{\vee,\top,\bot}$ be defined like the category $\hat{\mathbf{L}}_\vee$ save that it involves also $\hat{\kappa}$ and the equations $(\hat{\kappa})$ and $(\hat{k}\bot)$, and let the category $\check{\mathbf{L}}_{\wedge,\top,\bot}$ be defined like the category $\check{\mathbf{L}}_\wedge$ save that it involves also $\check{\kappa}$ and the equations $(\check{\kappa})$ and $(\check{k}\top)$. Composition Elimination is provable for $\hat{\mathbf{L}}_{\vee,\top,\bot}$ and $\check{\mathbf{L}}_{\wedge,\top,\bot}$ by abbreviating the proof of Composition Elimination for $\mathbf{L}_{\top,\bot}$, in the same way as we abbreviated the proof of Composition Elimination for $\mathbf{L}$ in order to obtain Composition Elimination for $\hat{\mathbf{L}}_\vee$.

An arrow term of $\mathbf{L}_{\top,\bot}$ is in *standard form* when it is of the form $g \circ f$ for f an arrow term $\hat{\mathbf{L}}_{\vee,\top,\bot}$ and g an arrow term of $\check{\mathbf{L}}_{\wedge,\top,\bot}$. We can then prove the following.

Standard-Form Lemma. *Every arrow term of* $\mathbf{L}_{\top,\bot}$ *is equal in* $\mathbf{L}_{\top,\bot}$ *to an arrow term in standard form.*

Proof. By categorial and bifunctorial equations, we may assume that we deal with a factorized arrow term f none of whose factors is a complex identity (i.e., f is a big composition of composition-free arrow terms none of which is equal to an identity arrow; see [14, Sections 2.6–7], for precise definitions of these notions) and every

factor of f is either an arrow term of $\hat{\mathbf{L}}_{\vee,\top,\bot}$, and then we call it a $\wedge$-*factor*, or an arrow term of $\check{\mathbf{L}}_{\wedge,\top,\bot}$, when we call it a $\vee$-*factor*.

Suppose $f : B \vdash C$ is a $\wedge$-factor and $g : A \vdash B$ is a $\vee$-factor. We show by induction on the length of $f \circ g$ that in $\mathbf{L}_{\top,\bot}$

$$(*) \quad f \circ g = g' \circ f' \quad \text{or} \quad f \circ g = f' \quad \text{or} \quad f \circ g = g'$$

for f' a $\wedge$-factor and g' a $\vee$-factor.

We will consider various cases for f. In all such cases, if g is $\check{w}_B$, then we use $(\check{w}\ nat)$. If f is $\hat{w}_B$, then we use $(\hat{w}\ nat)$. If f is $\hat{k}^i_{D,E}$ and g is $g_1 \wedge g_2$, then we use $(\hat{k}^i\ nat)$. If f is $f_1 \wedge f_2$ and g is $g_1 \wedge g_2$, then we use bifunctorial and categorial equations and the induction hypothesis.

If f is $f_1 \vee f_2$, then we have the following cases. If g is $\check{k}^i_{B_1,B_2}$, then we use $(\check{k}^i\ nat)$. If g is $g_1 \vee g_2$, then we use bifunctorial and categorial equations and the induction hypothesis.

Finally, cases where f is $\hat{\kappa}_B$ or g is $\check{\kappa}_B$ are taken care of by the equations $(\hat{\kappa})$ and $(\check{\kappa})$. This proves $(*)$, and it is clear that $(*)$ is sufficient to prove the lemma. $\dashv$

We can also prove Composition Elimination and an analogue of the Standard-Form Lemma for $\mathbf{L}_\bot$. Next we have the following lemmata for $\mathbf{L}_{\top,\bot}$ and $\mathbf{L}_\bot$.

Lemma 1. *If for f, g: $A \vdash B$ either A or B is isomorphic to $\top$ or $\bot$, then $f = g$.*

Proof. If A is isomorphic to $\bot$ or B is isomorphic to $\top$, then the matter is trivial. Suppose i: $B \vdash \bot$ is an isomorphism. Then from

$$\hat{k}^1_{\bot,\bot} \circ \langle i \circ f, i \circ g \rangle = \hat{k}^2_{\bot,\bot} \circ \langle i \circ f, i \circ g \rangle$$

we obtain $i \circ f = i \circ g$, which yields $f = g$. We proceed analogously if A is isomorphic to $\top$. $\dashv$

Lemma 2. *If for f, g: $A \vdash B$ we have $Gf = Gg = \emptyset$, then $f = g$.*

Proof. This proof depends on the Standard-Form Lemma above. We write down f in the standard form $f_2 \circ f_1$ for $f_1 : A \vdash C$ and g in the standard form $g_2 \circ g_1$ for $g_1 : A \vdash D$. Since $\check{k}^i$ and $\check{\kappa}$ do not occur in f_1, for every occurrence z of a propositional letter in C we have an occurrence x of that propositional letter in A such that $(x, z) \in Gf_1$, and since $\hat{k}^i$ and $\hat{\kappa}$ do not occur in f_2, for every occurrence z of a propositional letter in C we have an occurrence y of that propositional letter in B such that $(z, y) \in Gf_2$. So if C were not letterless, then Gf would not be empty. We conclude analogously that D, as well as C, is a letterless formula.

If both C and D are isomorphic to $\top$ or $\bot$, then we have an isomorphism $i : C \vdash D$, and $f = f_2 \circ i^{-1} \circ i \circ f_1$. By Lemma 1, we have $i \circ f_1 = g_1$ and $f_2 \circ i^{-1} = g_2$, from which $f = g$ follows. If i: $C \vdash \bot$ and j: $\top \vdash D$ are isomorphisms, then by Lemma 1 we have

$$f_2 \circ f_1 = g_2 \circ j \circ \hat{\kappa}_\bot \circ i \circ f_1 = g_2 \circ g_1,$$

and so $f = g$. (Note that $\hat{\kappa}_\bot = \check{\kappa}_\top$.) $\dashv$

We can then prove the following.

Sesquicartesian Coherence. *The functor G from $\mathbf{L}_\perp$ to* Rel *is faithful.*

Proof. We have Lemma 2 for the case when $Gf = Gg = \emptyset$. When $Gf = Gg \neq \emptyset$, we proceed as in the proof of Lattice Coherence, appealing if need there is to Lemma 2, until we reach the case when A is $A_1 \wedge A_2$ or a letter, and B is $B_1 \vee B_2$ or a letter, but A and B are not both letters. In that case, by Composition Elimination, the arrow term f is equal in $\mathbf{L}_\perp$ either to an arrow term of the form $f' \circ \hat{k}^i_{A_1,A_2}$, or to an arrow term of the form $\check{k}^i_{B_1,B_2} \circ f'$. Suppose $f = f' \circ \hat{k}^1_{A_1,A_2}$. Then for every $(x,y) \in Gf$ we have that x is in A_1. (We reason analogously when $f = f' \circ \hat{k}^2_{A_1,A_2}$.)

By Composition Elimination too, g is equal in $\mathbf{L}_\perp$ either to an arrow term of the form $g' \circ \hat{k}^i_{A_1,A_2}$, or to an arrow term of the form $\check{k}^i_{B_1,B_2} \circ g'$. In the first case we must have $g = g' \circ \hat{k}^1_{A_1,A_2}$, because $Gg = G(f' \circ \hat{k}^1_{A_1,A_2}) \neq \emptyset$, and then we apply the induction hypothesis to derive $f' = g'$ from $Gf' = Gg'$. Hence $f = g$ in $\mathbf{L}_\perp$.

Suppose $g = \check{k}^1_{B_1,B_2} \circ g'$. (We reason analogously when $g = \check{k}^2_{B_1,B_2} \circ g'$.) Let $f'' \colon A_1 \vdash B_1 \vee B_2''$ be the substitution instance of $f' \colon A_1 \vdash B_1 \vee B_2$ obtained by replacing every occurrence of propositional letter in B_2 by $\perp$. There is an isomorphism $i \colon B_2'' \vdash \perp$, and f'' exists because in Gf, which is equal to $G(\check{k}^1_{B_1,B_2} \circ g')$, there is no pair (x,y) with y in B_2. So we have an arrow $f''' \colon A_1 \vdash B_1$, which we define as $[\mathbf{1}_{B_1}, \check{\kappa}_{B_1}] \circ (\mathbf{1}_{B_1} \vee i) \circ f''$. It is easy to verify that $G(\check{k}^1_{B_1,B_2} \circ f''') = Gf'$, and that $G(f''' \circ \hat{k}^1_{A_1,A_2}) = Gg'$. By the induction hypothesis, we obtain $\check{k}^1_{B_1,B_2} \circ f''' = f'$ and $f''' \circ \hat{k}^1_{A_1,A_2} = g'$, from which we derive $f = g$. We reason analogously when $f = \check{k}^i_{B_1,B_2} \circ f'$. ⊣

From Sesquicartesian Coherence we infer coherence for $\mathbf{L}_\top$, which is isomorphic to $\mathbf{L}_\perp^{op}$.

7. Restricted coherence for dicartesian categories

For dicartesian categories we can prove easily a simple restricted coherence result, which was sufficient for the needs of [14]. A more general, but still restricted, coherence result with respect to *Rel*, falling short of full coherence, may be found in the revised version of [10, Section 7]. We present first the simple restricted coherence result, and will deal with the more general restricted coherence result later on.

We define inductively formulae of $\mathcal{L}_{\top,\perp}$ in *disjunctive normal form* (*dnf*): every $\vee$-free formula is in *dnf*, and if A and B are both in *dnf*, then $A \vee B$ is in *dnf*. We define dually formulae of $\mathcal{L}_{\top,\perp}$ in *conjunctive normal form* (*cnf*): every $\wedge$-free formula is in *cnf*, and if A and B are both in *cnf*, then $A \wedge B$ is in *cnf*.

Restricted Dicartesian Coherence. *Let $f,g \colon A \vdash B$ be arrow terms of $\mathbf{L}_{\top,\perp}$ such that A is in* dnf *and B in* cnf. *If $Gf = Gg$, then $f = g$ in $\mathbf{L}_{\top,\perp}$.*

Proof. If $Gf = Gg = \emptyset$, then we apply Lemma 2. If $Gf = Gg \neq \emptyset$, then we proceed as in the proof of Lattice Coherence, by induction on the sum of the lengths of A and B, appealing if need there is to Lemma 2, until we reach the case when A is

$A_1 \wedge A_2$ or a letter, and B is $B_1 \vee B_2$ or a letter, but A and B are not both letters. In that case there is no occurrence of $\vee$ in A and no occurrence of $\wedge$ in B. We then rely on the composition-free form of f and g in $\mathbf{L}_{\top,\bot}$ and on the equation $(\hat{K}\check{K})$. ⊣

To improve upon this result we need the following lemma for $\mathbf{L}_{\top,\bot}$, and the definitions that follow. This lemma is analogous up to a point to the Invertibility Lemma for $\vee$.

Lemma 3. *Let $f : A \vdash B_1 \vee B_2$ be a Gentzen term such that $Gf \neq \emptyset$ and $\vee$ does not occur in A. If for every $(x, y) \in Gf$ we have that y is in B_1, then there is a Gentzen term $g : A \vdash B_1$ such that $Gf = G\check{K}^1_{B_2} g$.*

Proof. We proceed by induction on the length of A. Suppose f is a composition-free Gentzen term. If A is a propositional letter, then by the assumption on Gf we have that f is of the form $\check{K}^1_{B_2} f'$, and we can take that g is f'.

If A is not a propositional letter and f is not of the form $\check{K}^1_{B_2} f'$ (by the assumption on Gf, the Gentzen term f cannot be of the form $\check{K}^2_{B_1} f'$), then, since $\vee$ does not occur in A, we have that f is of the form $\hat{K}^i_{A''} f'$ for $f' : A' \vdash B_1 \vee B_2$. Note that $Gf' \neq \emptyset$ and $\vee$ does not occur in A'. Since for every (x, y) in Gf' we have that y is in B_1, we may apply the induction hypothesis to f' and obtain $g' : A' \vdash B_1$ such that $Gf' = G\check{K}^1_{B_2} g'$. By relying on the equation $(\hat{K}\check{K})$, we can take that g is $\hat{K}^i_{A''} g'$. ⊣

A formula C of $\mathcal{L}_{\top,\bot}$ is called a *contradiction* when there is in $\mathbf{L}_{\top,\bot}$ an arrow of the type $C \vdash \bot$. For every formula that is not a contradiction there is a substitution instance isomorphic to $\top$. Suppose C is not a contradiction, and let $C^\top$ be obtained from C by substituting $\top$ for every propositional letter. If $C^\top$ were not isomorphic to $\top$, then since every letterless formula of $\mathcal{L}_{\top,\bot}$ is isomorphic in $\mathbf{L}_{\top,\bot}$ either to $\top$ or to $\bot$, we would have an isomorphism $i : C^\top \vdash \bot$. Since there is obviously an arrow $u : C \vdash C^\top$ formed by using $\hat{\kappa}_p$, we would have $i \circ u : C \vdash \bot$, and C would be a contradiction.

A formula C of $\mathcal{L}_{\top,\bot}$ is called a *tautology* when there is in $\mathbf{L}_{\top,\bot}$ an arrow of the type $\top \vdash C$. For every formula that is not a tautology there is a substitution instance isomorphic to $\bot$. (This is shown analogously to what we had in the preceding paragraph.)

A formula of $\mathcal{L}_{\top,\bot}$ is called $\bot$-*normal* when for every subformula $D \wedge C$ or $C \wedge D$ of it with C a contradiction, there is no occurrence of $\vee$ in D. A formula of $\mathcal{L}_{\top,\bot}$ is called $\top$-*normal* when for every subformula $D \vee C$ or $C \vee D$ of it with C a tautology, there is no occurrence of $\wedge$ in D.

We can now formulate our second partial coherence result for dicartesian categories.

Restricted Dicartesian Coherence II. *If $f, g : A \vdash B$ are terms of $\mathbf{L}_{\top,\bot}$ such that $Gf = Gg$ and either A is $\bot$-normal or B is $\top$-normal, then $f = g$ in $\mathbf{L}_{\top,\bot}$.*

Proof. Suppose A is $\bot$-normal. Lemma 2 covers the case when $Gf = Gg = \emptyset$. So we assume $Gf = Gg \neq \emptyset$, and proceed as in the proof of Sesquicartesian Coherence by induction on the sum of the lengths of A and B. The basis of this induction and the cases when A is of the form $A_1 \vee A_2$ or B is of the form $B_1 \wedge B_2$ are settled as in the proof of Sesquicartesian Coherence.

Suppose A is $A_1 \wedge A_2$ or a propositional letter and B is $B_1 \vee B_2$ or a propositional letter, but A and B are not both propositional letters. (The cases when A or B is a constant object are excluded by the assumption that $Gf = Gg \neq \emptyset$.) We proceed then as in the proof of Sesquicartesian Coherence until we reach the case when $f = f' \circ \hat{k}^1_{A_1,A_2}$ and $g = \check{k}^1_{B_1,B_2} \circ g'$.

Suppose A_2 is not a contradiction. Then there is an instance $A_2^\top$ of A_2 and an isomorphism $i : \top \vdash A_2^\top$. (To obtain $A_2^\top$ we substitute $\top$ for every letter in A_2.) Let $g'' : A_1 \wedge A_2^\top \vdash B_1$ be the substitution instance of $g' : A_1 \wedge A_2 \vdash B_1$ obtained by replacing every occurrence of propositional letter in A_2 by $\top$. Such a term exists because in Gg, which is equal to $G(f' \circ \hat{k}^1_{A_1,A_2})$, there is no pair (x, y) with x in A_2.

So we have an arrow $g''' = g'' \circ (\mathbf{1}_{A_1} \wedge i) \circ \langle \mathbf{1}_{A_1}, \hat{\kappa}_{A_1} \rangle : A_1 \vdash B_1$. It is easy to verify that $G(\check{k}^1_{B_1,B_2} \circ g''') = Gf'$ and that $G(g''' \circ \hat{k}^1_{A_1,A_2}) = Gg'$. By the induction hypothesis we obtain $\check{k}^1_{B_1,B_2} \circ g''' = f'$ and $g''' \circ \hat{k}^1_{A_1,A_2} = g'$, from which we derive $f = g$.

Suppose A_2 is a contradiction. Then by the assumption that A is $\bot$-normal we have that $\vee$ does not occur in A_1. We may apply Lemma 3 to $f' : A_1 \vdash B_1 \vee B_2$ to obtain $f''' : A_1 \vdash B_1$ such that $Gf' = G(\check{k}^1_{B_1,B_2} \circ f''')$. It is easy to verify that then $Gg' = G(f''' \circ \hat{k}^1_{A_1,A_2})$, and we may proceed as in the proof of Sesquicartesian Coherence.

We proceed analogously when B is $\top$-normal, relying on a lemma dual to Lemma 3. $\dashv$

Consider the following definitions:

$$A^0_\bot = A \wedge \bot, \qquad A^{n+1}_\bot = (A^n_\bot \vee \top) \wedge \bot,$$
$$f^0_\bot = f \wedge \mathbf{1}_\bot, \qquad f^{n+1}_\bot = (f^n_\bot \vee \mathbf{1}_\top) \wedge \mathbf{1}_\bot,$$
$$A^0_\top = A \vee \top, \qquad A^{n+1}_\top = (A^n_\top \wedge \bot) \vee \top,$$
$$f^0_\top = f \vee \mathbf{1}_\top, \qquad f^{n+1}_\top = (f^n_\top \wedge \mathbf{1}_\bot) \vee \mathbf{1}_\top.$$

Then for f^n being

$$(\check{k}^1_{A,\top} \wedge \mathbf{1}_\bot)^n_\top \circ \hat{k}^1_{(A\wedge\bot)^n_\top,\bot} : A^{n+1}_\bot \vdash A^{n+1}_\top$$

and g^n being

$$\check{k}^1_{(A\vee\top)^n_\bot,\top} \circ (\hat{k}^1_{A,\bot} \vee \mathbf{1}_\top)^n_\bot : A^{n+1}_\bot \vdash A^{n+1}_\top$$

we have $Gf^n = Gg^n$, but we suppose that $f^n = g^n$ does not hold in $\mathbf{L}_{\top,\bot}$. The equation $f^0 = g^0$ is

$$((\check{k}^1_{A,\top} \wedge \mathbf{1}_\bot) \vee \mathbf{1}_\top) \circ \hat{k}^1_{(A\wedge\bot)\vee\top,\bot} = \check{k}^1_{(A\vee\top)\wedge\bot,\top} \circ (\hat{k}^1_{A,\bot} \vee \mathbf{1}_\top) \wedge \mathbf{1}_\bot :$$
$$((A \wedge \bot) \vee \top) \wedge \bot \vdash ((A \vee \top) \wedge \bot) \vee \top.$$

Note that $A_{\bot}^{n+1}$ is not $\bot$-normal, and $A_{\top}^{n+1}$ is not $\top$-normal.

We don't know whether it is sufficient to add to $\mathbf{L}_{\top,\bot}$ the equations $f^n = g^n$ for every $n \geqslant 0$ in order to obtain full coherence for the resulting category.

As a corollary of Restricted Dicartesian Coherence II, we obtain that if $f, g : A \vdash B$ are terms of $\mathbf{L}_{\top,\bot}$ such that $Gf = Gg$, while A and B are isomorphic either to formulae of $\mathcal{L}$ (i.e. to formulae in which $\top$ and $\bot$ do not occur) or to letterless formulae, then $f = g$ in $\mathbf{L}_{\top,\bot}$. This corollary is analogous to the restricted coherence result for symmetric monoidal closed categories of Kelly and Mac Lane in [19] (see [15, Section 3.1]).

8. Maximality

A syntactically built category such as $\mathbf{L}$ and $\mathbf{L}_{\top,\bot}$ is called *maximal* when adding any new axiomatic equation between arrow terms of this category yields a category that is a preorder. The new axiomatic equation is supposed to be closed under substitution for propositional letters, as the equations of $\mathbf{L}$ and $\mathbf{L}_{\top,\bot}$ were. (This notion of maximality for syntactical categories is defined more precisely in [14, Section 9.3].) Maximality is an interesting property when the initial category, like $\mathbf{L}$ and $\mathbf{L}_{\top,\bot}$ here, is not itself a preorder. We will deal in subsequent sections with maximality for $\mathbf{L}$ and $\mathbf{L}_{\top,\bot}$.

The maximality property above is analogous to the property of usual formulations of the classical propositional calculus called *Post completeness*. That this calculus is Post complete means that if we add to it any new axiom-schema in the language of the calculus, then we can prove every formula. An analogue of Böhm's Theorem in the typed lambda calculus implies, similarly, that the typed lambda calculus cannot be extended without falling into triviality, i.e. without every equation (between terms of the same type) becoming derivable (see [26], [8] and references therein; see [1, Section 10.4], for Böhm's Theorem in the untyped lambda calculus).

Let us now consider several examples of common algebraic structures with analogous maximality properties. First, we have that semilattices are maximal in the following sense.

Let a and b be terms made exclusively of variables and of a binary operation $\cdot$, which we interpret as meet or join. That the equation $a = b$ holds in a semilattice S means that *every* instance of $a = b$ obtained by substituting names of elements of S for variables holds in S. Suppose $a = b$ does not hold in a free semilattice S_F (so it is not the case that $a = b$ holds in every semilattice). Hence there must be an instance of $a = b$ obtained by substituting names of elements of S_F for variables such that this instance does not hold in S_F. It is easy to conclude that in $a = b$ there must be at least two variables, and that S_F must have at least two free generators. Then every semilattice in which $a = b$ holds is trivial—namely, it has a single element.

Here is a short proof of that. If $a = b$ does not hold in S_F, then there must be a variable x in one of a and b that is not in the other. Then from $a = b$, by substituting y for every variable in a and b different from x, and by applying the

semilattice equations, we infer either $x = y$ or $x \cdot y = y$. If we have $x = y$, we are done, and, if we have $x \cdot y = y$, then we have also $y \cdot x = x$, and hence $x = y$.

Semilattices with unit, distributive lattices, distributive lattices with top and bottom, and Boolean algebras are maximal in the same sense. The equations $a = b$ in question are equations between terms made exclusively of variables and the operations of the kind of algebra we envisage: semilattices with unit, distributive lattices, etc. That such an equation holds in a particular structure means, as above, that every substitution instance of it holds. However, the number of variables in $a = b$ and the number of generators of the free structure mentioned need not always be at least two.

If we deal with semilattices with unit $\mathbf{1}$, then $a = b$ must have at least one variable, and the free semilattice with unit must have at least one free generator. We substitute $\mathbf{1}$ for every variable in a and b different from x in order to obtain $x = \mathbf{1}$, and hence triviality. So semilattices with unit are maximal in the same sense.

The same sort of maximality can be proven for distributive lattices, whose operations are $\wedge$ and $\vee$, which we call conjunction and disjunction, respectively. Then every term made of $\wedge$, $\vee$ and variables is equal to a term in disjunctive normal form (i.e. a multiple disjunction of multiple conjunctions of variables; see the preceding section for a precise definition), and to a term in conjunctive normal form (i.e. a multiple conjunction of multiple disjunctions of variables; see the preceding section). These normal forms are not unique. If $a = b$, in which we must have at least two variables, does not hold in a free distributive lattice D_F with at least two free generators, then either $a \leqslant b$ or $b \leqslant a$ does not hold in D_F. Suppose $a \leqslant b$ does not hold in D_F. Let a' be a disjunctive normal form of a, and let b' be a conjunctive normal form of b. So $a' \leqslant b'$ does not hold in D_F. From that we infer that for a disjunct a'' of a' and for a conjunct b'' of b' we do not have $a'' \leqslant b''$ in D_F. This means that there is no variable in common in a'' and b''; otherwise, the conjunction of variables a'' would be lesser than or equal in D_F to the disjunction of variables b''. If in a distributive lattice $a = b$ holds, then $a'' \leqslant b''$ holds too, and hence, by substitution, we obtain $x \leqslant y$. So $x = y$.

For distributive lattices with top $\top$ and bottom $\bot$, we proceed analogously via disjunctive and conjunctive normal form. Here $a = b$ may be even without variables, and the free structure may have even an empty set of free generators. The additional cases to consider are when in $a'' \leqslant b''$ we have that a'' is $\top$ and b'' is $\bot$. In any case, we obtain $\top \leqslant \bot$, and hence our structure is trivial.

The same sort of maximality can be proven for Boolean algebras, i.e. complemented distributive lattices. Boolean algebras must have top and bottom. In a disjunctive normal form now the disjuncts are conjunctions of variables x or terms $\bar{x}$, where $^-$ is complementation, or the disjunctive normal form is just $\top$ or $\bot$; analogously for conjunctive normal forms. Then we proceed as for distributive lattices with an equation $a = b$ that may be even without variables, until we reach that $a'' \leqslant b''$, which does not hold in a free Boolean algebra B_F, whose set of free generators may be even empty, holds in our Boolean algebra. If x is a conjunct of a'', then in b'' we cannot have a disjunct x; but we may have a disjunct $\bar{x}$. The same

holds for the conjuncts $\bar{x}$ of a''. It is excluded that both x and $\bar{x}$ are conjuncts of a'', or disjuncts of b''; otherwise, $a'' \leqslant b''$ would hold in B_F. Then for every conjunct x in a'' and every disjunct $\bar{y}$ in b'' we substitute $\top$ for x and y, and for every other variable we substitute $\bot$. In any case, we obtain $\top \leqslant \bot$, and hence our Boolean algebra is trivial. This is essentially the proof of Post completeness for the classical propositional calculus, due to Bernays and Hilbert (see [28, Section 2.4], and [16, Section I.13]), from which we can infer the ordinary completeness of this calculus with respect to valuations in the two-element Boolean algebra—namely, with respect to truth tables—and also completeness with respect to any nontrivial model.

As examples of common algebraic structures that are not maximal in the sense above, we have semigroups, commutative semigroups, lattices, and many others. What is maximal for semilattices and is not maximal for lattices is the equational theory of the structures in question. The equational theory of semilattices cannot be extended without falling into triviality, while the equational theory of lattices can be extended with the distributive law, for example.

The notions of maximality envisaged in this section were extreme (or should we say "maximal"), in the sense that we envisaged collapsing only into preorder. For semilattices, distributive lattices, etc., this is also preorder for a one-object category. We may, however, envisage relativizing our notion of maximality by replacing preorder with a weaker property, such that structures possessing it are trivial, but not so trivial (cf. [7, Section 4.11]). We will encounter maximality in such a relative sense in the last section.

As an example of relative maximality in a common algebraic structure we can take symmetric groups. Consider the standard axioms for the symmetric group $\mathcal{S}_n$, where $n \geqslant 2$, with the generators s_i, for $i \in \{1, \ldots, n-1\}$, corresponding to transpositions of immediate neighbours (see [6, Section 6.2]). If to $\mathcal{S}_n$ for $n \geqslant 5$ we add an equation $a = \mathbf{1}$ where a is built exclusively of the generators s_i of $\mathcal{S}_n$ with composition, and $a = \mathbf{1}$ does not hold in $\mathcal{S}_n$, then we can derive $s_i = s_j$. This does not mean that the resulting structure will be a one-element structure, i.e. the trivial one-element group. It will be such if a is an odd permutation, and if a is an even permutation, then we will obtain a two-element structure, which is $\mathcal{S}_2$. This can be inferred from facts about the normal subgroups of $\mathcal{S}_n$. Simple groups are maximal in the nonrelative sense, envisaged above for semilattices.

9. Maximality of lattice categories

We will show in this section that $\mathbf{L}$ is maximal in the sense specified at the beginning of the preceding section; namely, in the interesting way. (We take over this result from [11, Section 5], and [14, Section 9.5].)

Suppose A and B are formulae of $\mathcal{L}$ in which only p occurs as a letter. If for some arrow terms $f_1, f_2 : A \vdash B$ of $\mathbf{L}$ we have $Gf_1 \neq Gf_2$, then for some x in A and some y in B we have $(x, y) \in Gf_1$ and $(x, y) \notin Gf_2$, or vice versa. Suppose $(x, y) \in Gf_1$ and $(x, y) \notin Gf_2$.

For every subformula C of A and every formula D let A^C_D be the formula obtained from A by replacing the particular occurrence of the formula C in A by D. It can be shown that for every subformula $A_1 \vee A_2$ of A we have an arrow term $h\colon A^{A_1 \vee A_2}_{A_j} \vdash A$ of $\mathbf{L}$, built by using $\check{k}^j_{A_1,A_2}$, such that there is an x' in $A^{A_1 \vee A_2}_{A_j}$ for which $(x', x) \in Gh$. Hence, for such an h, we have $(x', y) \in G(f_1 \circ h)$ and $(x', y) \notin G(f_2 \circ h)$.

We compose f_i repeatedly with such arrow terms until we obtain the arrow terms $f'_i\colon p \wedge \ldots \wedge p \vdash B$ of $\mathbf{L}$ such that parentheses are somehow associated in $p \wedge \ldots \wedge p$ and for some z in $(p \wedge \ldots \wedge p)$ we have $(z, y) \in Gf'_1$ and $(z, y) \notin Gf'_2$. The formula $p \wedge \ldots \wedge p$ may also be only p. We may further compose f'_i with other arrow terms of $\mathbf{L}$ in order to obtain the arrow terms f''_i of type $p \wedge A' \vdash B$ or $p \vdash B$ such that A' is of the form $p \wedge \ldots \wedge p$ with parentheses somehow associated. Let us use 0 to denote the first occurrence of a propositional letter in a formula, counting from the left. So we have $(0, y) \in Gf''_1$ but $(0, y) \notin Gf''_2$.

By working dually on B we obtain the arrow terms f'''_i of $\mathbf{L}$ of type $p \wedge A' \vdash p \vee B'$, for A' of the form $p \wedge \ldots \wedge p$ and B' of the form $p \vee \ldots \vee p$, or of type $p \wedge A' \vdash p$, or of type $p \vdash p \vee B'$, such that $(0, 0) \in Gf'''_1$ and $(0, 0) \notin Gf'''_2$. (We cannot obtain that f'''_1 and f'''_2 are of type $p \vdash p$, since, otherwise, by Composition Elimination for $\mathbf{L}$, f'''_2 would not exist.)

There is an arrow term $h^\wedge\colon p \vdash p \wedge \ldots \wedge p$ of $\mathbf{L}$ defined by using $\hat{w}$ such that for every $x \in G(p \wedge \ldots \wedge p)$ we have $(0, x) \in Gh^\wedge$. We define analogously with the help of $\check{w}$ an arrow term $h^\vee\colon p \vee \ldots \vee p \vdash p$ of $\mathbf{L}$ such that for every x in $p \vee \ldots \vee p$ we have $(x, 0) \in Gh^\vee$. The arrow terms $h^\wedge$ and $h^\vee$ may be $\mathbf{1}_p\colon p \vdash p$.

If f'''_i is of type $p \wedge A' \vdash p \vee B'$, let $f^\dagger_i\colon p \wedge p \vdash p \vee p$ be defined by

$$f^\dagger_i =_{df} (\mathbf{1}_p \vee h^\vee) \circ f'''_i \circ (\mathbf{1}_p \vee h^\wedge).$$

By Composition Elimination for $\mathbf{L}$, we have that $Gf^\dagger_i$ must be a singleton. Let us use 1 to denote the second occurrence of a propositional letter in a formula, counting from the left. If $(1, 0)$ or $(1, 1)$ belongs to $Gf^\dagger_2$, then for $f^*_i : p \wedge p \vdash p$ defined as $\check{w}_p \circ f^\dagger_i$ we have $(0, 0) \in Gf^*_1$ and $(0, 0) \notin Gf^*_2$. If $(0, 1)$ or $(1, 1)$ belongs to $Gf^\dagger_2$, then for $f^*_i : p \vdash p \vee p$ defined as $f^\dagger_i \circ \hat{w}_p$ we have $(0, 0) \in Gf^*_1$ and $(0, 0) \notin Gf^*_2$.

If f'''_i is of type $p \wedge A' \vdash p$, then for $f^*_i : p \wedge p \vdash p$ defined as $f'''_i \circ (\mathbf{1}_p \vee h^\wedge)$ we have $(0, 0) \in Gf^*_1$ and $(0, 0) \notin Gf^*_2$.

If f'''_i is of type $p \vdash p \vee B'$, then for $f^*_i : p \vdash p \vee p$ defined as $(\mathbf{1}_p \vee h^\vee) \circ f'''_i$ we have $(0, 0) \in Gf^*_1$ and $(0, 0) \notin Gf^*_2$. In all that we have by Composition Elimination for $\mathbf{L}$ that Gf^*_i must be a singleton.

In cases where f^*_i is of type $p \wedge p \vdash p$, by Composition Elimination for $\mathbf{L}$, by the conditions on Gf^*_1 and Gf^*_2, and by the functoriality of G, we obtain in $\mathbf{L}$ the equation $f^*_i = \hat{k}^i_{p,p}$. (This follows from Lattice Coherence too.) So in $\mathbf{L}$ extended with $f_1 = f_2$ we can derive the equation

$$(\hat{k}\hat{k}) \quad \hat{k}^1_{p,p} = \hat{k}^2_{p,p}.$$

In cases where f^*_i is of type $p \vdash p \vee p$, we conclude analogously that we have in $\mathbf{L}$ the equation $f^*_i = \check{k}^i_{p,p}$, and so in $\mathbf{L}$ extended with $f_1 = f_2$ we can derive

$$(\check{k}\check{k}) \quad \check{k}^1_{p,p} = \check{k}^2_{p,p}.$$

If either of $(\hat{k}\hat{k})$ and $(\check{k}\check{k})$ holds in a lattice category $\mathcal{A}$, then $\mathcal{A}$ is a preorder.

It remains to remark that if for some arrow terms g_1 and g_2 of $\mathbf{L}$ of the same type we have that $g_1 = g_2$ does not hold for $\mathbf{L}$, then by Lattice Coherence we have $Gg_1 \neq Gg_2$. If we take the substitution instances g_1' of g_1 and g_2' of g_2 obtained by replacing every letter by a single letter p, then we obtain again $Gg_1' \neq Gg_2'$. If $g_1 = g_2$ holds in a lattice category $\mathcal{A}$, then $g_1' = g_2'$ holds too, and $\mathcal{A}$ is a preorder, as we have shown above. This concludes the proof of maximality for $\mathbf{L}$. (In the original presentation of this proof in [11, Section 5], there are some slight inaccuracies in the definition of f_i^*.)

10. Relative maximality of dicartesian categories

The category $\mathbf{L}_{\top,\bot}$ is not maximal in the sense in which $\mathbf{L}$ is. This is shown by the following counterexample.

Let Set_* be the category whose objects are sets with a distinguished element $*$, and whose arrows are $*$-preserving functions f between these sets; namely, $f(*) = *$. This category is isomorphic to the category of sets with partial functions. The following definitions serve to show that Set_* is a category in which we can interpret the objects and arrow terms of $\mathbf{L}_{\top,\bot}$:

$$\mathrm{I} = \{*\}, \qquad a' = \{(x,*) \mid x \in a - I\}, \qquad b'' = \{(*,y) \mid y \in b - \mathrm{I}\},$$

$$\begin{aligned} a \otimes b &= ((a - \mathrm{I}) \wedge (b - \mathrm{I})) \cup \mathrm{I}, \\ a \boxtimes b &= (a \otimes b) \cup a' \cup b'', \\ a \boxplus b &= a' \cup b'' \cup \mathrm{I}. \end{aligned}$$

Note that $a \boxtimes b$ is isomorphic in Set to the cartesian product $a \times b$; the element $*$ of $a \boxtimes b$ corresponds to the element $(*,*)$ of $a \times b$.

The functions $\hat{k}^i_{a_1,a_2} : a_1 \boxtimes a_2 \to a_i$, for $i \in \{1,2\}$, are defined by

$$\hat{k}^i_{a_1,a_2}(x_1,x_2) = x_i, \qquad \hat{k}^i_{a_1,a_2}(*) = *;$$

for $f_i : c \to a_i$, the function $\langle f_1, f_2 \rangle : c \to a_1 \boxtimes a_2$ is defined by

$$\langle f_1, f_2 \rangle(z) = \begin{cases} (f_1(z), f_2(z)) & \text{if } f_1(z) \neq * \text{ or } f_2(z) \neq * \\ * & \text{if } f_1(z) = f_2(z) = *; \end{cases}$$

and the function $\hat{\kappa}_a : a \to \mathrm{I}$ is defined by $\hat{\kappa}_a(x) = *$. Having in mind the isomorphism between $a \boxtimes b$ and $a \times b$ mentioned above, the functions $\hat{k}^i_{a_1,a_2} : a_1 \boxtimes a_2 \to a_i$ correspond to the projection functions, while $\langle \_, \_ \rangle$ corresponds to the usual pairing operation on functions.

The functions $\check{k}^i_{a_1,a_2} : a_i \to a_1 \boxplus a_2$ are defined by

$$\begin{aligned} &\check{k}^1_{a_1,a_2}(x) = (x,*), \quad \check{k}^2_{a_1,a_2}(x) = (*,x), \quad \text{for } x \neq *, \\ &\check{k}^i_{a_1,a_2}(*) = *; \end{aligned}$$

for $g_i\colon a_i \to c$, the function $[g_1, g_2]\colon a_1 \boxplus a_2 \to c$ is defined by

$$[g_1, g_2](x_1, x_2) = g_i(x_i), \text{ for } x_i \neq *,$$
$$[g_1, g_2](*) = *;$$

finally, the function $\check{\kappa}_a : \mathrm{I} \to a$ is defined by $\check{\kappa}_a(*) = *$.

If we take that $\wedge$ is $\boxtimes$ and $\vee$ is $\boxplus$ then it can be checked in a straightforward manner that Set_* and Set_* without I are lattice categories, and if in Set_* we take further that both $\top$ and $\bot$ are I, then Set_* is a dicartesian category.

Consider now the category $Set_*^{\emptyset}$, which is obtained by adding to Set_* the empty set $\emptyset$ as a new object, and the empty functions $\emptyset_a : \emptyset \to a$ as new arrows. The identity arrow $\mathbf{1}_\emptyset$ is $\emptyset_\emptyset$. For $Set_*^{\emptyset}$, we enlarge the definitions above by

$$\begin{aligned}
\emptyset \boxtimes a &= a \boxtimes \emptyset = \emptyset,\\
\emptyset \boxplus a &= a \boxplus \emptyset = a,\\
\hat{k}^i_{a_1,a_2} &= \emptyset_{a_i}, \text{ for } a_1 = \emptyset \text{ or } a_2 = \emptyset,\\
\langle \emptyset_{a_1}, \emptyset_{a_2} \rangle &= \emptyset_{a_1 \boxtimes a_2},\\
\hat{\kappa}_\emptyset &= \emptyset_{\mathrm{I}},\\
\check{k}^i_{a_1,a_2} &= \emptyset_{a_1 \boxplus a_2}, \text{ for } a_i = \emptyset,\\
[f_1, \emptyset_c] &= f_1, \qquad [\emptyset_c, f_2] = f_2,
\end{aligned}$$

and define now the function $\check{\kappa}_a : \emptyset \to a$ by $\check{\kappa}_a = \emptyset_a$. Then it can be checked that $Set_*^{\emptyset}$ where $\wedge$ is $\boxtimes$ and $\vee$ is $\boxplus$ as before, while $\top$ is I and $\bot$ is $\emptyset$, is a dicartesian category too.

In $\mathbf{L}_{\top,\bot}$ the equation $\hat{k}^1_{p,\bot} = \check{\kappa}_p \circ \hat{k}^2_{p,\bot}$ does not hold, because $G\hat{k}^1_{p,\bot} \neq \emptyset$ and $G(\check{\kappa}_p \circ \hat{k}^2_{p,\bot}) = \emptyset$, but in $Set_*^{\emptyset}$ this equation holds, because both sides are equal to $\emptyset_\emptyset$. Since $Set_*^{\emptyset}$ is not a preorder, we can conclude that $\mathbf{L}_{\top,\bot}$ is not maximal.

Although this maximality fails, the category $\mathbf{L}_{\top,\bot}$ may be shown maximal in a relative sense. This relative maximality result, which we are going to demonstrate now, says that every dicartesian category that satisfies an equation $f = g$ between arrow terms of $\mathbf{L}_{\top,\bot}$ such that $Gf \neq Gg$ (which implies that $f = g$ is not in $\mathbf{L}_{\top,\bot}$) satisfies also some particular equations. These equations do not give preorder in general, but a kind of "contextual" preorder. Moreover, when $\mathbf{L}_{\top,\bot}$ is extended with some of these equations we obtain a maximal category.

If for some arrow terms $f_1, f_2 : A \vdash B$ of $\mathbf{L}_{\top,\bot}$ we have $Gf_1 \neq Gf_2$, then for some x in A and some y in B we have $(x,y) \in Gf_1$ and $(x,y) \notin Gf_2$, or vice versa. Suppose $(x,y) \in Gf_1$ and $(x,y) \notin Gf_2$. Suppose x is an occurrence of p, so that y must be an occurrence of p too.

Let A' be the formula obtained from the formula A by replacing x by $p \wedge \bot$, and every other occurrence of letter or $\top$ by $\bot$. Dually, let B' be the formula obtained from B by replacing y by $p \vee \top$, and every other occurrence of letter or $\bot$ by $\top$. Let us use 0, as in the preceding section, to denote the first occurrence of a propositional letter in a formula, counting from the left. Then it can be shown that there is an arrow term $h^A : A' \vdash A$ of $\mathbf{L}_{\top,\bot}$ such that $Gh^A = \{(0,x)\}$, and an arrow term $h^B : B \vdash B'$ of $\mathbf{L}_{\top,\bot}$ such that $Gh^B = \{(y,0)\}$. We build h^A with

$\hat{k}^1_{p,\bot} : p \wedge \bot \vdash p$ and instances of $\check{\kappa}_C : \bot \vdash C$, with the help of the operations $\wedge$ and $\vee$ on arrow terms. Analogously, h^B is built with $\check{k}^1_{p,\top} : p \vdash p \vee \top$ and instances of $\hat{\kappa}_C : C \vdash \top$. It can also be shown that there are arrow terms $j^A : p \wedge \bot \vdash A'$ and $j^B : B' \vdash p \vee \top$ of $\mathbf{L}_{\top,\bot}$ such that $Gj^A = Gj^B = \{(0,0)\}$. These arrow terms stand for isomorphisms of $\mathbf{L}_{\top,\bot}$.

Then it is clear that for f'_i being

$$j^B \circ h^B \circ f_i \circ h^A \circ j^A : p \wedge \bot \vdash p \vee \top,$$

with $i \in \{1,2\}$, we have $Gf'_1 = \{(0,0)\}$, while $Gf'_2 = \emptyset$. Hence, by Composition Elimination for $\mathbf{L}_{\top,\bot}$ and by the functoriality of G, we obtain in $\mathbf{L}_{\top,\bot}$ the equations

$$\begin{aligned} f'_1 &= \check{k}^1_{p,\top} \circ \hat{k}^1_{p,\bot}, \\ f'_2 &= \check{\kappa}_{p \vee \top} \circ \hat{k}^2_{p,\bot} \;=\; \check{k}^2_{p,\top} \circ \hat{\kappa}_{p \wedge \bot}. \end{aligned}$$

(This follows from Restricted Dicartesian Coherence too.) If we write $\mathbf{0}_{\bot,\top}$ for $\hat{\kappa}_\bot$, which is equal to $\check{\kappa}_\top$ in $\mathbf{L}_{\top,\bot}$, then in $\mathbf{L}_{\top,\bot}$ we have

$$f'_2 = \check{k}^2_{p,\top} \circ \mathbf{0}_{\bot,\top} \circ \hat{k}^2_{p,\bot}.$$

So in $\mathbf{L}_{\top,\bot}$ extended with $f_1 = f_2$ we can derive

$$(\hat{k}\check{k}) \quad \check{k}^1_{p,\top} \circ \hat{k}^1_{p,\bot} \;=\; \check{k}^2_{p,\top} \circ \mathbf{0}_{\bot,\top} \circ \hat{k}^2_{p,\bot}.$$

The equation

$$(\hat{k}\check{\kappa}) \quad \hat{k}^1_{p,\bot} \;=\; \check{\kappa}_p \circ \hat{k}^2_{p,\bot},$$

which holds in $Set^{\emptyset}_*$, and which we have used above for showing the nonmaximality of $\mathbf{L}_{\top,\bot}$, clearly yields $(\hat{k}\check{k})$, which hence holds in $Set^{\emptyset}_*$, and which hence we could have also used for showing this nonmaximality.

If we refine the procedure above by building A' and B' out of A and B more carefully, then in some cases we could derive $(\hat{k}\check{\kappa})$ or its dual

$$(\check{k}\hat{\kappa}) \quad \check{k}^1_{p,\top} \;=\; \check{k}^2_{p,\top} \circ \hat{\kappa}_p$$

instead of $(\hat{k}\check{k})$. We do not replace x by $p \wedge \bot$ in building A', and we can proceed more selectively with other occurrences of letters and $\top$ in A, in order to obtain an A' isomorphic to p if possible. We can proceed analogously when we build B' out of B to obtain a B' isomorphic to p if possible.

Note that we have the following:

$$\begin{aligned} \check{\kappa}_{p \wedge \bot} \circ \hat{k}^2_{p,\bot} &= \langle \check{\kappa}_p, \mathbf{1}_\bot \rangle \circ \hat{k}^2_{p,\bot} \\ &= \langle \hat{k}^1_{p,\bot}, \hat{k}^2_{p,\bot} \rangle, \text{ with } (\hat{k}\check{\kappa}), \\ &= \mathbf{1}_{p \wedge \bot}. \end{aligned}$$

In the other direction, it is clear that the equation derived yields $(\hat{k}\check{\kappa})$. So with $(\hat{k}\check{\kappa})$ we have that $C \wedge \bot$ and $\bot$ are isomorphic, and, analogously, with $(\check{k}\hat{\kappa})$ we have that $C \vee \top$ and $\top$ are isomorphic. It can be shown that the natural logical category defined as $\mathbf{L}_{\top,\bot}$ save that we assume in addition both $(\hat{k}\check{\kappa})$ and $(\check{k}\hat{\kappa})$ is maximal. (This is achieved by eliminating letterless subformulae from C and D in

$g_1, g_2 : C \vdash D$ such that $Gg_1 \neq Gg_2$, and falling upon the argument used for the maximality of $\mathbf{L}$ in the preceding section.)

If $f : a \vdash b$ is any arrow of a dicartesian category $\mathcal{A}$ and $(\hat{k}\check{k})$ holds in $\mathcal{A}$, then we have in $\mathcal{A}$

$$\begin{aligned} \check{k}^1_{b,\top} \circ f \circ \hat{k}^1_{a,\bot} &= \check{k}^1_{b,\top} \circ \hat{k}^1_{b,\bot} \circ (f \wedge \mathbf{1}_\bot) \\ &= \check{k}^2_{b,\top} \circ \mathbf{0}_{\bot,\top} \circ \hat{k}^2_{a,\bot}, \end{aligned}$$

and hence for $f, g : a \vdash b$ we have in $\mathcal{A}$

$$(\hat{k}\check{k}fg) \quad \check{k}^1_{b,\top} \circ f \circ \hat{k}^1_{a,\bot} = \check{k}^1_{b,\top} \circ g \circ \hat{k}^1_{a,\bot}.$$

So, although $\mathbf{L}_{\top,\bot}$ is not maximal, it is maximal in the relative sense that every dicartesian category that satisfies an equation $f = g$ between arrow terms of $\mathbf{L}_{\top,\bot}$ such that $Gf \neq Gg$ satisfies also $(\hat{k}\check{k})$ and $(\hat{k}\check{k}fg)$. Some of these dicartesian categories may satisfy more than just $(\hat{k}\check{k})$ and $(\hat{k}\check{k}fg)$. They may satisfy $(\hat{k}\hat{\kappa})$ or $(\check{k}\hat{\kappa})$, which yields

$$f \circ \hat{k}^1_{a,\bot} = g \circ \hat{k}^1_{a,\bot} \qquad \text{or} \qquad \check{k}^1_{b,\top} \circ f = \check{k}^1_{b,\top} \circ g,$$

and some may be preorders.

Acknowledgement. We are grateful to Slobodanka Janković for a helpful stylistic suggestion. Work on this paper was supported by the Ministry of Science of Serbia (Grant 144013).

References

[1] H. P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, North-Holland, Amsterdam, 1981

[2] J. Bénabou, *Catégories avec multiplication*, C. R. Acad. Sci., Paris, Sér. I Math. 256 (1963), 1887–1890

[3] G. Burde and H. Zieschang, *Knots*, de Gruyter, Berlin, 1985

[4] C. C. Chang and H. J. Keisler, *Model Theory*, North-Holland, Amsterdam, 1973

[5] J. R. B. Cockett and R. A. G. Seely, *Finite sum-product logic*, Theory Appl. Categ. 8 (2001), 63–99

[6] H. S. M. Coxeter and W. O. J. Moser, *Generators and Relations for Discrete Groups*, Springer, Berlin, 1957

[7] K. Došen, *Cut Elimination in Categories*, Kluwer, Dordrecht, 1999

[8] K. Došen and Z. Petrić, *The maximality of the typed lambda calculus and of cartesian closed categories*, Publ. Inst. Math., Nouv. Sér. 68(82) (2000), 1–19 (available at: http://arXiv.org/math.CT/9911073)

[9] ———, *The maximality of cartesian categories*, Math. Log. Q. 47 (2001), 137–144 (available at: http://arXiv.org/math.CT/911059)

[10] ———, *Coherent bicartesian and sesquicartesian categories*; in: *Proof Theory in Computer Science* (R. Kahle et al., editors), Lect. Notes Comput. Sci. 2183, Springer, Berlin, 2001, pp. 78–92 (revised version of 2006, with major corrections, available at: http://arXiv.org/math.-CT/0006091)

[11] ———, *Bicartesian coherence*, Stud. Log. 71 (2002), 331–353 (version with some corrections in the proof of maximality available at: http://arXiv.org/math.CT/0006052)

[12] ———, *Generality of proofs and its Brauerian representation*, J. Symb. Log. 68 (2003), 740–750 (available at: http://arXiv.org/math.LO/0211090)

[13] ———, *A Brauerian representation of split preorders*, Math. Log. Q. 49 (2003), 579–586 (available at: http://arXiv.org/math.LO/0211277)

[14] ———, *Proof-Theoretical Coherence*, KCL Publications (College Publications), London, 2004 (revised version available at: http://www.mi.sanu.ac.yu/~kosta/publications.htm)

[15] ———, *Proof-Net Categories*, Polimetrica, Monza, 2007 (preprint available at: http://-www.mi.sanu.ac.yu/~kosta/pn.pdf, 2005)

[16] D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik*, Springer, Berlin, 1928 (English translation of the second edition from 1938, *Principles of Mathematical Logic*, Chelsea, New York, 1950)

[17] G. M. Kelly and M. L. Laplaza, *Coherence for compact closed categories*, J. Pure Appl. Algebra 19 (1980), 193–213

[18] G. M. Kelly et al., editors, *Coherence in Categories*, Lect. Notes Math. 281, Springer, Berlin, 1972

[19] G. M. Kelly and S. Mac Lane, *Coherence in closed categories*, J. Pure Appl. Algebra 1 (1971), 97–140, 219

[20] J. Lambek, *Deductive systems and categories I: Syntactic calculus and residuated categories*, Math.l Syst. Theory 2 (1968), 287–318

[21] ———, *Deductive systems and categories II: Standard constructions and closed categories* in: *Category Theory, Homology Theory and their Applications I*, Lect. Notes Math. 86, Springer, Berlin, 1969, pp. 76–122

[22] J. Lambek and P. J. Scott, *Introduction to Higher Order Categorical Logic*, Cambridge University Press, Cambridge, 1986

[23] S. Mac Lane, *Natural associativity and commutativity*, Rice Univ. Stud. Papers in Math. 49 (1963), 28–46

[24] Z. Petrić, *G-Dinaturality*, Ann. Pure Appl. Logic 122 (2003), 131–173 (available at: http://-arXiv.org/math.CT/0012019)

[25] A. Preller and P. Duroux, *Normalisation of the theory* $\mathbf{T}$ *of Cartesian closed categories and conservativity of extensions* $\mathbf{T}[x]$ *of* $\mathbf{T}$, Theor. Inform. Appl. 33 (1999), 227–257

[26] A. K. Simpson, *Categorical completeness results for the simply-typed lambda-calculus*; in: *Typed Lambda Calculi and Applications* (M. Dezani-Ciancaglini and G. Plotkin, editors), Lect. Notes Comput. Sci. 902, Springer, Berlin, 1995, pp. 414–427

[27] M. E. Szabo, *A counter-example to coherence in cartesian closed categories*, Canad. Math. Bull. 18 (1975), 111–114

[28] R. Zach, *Completeness before Post: Bernays, Hilbert, and the development of propositional logic*, Bull. Symb. Log. 5 (1999), 331–366

Zoran Ognjanović, Miodrag Rašković
and Zoran Marković

PROBABILITY LOGICS

Abstract. The paper summarizes the results of the authors in formalization of uncertain reasoning. A number of probability logics is considered. Their axiomatizations, completeness, compactness and decidability are addressed. Some possible applications of probability logics are analyzed. A historical overview of related works is given.

Mathematics Subject Classification (2000): 68T37, 03B60, 03B70, 68T15, 68T27, 68T30, 03B35, 03B44, 03B45, 03B42

Keywords: probability logic, conditional probability, approximate probability, non-standard analysis, strong completeness, decidability, default reasoning

CONTENTS

1. Introduction

The problem of reasoning with uncertain knowledge is an ancient problem dating, at least, from Leibnitz and Boole. However, in the last decades there is a growing interest in the field connected with applications to computer science and artificial intelligence. Researchers from those areas have studied uncertain reasoning using different tools. Some of the proposed formalisms for representing, and

reasoning with, uncertain knowledge are based on probabilistic logics. That approach extends the classical (propositional or first order) calculus with expressions that speak about probability, while formulas remain true or false. Thus, one is able to make statements of the form (in our notation) $P_{\geqslant s}\alpha$ with the intended meaning "the probability of α is at least s".

The probability operators behave like modal operators and the corresponding semantics consists in special types of Kripke models (possible worlds) with addition of probability measures defined over the worlds. One of the main proof-theoretical problems with that approach is providing an axiom system which would be strongly complete ("every consistent set of formulas has a model", in contrast to the weak completeness "every consistent formula has a model"). This results from the inherent non-compactness of such systems. Namely, in such languages it is possible to define an inconsistent infinite set of formulas, every finite subset of which is consistent (e.g., $\{\neg P_{=0}\alpha\} \cup \{P_{<1/n}\alpha : n \text{is a positive integer}\}$). As it was pointed in [85, 125], there is an unpleasant consequence of finitary axiomatization in that case: there exist unsatisfiable sets of formulas that are consistent with respect to the assumed finite axiomatic system (since all finite subsets are consistent and deductions are finite sequences). Another important theoretical problem is related to the decidability issue.

In this paper we present a number of probabilistic logic. The main differences between the logics are:

- some of the logics are infinitary[1], while the others are finitary,
- the corresponding languages contain different kinds of probabilistic operators, both for unconditional and conditional probability,
- some of the logics are propositional, while the others are based on the first-order logic,
- for most of the logics we start from classical logic, but in some cases the basic logic can be intuitionistic or temporal,
- in some of the logics iterations of probabilistic operators are not allowed,
- for some of the logics restrictions of the following kinds are used: only probability measures with fixed finite range are allowed in models, only one probability measure on sets of possible worlds is allowed in a model, the measures are allowed to be finitely additive.

For all these logics we give the corresponding axiomatizations, prove completeness, and discuss their decidability. More precisely, we consider the following logics (the notation was taken from the corresponding papers):

- LPP_1 (L for logic, the first P for propositional, and the second P for probability), probability logic which starts from classical propositional logic, with iterations of the probability operators and real-valued probability functions [83, 85],

[1] In this paper the terms finitary and infinitary concern meta language only. Object languages are countable, formulas are finite (except where it is explicitly said), while only proofs are allowed to be infinite.

- $LPP_1^{\mathrm{Fr}(n)}$ and LPP_1^S that are similar to LPP_1, but with probability functions restricted to have ranges $\{0, 1/n, \ldots, (n-1)/n, 1\}$ and S, respectively [81, 83, 85],
- $LPP_1^{A,\omega_1,\mathrm{Fin}}$, probability logic similar to $LPP_1^{\mathrm{Fr}(n)}$, but with probability functions restricted to have arbitrary finite ranges [26],
- LPP_1^{LTL}, probability logic similar to LPP_1, but the basic logic is discrete linear-time logic LTL [82, 83, 91],
- LPP_2, $LPP_2^{\mathrm{Fr}(n)}$, $LPP_2^{A,\omega_1,\mathrm{Fin}}$ and LPP_2^S, probability logics similar to the above logics, but without iterations of the probability operators [83, 85, 106],
- $LPP_{2,P,Q,O}$, probability logic which extends LPP_2 by having a new kind of probabilistic operators of the form Q_F, with the intended meaning "the probability belongs to the set F" [84],
- $LPP_{2,\preceq}$ and $LPP_{2,\preceq}^{\mathrm{Fr}(n)}$, probability logics similar to LPP_2 and $LPP_2^{\mathrm{Fr}(n)}$, but allowing reasoning about qualitative probabilities [93],
- LPP_2^I, probability logics similar to LPP_2, but the basic logic is propositional intuitionistic logic [74, 75, 76],
- $LFOP_1$, $LFOP_1^{\mathrm{Fr}(n)}$, $LFOP_1^{A,\omega_1,\mathrm{Fin}}$, $LFOP_1^S$ and $LFOP_2$, first-order counterparts of the above logics [85, 110],
- $LPCP_2^{S,\approx}$, propositional Kolmogorov's style-conditional probability logic, without iterations of the probability operators, with probability functions restricted to have the range S and probability operators that can express approximate probabilities [88, 92, 112, 113, 114], and
- $LPCP_2^{\mathrm{Chr}}$, propositional conditional probability logic, which corresponds to de Finetti's view on coherent conditional probabilities [50, 90].

The rest of the paper is organized in the following way. In section 2 we give a short overview of studies relating logic and probability until the mid 1980's, and the work of H. J. Keisler and N. Nilsson [41, 42, 78, 116, 122]. Syntax and semantics, an infinitary axiomatization, the corresponding extended completeness, decidability and complexity of LPP_2 are presented in Section 3.1. As a semantics we introduce a class of models that combine properties of Kripke models and probabilities defined on sets of possible worlds. We consider the class of so called measurable models (which means that all sets of possible worlds definable by classical formulas are measurable) and some of its subclasses: in the first case all subsets of worlds are measurable, then probabilities are required to be σ-additive, while models in the last subclass satisfy that only empty set has the zero probability. The proposed axiomatization is infinitary, i.e., there is an inference rule with countably many premisses and one conclusion. That rule corresponds to the following property of real numbers: if the probability is arbitrary close to s, it is at least s. Thus, proofs with countably many formulas are allowed. The proof of extended completeness follows Henkin procedure: starting from a consistent set we construct its maximal consistent extension and the corresponding canonical model which satisfies the considered set of formulas. Decidability of LPP_2 is proved by

reducing the satisfiability problem to linear programming problem. Since the related linear systems can be of exponential sizes, in the same section we describe some heuristical approaches (genetic algorithms and variable neighborhood search) to the probabilistic satisfiability problem [51, 86, 87, 89]. Some variants of LPP_2 ($LPP_2^{\mathrm{Fr}(n)}$, $LPP_2^{A,\omega_1,\mathrm{Fin}}$ and LPP_2^S obtained by putting some restrictions on ranges of probability functions) and the logic LPP_1 are considered in the sections 4 and 5, respectively. In Section 6 we consider some extensions of the basic probability language. The first extension, $LPP_{2,P,Q,O}$, contains probability operators of the form Q_F with the intended meaning "the probability belongs to the set F". It turns out that in a general case neither $P_{\geqslant}$-operators are definable from Q_F-operators, nor are Q_F-operators operators definable from $P_{\geqslant}$-operators. Then, we discuss two logics that allow expressing qualitative probabilities: $LPP_{2,\preceq}$ and $LPP_{2,\preceq}^{\mathrm{Fr}(n)}$. It is proved elsewhere that the set of probability first-order valid formulas is not recursively enumerable and that no recursive complete axiomatization is possible. In Section 7 we extend our approach for the propositional case and give a complete infinitary first order axiomatization. That section also contains a discussion on the (dis)similarities between probability and modal logics. Intuitionistic and temporal probability logics are presented in Section 8. Two logics with conditional probabilities ($LPCP_2^{S,\approx}$ and $LPCP_2^{\mathrm{Chr}}$), and their applications are described in Section 9. One of the infinitary inference rules for $LPCP_2^{S,\approx}$ enables us to syntactically define the range of probability functions. In the case of $LPCP_2^{S,\approx}$, that range is the unit interval of a recursive non-archimedean field which makes it possible to express statements about approximate probabilities: $CP_{\approx s}(\alpha,\beta)$ which means "the conditional probability of α given β is approximately s". Furthermore, formulas of the form $CP_{\approx 1}(\alpha,\beta)$ can be used to model defaults, i.e., expressions of the form "if β, then generally α". It relates $LPCP_2^{S,\approx}$ with the well known system P which forms a core of default reasoning. It is proved that if we restrict attention only to formulas of the form $CP_{\approx 1}(\alpha,\beta)$, the resulting system coincides with P when we work only with finite sets of assumptions. If we allow inferences from infinite sets of such "defaults", our system is somewhat stronger. The main advantage, however, is ability to use $LPCP_2^{S,\approx}$ to combine uncertain knowledge and defaults. Finally, Section 10 discusses some of the more recent related papers.

2. History

Gottfried Wilhelm Leibnitz (1646–1716) investigated universal basis for all sciences and tried to establish logic as a generalized mathematical calculus. He considered probabilistic logic as a tool for the uncertainty estimation, and defined probability as a measure of knowledge. In some of his essays [67, 68, 69] Leibnitz suggested that tools developed for analyzing games of chance should be applied in developing a new kind of logic treating degrees of probability which, in turn, could be used to make rational decisions on conflicting claims. He distinguished two calculi. The first one, forward calculus, was concerned with estimating the probability of an event if the probabilities of its conditions are known. In the second one, called reverse calculus, estimations of probabilities of causes, when the probability

of their consequence is known, were considered. Leibnitz's logical works were for the most part published long after his death (by L. Couturat in the early 1900s). However, Leibnitz had some successors, the most important of whom, when the probabilistic logic is in question, were the brothers Jacobus (1654–1705) and Johann (1667–1748) Bernoulli, Thomas Bayes (1702–1761), Johann Heinrich Lambert (1728–1777), Pierre Simon de Laplace (1749–1827), Bernard Bolzano (1781–1848), Augustus De Morgan (1806–1871), George Boole (1815–1864), John Venn (1834–1923), Hugh MacColl (1837–1909), Charles S. Peirce (1839–1887), Platon Sereyevich Poretskiy (1846–1907), etc. We shall briefly mention some of their results.

Jacobus Bernoulli in his unfinished work [7, Part IV, Chapter III], was the first who made advance along the Leibnitz's ideas. Using Huygen's notion of expectation, i.e., the value of a gamble in games of chance, he offered a procedure for determining numerical degrees of certainty of conjectures produced by arguments. The word argument was used to represent statements as well as the implication relation between premises and conclusions. He divided arguments into categories according to whether the premises, and the argumentation from premises to conclusions are contingent or necessary. For example, if an argument exist contingently (i.e., it is true in $b > 0$ cases, while it is not in $c > 0$ cases) and implies a conclusion necessarily, then such an argument establishes $\frac{b}{b+c}$ as the certainty of the conclusion. Bernoulli also discussed the question of computing the degree of certainty when there were more then one argument for the same conclusion.

J. H. Lamber in [65], analyzed syllogistic inference of the form "if three quarters of the A's are B's, and C is A, then with probability $\frac{3}{4}$, C is B". In [5], written by T. Bayes, there was the first occurrence of a result involving conditional probability. In modern notation, he considered the problem of finding the conditional probability $P(A|B)$ where A is the proposition "$P(E) \in [a, b]$", while B is the proposition "an event E happened p and failed q times in $p + q$ independent trials". For B. Bolzano [11] logic was a theory of science, while probability was a part of logic. Using contemporary language it can be said that he understood validity of a proposition $A(x)$ as a measure of the set $\{c : \models A(c)\}$, i.e., as the ratio $\frac{|\{x:x\in U \wedge U \models A(x)\}|}{|\{x:x\in U\}|}$. Relative validity was a relation between propositions and had the same properties as what we call conditional probability. Bolzano derived a number of theorems regarding relative validity. A. De Morgan devoted a chapter of [21], to probability inference offering a defense for the numerical probabilistic approach as a part of logic. Instead of giving a systematic treatment of the field, he rather described some problems and tried to apply logical concepts to them. It is interesting that De Morgan made some mistakes, mainly due to his ignoring of (in)dependence of events.

The calculus inaugurated by G. Boole in [12, 13] initiated rapid development of mathematical logic. Boole sought to make his system the basis of a logical calculus as well as a more general method for the application in the probability theory. He wrote "... Every system of interpretation which does not affect the truth of the relations supposed is equally admissible, and it is thus that the same process may under one scheme of interpretation represent the solution of a question on the

properties of numbers, under another that of a geometrical problems, and under the third that of a problem of dynamics or optics..." Since 1854 Boole concentrated on unification of various elements of truth. He hoped to continue the advancement toward probable indications concerning the nature and structure of human thought. The most general problem (originally called "general problem in the theory of probability") Boole claimed that he could solve, concerned an arbitrary set of logical functions $\{f_1(x_1,\ldots,x_m),\ \ldots,\ f_k(x_1,\ldots,x_m),\ F(x_1,\ldots,x_m)\}$ and the corresponding probabilities $p_1 = P(f_1(x_1,\ldots,x_m)),\ \ldots,\ p_k = P(f_k(x_1,\ldots,x_m))$, and asked for $P(F(x_1,\ldots,x_m))$ in terms of $p_1,\ldots,p_k$. He explained the relation between the logic of classical connectives and the formal probability properties of compound events using the following assumptions. He restricted disjunctions to the exclusive ones, and believed that any compound proposition can be expressed in terms of, maybe ideal, simple and independent components. Thus, the probability of an or-compound is equal to the sum of the components, while the probability of an and-compound is equal to the product of the components. In such a way, it was possible to convert logical functions of events into a system of algebraic functions of the corresponding probabilities. Boole tried to solve such systems using a procedure equivalent to Fourier–Motzkin elimination. His procedure, although not entirely successful, provided a basis for probabilistic inferences. In [40, 41] a rationale and a correction for the Boole's procedure were given using the linear programming approach. It was noted that analytical expressions of the lower and upper bounds of the probabilities could be obtained.

The successors of Boole tried to improve the form of Boole's ideas. One of them was P.S. Poretskiy [96]. C.S. Pierce in [95] and H. MacColl in [71] clarified the notion of conditional probability, as the chance that a statement is true on the assumption that another statement is true, and introduced the corresponding symbol x_a ($P(x|a)$, in the contemporary formal language).

McColl also developed, contemporaneously with Frege, propositional logic as a branch of logic independent of the class calculus or term logic of the traditional syllogisms. He was the first author who made an attempt, in [72], to augment the two-valued logical formalism with a third truth value. It was a system of propositional logic with certain, impossible, and variable propositions. The propositions of the former two types are either necessary true or necessary false, while the propositions of the last type are sometimes true and sometimes false. MacColl's idea of proceeding along the probabilistic lines in the development of many-valued logic is of particular interest because he applied the calculus of variable propositions to the calculus of probabilities. His truth-values, like probabilities, cannot be combined in a truth-functional way. For example, if p is a variable proposition, so are $p \wedge p$ and $\neg p$, while $p \wedge \neg p$ is impossible rather than variable. Later systems, for example the ones of Lukasiewicz, were deficient in this respect.

In the 1870's J. Venn developed the idea of extending the frequency of occurrence concept of probability to logic. Venn thought that probability logic is the logic of sequence of statements. A single element sequence of this type attributes to the given proposition one of two values 0 or 1, while an infinite sequence attributes any real number which lies in the interval $[0, 1]$. Some of the traditional logicians were

dissatisfied with the inclusion of the induction in the definition of the concept of probability, but the others continued to work in that direction.

During the first half of XX century there were at least three directions in the development of theory of probability. The researchers that belonged to the first one, Richard von Miss (1883–1953) and Hans Reichenbach (1891–1953), for example, regarded probability as a relative frequency and derived rules of the theory from that interpretation. The second approach was characterized by the development of formal calculus of probability. Some of the corresponding authors were Georg Bohlmann (1869–1928) [10], Sergei Natanovich Bernstein (1880–1968) [8], andÉmil Borel (1871–1956) [14, 15]. These investigations culminated in A. N. Kolmogorov's (1903–1987) axiomatization of probability [60]. Finally, some of the researcher, like John M. Keynes (1883–1946) [59], Hans Reichenbach [115, 116], and Rudolf Carnap (1891–1970) [18, 19] continued Boole's approach connecting probability and logic.

In work of J. Keynes probability was seen as an undefined primitive concept. He presented an axiomatic analysis of a relation between propositions which behaved like conditional probability. That axiomatic system is not acceptable, at least from the point of the recent logical standards. For example, no specification of syntax was given, there were no inference rules, etc.

R. Carnap's work on logical foundations of probability was an attempt to develop a pure logical concept of probability. Carnap connected the concepts of inductive reasoning, probability and confirmation. He was among the first researchers who clearly acknowledged that there are two distinct concepts of probability. The concept of probability as the relative frequency (in the long run) which is used in statistical investigations is empirical in nature and, therefore, unsuitable for the development of inductive logic. For the development of inductive logic, which in his view is the same as probability logic, he needed the logical concept of probability as a degree of confirmation of some hypothesis on the basis of some evidence, i.e., a logical relation between two propositions, denoted by $c(h, e)$. Carnap fixed an unary first order language to express h and e, and studied properties of c. Even though Carnap's work was not completely successful, it stimulated a line of research on probabilistic first-order logics [33, 34, 120, 123]. In [33] there was a generalization of the notion of a model for a first-order language in which probability values replaced truth-values, and some kind of completeness theorem was proven. Similarly, in [34] a first order language L of arithmetic and a set of its models were considered. To every sentence the set of models in which it is true was associated, and the probability was defined on such definable sets. Then, they studied random sequences and some other notions from the theory of probability defined over L. In [120] the ideas from [33] were extended to infinitary languages. Boolean algebras with attached probability measures were considered as suitable models for reasoning about probability. Let I and m denote an interpretation and a probability defined on a Boolean algebra, respectively. A probability assertion A is a tuple $(a, s_1, \ldots, s_n)$, where a is a formula of the language of real closed fields with n free variables, while s_i's are sentences of an infinitary first order language. A speaks about probabilities such that it holds in a model if $a(m(I(s_1)), \ldots, m(I(s_n)))$ is true in the reals. Then, a probability assertion A is a consequence of a set T of

assertions if A holds in every model of T. In [120] a number of results about such a consequence relation were proved.

H. Reichenbach investigated the logical structure of probability statements from the philosophical and technical points of view. He introduced a fundamental probability relation between classes and real numbers using formulas of the form $P(A,B) = p$ which could be read as "for every i, if x_i belongs to the class A, then y_i belongs to the class B with probability p". Reichenbach gave a frequency interpretation for the probability relation, and the corresponding axioms $((A \to B) \to (P(A,B) = 1)$, for example). If $x_i \in A$ for every i, he used $P(B) = p$ instead of $P(A,B) = p$, and constructed truth tables for the classical connectives with a continuous scale of truth (if $P(A) = p$, $P(B) = q$, and $P(A,B) = u$, then $P(A \vee B) = p + q - pu$, for example). However, as can be seen, the value of $P(A \vee B) = p + q - pu$ depends on three values, i.e., on $P(A)$, $P(B)$, and $P(A,B)$, and not on $P(A)$ and $P(B)$ only, as it is the case in the classical two-valued logic.

Aleksandar Kron (1938–2000), Belgrade's logician and philosopher, studied relationship between multi-valued logics and probability theory [64]. He considered a unary operation generating a Boolean algebra of sets of formulas, and a probability function defined on that algebra, and gave some statement connecting notions from probability theory (conditional probability, independence) and logic (implication, proof).

In spite of the mentioned works of Reichenbach, Carnap and their followers, the mainstreams of development of logic and probability theory were almost separated during second half of XX century. Namely, in the last quarter of XIX century, independently of the algebraic approach, there was a development of mathematical logic inspired by the need of giving axiomatic foundations of mathematics. The main representative of that effort was Gottlob Frege (1848–1925). He tried to explain the fundamental logical relationships between the concepts and propositions of mathematics. Truth-values, as special kinds of abstract values, were described by Frege according to whom every proposition is a name for truth or falsity. It is clear that, according to Frege, the truth values had a special status that had nothing to do with probabilities. That approach culminated with Kurt Gödel's (1904–1977) proof of the completeness for the first order logic [37]. Since those works, the first order logic played the central role in the logical community for many years, and only in the late 70's a wider interest in probability logics reappeared.

The most important advancement in probability logic, after work of Leibnitz and Boole, was made by H. Jerome Keisler. The purpose of his famous paper [54] was to give model-theoretic approach to probability theory. Also it is important to emphasize that in this paper he makes use of nonstandard analysis as an useful method.

Keisler introduced several probability quantifiers, as for example $Px > r$. The formula $(Px > r)\phi(x)$ means that the set $\{x : \phi(x)\}$ has probability greater than r. A recursive axiomatization for that kind of logics (the main one denoted by L_{AP}) was given by D. Hoover [46]. He used admissible and countable fragments of infinitary predicate logic (but without ordinary quantifiers $\forall$ and $\exists$). In the following years Keisler and Hoover made very important contributions in the field. They

proved Completeness theorem for various kinds of models (probability, graded, analytic, hyperfinite etc.) and many other model-theoretical theorems. The development of probability model theory has engendered the need for the study of logics with greater expressive power than that of the logic L_{AP}. The logic L_{AI}, introduced in [55] as an equivalent of the logic L_{AP}, allows us to express many properties of random variables in an easier way. In this logic the quantifiers $\int \ldots dx$ are incorporated instead of the quantifiers $Px > r$. The completeness proof for L_{AI} used the Loeb construction of the Daniell integral (see also [22, 23, 24, 25]).

The logic L_{AI} is not rich enough to express probabilistic notions involving conditional expectations of random variables with respect to σ-algebras, such as martingale, Markov process, Brownian motion, stopping time, optional stochastic process, etc. These properties can be naturally expressed in a language with both integral quantifiers and conditional expectation operators. The logics LAE and Lad introduced by Keisler in [55], are appropriate for the study of random variables and stochastic processes. The model theory of these logics has been developed further by Hoover in [47], Keisler in [57, 58], Rodenhausen in [118] and Fajardo in [29].

In [97] Rašković introduced new L_{AM} logic which, instead of probability measure, has σ-finite one and give the method how to transfer results from L_{AP} to L_{AM}. In a series of papers [98, 99, 101, 104, 108], he also gave answers to a number of problems proposed by Keisler in [55]. In [98, 99] a new method of using Barwise compactness theorem [4] in proving completeness theorems was presented. It is difficult to mix ordinary and probability quantifiers because of the fact that projection of a measurable set can be nonmeasurable. As a consequence of that it is hard (if not impossible) to expect adequate logic in its full strength. But some effort in that direction has been made in [100, 102, 103, 105]. The notion of a cylindric probability algebra can be considered as a common algebraic abstraction from a geometry associated with basic set-theoretic notions on the one hand and the theory of deductive systems of probability logic on the other. These two sources are connected because models of deductive systems of probability logic give rise in natural way to probability structures within set-theoretical algebras. As is well known, the theory of Boolean algebras is related to the sentential calculus, and theory of cylindric algebras to the first-order predicate logic. The theory of cylindric probability algebras, designed to provide an apparatus for an algebraic study of probability logics, is presented in [49, 109, 111] analogously to Boolean algebras and cylindric algebras. The model theory for probability logic with undetermined finite range is given in [104]. Continuous time probability logic L^t_{AP}, developed in [107], is a logic appropriate for the study of a space with a family of continuous time probability measures. The set of universal conjunctive formulas of L^t_{AP} is the least set containing all quantifier-free formulas and closed under arbitrary $\wedge$, finite $\vee$, and quantifiers $(Px > r)$, $r \in Q \cap [0,1]$. The completeness theorem and finite compactness theorem (for universal conjunctive formulas) were proven.

Since the middle of 1980's the interest in probabilistic logics started growing because of development of many fields of application of reasoning about uncertain knowledge: in economics, artificial intelligence, computer science, philosophy etc. Researchers attempt to combine probability-based and logic-based approaches

to knowledge representation. In the logical framework for modelling uncertainty, probabilities express degrees of belief. For example, one can say that "probability that Homer wrote Iliad is at most a half" expressing one's disbelief that Homer is the real author of Iliad. The first of those papers is [79] (see also: [80]) which resulted from the work on developing an expert system in medicine, where N. Nilsson tried to give a logic with probabilistic operators as a well-founded framework for uncertain reasoning. Sentences of the logic spoke about probabilities. He was able to express a probabilistic generalization of modus ponens as "if α holds with the probability s, and β follows from α with the probability t, then the probability of β is r".

3. Logic LPP_2

In this section we present the logic LPP_2. We describe its syntax and some classes of models, give an infinitary axiomatization and prove that it is sound and complete with respect to the mentioned classes of models.

3.1. Syntax. Let S be the set of all rational numbers from $[0,1]$. The language of LPP_2 consists of the denumerable set $\phi = \{p, q, r, \ldots\}$ of primitive propositions, classical propositional connectives $\neg$, and $\wedge$, and a list of probability operators $P_{\geqslant s}$ for every $s \in S$. The set For_C of all classical propositional formulas over the set ϕ is defined as usual. The formulas from the set For_C will be denoted by α, β,... If $\alpha \in \mathrm{For}_C$ and $s \in S$, then $P_{\geqslant s}\alpha$ is a *basic probability formula*. The set For_P of all probability formulas is the smallest set

- containing all basic probability formulas, and
- closed under formation rules: if $A, B \in \mathrm{For}_P$, then $\neg A$, $A \wedge B \in \mathrm{For}_P$.

The formulas from the set For_P will be denoted by A, B,... Let $\mathrm{For}_{LPP_2} = \mathrm{For}_C \cup \mathrm{For}_P$. The formulas from the set For_{LPP_2} will be denoted by Φ, Ψ,...

We use the usual abbreviations for the other classical connectives, and also denote:

- $\neg P_{\geqslant s}\alpha$ by $P_{<s}\alpha$,
- $P_{\geqslant 1-s}\neg\alpha$ by $P_{\leqslant s}\alpha$,
- $\neg P_{\leqslant s}\alpha$ by $P_{>s}\alpha$,
- $P_{\geqslant s}\alpha \wedge P_{\leqslant s}\alpha$ by $P_{=s}\alpha$, and
- both $\alpha \wedge \neg\alpha$ and $A \wedge \neg A$ by $\bot$, letting the context determine the meaning.

As it can be seen, neither mixing of pure propositional formulas and probability formulas, nor nested probability operators are allowed. Thus, $\alpha \wedge P_{\geqslant s}\beta$ and $P_{\geqslant s}P_{\geqslant r}\alpha$ do not belong to the set For_{LPP_2}.

Let $p_1, \ldots, p_n$ be a list of all primitive propositions from $\Phi \in \mathrm{For}_{LPP_2}$. An *atom* a of Φ is a formula of the form $\pm p_1 \wedge \ldots \wedge \pm p_n$, where $\pm p_i$ is either p_i, or $\neg p_i$.

3.2. Semantics. The semantics for For_{LPP_2} will be based on the possible-world approach.

Definition 1. An *LPP_2-model* is a structure $\mathbf{M} = \langle W, H, \mu, v\rangle$ where:

- W is a nonempty set of objects called worlds,

- H is an algebra of subsets of W, and
- μ is a finitely additive measure, $\mu : H \to [0,1]$,
- $v : W \times \phi \to \{\text{true}, \text{false}\}$ provides for each world $w \in W$ a two-valued evaluation of the primitive proposition, that is $v(w,p) \in \{\text{true}, \text{false}\}$, for each primitive proposition $p \in \phi$ and each world $w \in W$; a truth-evaluation $v(w,\cdot)$ is extended to classical propositional formulas as usual.

If $\mathbf{M}$ is an LPP_2-model and $\alpha \in \text{For}_C$, the set $\{w : v(w,\alpha) = \text{true}\}$ is denoted by $[\alpha]_{\mathbf{M}}$. We will omit the subscript $\mathbf{M}$ from $[\alpha]_{\mathbf{M}}$ and write $[\alpha]$ if $\mathbf{M}$ is clear from the context. An LPP_2-model $\mathbf{M} = \langle W, H, \mu, v\rangle$ is *measurable* if $[\alpha]_{\mathbf{M}} \in H$ for every formula $\alpha \in \text{For}_C$. In this section we focus on the class of all measurable LPP_2-models (denoted by $LPP_{2,\text{Meas}}$).

Definition 2. *The satisfiability relation* $\models \subseteq LPP_{2,\text{Meas}} \times \text{For}_{LPP_2}$ fulfills the following conditions for every $LPP_{2,\text{Meas}}$-model $\mathbf{M} = \langle W, H, \mu, v\rangle$:

- if $\alpha \in \text{For}_C$, $\mathbf{M} \models \alpha$ iff for every $w \in W$, $v(w,\alpha) = \text{true}$,
- if $\mathbf{M} \models P_{\geqslant s}\alpha$ iff $\mu([\alpha]) \geqslant s$,
- if $A \in \text{For}_P$, $\mathbf{M} \models \neg A$ iff $\mathbf{M}, \not\models A$,
- if $A, B \in \text{For}_P$, $\mathbf{M} \models A \wedge B$ iff $\mathbf{M} \models A$ and $\mathbf{M} \models B$. □

Definition 3. A formula $\Phi \in \text{For}_{LPP_2}$ is *satisfiable* if there is an $LPP_{2,\text{Meas}}$-model $\mathbf{M}$ such that $\mathbf{M} \models \Phi$; Φ is *valid* if for every $LPP_{2,\text{Meas}}$-model $\mathbf{M}$, $\mathbf{M} \models \Phi$; a set of T formulas is satisfiable if there is an $LPP_{2,\text{Meas}}$-model $\mathbf{M}$ such that $\mathbf{M} \models \Phi$ for every $\Phi \in T$.

Example 4. Consider the set $T = \{\neg P_{=0}\alpha\} \cup \{P_{<1/n}\alpha : n \text{ is a positive integer}\}$. Although every finite subset of T is $LPP_{2,\text{Meas}}$-satisfiable, the set T itself is not. So, the compactness theorem "If every finite subset of T is satisfiable, then T is satisfiable" does not hold for LPP_2. □

Example 5. Note that the classical formulas do not behave in the usual way: for some α and $\beta \in \text{For}_C$ and an $LPP_{2,\text{Meas}}$-model $\mathbf{M}$ it can be $\mathbf{M} \models \alpha \vee \beta$, but that neither $\mathbf{M} \models \alpha$, nor $\mathbf{M} \models \beta$. Similarly, it can be simultaneously $\mathbf{M} \not\models \alpha$ and $\mathbf{M} \not\models \neg\alpha$. Nevertheless, the set of all classical formulas that are valid with respect to the above given semantics and the set of all classical valid formulas coincide, because every world from an arbitrary $LPP_{2,\text{Meas}}$-model can be seen as a classical propositional interpretation.

In the sequel we will also consider the following classes of LPP_2-models:

$$LPP_{2,\text{Meas,All}}, \quad LPP_{2,\text{Meas},\sigma} \quad \text{and} \quad LPP_{2,\text{Meas,Neat}}.$$

A model $\mathbf{M} = \langle W, H, \mu, v\rangle$ belongs to the first class if H is the power set of W, i.e., if every subset of W is μ-measurable. A model $\mathbf{M}$ belongs to the second class if it is a σ-additive measurable model, i.e., if μ is a σ-additive probability measure. A model $\mathbf{M}$ belongs to the second class if it is a measurable model such that $\mu(H_1) = 0$ iff $H_1 = \emptyset$, i.e., if only the empty set has the zero probability.

3.3. Complete Axiomatization. The set of all valid formulas can be characterized by the following set of axiom schemata:

(1) all instances of the classical propositional tautologies
(2) $P_{\geqslant 0}\alpha$
(3) $P_{\leqslant r}\alpha \to P_{<s}\alpha$, $s > r$
(4) $P_{<s}\alpha \to P_{\leqslant s}\alpha$
(5) $(P_{\geqslant r}\alpha \wedge P_{\geqslant s}\beta \wedge P_{\geqslant 1}(\neg(\alpha \wedge \beta))) \to P_{\geqslant \min(1,r+s)}(\alpha \vee \beta)$
(6) $(P_{\leqslant r}\alpha \wedge P_{<s}\beta) \to P_{<r+s}(\alpha \vee \beta)$, $r + s \leqslant 1$

and inference rules:

(1) From Φ and $\Phi \to \Psi$ infer Ψ.
(2) From α infer $P_{\geqslant 1}\alpha$.
(3) From $A \to P_{\geqslant s-\frac{1}{k}}\alpha$, for every integer $k \geqslant \frac{1}{s}$, and $s > 0$ infer $A \to P_{\geqslant s}\alpha$.

We denote this axiomatic system by Ax_{LPP_2}.

Definition 6. A formula Φ is *deducible from a set* T of formulas (denoted by $T \vdash \Phi$) if there is an at most denumerable sequence of formulas $\Phi_0, \Phi_1, \ldots, \Phi$, such that every Φ_i is an axiom or a formula from the set T, or it is derived from the preceding formulas by an inference rule. A *proof* for Φ from T is the corresponding sequence of formulas. A formula Φ is a *theorem* (denoted by $\vdash \Phi$) if it is deducible from the empty set. □

Definition 7. A set T of formulas is *consistent* if there are at least a formula from For_C, and at least a formula from For_P that are not deducible from T, otherwise T is *inconsistent*. A consistent set T of formulas is said to be *maximal consistent* if the following holds:

- for every $\alpha \in \mathrm{For}_C$, if $T \vdash \alpha$, then $\alpha \in T$ and $P_{\geqslant 1}\alpha \in T$, and
- for every $A \in \mathrm{For}_P$, either $A \in T$ or $\neg A \in T$.

A set T of formulas is *deductively closed* if for every $\Phi \in \mathrm{For}_{LPP_2}$, if $T \vdash \Phi$, then $\Phi \in T$.

Alternatively, we can say that T is inconsistent iff $T \vdash \bot$. Also, note that classical and probability formulas are handled in different ways in Definition 7: it is not required that for every classical formula α, either α or $\neg\alpha$ belongs to a maximal consistent set, as it is done for formulas from For_P.

Let us now discuss the above axioms and rules. First note that, by Axiom 1, the classical propositional logic is a sublogic of LPP_2. It is also easy to see that every LPP_2-proof consists of two parts (one of them may be empty). In the first one only classical formulas are involved, while the second one uses formulas from For_P. Two parts are separated by some applications of Rule 2. There is no inverse rule, so we can pass from the classical to the probability level, but we cannot come back. It follows that LPP_2-logic is a conservative extension of the classical propositional logic. The axioms 2– 6 concern the probabilistic aspect of LPP_2. Axiom 2 announces that every formula is satisfied by a set of worlds of the measure at least 0. By substituting $\neg\alpha$ for α in the axiom, the formula $P_{\geqslant 0}\neg\alpha$ is obtained. According to our definition of the operator $P_{\leqslant 1}$, we have the following instance of Axiom 2:

$2'$. $P_{\leqslant 1}\alpha$ $(= P_{\geqslant 1-s}\neg\alpha$, for $s = 1)$.

It forces that every formula is satisfied by a set of time instants of the measure at most 1, and gives the upper bound for probabilities of formulas in $LPP_{2,\text{Meas}}$-models. In a similar way,the axioms 3 and 4 are equivalent to

$3'$. $P_{\geqslant t}\alpha \to P_{>s}\alpha$, $t > s$

$4'$. $P_{>s}\alpha \to P_{\geqslant s}\alpha$

respectively. The axioms 5 and 6 correspond to the additivity of measures. For example, in Axiom 5, if sets of worlds that satisfy α and β are disjoint, then the measure of the set of worlds that satisfy $\alpha \vee \beta$ is the sum of the measures of the former two sets. Rule 1 is classical Modus Ponens. Rule 2 can be considered as the rule of necessitation in modal logics, but it can be applied on the classical propositional formulas only. Rule 3 is the only infinitary inference rule in the system, i.e., it has a countable set of assumptions and one conclusion. It corresponds to the Archimedean axiom for real numbers and intuitively says that if the probability is arbitrary close to s, then it is at least s.

3.4. Soundness and completeness.

3.4.1. Soundness. Soundness of our system follows from the soundness of classical propositional logic, as well as from the properties of probabilistic measures, so we give only a sketch of a straightforward but tedious proof.

Theorem 8 (Soundness). *The axiomatic system Ax_{LPP_2} is sound with respect to the class of $LPP_{2,\text{Meas}}$-models.*

Proof. We can show that every instance of an axiom schemata holds in every model, while the inference rules preserve the validity. For example, let us consider Axiom 5. Suppose that $P_{\geqslant r}\alpha$, $P_{\geqslant s}\beta$, and $P_{\geqslant 1}\neg(\alpha \vee \beta)$ hold in a model $\mathbf{M} = \langle W, H, \mu, v\rangle$. It means that $\mu([\alpha]) \geqslant r$, $\mu([\beta]) \geqslant s$, and that $[\alpha]$ and $[\beta]$ are disjoint sets. By the definition of finitely additive measures, the measure of $[\alpha] \cup [\beta]$ (which is $[\alpha \vee \beta]$) is $\mu([\alpha]) + \mu([\beta])$. Hence, $M \vDash P_{\geqslant \min(1, r+s)}(\alpha \vee \beta)$, and Axiom 5 holds in M. The other axioms can be proved to be valid in a similar way.

Rule 1 is validity-preserving for the same reason as in classical logic. Consider Rule 2 and suppose that a formula $\alpha \in \text{For}_C$ is valid. Then, for every model $\mathbf{M} = \langle W, H, \mu, v\rangle$, $[\alpha] = W$, and $\mu([\alpha]) = 1$. Hence, $P_{\geqslant 1}\alpha$ is valid too. Rule 3 preserves validity because of the properties of the set of real numbers. □

3.4.2. Completeness. In the proof of the completeness theorem the following strategy is applied. We start with a form of Deduction theorem (Theorem 9) and some other auxiliary statements (the lemmas 10, 11, 12). Then, we show how to extend a consistent set T of formulas to a maximal consistent set T^* (Theorem 13). Finally, the canonical model $\mathbf{M}_T$ is constructed using the set T^* (Theorem 14) such that $\mathbf{M}_T \vDash \varphi$ iff $\varphi \in T^*$ (Theorem 15).

Theorem 9 (Deduction theorem). *If T is a set of formulas and $\varphi, \psi \in \text{For}_C$ or $\varphi, \psi \in \text{For}_P$, then*

$$T \cup \{\varphi\} \vdash \psi \text{ iff } T \vdash \varphi \to \psi.$$

Proof. The implication from right to left can prove exactly in the same way as in the classical propositional case. For the other direction we use the transfinite induction on the length of the proof of ψ from $T \cup \{\varphi\}$. The cases when either $\vdash \psi$ or $\varphi = \psi$ or ψ is obtained by application of Modus Ponens (Rule 1) are standard.

Thus, let us consider the case where $\psi = P_{\geqslant 1}\alpha$ is obtained from $T \cup \{\varphi\}$ by an application of Rule 2, and $\varphi \in \mathrm{For}_P^S$. In that case:

$T, \varphi \vdash \alpha$
$T, \varphi \vdash P_{\geqslant 1}\alpha$ by Rule 2

However, since $\alpha \in \mathrm{For}_C$, and $\varphi \in \mathrm{For}_P^S$, φ does not affect the proof of α from $T \cup \{\varphi\}$, and we have:

(1) $T \vdash \alpha$
(2) $T \vdash P_{\geqslant 1}\alpha$ by Rule 2
(3) $T \vdash P_{\geqslant 1}\alpha \to (\varphi \to P_{\geqslant 1}\alpha)$
(4) $T \vdash \varphi \to P_{\geqslant 1}\alpha$ by Rule 1.

Next, let us consider the case where $\psi = A \to P_{\geqslant s}\alpha$ is obtained from $T \cup \{\varphi\}$ by an application of Rule 3, and $\varphi \in \mathrm{For}_P$. Then:

(1) $T, \varphi \vdash A \to P_{\geqslant s-\frac{1}{k}}\alpha$, for every integer $k \geqslant \frac{1}{s}$
(2) $T \vdash \varphi \to (A \to P_{\geqslant s-\frac{1}{k}}\alpha)$, for $k \geqslant \frac{1}{s}$, by the induction hypothesis
(3) $T \vdash (\varphi \wedge A) \to P_{\geqslant s-\frac{1}{k}}\alpha$, for $k \geqslant \frac{1}{s}$
(4) $T \vdash (\varphi \wedge A) \to P_{\geqslant s}\alpha$, from (3) by Rule 3
(5) $T \vdash \varphi \to \psi$. □

Lemma 10.

(1) $\vdash P_{\geqslant 1}(\alpha \to \beta) \to (P_{\geqslant s}\alpha \to P_{\geqslant s}\beta)$,
(2) *if* $\vdash \alpha \leftrightarrow \beta$, *then* $\vdash P_{\geqslant s}\alpha \leftrightarrow P_{\geqslant s}\beta$,
(3) $\vdash P_{\geqslant s}\alpha \to P_{\geqslant r}\alpha$, $s \geqslant r$,
(4) $\vdash P_{\leqslant r}\alpha \to P_{\leqslant s}\alpha$, $s \geqslant r$.

Proof. (1) First note that using Rule 2, from $\vdash \neg\alpha \vee \neg\bot$, we obtain

$$\vdash P_{\geqslant 1}(\neg\alpha \vee \neg\bot), \tag{1}$$

and similarly, from $\vdash (\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha$ we have

$$\vdash P_{\geqslant 1}((\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha). \tag{2}$$

By Axiom 5, we have $\vdash (P_{\geqslant s}\alpha \wedge P_{\geqslant 0}\bot \wedge P_{\geqslant 1}(\neg\alpha \vee \neg\bot)) \to P_{\geqslant s}(\alpha \vee \bot)$. Since $\vdash P_{\geqslant 0}\bot$ by Axiom 2, from (1) it follows that

$$\vdash P_{\geqslant s}\alpha \to P_{\geqslant s}(\alpha \vee \bot). \tag{3}$$

The expression $P_{\geqslant s}(\alpha \vee \bot)$ denotes $P_{\geqslant s}\neg(\neg\alpha \wedge \neg\bot)$, $P_{\geqslant 1-(1-s)}\neg(\neg\alpha \wedge \neg\bot)$, and $P_{\leqslant 1-s}(\neg\alpha \wedge \neg\bot)$. Similarly, $\neg P_{\geqslant s}\neg\neg\alpha$ denotes $P_{<s}\neg\neg\alpha$. By Axiom 6, we have

$$\vdash (P_{\leqslant 1-s}(\neg\alpha \wedge \neg\bot) \wedge P_{<s}\neg\neg\alpha) \to P_{<1}((\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha).$$

Since $P_{\geqslant 1}((\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha)$ denotes $\neg P_{<1}((\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha)$, from (2) we obtain

$$\vdash (P_{\leqslant 1-s}(\neg\alpha \wedge \neg\bot) \wedge P_{<s}\neg\neg\alpha) \to (P_{<1}((\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha) \wedge \neg P_{<1}((\neg\alpha \wedge \neg\bot) \vee \neg\neg\alpha)).$$

It follows that $\vdash P_{\leqslant 1-s}(\neg\alpha \wedge \neg\bot) \to \neg P_{<s}\neg\neg\alpha$, i.e.,

$$\vdash P_{\geqslant s}(\alpha \vee \bot) \to P_{\geqslant s}\neg\neg\alpha. \tag{4}$$

From (3) and (4) we obtain $\vdash P_{\geqslant s}\alpha \to P_{\geqslant s}\neg\neg\alpha$. The negation of the formula $P_{\geqslant 1}(\alpha \to \beta) \to (P_{\geqslant s}\alpha \to P_{\geqslant s}\beta)$ is equivalent to $P_{\geqslant 1}(\neg\alpha \vee \beta) \wedge P_{\geqslant s}\alpha \wedge P_{<s}\beta$. Since $\vdash P_{\geqslant s}\alpha \to P_{\geqslant s}\neg\neg\alpha$, this formula implies $P_{\geqslant 1}(\neg\alpha \vee \beta) \wedge P_{\geqslant s}\neg\neg\alpha \wedge P_{<s}\beta$ which can be rewritten as $P_{\geqslant 1}(\neg\alpha \vee \beta) \wedge P_{\leqslant 1-s}\neg\alpha \wedge P_{<s}\beta$. From:

- Axiom 6, $P_{\leqslant 1-s}\neg\alpha \wedge P_{<s}\beta \to P_{<1}(\neg\alpha \vee \beta)$, and
- $P_{<1}\alpha = \neg P_{\geqslant 1}\alpha$,

we have

$$\vdash \neg(P_{\geqslant 1}(\alpha \to \beta) \to (P_{\geqslant s}\alpha \to P_{\geqslant s}\beta)) \to P_{\geqslant 1}(\neg\alpha \vee \beta) \wedge \neg P_{\geqslant 1}(\neg\alpha \vee \beta),$$

a contradiction. It follows that

$$\vdash P_{\geqslant 1}(\alpha \to \beta) \to (P_{\geqslant s}\alpha \to P_{\geqslant s}\beta).$$

(2) It is an easy consequence of Lemma 10(1).

(3) This formula expresses monotonicity of probabilities. From Axiom 3' $P_{\geqslant s}\alpha \to P_{>r}\alpha$, $s > r$, and Axiom 4' $P_{>r}\alpha \to P_{\geqslant r}\alpha$, we obtain $\vdash P_{\geqslant s}\alpha \to P_{\geqslant r}\alpha$ for $s > r$. If $s = r$, the formula is trivially a theorem of the form $\vdash \varphi \to \varphi$.

(4) Similarly as (3). □

Lemma 11. *Let T be a consistent set of formulas.*

(1) *For any formula $A \in \mathrm{For}_P$, either $T \cup \{A\}$ is consistent or $T \cup \{\neg A\}$ is consistent.*

(2) *If $\neg(\alpha \to P_{\geqslant s}\beta) \in T$, then there is some $n > \frac{1}{s}$ such that $T \cup \{\alpha \to \neg P_{\geqslant s-\frac{1}{n}}\beta\}$ is consistent.*

Proof. (1) The proof is standard: if $T \cup \{A\} \vdash \bot$, and $T \cup \{\neg A\} \vdash \bot$, by Deduction Theorem we have $T \vdash \bot$.

(2) Suppose that for every $n > \frac{1}{s}$:

$$T, \alpha \to \neg P_{\geqslant s-\frac{1}{n}}\beta \vdash \bot.$$

By Deduction Theorem, and manipulation at the propositional level, we have

$$T \vdash \alpha \to P_{\geqslant s-\frac{1}{n}}\beta,$$

for every $n > \frac{1}{s}$. By application of Rule 3 we obtain $T \vdash \alpha \to P_{\geqslant s}\beta$, a contradiction with the fact that $\neg(\alpha \to P_{\geqslant s}\beta) \in T$. □

Lemma 12. *Let T be a maximal consistent set of formulas. Then,*

(1) *for any formula $A \in \mathrm{For}_P$, exactly one member of $\{A, \neg A\}$ is in T,*

(2) *for all formulas $A, B \in \mathrm{For}_P$, $A \vee B \in T$ iff $A \in T$ or $B \in T$,*

(3) *for all formulas φ, ψ, where either $\varphi, \psi \in \mathrm{For}_C$ or $\varphi, \psi \in \mathrm{For}_P$, $\varphi \wedge \psi \in T$ iff $\{\varphi, \psi\} \subset T$,*

(4) *for every $\varphi \in \mathrm{For}_{LPP_2}$, if $T \vdash \varphi$, then $\varphi \in T$,*

(5) *for all formulas φ, ψ, where either $\varphi, \psi \in \mathrm{For}_C$ or $\varphi, \psi \in \mathrm{For}_P$, if $\{\varphi, \varphi \to \psi\} \subset T$, then $\psi \in T$,*

(6) *for all formulas* φ, ψ, *where either* $\varphi, \psi \in \mathrm{For}_C$ *or* $\varphi, \psi \in \mathrm{For}_P$, *if* $\varphi \in T$ *and* $\vdash \varphi \to \psi$, *then* $\psi \in T$,

(7) *for any formula* α, *if* $t = \sup_s\{P_{\geqslant s}\alpha \in T\}$, *and* $t \in S$, *then* $P_{\geqslant t}\alpha \in T$.

Proof. Proofs (1)–(6) are standard.

(7) Let $t = \sup_s\{P_{\geqslant s}\alpha \in T\} \in S$. By the monotonicity of the measure (Lemma 10(12)), for every $s \in S$, $s < t$, $T \vdash P_{\geqslant s}\alpha$. Using Rule 3 we have $T \vdash P_{\geqslant t}\alpha$. Since T is a maximal consistent set, it follows from Lemma 12(4) that $P_{\geqslant t}\alpha \in T$. □

Theorem 13. *Every consistent set can be extended to a maximal consistent set.*

Proof. Let T be a consistent set, $Cn_C(T)$ the set of all classical formulas that are consequences of T, and $A_0, A_1, \ldots$ an enumeration of all formulas from For_P. We define a sequence of sets T_i, $i = 0, 1, 2, \ldots$ such that:

(1) $T_0 = T \cup Cn_C(T) \cup \{P_{\geqslant 1}\alpha : \alpha \in Cn_C(T)\}$

(2) for every $i \geqslant 0$,

  (a) if $T_i \cup \{A_i\}$ is consistent, then $T_{i+1} = T_i \cup \{A_i\}$, otherwise

  (b) if A_i is of the form $\beta \to P_{\geqslant s}\gamma$, then $T_{i+1} = T_i \cup \{\neg A_i, \beta \to \neg P_{\geqslant s-\frac{1}{n}}\gamma\}$, for some positive integer n, so that T_{i+1} is consistent, otherwise

  (c) $T_{i+1} = T_i \cup \{\neg A_i\}$.

(3) $\mathcal{T} = \bigcup_{i=0}^{\infty} T_i$.

The set T_0 is consistent since it is contains consequences of an consistent set, and similarly for the other members of the family of sets, by Lemma 12 each T_i, $i > 0$, is consistent.

It remains to show that $\mathcal{T}$ is maximal and consistent. The steps 1 and 2 of the above construction fulfill all requirements from Definition 7 which guarantees that $\mathcal{T}$ is maximal. We continue by showing that $\mathcal{T}$ is a deductively closed set which does not contain all formulas, and, as a consequence, that $\mathcal{T}$ is consistent.

First of all, $\mathcal{T}$ does not contain all formulas. If $\alpha \in \mathrm{For}_C$, by the construction of T_0, α and $\neg\alpha$ cannot be simultaneously in T_0. For a formula $A \in \mathrm{For}_P$ the set $\mathcal{T}$ does not contain both $A = A_i$ and $\neg A = A_j$, because $T_{\max(i,j)+1}$ is consistent.

I remains to show that $\mathcal{T}$ is deductively closed. If a formula $\alpha \in \mathrm{For}_C$ and $\mathcal{T} \vdash \alpha$, then by the construction of T_0, $\alpha \in \mathcal{T}$ and $P_{\geqslant 1}\alpha \in \mathcal{T}$. Let $A \in \mathrm{For}_P$. It can be proved by the induction on the length of the inference that if $\mathcal{T} \vdash A$, then $A \in \mathcal{T}$. Note that if $A = A_j$ and $T_i \vdash A$, it must be $A \in \mathcal{T}$ because $T_{\max(i,j)+1}$ is consistent. Suppose that the sequence $\varphi_1, \varphi_2, \ldots, A$ forms the proof of A from $\mathcal{T}$. If the sequence is finite, there must be a set T_i such that $T_i \vdash A$, and $A \in \mathcal{T}$. Thus, suppose that the sequence is countably infinite. We can show that for every i, if φ_i is obtained by an application of an inference rule, and all the premises belong to $\mathcal{T}$, then it must be $\varphi_i \in \mathcal{T}$. If the rule is a finitary one, then there must be a set T_j which contains all the premises and $T_j \vdash \varphi_i$. Reasoning as above, we conclude $\varphi_i \in \mathcal{T}$. Next, we consider the only infinitary rule 3. Let $\varphi_i = B \to P_{\geqslant s}\alpha$ be obtained from the set of premises $\{\varphi_i^k = B \to P_{\geqslant s_k}\gamma : s_k \in S\}$. By the induction hypothesis, $\varphi_i^k \in \mathcal{T}$ for every k. If $\varphi_i \notin \mathcal{T}$, by the step 2b of the construction, there are some l and j such that $\neg(B \to P_{\geqslant s}\alpha), B \to \neg P_{\geqslant s-\frac{1}{l}}\gamma \in T_j$. It means that for some $j' \geqslant j$:

- $B \wedge \neg P_{\geqslant s}\alpha \in T_{j'}$,
- $B \in T_{j'}$,
- $\neg P_{\geqslant s-\frac{1}{i}}\gamma, P_{\geqslant s-\frac{1}{i}}\gamma \in T_{j'}$,

which is in contradiction with consistency of $T_{j'}$. □

The set $\mathcal{T}$ is used to define a tuple $\mathbf{M}_T = \langle W, H, \mu, v\rangle$, where:

- $W = \{w \vDash Cn_C(T)\}$ contains all classical propositional interpretations that satisfy the set $Cn_C(T)$ of all classical consequences of the set T,
- $[\alpha] = \{w \in W : w \vDash \alpha\}$ and $H = \{[\alpha] : \alpha \in \mathrm{For}_C\}$,
- $\mu : H \to [0,1]$ such that $\mu([\alpha]) = \sup_s\{P_{\geqslant s}\alpha \in \mathcal{T}\}$, and
- for every world w and every primitive proposition $p \in \phi$, $v(w,p) =$ true iff $w \vDash p$.

The next theorem states that $\mathbf{M}_T$ is an $LPP_{2,\mathrm{Meas}}$-model.

Theorem 14. *Let $\mathbf{M}_T = \langle W, H, \mu, v\rangle$ be defined as above and $\alpha, \beta \in \mathrm{For}_C$. Then, the following hold:*

(1) *H is an algebra of subsets of W,*
(2) *If $[\alpha] = [\beta]$, then $\mu([\alpha]) = \mu([\beta])$,*
(3) $\mu([\alpha]) \geqslant 0$.
(4) $\mu(W) = 1$ *and* $\mu(\emptyset) = 0$.
(5) $\mu([\alpha]) = 1 - \mu([\neg\alpha])$.
(6) $\mu([\alpha] \cup [\beta]) = \mu([\alpha]) + \mu([\beta])$, *for all disjoint* $[\alpha]$ *and* $[\beta]$.

Proof. (1) Let α, α_1, α_2,... α_n be formulas from For_C. It is not hard to see that the following hold:

- $W = [\alpha \vee \neg\alpha]$, and $W \in H$,
- if $[\alpha] \in H$, then its complement $[\neg\alpha]$ belongs to H, and
- if $[\alpha_1],\ldots, [\alpha_n] \in H$, then the union $[\alpha_1] \cup \ldots \cup [\alpha_n] \in H$ because $[\alpha_1] \cup \ldots \cup [\alpha_n] = [\alpha_1 \vee \ldots \vee \alpha_n]$.

Thus, H is an algebra of subsets of W.

(2) It is enough to prove that $[\alpha] \subset [\beta]$ implies $\mu([\alpha]) \leqslant \mu([\beta])$. By the completeness of the propositional logic, $[\alpha] \subset [\beta]$ means that $\alpha \to \beta \in Cn_C(T)$ and $P_{\geqslant 1}(\alpha \to \beta) \in \mathcal{T}$. By Lemma 10(1) we have that for every $s \in S$, $P_{\geqslant s}\alpha \to P_{\geqslant s}\beta \in \mathcal{T}$. Thus, $\mu([\alpha]) \leqslant \mu([\beta])$.

(3) Since $P_{\geqslant 0}\alpha$ is an axiom, $\mu([\alpha]) \geqslant 0$.

(4) Since $p \vee \neg p \in Cn_C(T)$ and $P_{\geqslant 1}(p \vee \neg p) \in \mathcal{T}$ for every $p \in \phi$, we have $W = [p \vee \neg p]$ and $\mu(W) = 1$. On the other hand, obviously, $\mu(\emptyset) \geqslant 0$. Since $P_{\geqslant 1}(p \vee \neg p) = P_{\geqslant 1-0}(p \vee \neg p) = P_{\leqslant 0}\neg(p \vee \neg p) = P_{\leqslant 0}(p \wedge \neg p) = \neg P_{>0}(p \wedge \neg p)$, by Axiom 3', $\sup_s\{P_{\geqslant s}(p \wedge \neg p) \in \mathcal{T}\} = 0$, and $\mu(\emptyset) = 0$.

(5) Let $r = \mu([\alpha]) = \sup_s\{P_{\geqslant s}\alpha \in \mathcal{T}\}$. Suppose that $r = 1$. By Lemma 12(7), $P_{\geqslant 1}\alpha) \in \mathcal{T}$. Thus, $\neg P_{>0}\neg\alpha (= P_{\leqslant 0}\neg\alpha = P_{\geqslant 1}\alpha)$ belongs to $\mathcal{T}$. If for some $s > 0$, $P_{\geqslant s}\neg\alpha \in \mathcal{T}$, by Axiom 3' it must be $P_{>0}\neg\alpha \in w$, a contradiction. It follows that $\mu([\neg\alpha]) = 1$. Next, suppose that $r < 1$. Then, for every rational number $r' \in (r,1]$, $\neg P_{\geqslant r'}\alpha = P_{<r'}\alpha$, and $P_{<r'}\alpha \in \mathcal{T}$. By Axiom 4, $P_{\leqslant r'}\alpha$ and $P_{\geqslant 1-r'}\neg\alpha$

belong to $\mathcal{T}$. On the other hand, if there is a rational number $r'' \in [0, r)$ such that $P_{\geqslant 1-r''}\neg\alpha \in \mathcal{T}$, then $\neg P_{>r''}\alpha \in \mathcal{T}$, a contradiction. Hence, $\sup_s\{P_{\geqslant s}(\neg\alpha) \in \mathcal{T}\} = 1 - \sup_s\{P_{\geqslant s}\alpha \in \mathcal{T}\}$, i.e., $\mu([\alpha]) = 1 - \mu([\neg\alpha])$.
(6) Let $[\alpha]\cap[\beta] = \emptyset$, $\mu([\alpha]) = r$ and $\mu([\beta]) = s$. Since $[\beta] \subset [\neg\alpha]$, by the above steps (2) and (5), we have $r + s \leqslant r + (1 - r) = 1$. Suppose that $r > 0$, and $s > 0$. By the well known properties of the supremum, for every rational number $r' \in [0, r)$, and every rational number $s' \in [0, s)$, we have $P_{\geqslant r'}\alpha$, $P_{\geqslant s'}\beta \in \mathcal{T}$. It follows by the axiom 5 that $P_{\geqslant r'+s'}(\alpha \vee \beta) \in \mathcal{T}$. Hence, $r + s \leqslant t_0 = \sup_t\{P_{\geqslant t}(\alpha \vee \beta) \in \mathcal{T}\}$. If $r + s = 1$, then the statement trivially holds. Suppose $r + s < 1$. If $r + s < t_0$, then for every rational number $t' \in (r + s, t_0)$ we have $P_{\geqslant t'}(\alpha \vee \beta) \in \mathcal{T}$. We can choose rational numbers $r'' > r$ and $s'' > s$ such that:

$$\neg P_{\geqslant r''}\alpha, \quad P_{<r''}\alpha \in \mathcal{T}, \quad \neg P_{\geqslant s''}\beta, \quad P_{<s''}(\beta) \in \mathcal{T} \quad \text{and} \quad r'' + s'' = t' \leqslant 1.$$

By Axiom 4, $P_{\leqslant r''}\alpha \in \mathcal{T}$. Using Axiom 6 we have

$$P_{<r''+s''}(\alpha \vee \beta) \in \mathcal{T}, \quad \neg P_{\geqslant r''+s''}(\alpha \vee \beta) \in \mathcal{T} \quad \text{and} \quad \neg P_{\geqslant t'}(\alpha \vee \beta) \in \mathcal{T},$$

a contradiction. Hence, $r+s = t_0$ and $\mu([\alpha]\cup[\beta]) = \mu([\alpha])+\mu([\beta])$. Finally suppose that $r = 0$ or $s = 0$. Then we can reason as above, with the only exception that $r' = 0$ or $s' = 0$. □

Theorem 15 (Extended completeness theorem for $LPP_{2,\text{Meas}}$). *A set T of formulas is Ax_{LPP_2}-consistent iff it is $LPP_{2,\text{Meas}}$-satisfiable.*

Proof. The ($\Leftarrow$)-direction follows from the soundness of the above axiomatic system. In order to prove the ($\Rightarrow$)-direction we can construct the $LPP_{2,\text{Meas}}$-model $\mathbf{M}_T$, and show that for every $\varphi \in \text{For}_{LPP_2}$, $\mathbf{M}_T \models \varphi$ iff $\varphi \in \mathcal{T}$.

To begin the induction, let $\varphi = \alpha \in \text{For}_C$. If $\alpha \in Cn_C(T)$, then by the definition of $\mathbf{M}_T$, $\mathbf{M}_T \models \alpha$. Conversely, if $\mathbf{M}_T \models \alpha$, by the completeness of classical propositional logic, $\alpha \in Cn_C(T)$.

Next, let $\varphi = P_{\geqslant s}\alpha$. If $P_{\geqslant s}\alpha \in \mathcal{T}$, then $\sup_r\{P_{\geqslant r}(\alpha) \in \mathcal{T}\} = \mu([\alpha]) \geqslant s$, and $\mathbf{M}_T \models P_{\geqslant s}\alpha$. For the other direction, suppose that $\mathbf{M}_T \models P_{\geqslant s}\alpha$, i.e., that $\sup_r\{P_{\geqslant r}(\alpha) \in \mathcal{T}\} \geqslant s$. If $\mu([\alpha]) > s$, then, by the well known property of supremum and monotonicity of μ, $P_{\geqslant s}\alpha \in \mathcal{T}$. If $\mu([\alpha]) = s$, then by Lemma 12(7), $P_{\geqslant s}\alpha \in \mathcal{T}$.

Let $\varphi = \neg A \in \text{For}_P$. Then $\mathbf{M}_T \models \neg A$ iff $\mathbf{M}_T \not\models A$ iff $A \notin \mathcal{T}$ iff (by Lema 12(1)) $\neg A \in \mathcal{T}$.

Finally, let $\varphi = A \wedge B \in \text{For}_P$. $\mathbf{M}_T \models A \wedge B$ iff $\mathbf{M}_T \models A$ and $\mathbf{M}_T \models B$ iff A, $B \in \mathcal{T}$ iff (by Lema 12(3)) $A \wedge B \in \mathcal{T}$. □

In the last part of this section the canonical model $\mathbf{M}_T$ from Theorem 15 will be used as a weak model, i.e., as a tool in proving completeness with respect to the classes: $LPP_{2,\text{Meas,All}}$, $LPP_{2,\text{Meas},\sigma}$ and $LPP_{2,\text{Meas,Neat}}$.

Theorem 16 (Extended completeness theorem for $LPP_{2,\text{Meas,All}}$). *A set T of formulas is Ax_{LPP_2}-consistent iff it is $LPP_{2,\text{Meas,All}}$-satisfiable.*

Proof. The proof can be obtained by applying the extension theorem for additive measure[2]. on the measure μ from the weak canonical model $\mathbf{M}_T$. Thus, there is a finitely additive measure $\overline{\mu}$ defined on the power set of W that is an extension of the measure μ. □

Theorem 17 (Extended completeness theorem for $LPP_{2,\mathrm{Meas},\sigma}$). *A set T of formulas is Ax_{LPP_2}-consistent iff it is $LPP_{2,\mathrm{Meas},\sigma}$-satisfiable.*

Proof. By the Loeb process and a bounded elementary embedding [46] we can transform the weak canonical model $\mathbf{M}_T$ into a σ-additive probability model ${}^*\mathbf{M}_T$ such that for every formula Φ, $\mathbf{M}_T \models \Phi$ iff ${}^*\mathbf{M}_T \models \Phi$. □

Theorem 18 (Extended completeness theorem for $LPP_{2,\mathrm{Meas,Neat}}$). *A set T of formulas is Ax_{LPP_2}-consistent iff it is $LPP_{2,\mathrm{Meas,Neat}}$-satisfiable.*

Proof. In this proof we use a slightly changed construction of the set $\mathcal{T}$ from Theorem 13. Using the same notation as above, the sequence of sets T_i, $i = 0, 1, 2, \ldots$ is now defined in the following way:

(1) $T_0 = T \cup Cn_C(T) \cup \{P_{\geqslant 1}\alpha : \alpha \in Cn_C(T)\}$
(2) for every $i \geqslant 0$,
   (a) if $T_i \cup \{A_i\}$ is consistent, then $T_{i+1} = T_i \cup \{A_i\}$, otherwise
   (b) if A_i is of the form $\beta \to P_{\geqslant s}\gamma$, then $T_{i+1} = T_i \cup \{\neg A_i, \beta \to \neg P_{\geqslant s-\frac{1}{n}}\gamma\}$, for some positive integer n, so that T_{i+1} is consistent, otherwise
   (c) $T_{i+1} = T_i \cup \{\neg A_i\}$.
   (d) if T_i is enlarged by a formula of the form $P_{=0}\alpha$, add $\neg\alpha$ to T_{i+1} as well.
(3) $\mathcal{T} = \bigcup_{i=0}^{\infty} T_i$.

As it can be seen, the only new step is 2d. We can show that it produces consistent sets, too. So, suppose that for some $\alpha \in \mathrm{For}_C$, $(T_i \cup \{P_{=0}\alpha\}) \cup \{\neg\alpha\} \vdash \bot$. By Deduction theorem, we have that $T_i \cup \{P_{=0}\alpha\} \vdash \alpha$. Since $\alpha \in \mathrm{For}_C$, α belongs to $Cn_C(T)$, and by the construction, we have that $P_{\geqslant 1}\alpha \in T_0$ which leads to inconsistency of $T_i \cup \{P_{=0}\alpha\}$ since:

(1) $T_i, P_{=0}\alpha \vdash P_{\geqslant 1}\alpha$, since $P_{\geqslant 1}\alpha \in T_i$,
(2) $T_i, P_{=0}\alpha \vdash P_{\leqslant 0}\alpha$, by the definition of $P_{=0}$,
(3) $T_i, P_{=0}\alpha \vdash P_{<1}\alpha$, by Axiom 3

and $P_{<1}\alpha = \neg P_{\geqslant 1}\alpha$. The rest of the completeness proof is the same as in Theorem 17. □

The situation that the axiomatic system Ax_{LPP_2} is sound and complete with respect to three different classes of models is similar to the one from the modal framework where, for example, the system K is characterized by the class of all

[2]**Theorem 3.2.10** from [9]. Let C be an algebra of subsets of a set Ω and $\mu(w)$ a positive bounded charge-a finitely additive measure-on C. Let F be an algebra on Ω containing C. Then there exists a positive bounded charge $\overline{\mu(w)}$ on F such that $\overline{\mu(w)}$ is an extension of $\mu(w)$ from C to F and that the range of $\overline{\mu(w)}$ is a subset of the closure of the range of $\mu(w)$ on C

models, but also by the class of all irreflexive models. In other words, LPP_2-formulas cannot express the differences between the mentioned classes of probability models.

3.5. Decidability and Complexity. In this subsection we will consider the problem of satisfiability of For_{LPP_2} formulas. Since there is a procedure for deciding satisfiability and validity for classical propositional formulas, we will consider For_P-formulas only.

So, let $A \in \mathrm{For}_P$. Recall that an atom a of A is a formula of the form $\pm p_1 \wedge \ldots \wedge \pm p_n$, where $\pm p_i$ is either p_i, or $\neg p_i$, and $p_1, \ldots, p_n$ are all primitive propositions appearing in A. Note that for different atoms a_i and a_j we have $\vdash a_i \to \neg a_j$. Thus, in every $LPP_{2,\mathrm{Meas}}$-model $\mu(a_i \vee a_j) = \mu(a_i) + \mu(a_j)$. It is easy, using propositional reasoning and Lemma 10(2), to show that A is equivalent to a formula

$$DNF(A) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} X^{i,j}(p_1, \ldots, p_n)$$

called a disjunctive normal form of$A$, where:

- $X^{i,j}$ is a probability operator from the set $\{P_{\geqslant s_{i,j}}, P_{<s_{i,j}}\}$, and
- $X^{i,j}(p_1, \ldots, p_n)$ denotes that the propositional formula which is in the scope of the probability operator $X^{i,j}$ is in the complete disjunctive normal form, i.e., the propositional formula is a disjunction of the atoms of A.

Theorem 19 (Decidability theorem). *The logic LPP_2 is decidable.*

Proof. As it is noted above, a For_P-formula A is equivalent to

$$DNF(A) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} X^{i,j}(p_1, \ldots, p_n).$$

A is satisfiable iff at least one disjunct from $DNF(A)$ is satisfiable. Let the measure of the atom a_i be denoted by y_i. We use an expression of the form $a_t \in X(p_1, \ldots, p_n)$ to denote that the atom a_t appears in the propositional part of $X(p_1, \ldots, p_n)$. A disjunct $D = \bigwedge_{j=1}^{k} X^j(p_1, \ldots, p_n)$ from $DNF(A)$ is satisfiable iff the following system of linear equalities and inequalities is satisfiable:

$$\begin{gathered}
\sum_{i=1}^{2^n} y_i = 1 \\
y_i \geqslant 0 \text{ for } i = 1, \ldots, 2^n \\
\sum_{a_t \in X^1(p_1,\ldots,p_n) \in D} y_t \begin{cases} \geqslant s_1 & \text{if } X^1 = P_{\geqslant s_1} \\ < s_1 & \text{if } X^1 = P_{<s_1} \end{cases} \\
\ldots \\
\sum_{a_t \in X^k(p_1,\ldots,p_n) \in D} y_t \begin{cases} \geqslant s_k & \text{if } X^k = P_{\geqslant s_k} \\ < s_k & \text{if } X^k = P_{<s_k} \end{cases}
\end{gathered} \tag{5}$$

Since the problem of $LPP_{2,\text{Meas}}$-satisfiability of A is reduced to the linear systems solving problem, the satisfiability problem for LPP_2-logic is decidable. Finally, since A is $LPP_{2,\text{Meas}}$-valid iff $\neg A$ is not $LPP_{2,\text{Meas}}$-satisfiable, the validity problem is also decidable. □

We can show that the $LPP_{2,\text{Meas}}$-satisfiability problem is NP-complete.

Theorem 20. *The $LPP_{2,\text{Meas}}$-satisfiability problem is* NP*-complete.*

Proof. The lower bound follows from the complexity of the same problem for classical propositional logic. The upper bound is a consequence of the NP-complexity of the satisfiability problem for weight formulas from [27, Theorem 2.9][3]. □

3.6. A heuristical approach to the $LPP_{2,\text{Meas}}$-satisfiability problem. Since the $LPP_{2,\text{Meas}}$-satisfiability problem is NP-complete, it is natural to try to solve its instances using heuristics. In this section we describe such an approach which is based on genetic algorithms.

Genetic algorithms (GA) use populations of individuals. Each individual (also called chromosome) is seen as a possible solution in the search space for the particular problem. Thus, a GA can be seen as a searching procedure for the global optima of the corresponding problem. Individuals are represented by genetic code over a finite alphabet. An evaluation function assigning fitness values to individuals has to be defined. Fitness values indicate quality of the corresponding individuals, while average fitness of entire populations may be good measures of obtained quality of the procedures. GA's consist of applications of the genetic operators to populations that must ensure that average fitness values are continually improved from each generation to subsequent. Basic genetic operators are selection, crossover and mutation, but some additional operators such as inversion, local search, etc., may be used.

Selection mechanism favourizes highly fitted individuals (as well as parts of genetic code of individuals, i.e., genes) to have better chances for reproduction into

---

[3]Statements about complexity of the satisfiability problem for weight formulas from [27]. $|A|$ and $||A||$ denote the length of A (the number of symbols required to write A), and the length of the longest coefficient appearing in A, when written in binary, respectively. The size of a rational number a/b, where a and b are relatively prime, is defined to be the sum of lengths of a and b, when written in binary.

Theorem 2.6 Suppose A is a weight formula that is satisfied in some measurable probability structure. Then A is satisfied in a structure (S, H, μ, v) with at most $|A|$ states where every set of states is measurable, and where the probability assigned to each state is a rational number with size $O(|A|||A|| + |A| \log(|A|))$.

Lemma 2.7 If a system of r linear equalities and/or inequalities with integer coefficients each of length at most l has a nonnegative solution, then it has a nonnegative solution with at most r entries positive, and where the size of each member of the solution is $O(rl + r \log(r))$.

Lemma 2.8 Let A be a weight formula. Let $M = (S, H, \mu, v)$ and $M_0 = (S, H, \mu, v')$ be probability structures with the same underlying probability space (S, H, μ). Assume that $v(w, p) = v'(w, p)$ for every state w and every primitive proposition p that appears in A. Then $M \models A$ iff $M_0 \models A$.

Theorem 2.9 The problem of deciding whether a weight formula is satisfiable in a measurable probability structure is NPcomplete.

```
InputData();
PopulationInit();
while(not FinishedGA()){
      for (i = 0 ; i < N_pop ; i++) p_i = ObjectiveFunction();
      HeuristicImprovement();
      ComputeFitnesses();
      Selection();
      Crossover();
      Mutation();
}
OutputResults();
```

FIGURE 1. A general description of GA's

next generations. On the other hand, chances for reproduction for less fitted members are reduced, and they are gradually wiped out from populations. Crossover operator partitions a population into a set of pairs of individuals named parents. For each pair a recombination of their genetic material is performed with some probability. In that way nondeterministic exchange of genetic material in populations is obtained. Multiple usage of selection and crossover operators may produce that the variety of genetic materials is lost. It means that some areas of search spaces become not reachable. This usually causes the convergence in local optimums far from the global optimal values. Mutation operator can help to avoid this shortcoming. Parts of individuals (genes) can be changed with some small probability to increase diversibility of genetic material. An initial population is usually generated by random, although sometimes it may be fully or partially produced by an initial heuristic. A general description of GA's is given in Figure 1, where N_{pop} and p_i denote the number of individuals and their objective values, respectively. The objective value of an individual corresponds to the value which the individual owns in the case of the considered problem. The for-loop is repeated until a finishing criterion (the global optima is found, the maximal number of iterations is reached,...) is satisfied. Since the procedure is not complete, if the maximal number of iterations is reached, we do not know whether the considered problem is solvable. HeuristicImprovement() can be optionally included to improve efficiency of GA and/or to help the procedure to escape from local optima.

In this section, we slightly change syntax of probabilistic formulas. Namely, as we will mention below in Section 10, sometimes is suitable to consider boolean combinations of basic weight formulas of the form: $a_1w(\alpha_1) + \cdots + a_nw(\alpha_n) \geqslant c$, where a_i's and c are rational numbers, and α_i's are classical propositional formulas containing primitive propositions from ϕ. The intended meaning of $w(\alpha)$ is "the probability of α". Note that $w(\alpha) \geqslant s$ can be written as $P_{\geqslant s}\alpha$ in our notation. A weight literal is an expression of the form $\sum_i a_i w(\alpha_i) \geqslant c$ or $\sum_i a_i w(\alpha_i) < c$. The logic that allows such kind of formulas is still NP-complete-which can be proved as

above, i.e., by reducing the $LPP_{2,\text{Meas}}$-satisfiability problem to linear programming problem – so by using this logic we just add some expressiveness to our language.

Since For$_P$-formulas can be equivalently translated into their disjunctive normal forms, and a disjunction is satisfiable if at least one disjunct is satisfiable, in the sequel we will only consider formulas of the following form:

$$\bigwedge_{j=1}^{k} a_1^j w(\text{CDNF}(\alpha_1^j)) + \cdots + a_{n_j}^j w(\text{CDNF}(\alpha_{n_j}^j)) rho_j \; c^j,$$

where $\rho_j \in \{\geqslant, <\}$, a_i^j's and c^j are rational numbers, and CDNF(α) denotes the complete disjunctive normal form of α. We say that such a formula is in the weight conjunctive form (wfc-form). Also, we will use $at \in \text{CDNF}(\alpha)$ to denote that the atom at appears in CDNF(α).

Example 21. The formula $w(p \to q) + w(p) \geqslant 1.7 \wedge w(q) \geqslant 0.6$ is satisfiable since the same holds for the linear system

$$\begin{aligned}
&\mu(p \wedge q) + \mu(p \wedge \neg q) + \mu(\neg p \wedge q) + \mu(\neg p \wedge \neg q) = 1\\
&\mu(p \wedge q) \geqslant 0\\
&\mu(p \wedge \neg q) \geqslant 0\\
&\mu(\neg p \wedge q) \geqslant 0\\
&\mu(\neg p \wedge \neg q) \geqslant 0\\
&\mu(p \wedge \neg q) + \mu(\neg p \wedge q) + \mu(\neg p \wedge \neg q) + 2\mu(p \wedge q) \geqslant 1.7\\
&\mu(p \wedge q) + \mu(\neg p \wedge q) \geqslant 0.6.
\end{aligned}$$

□

The input for the $LPP_{2,\text{Meas}}$-satisfiability checker based on genetic algorithms is a weight formula f in the wfc-form with L weight literals. Without loss of generality, we demand that classical formulas appearing in weight terms are in disjunctive normal form. Let $\phi(f) = \{p_1, \ldots, p_N\}$ denote the set of all primitive propositions from f, and $|\phi(f)| = N$.

An individual M consists of L pairs of the form (atom, probability) that describe a probabilistic model. The first coordinate is given as a bit string of length N, where 1 at the position i denotes $\neg p_i$, while 0 denotes p_i. Probabilities are represented by floating point numbers.

For an individual $M = \big((at_1, \mu(at_1)), \ldots, (at_N, \mu(at_N))\big)$, the linear system is equivalent to: $\bigvee_{i=1}^{L} \big(\sum_{j=1}^{L} a_{ij}\mu(at_j)\big)\rho_i\, c_i\big)$. Note that it is possible that some $a_{ij} = 0$, though $[a_{ij}]$ matrix is usually not sparse.

The individuals are evaluated using function $d(M)$, which measures a degree of unsatisfiability of an individual M. Function $d(M)$ is defined as the distance between left and right hand side values of the weight literals not satisfied in the model described by M:

$$d(M) = \sqrt{\sum_{M \not\models t_i \; \rho_i \; c_i} \left[a_1^i \sum_{at \in \text{CDNF}(\alpha_1^i)} \mu(at) + \cdots + a_{n_i}^i \sum_{at \in \text{CDNF}(\alpha_{n_i}^i)} \mu(at) - c_i\right]^2}.$$

If $d(M) = 0$, all the inequalities in the linear system are satisfied, hence the individual M is a solution.

Some features of GA have been set for all tests:

- the population consists of 10 individuals,
- one set of tests has been performed with a population of 20 individuals,
- selection is performed using the rank-based roulette operator (with the rank from 2.5 for the best individual to 1.6 for the worst individual-the step is 0.1),
- The crossover operator is one-point, with the probability 0.85
- the elitist strategy with one elite individual is used in the generation replacement scheme,
- multiple occurrences of an individual are removed from the population.

Two problem-specific *two-parts* mutation operator were used. The first operator (*TP1*) features two different probabilities of mutation for the two parts (*atoms, probabilities*) of an individual; after mutation, the real numbers in *probabilities* part of an individual have to be scaled since their sum must equal 1. The second operator (*TP2*) is a combination of ordinary mutation on *atoms* part, and a special mutation on *probabilities* part of an individual. Instead of performing mutation on two bits in the representation of *probabilities* part, two members p_{i1}, p_{i2} of *probabilities* part are chosen randomly and then replaced with random p'_{i_1}, p'_{i_2}, such that $p_{i_1} + p_{i_2} = p'_{i_1} + p'_{i_2}$ and $0 \leqslant p'_{i_1}, p'_{i_2} \leqslant 1$. The sum of probabilities does not change and no scaling is needed.

We have experimented with the following choices in the local search procedure:

LS1 (LS denotes "local search"): For an individual M all the weight literals are divided into two sets: the first set (B) contains all satisfied literals, while the second one (W) contains all the remaining literals. The literal $t_B\ \rho_B\ c_B \in B$ (called the best one) with the biggest difference $|\mu(t_B) - c_B|$ between the left and the right side, and the literal $t_W\ \rho_W\ c_W \in W$(the worst one) with the biggest difference $|\mu(t_W) - c_W|$ are found. Two sets of atoms are determined: the first set $B_{\mathrm{At}(f)}$ contains all the atoms from M satisfying at least one classical formula α_i^B from $t_B = a_1^B w(\alpha_1^B) + \cdots + a_{k_B}^B w(\alpha_{k_B}^B)$, while the second one $W_{\mathrm{At}(f)}$ contains all the atoms from M satisfying at least one classical formula α_i^W from $t_W = a_1^W w(\alpha_1^W) + \cdots + a_{k_W}^W w(\alpha_{k_W}^W)$. The probabilities of a randomly selected atom from $B_{\mathrm{At}(f)} \setminus W_{\mathrm{At}(f)}$ and a randomly selected atom from $W_{\mathrm{At}(f)} \setminus B_{\mathrm{At}(f)}$ are changed so that $t_B\ \rho_B\ c_B$ remains satisfied, while the distance $|\mu(t_W) - c_W|$ is decreased or $t_W\ \rho_W\ c_W$ is satisfied.

LS2: For na individual M, the *worst* weight literal $t_W\ \rho_W\ c_W$ from W (the set of unsatisfied literals) with the biggest difference $|\mu(t_W) - c_W|$ is found. The literal can be represented as $\sum_{j=1}^{L} a_{Wj}\mu(at_j)\rho_W c_W$. We try to change the vector of probabilities $[\mu(at_j)]$, so that the linear equation $\sum_{j=1}^{L} a_{Wj}\mu(at_j) = c_W$ is satisfied. The equation $\sum_{j=1}^{L} a_{Wj}\mu(at_j) = c_W$ represents a hyper-plane in R^n while $[a_{Wj}]$ denotes a vector normal to the hyper-plane. The projection of $[\mu(at_j)]$ to the hyper-plane, which satisfies the equation, is $[\mu'(at_j)] = [\mu(at_j)] + k_W[a_{Wj}]$. The

calculation of k and the projection vector is simple and straightforward, and gives

$$k = \frac{c_w - a_W \circ [\mu(at_j)]}{|a_W|^2} = \frac{c_w - \sum_{j=1}^{L} \mu(at_j) a_{W_j}}{\sum_{j=1}^{L} {a_{W_j}}^2}.$$

We set the new vector of probabilities to be

$$[\mu''(at_j)] = \frac{[\max\{\mu'(at_j), 0\}]}{\sum_{k=1}^{L} \max\{\mu'(at_k), 0\}}$$

(negative coordinates are replaced with 0, and the vector is scaled so that the sum of its coordinates $\sum_{j=1}^{L} \mu''(at_j)$ equals 1).

LS3 is similar to LS2, with the difference being made when choosing the weight literal $t_W\ \rho_W\ c_W$ from W (the set of unsatisfied literals). The chosen literal is the one with the smallest difference $|\mu(t_W) - c_W|$; it is the *best bad literal*.

LS4 is similar to LS2 and LS3. Instead of calculating the projection $[\mu'(at_j)] = [\mu(at_j)] + k_W[a_{W_j}]$ for one chosen weight literal $t_W\ \rho_W\ c_W$ from W, we calculate $k_{W_i}[a_{W_i j}]$ for each literal $t_{W_i}\ \rho_{W_i}\ c_{W_i}$ from W (the set of unsatisfied literals) and calculate the *intermediate* vector $[\mu'(at_j)]$, by adding the linear combination to the original vector: $[\mu'(at_j)] = [\mu(at_j)] + \sum_{W_i} k_{W_i}[a_{W_i j}]$. The new vector of probabilities $[\mu''(at_j)]$ is then calculated in same fashion as in LS2.

In our methodology, introduced in [86], the performance of the system is evaluated on a set of PSAT-instances, i.e., on a set of randomly generated formulas in the wfc-form (with classical formulas in disjunctive normal form). The advantage of this approach is that a formula can be randomly generated according to the following parameters: N-the number of propositional letters, L-the number of weight literals, S-the maximal number of summands in weight terms, and D-the maximal number of disjuncts in DNF's of classical formulas. The considered set of test problems contains 27 satisfiable formulas. Three PSAT-instances were generated for each of 9 pairs of (N, L), where $N \in \{50, 100, 200\}$, and $L \in \{N, 2N, 5N\}$. For every instance $S = D = 5$. Having the above parameters, L atoms and their probabilities (with the constraint that the sum of probabilities must be equal to 1) are chosen. Next, a formula f containing L basic weight formulas is generated. It contains primitive propositions from the set $\{p_1, \ldots, p_N\}$ only. Every weight literal contains at most S summands in its weight term. Every classical formula is in disjunctive normal form with at most D disjuncts, while every disjunct is a conjunction of at most N literals. For every weight term t coefficients are chosen, and the value of t is computed. Next, the sum $sp(t)$ of positive coefficients and the sum $sn(t)$ of negative coefficients are computed. Finally, the right side value of the weight literals between $sp(t)$ and $sn(t)$, and the relation sign are chosen such that f is satisfiable.

We prefer to test more problem instances of different sizes (even very large scale instances) rather than making more trials on a smaller set of instances (of smaller or average size). Since the tests are of large sizes, the necessity to perform them in a reasonable time imposed to set the maximal number of generations to be: 10000 for $N = 50$, 7000 for $N = 100$ and 5000 for $N = 200$.

TABLE 1. Average time (rounded to seconds) used by the test computer to execute *successful* tests for some selected parameters. (*Note: Value 0 means that the average time was less than half second.*)

L, N, inst. no.	Table 1 TP2(12,4) No LS		Table 2 TP2(12,4) 10 individuals LS's applied in each generation				Table 3 TP2(12,4) 10 individuals LS's applied in each third generation				Table 5 10 individuals Combination of LS's applied in each generation		
	10 ind.	20 ind.	LS1	LS2	LS3	LS4	LS1	LS2	LS3	LS4	TP2(12,4)	TP2(24,8)	TP2(48,16)
50, 50, 1	0	1	0	0	0	0	0	0	0	0	0	0	0
50, 50, 2	0	1	0	0	0	0	0	0	0	0	0	0	0
50, 50, 3	0	1	0	1	1	0	0	0	0	0	0	0	0
50, 100, 1	1	1	1	0	0	2	1	0	0	1	1	0	1
50, 100, 2	1	2	1	1	2	2	1	1	1	2	2	2	3
50, 100, 3	3	3	1	2	7	10	1	2	3	4	1	3	3
50, 250, 1	16	20	28	16	16	39	22	14	11	21	40	35	42
50, 250, 2	51	56	24	38	34	97	26	35	30	50	68	70	132
50, 250, 3	18	20	18	9	17	25	10	8	13	14	15	16	19
100, 100, 1	0	1	0	0	0	0	0	0	0	0	0	0	0
100, 100, 2	0	1	0	0	0	0	0	0	0	0	0	0	0
100, 100, 3	0	1	0	0	0	1	1	0	0	0	1	1	1
100, 200, 1	8	12	10	3	8	9	6	3	8	7	5	5	7
100, 200, 2	2	3	1	3	2	4	1	3	1	4	4	1	2
100, 200, 3	1	3	4	1	2	26	2	1	1	2	2	2	2
100, 500, 1	187	236	170	130	149	384	94	145	244	228	269	294	271
100, 500, 2	295	309	242	241	298	333	169	306	151	228	236	260	480
100, 500, 3	484	575	326	509	416	775	296	390	355	461	1019	777	671
200, 200, 1	58	91	71	108	56	134	34	78	66	3471	146	270	202
200, 200, 2	5	6	11	7	7	14	11	7	10	9	13	11	9
200, 200, 3	2	3	4	1	2	2	4	1	1	4	4	2	3
200, 400, 1	12	11	4	6	5	25	6	7	5	14	8	11	7
200, 400, 2	238	286	N/A	195	163	484	N/A	171	161	296	479	686	1128
200, 400, 3	205	230	N/A	174	205	247	N/A	153	201	208	419	334	374
200, 1000, 1	1593	2173	3064	888	1347	2972	2307	811	1271	1865	2363	2087	2032
200, 1000, 2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	19582	19977
200, 1000, 3	1489	1861	3298	1364	792	3548	2456	1135	1080	2023	2818	2770	2778

As an illustration of the corresponding results we give Table 1 which contains the average running time of successful tests as measured on our test computer (a Pentium P4 2.4GHz, 512MB-based Linux station). The table shows running times only for selected tests. Columns 2 and 3 show times for tests without LS's, with different population size (10 individuals vs 20 individuals). Increased population size does result in smaller number of iterations needed to find the solution, but the computational cost for each iteration is increased and the overall computational cost is greater than with smaller population size. In columns 4-7 and 8-11 we can compare the efficiency of various LS's. It is clear that LS2 and LS3 are more efficient than LS1 and LS4 when used for large problem instances, however it is not clear which of them is the most efficient. The running times in columns 8–11 (LS's applied in each third generation) are on average smaller than times in columns 4-7 (LS's applied in each generation). However, this does not mean that the principle of reducing application of LS's to each third generation is always more efficient. Finally, columns 12-14 show execution times for tests using combination of LS's. Combined usage of LS's is not justified in terms of time efficiency, but it is justified in terms of increased success rate. Higher mutation rate in this setup leads to better time efficiency and higher success rate, except for a few less complex problem instances.

4. Some variants of the logic LPP_2

The lack of compactness in the presence of a finitary axiomatization might cause a logical problem: there are consistent sets of formulas that have no model. Example 4 contains such a set for LPP_2. One way to avoid consistency of unsatisfiable sets is to employ infinitary logic as we do above. On the other hand, the lack of compactness motivates also investigations of models in which probabilities have a fixed finite range in which case a finitary axiomatization does not imply the above problem any more. In this section we present three logics inspired by the idea of restricting the range of probability measures. In the first logic (denoted $LPP_2^{\mathrm{Fr}(n)}$) we give a finitary sound and complete axiomatization with respect to a class of models with measures which have a fixed finite range of the form $\{0, 1/n, 2/n, \ldots, n-1/n, 1\}$. Then we introduce another logic (denoted $LPP_2^{A,\omega_1,\mathrm{Fin}}$) in which the assumption about the range of the measure is relaxed, and we consider the class of all probabilistic models whose measures have arbitrary finite ranges (without the requirement that the range is fixed in advance). Finally, we analyze the logic LPP_2^S. It involves a rule that enables us to syntactically define the range of the probability function which will appear in the interpretation.

4.1. Logic $LPP_2^{\mathrm{Fr}(n)}$. Let n be a fixed positive integer, and Range $= \{0, 1/n, \ldots, (n-1)/n, 1\}$. If $s \in [0,1)$, then s^+ denotes $\min\{r \in \text{Range} : s < r\}$. If $s \in (0,1]$, $s^- = \max\{r \in \text{Range} : s > r\}$. The most of the notions defined in Section 3 are also used for the logic $LPP_2^{\mathrm{Fr}(n)}$. The main, but important, differences are:

- in Definition 1-the finitely additive measure μ maps the algebra H to Range and

- in Definition 6-proofs are finite sequences of formulas.

Note that $LPP_2^{\mathrm{Fr}(n)}$-models are given relatively to n, and that different choices of n produce different logics. The axiomatic system $Ax_{LPP_2^{\mathrm{Fr}(n)}}$ contains all the axioms from the system Ax_{LPP_2}, and the inference rules 1 and 2 (but note Rule 3), as well as the following new axiom:

(7) $P_{>s}\alpha \to P_{\geqslant s^+}\alpha$

Since the only infinitary inference rule from Ax_{LPP_2} (Rule 3) is not included in $Ax_{LPP_2^{\mathrm{Fr}(n)}}$, it is a finitary axiomatic system. Nevertheless, many statements from the previous section still hold. The next lemma states that Axiom 7 implies that the range of measures must be the set Range.

Lemma 22. *Let α be a sentence. Then:*

(1) $\vdash P_{<r}\alpha \to P_{\leqslant r^-}\alpha$,
(2) $\vdash P_{>r}\alpha \leftrightarrow P_{\geqslant r^+}\alpha$,
(3) $\vdash P_{\leqslant r^-}\alpha \leftrightarrow P_{<r}\alpha$,
(4) $\vdash \bigvee_{s\in \mathrm{Range}} P_{=s}\alpha$,
(5) $\vdash \underline{\vee}_{s\in \mathrm{Range}} P_{=s}\alpha$, *where $\underline{\vee}$ denotes the exclusive disjunction.*

Proof. (1) The considered formula is equivalent to Axiom 7 because $P_{>r}\alpha = \neg P_{\leqslant r}\alpha = \neg P_{\geqslant 1-r}\neg\alpha = P_{<1-r}\neg\alpha$, and $P_{\geqslant r^+}\alpha = P_{\geqslant 1-(1-r^+)}\alpha = P_{\leqslant 1-r^+}\neg\alpha = P_{\leqslant (1-r)^-}\neg\alpha$.

(2) The formula is obtained from the axioms 7 and 3'.

(3) The formula is obtained from Axiom 3, and Lemma 22(1).

(4) From Axiom 2' $P_{\leqslant 1}\alpha$ $(= \neg P_{>1}\alpha)$, we have $\vdash (P_{\geqslant 1}\alpha \vee \neg P_{\geqslant 1}\alpha) \wedge \neg P_{>1}\alpha$. Thus,

$$\vdash (P_{\geqslant 1}\alpha \wedge \neg P_{>1}\alpha) \vee (\neg P_{\geqslant 1} \wedge \neg P_{>1}\alpha).$$

From $P_{\geqslant 1}\alpha \wedge \neg P_{>1}\alpha = P_{=1}\alpha$, and $\vdash P_{<1}\alpha \to P_{\leqslant 1}\alpha$, we have $\vdash P_{=1}\alpha \vee P_{<1}\alpha$. From $\vdash P_{<1}\alpha \leftrightarrow ((P_{\geqslant 1-}\alpha \vee \neg P_{\geqslant 1-}\alpha) \wedge P_{<1}\alpha)$, $\vdash (P_{\geqslant s}\alpha \to P_{\geqslant s^-}\alpha) \leftrightarrow (P_{<s^-}\alpha \to P_{<s}\alpha)$, we have

$\vdash P_{<1}\alpha \leftrightarrow ((P_{\geqslant 1-}\alpha \wedge \neg P_{>1-}\alpha) \vee (P_{<1-}\alpha \wedge P_{<1}\alpha))$, and $\vdash P_{=1}\alpha \vee P_{=1-}\alpha \vee P_{<1-}\alpha$.

In such a way we obtain $\vdash (\bigvee_{s\in \mathrm{Range}} P_{=s}\alpha) \vee P_{<0}\alpha$. Since $\vdash \neg P_{<0}\alpha$, we finally have $\vdash \bigvee_{s\in \mathrm{Range}} P_{=s}\alpha$.

(5) From $P_{=r}\alpha = P_{\geqslant r}\alpha \wedge \neg P_{>r}\alpha$, and the axiom 3, we have $\vdash P_{=r}\alpha \to \neg P_{=s}\alpha$, for $s > r$. Similarly, by the axiom 3', we have $\vdash P_{=r}\alpha \to \neg P_{=s}\alpha$, for $s < r$. It follows that $\vdash P_{=r}\alpha \to \neg P_{=s}\alpha$, for $r \neq s$, and $\vdash \underline{\vee}_{s\in \mathrm{Range}} P_{=s}\alpha$. □

The completeness proofs for the classes:

$$LPP_{2,\mathrm{Meas}}^{\mathrm{Fr}(n)}, \quad LPP_{2,\mathrm{Meas,All}}^{\mathrm{Fr}(n)}, \quad LPP_{2,\mathrm{Meas},\sigma}^{\mathrm{Fr}(n)} \quad \text{and} \quad LPP_{2,\mathrm{Meas,Neat}}^{\mathrm{Fr}(n)}$$

are similar to the corresponding proofs from the previous section. In the sequel we sketch this proof and emphasize some modified steps.

We begin as in the statements 8, 9 and 10. In the counterpart of Theorem 13 we do not use the step 2b of the construction of a maximal consistent set, but otherwise follow the corresponding proof. Then, the statements 12(1)–12(6) obviously hold,

while Lemma 12(7) needs some explanation. By Lemam 22(5), the supremum s of the set $\{r : P_{\geqslant r}\alpha \in T\}$ must be in the set Range. Also, for that s, it must be $P_{\geqslant s}\alpha \in T$, where T is the considered maximal consistent set. Thus, Lemma 12(7) holds. A canonical model $\mathbf{M}_T = \langle W, H, \mu, v\rangle$ is introduced as above. Note that in the counterpart of Theorem 14 for every formula $\alpha \in \mathrm{For}_C$, $\sup\{r : P_{\geqslant r}\alpha \in T\}$ is the same as $\max\{r : P_{\geqslant r}\alpha \in T, r \in \mathrm{Range}\}$, because the set Range is finite. Theorems 15– 18 can be now proved similarly as it is done above.

Theorem 23 announces a property that does not hold for the systems considered in the previous section. Another difference between logics from this and the previous sections is illustrated in Example 24.

Theorem 23 (Compactness theorem for $LPP_2^{\mathrm{FR}(n)}$). *Let L be any class of models considered in this section and T be a set of formulas. If every finite subset of T is L-satisfiable, then T is L-satisfiable.*

Proof. If T is not L-satisfiable, then it is not $Ax_{LPP_2^{\mathrm{FR}(n)}}$-consistent. It follows that $T \vdash \bot$. Since the axiomatic system $Ax_{LPP_2^{\mathrm{FR}(n)}}$ is finitary one, there must be a finite set $T' \subset T$ such that $T' \vdash \bot$. It is a contradiction because every finite subset of T is both L-satisfiable and $Ax_{LPP_2^{\mathrm{FR}(n)}}$-consistent. □

Example 24. For every positive integer n and Range defined as above, it is easy to construct an $LPP_{2,\mathrm{Meas}}$-model $\mathbf{M} = \langle W, H, \mu, v\rangle$ which does not satisfy that Axiom 7. For example, let $n = 3$, and $p \in \phi$:

- $W = \{w_1, w_2\}$,
- H is the power set of W
- $\mu(w_1) = 1/2$, $\mu(w_2) = 1/2$ and
- $v(w_1, p) = \mathrm{true}$, $v(w_2, p) = \mathrm{false}$.

Since $\mu([p]_{\mathbf{M}}) = 1/2$, obviously $\mathbf{M} \models P_{>1/3}p$, and $\mathbf{M} \not\models P_{\geqslant 2/3}p$, so the instance $P_{>1/3}p \to P_{\geqslant 2/3}p$ of Axiom 7 does not hold in $\mathbf{M}$. □

Finally, decidability of the satisfiability problem for the classes of models considered in this section can be proved similarly as Theorem 19. Only, note that the measures of atoms must be in the set Range. Since that set is always finite, there are only finitely many possibilities for such distributions, and decidability easily follows.

4.2. Logic $LPP_2^{A,\omega_1,\mathrm{Fin}}$. In Section 4.1 the considered measures have a fixed finite range. Using ideas from [98], that assumption is relaxed, and we prove the completeness theorem with respect to the class of all probabilistic models whose measures have arbitrary finite ranges (without the requirement that the range is fixed in advance). In the sequel some notions from [4] are used.

Let A be a countable admissible set and $\omega \in A$. We use $LPP_2^{A,\omega_1,\mathrm{Fin}}$ to denote our logic. The language of $LPP_2^{A,\omega_1,\mathrm{Fin}}$ is a subset of A. It is the classical propositional language ($\neg$, $\wedge$, $\vee$) augmented by a list of unary probabilistic operators of the form $P_{\geqslant s}$, for every $s \in [0,1] \cap A$. An important characteristic of $LPP_2^{A,\omega_1,\mathrm{Fin}}$ is that the conjunction symbol and the disjunction symbol may be applied to *finite or*

countable sets of probability formulas. It means that if $G \in A$ is a set of formulas of $LPP_2^{A,\omega_1,\mathrm{Fin}}$, then: $\bigwedge_{B\in G}\Phi$ and $\bigvee_{B\in G}\Phi$ and are also $LPP_2^{A,\omega_1,\mathrm{Fin}}$-formulas (but note that all formulas from the set G must be from For_P). For an $LPP_2^{A,\omega_1,\mathrm{Fin}}$-formula Φ, the formula $\Phi\neg$ is obtained by moving a negation inside the formula Φ over the classical connectives. For example, $(\bigwedge_{\Phi\in G}\Phi)\neg$ denotes $\bigvee_{\Phi\in G}\neg\Phi$, and similarly for the other classical connectives.

Here we consider a particular subclass of the class $LPP_{2,\mathrm{Meas}}$ of all measurable probabilistic models. We denote it $LPP_{2,\mathrm{Meas}}^{A,\omega_1,\mathrm{Fin}}$,and it contains all measurable models whose measures have finite ranges. The satisfaction relation $\models$ generalizes the corresponding relation from Definition 2. The new cases are related to infinitary formulas:

- if G is a finite or countable set of For_P-formulas, $\mathbf{M} \models \bigwedge G$ iff for every $B \in G$, $\mathbf{M} \models B$, and
- if G is a finite or countable set of For_P-formulas, $\mathbf{M} \models \bigvee G$ iff there is some $b \in G$ so that $\mathbf{M} \models B$.

The axiomatic system $Ax_{LPP_2^{A,\omega_1,\mathrm{Fin}}}$ contains all the axioms and rules from the system Ax_{LPP_2}, and also the following new axioms:

(7) $(\neg\Phi) \leftrightarrow (\Phi\neg)$
(8) $(\bigwedge_{B\in G} B) \to C$, $C \in G$, $G \in A$, G is a set of probability formulas
(9) $\bigvee_{c>0}\bigwedge_{\alpha\in G}(P_{>0}\alpha \to P_{>c}\alpha)$, $G \in A$, G is a set of classical propositional formulas

and the rule

(4) From $B \to C$, for all $C \in G$, infer $B \to \bigwedge_{C\in G} C$, G is a set of probability formulas

introduced in [53]. In the completeness proof a result[4] from [9] and the weak-strong model construction from [98] will be used. A weak model is an $LPP_{2,\mathrm{Meas}}^{A,\omega_1,\mathrm{Fin}}$-model defined above.

Theorem 25. *An $LPP_2^{A,\omega_1,\mathrm{Fin}}$-formula Φ is consistent iff it is satisfiable in a weak model in which every $LPP_2^{A,\omega_1,\mathrm{Fin}}$-theorem is true.*

Proof. The simpler direction follows from the soundness of the axiomatic system. For the other direction, let $A_1, A_2, \ldots$ be an enumeration of all $LPP_2^{A,\omega_1,\mathrm{Fin}}$-$\mathrm{For}_P$-formulas. We modify the construction from Theorem 13:

(1) $T_0 = \{\Phi\} \cup Cn_C(\Phi) \cup \{P_{\geqslant 1}\alpha : \alpha \in Cn_C(\Phi)\}$
(2) for every $i \geqslant 0$,
  (a) if $A_i = \bigvee_{B\in G} B$ and $T_i \cup \{A_i\}$ is consistent, then for some $B \in G$, $T_{i+1} = T_i \cup \{A_i\} \cup \{B\}$ such that T_{i+1} is consistent, otherwise
  (b) if $T_i \cup \{A_i\}$ is consistent, then $T_{i+1} = T_i \cup \{A_i\}$, otherwise
  (c) if A_i is of the form $\beta \to P_{\geqslant s}\gamma$, and $T_i \cup \{A_i\}$ is not consistent, then $T_{i+1} = T_i \cup \{\neg A_i, \beta \to \neg P_{\geqslant s-\frac{1}{n}}\gamma\}$, for some positive integer n, so that T_{i+1} is consistent, otherwise

[4] **Theorem 3.2.10 from** [9] If μ is a finitely additive measure and there is a real number $c \in (0,1)$ such that $\mu(\theta) > c$, whenever $\mu(\theta) \neq 0$, then μ has a finite range.□

(d) $T_{i+1} = T_i \cup \{\neg A_i\}$.

(3) $\mathcal{T} = \bigcup_{i=0}^{\infty} T_i$.

We can show that every T_i obtained by the new step in the construction (the step 2a) is also consistent. To prove that, suppose that $T_i \cup \{A_i\}$ is consistent, where $A = \bigvee_{B \in G} B$, but that for every $B \in G$, the set $T_{i+1} = T_i \cup \{A_i\} \cup \{B\}$ is not consistent. It means that

- $T_i \cup \{A_i\} \cup \{B\} \vdash \bot$, for every $B \in G$
- $T_i \cup \{A_i\} \vdash \neg B$, for every $B \in G$
- $T_i \cup \{A_i\} \vdash \bigwedge_{B \in G} \neg B$, by Rule 4
- $T_i \cup \{A_i\} \vdash \neg \bigvee_{B \in G} B$, by Axiom 7

which contradicts consistency of $T_i \cup \{A_i\}$. Then, we can follow the completeness prof for $LPP_{2,\mathrm{Meas}}$, and construct the canonical model $\mathbf{M}_\Phi$. The axioms guarantee that $\mathbf{M}_\Phi$ is a weak model in which every $LPP_2^{A,\omega_1,\mathrm{Fin}}$-theorem is true, and that $\mathbf{M}_\Phi \models \varphi$ iff $\varphi \in \mathcal{T}$. □

Note that, although in a weak model (since Axiom 9 holds) for every For_C-formula α the following condition is fulfilled:

$$\text{if } \mathbf{M} \models P_{>0}\alpha \text{ then } \mathbf{M} \models P_{>c}\alpha. \tag{6}$$

it may be the case that there is no single $c > 0$ such that the condition (6) holds for all formulas. Thus, we will now construct the corresponding strong model, i.e., a weak model $\mathbf{M}$ which satisfies that there is a $c > 0$ such that for every For_C-formula α the condition (6) holds. By Theorem 3.2.10 from [9] (see Footnote 4), measures from a strong model have finite ranges, and the model belongs to the $LPP_{2,\mathrm{Meas}}^{A,\omega_1,\mathrm{Fin}}$-class.

Theorem 26. *An $LPP_2^{A,\omega_1,\mathrm{Fin}}$-formula Φ is consistent iff it is satisfiable in a strong model in which every $LPP_2^{A,\omega_1,\mathrm{Fin}}$-theorem is true.*

Proof. Again, the simpler direction follows from the soundness of the axiomatic system. To prove another part of the statement we consider a language L_A containing:

- the following three kinds of variables:
  - variables for sets (X, Y, Z,...),
  - variables for elements (x, y, z,...),
  - variables for reals from $[0,1]$ (r, s,...), and
  - variables for positive reals greater than 1 (u,v,...)
- the predicates: $\leqslant$ for reals, $V(u,u)$, $E(x,X)$ and $\mu(X,r)$,
- a set constant symbol W_α for every $LPP_2^{A,\omega_1,\mathrm{Fin}}$-$\mathrm{For}_C$-formula α,
- a constant symbol r' for every real number $r \in [0,1] \cap A$, and
- two function symbols for additions and multiplications for reals.

The intended meaning of $E(x,X)$ is $x \in X$, $V(u,u)$ means that a formula Φ with the Gödel-number u (denoted $gb(\Phi) = u$) holds in the model, while $\mu(X,r)$ can be understood as "r is the measure of X". We use $\mu(X) \geqslant r$ to denote $(\exists s)(s \geqslant r \wedge \mu(X,s))$, and $V(\Phi)$ to denote $V(gb(\Phi), gb(\Phi))$.

We define a theory T of $L_{\omega_1\omega} \cap A$ which contains the following formulas:

(1) $(\forall X)(\forall Y)((\forall x)(E(x,X) \leftrightarrow E(x,Y)) \leftrightarrow X = Y)$
(2) $(\forall x)(E(x, W_{\alpha\wedge\beta}) \leftrightarrow (E(x,W_\alpha) \wedge E(x,W_\beta)))$ for every $\alpha \wedge \beta \in \mathrm{For}_C$
(3) $(\forall x)(E(x,W_{\neg\alpha}) \leftrightarrow \neg E(x,W_\alpha))$, for every $\alpha \in \mathrm{For}_C$
(4) $(\forall x)(E(x,W_{p\vee\neg p})$
(5) $V(\alpha) \leftrightarrow W_\alpha = W_{p\vee\neg p}$, for every $\alpha \in \mathrm{For}_C$
(6) $V(P_{\geqslant s}\alpha) \leftrightarrow \mu(W_\alpha) > s$, for every $\alpha \in \mathrm{For}_C$
(7) $V(\bigwedge_{B\in G} B) \leftrightarrow \bigwedge_{B\in G} V(B)$, for every set of probability formulas $G \in A$
(8) $V(\neg B) \leftrightarrow \neg V(B)$, for every $LPP_2^{A,\omega_1,\mathrm{Fin}}$-$\mathrm{For}_P$-formula B
(9) $(\forall X)(\exists_1 r)\mu(X,r)$
(10) $(\forall X)(\forall Y)((\mu(X,r) \wedge \mu(Y,s) \wedge \neg(\exists y)(E(y,X) \wedge E(y,Y))) \rightarrow$
$\rightarrow (\exists Z)((\forall y)((E(y,X) \vee E(y,Y)) \leftrightarrow E(y,Z)) \wedge \mu(Z, r+s)))$
(11) $(\forall X)((\forall y)E(y,X) \rightarrow \mu(X,1))$
(12) $(\exists r > 0)(\forall X)(\mu(X) > 0 \rightarrow \mu(X) > r)$
(13) Axioms for Archimedian fields for real numbers
(14) $(\forall x)E(x, W_\Psi)$ where Ψ is an axiom of $LPP_2^{A,\omega_1,\mathrm{Fin}}$
(15) $(\exists x)E(x, W_\Phi)$ where Φ is the formula from the formulation of the statement.

Let a standard model for L_A be $\langle W, H, F, V, E, \mu, +, *, \leqslant, W_\alpha, r\rangle_{\alpha\in\mathrm{For}_C, r\in F}$, where $H \subset 2^W$, $F = F' \cap [0,1]$, $F' \subset R$ a field, $V \subset R \times R$, $E \subset W \times H$, $\mu : H \to F$, $+, * : F^2 \to F$, $\leqslant \subset F^2$, and $W_\alpha \in H$.

Let $\mathbf{M} = \langle W, H, \mu, v\rangle$ be a weak model for $LPP_2^{A,\omega_1,\mathrm{Fin}}$. If we define $W_\alpha = \bigcup_{w\in W}[\alpha]_w$, and $H = \{W_\alpha : \alpha \in \mathrm{For}_C\}$, it easy to show that M can be transformed to a standard model. On the other hand, if Ψ is a consistent $LPP_2^{A,\omega_1,\mathrm{Fin}}$-formula, then there is a weak model in which it is satisfied, and consequently there is a standard model in which $V(\Psi)$ holds.

Let $T_0 \subset T$, $T_0 \in A$. Since Axiom 9 holds in the weak model M it follows that every T_0 has a model. Hence, by the Barwise compactness theorem, T has a model $\mathbf{M}' = \langle W, H, F, V, E, \mu, +, *, \leqslant, W_\alpha, r\rangle_{\alpha\in\mathrm{For}_C, r\in F}$. We define a strong model $\mathbf{M}'' = \langle W, H, \mu, v\rangle$ such that the following holds:

- for every $w \in W$, $v(w,p)$ = true iff $w \in W_p$ for every primitive proposition p,
- $H = \{W_\alpha : \alpha \in \mathrm{For}_C\}$,
- $\mu(X) = r$ iff $\mu(X,r)$ holds in $\mathbf{M}'$.

Since (15) holds in $\mathbf{M}'$, $\mathbf{M}'' \models \Phi$. □

Completeness also holds for Σ_1 definable theories, but we it is possible to show that it cannot be generalized to arbitrary theories.

4.3. Logic LPP_2^S. Another generalization of the logic $LPP_2^{A,\omega_1,\mathrm{Fin}}$ contains an infinitary rule which enables us to syntactically define the range S of the probability function which appears in the interpretation:

- From $A \rightarrow P_{\neq s}\alpha$, for every $s \in S$, infer $A \rightarrow \bot$.

However, we will skip all technical details here and discuss another logic which extends LPP_2^S in Subsection 9.1.

5. Logic LPP_1

In this section we will present the logic LPP_1 which extends LPP_2 so that iterations of the probabilistic operators are allowed. For example, $\alpha \wedge P_{\geqslant s}P_{\geqslant r}\beta$ is a formula of LPP_1. In that way we can express statements about higher order probabilities and mix classical and probabilistic formulas. More formally, the set For_{LPP_1} of formulas is the smallest set containing primitive propositions, and closed under formation rules: if α and β are formulas, then $P_{\geqslant s}\alpha$, $\neg\alpha$ and $\alpha \wedge \beta$ are formulas. The formulas from the set For_{LPP_1} will be denoted by α, β,...

The corresponding semantics can be given as follows:

Definition 27. An *LPP_1-model* is a structure $\mathbf{M} = \langle W, \mathrm{Prob}, v\rangle$ where:

- W is a nonempty set of objects called worlds,
- Prob is a probability assignment which assigns to every $w \in W$ a probability space, such that $\mathrm{Prob}(w) = \langle W(w), H(w), \mu(w)\rangle$, where:
  - $W(w)$ is a non empty subset of W,
  - $H(w)$ is an algebra of subsets of $W(w)$ and
  - $\mu(w) : H(w) \to [0,1]$ is a finitely additive probability measure.
- v assigns to every $w \in W$ a two-valued evaluation of the primitive propositions, i.e., for every $w \in W$, $v(w) : \phi \to \{\mathrm{true}, \mathrm{false}\}$.

Note that, in contrast to Definition 1, there are as many probability spaces (in a model $\mathbf{M} = \langle W, \mathrm{Prob}, v\rangle$) as the worlds (in the set W), i.e., for every world w there is a particular $\langle W(w), \mathrm{Prob}(w), \mu(w)\rangle$. As a consequence, the satisfiability relation is now defined between worlds and formulas:

Definition 28. *The satisfiability relation* $\models$ fulfills the following conditions for every LPP_1-model $\mathbf{M} = \langle W, \mathrm{Prob}, v\rangle$ and every world $w \in W$:

- if $p \in \phi$ is a primitive proposition, $\mathbf{M}, w \models \alpha$ iff $v(w)(p) = \mathrm{true}$,
- $\mathbf{M}, w \models \neg\alpha$ iff $\mathbf{M}, w \not\models \alpha$,
- $\mathbf{M}, w \models \alpha \wedge \beta$ iff $\mathbf{M}, w \models \alpha$ and $\mathbf{M}, w \models \beta$, and
- $\mathbf{M}, w \models P_{\geqslant s}\alpha$ iff $\mu(w)([\alpha]_{\mathbf{M},w}) \geqslant s$,

where $[\alpha]_{\mathbf{M},w}$ denotes the set $\{u \in W(w) : \mathbf{M}, u \models \alpha\}$. We will omit $\mathbf{M}$ from $\mathbf{M}, w \models \alpha$ and write $w \models \alpha$ if $\mathbf{M}$ is clear from the context. Similarly, we will write $[\alpha]_w$ instead of $[\alpha]_{\mathbf{M},w}$.

Similarly as above, we consider measurable models only. An LPP_1-model $\mathbf{M} = \langle W, \mathrm{Prob}, v\rangle$ is *measurable* if for every $w \in W$ the set $H(w) = \{[\alpha]_w : \alpha \in \mathrm{For}_{LPP_1}\}$. $LPP_{1,\mathrm{Meas}}$ denotes the class of all measurable LPP_1-models.

Definition 29. A formula $\alpha \in \mathrm{For}_{LPP_1}$ is *satisfiable* if there is a world w in an $LPP_{1,\mathrm{Meas}}$-model $\mathbf{M}$ such that $w \models \alpha$; α is *valid* if it is satisfied in each world in each $LPP_{1,\mathrm{Meas}}$-model. A set T of formulas is satisfiable if there is a world w in an $LPP_{1,\mathrm{Meas}}$-model $\mathbf{M}$ such that $w \models \alpha$ for every $\alpha \in T$.

5.0.1. Axiomatization, completeness, decidability. It is interesting that a sound and complete axiomatization with respect to the mention class $LPP_{1,\mathrm{Meas}}$ can be given by the axiomatic system Ax_{LPP_2} from Section 3. Of course, instances of

axiom schemata must obey the syntactical rules that hold in this section. However, the notions of deducibility and consistency introduced in the definitions 6 and 7 must be changed.

Definition 30. A formula α is *deducible from a set* T of formulas ($T \vdash \alpha$) if there is an at most denumerable sequence of formulas $\alpha_0, \alpha_1, \ldots, \alpha$, such that every α_i is an axiom or a formula from the set T, or it is derived from the preceding formulas by an inference rule, with the exception that Rule 2 can be applied to the theorems only. If $\emptyset \vdash \alpha$, we say that α is a theorem ($\vdash \alpha$).

Definition 31. A set T of formulas is *inconsistent* if $T \vdash \alpha$, for every formula α, otherwise it is *consistent.* Equivalently, T is inconsistent iff $T \vdash \bot$. A set T of formulas is *maximal* if for every formula α either $\alpha \in T$ or $\neg\alpha \in T$.

Now, the restriction from Definition 30 that Rule 2 can be applied to the theorems only guarantees that Deduction theorem for LPP_1 holds. Also, the counterparts of the statements 10–13 can be proved in the same way as above. The canonical model $\mathbf{M} = \langle W, \mathrm{Prob}, v\rangle$ can be defined such that:

- $W = \{w : w \text{ is a maximal consistent set of formulas}\}$,
- for every primitive proposition $p \in \phi$, and every $w \in W$, $v(w)(p) = \top$ iff $p \in w$, and
- for every $w \in W$, $\mathrm{Prob}(w) = \langle W(w), H(w), \mu(w)\rangle$ is defined as follows:
  - $W(w) = W$,
  - $H(w) = \{\{u : u \in W, \alpha \in u\} : \alpha \in \mathrm{For}_{LPP_1}\}$, and
  - $\mu(w)(\{u : u \in W, \alpha \in u\}) = \sup\{s : P_{\geqslant s}\alpha \in w\}$.

Similarly as above, we can prove that for every formula α and every world w, $\alpha \in w$ iff $w \vDash \alpha$. It follows that:

- for all α and w, $[\alpha]_w = \{u : u \in W, \alpha \in u\}$,
- for all w, $\mathrm{Prob}(w) = \langle W(w), H(w), \mu(w),\rangle$ is a probability space,
- the canonical model $\mathbf{M}$ is an $LPP_{1,\mathrm{Meas}}$-model, and
- every consistent set of formulas is satisfiable in some world from $\mathbf{M}$,

i.e., we obtain the extended completeness theorem for the class $LPP_{1,\mathrm{Meas}}$. Furthermore, reasoning as in the sections 3.4 and 4, we can prove completeness for the following classes of models $LPP_{1,\mathrm{Meas,All}}$, $LPP_{1,\mathrm{Meas},\sigma}$ and $LPP_{1,\mathrm{Meas,Neat}}$, and logics $LPP_1^{\mathrm{Fr}(n)}$, $LPP_1^{A,\omega_1,\mathrm{Fin}}$ and LPP_1^S.

Decidability and complexity of the satisfiability problem for the class $LPP_{1,\mathrm{Meas}}$ are analyzed in the sequel of this section.

Theorem 32. *If a formula α is satisfiable, then it is satisfiable in an $LPP_{1,\mathrm{Meas}}$-model with a finite number of worlds. The number of worlds in that model is at most 2^k, where k denotes the number of subformulas of α.*

Proof. Suppose that α holds in a world of an $LPP_{1,\mathrm{Meas}}$-model $\mathbf{M} = \langle W, \mathrm{Prob}, v\rangle$. Let $\mathrm{Subf}(\alpha)$ denote the set of all subformulas of α, and $k = |\mathrm{Subf}(\alpha)|$. Let $\approx$ denote the equivalence relation over W^2, such that $w \approx u$ iff for every $\beta \in \mathrm{Subf}(\alpha)$, $w \vDash \beta$ iff $u \vDash \beta$. The quotient set $W_{/\approx}$ is finite. From every class C_i we choose an element and denote it w_i. We consider the model $\mathbf{M}^* = \langle W^*, \mathrm{Prob}^*, v^*\rangle$, where:

- $W^* = \{w_i\}$,
- Prob* is defined as follows:
  - $W^*(w_i) = \{w_j \in W^* : (\exists u \in C_{w_j}) u \in W(w_i)\}$
  - $H^*(w_i)$ is the powerset of $W^*(w_i)$,
  - $\mu^*(w_i)(w_j) = \mu(w_i)(C_{w_j})$, and for any $D \subset H^*(w_i)$, $\mu^*(w_i)(D) = \sum_{w_j \in D} \mu^*(w_i)(w_j)$,
- $v^*(w_i)(p) = v(w_i)(p)$, for every primitive proposition $p \in \phi$.

It is easy to show that $\mathbf{M}^*$ is an $LPP_{1,\text{Meas}}$-model. For example, for every w_i, $\mu^*(w_i)$ is a finitely additive probability measure, since

$$\mu^*(w_i)(W^*(w_i)) = \sum_{w_j \in W^*(w_i)} \mu^*(w_i)(w_j) = \sum_{C_{w_j} \in W_{/\approx}} \mu^*(w_i)(C_{w_j}) = 1.$$

We can now show that for every $\beta \in \text{Subf}(\alpha)$, β is satisfiable in $\mathbf{M}$ iff it is satisfiable in $\mathbf{M}^*$. If $\beta \in \phi$, $\mathbf{M}, w \models \beta$ iff for $w_i \in C_w$, $\mathbf{M}, w_i \models \beta$ iff $\mathbf{M}^*, w_i \models \beta$. The cases related to $\wedge$ and $\neg$ can be proved as usual. Finally, let $\beta = P_{\geqslant s}\gamma$. Then, $\mathbf{M}, w \models P_{\geqslant s}\gamma$ iff for $w_i \in C_w$, $\mathbf{M}, w_i \models P_{\geqslant s}\gamma$ iff

$$s \leqslant \mu(w_i)([\gamma]_{\mathbf{M},w_i}) = \sum_{C_u : \mathbf{M}, u \models \gamma} \mu(w_i)(C_u) = \sum_{C_u : \mathbf{M}^*, u \models \gamma} \mu^*(w_i)(C_u) = \mu^*(w_i)([\gamma]_{\mathbf{M}^*,w_i})$$

iff $\mathbf{M}^*, w_i \models P_{\geqslant s}\gamma$.

Finally, it is clear that the number of different classes in $W_{/\approx}$ is at most 2^k, and the same holds for the number of worlds in $\mathbf{M}^*$. □

Theorem 33 (Decidability theorem). *The logic LPP_1 is decidable.*

Proof. As it is noted above, a formula α is $LPP_{1,\text{Meas}}$-satisfiable iff it is satisfiable in an $LPP_{1,\text{Meas}}$-model with at most 2^k worlds, where k denotes the number of subformulas of α. Observe that it does not necessary imply decidability of the satisfiability problem for the class $LPP_{1,\text{Meas}}$ because there are infinitely many such models. Nevertheless, the next procedure decides the satisfiability problem. The procedure is applied for every such $l \leqslant 2^k$.

Let $\text{Subf}(\alpha) = \{\beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_m\}$, and $k = n + m$. In every world w from $\mathbf{M}$ exactly one of the formulas of the form

$$\delta_w = \beta_1 \wedge \ldots \wedge \beta_n \wedge \neg\gamma_1 \wedge \ldots \wedge \neg\gamma_m$$

holds. For every $l \leqslant 2^k$ we will consider l formulas of the above form. The chosen formulas are not necessarily different, but at least one of the formulas must contain the examined formula α. Using probabilistic constraints (i.e., formulas of the form $P_{\geqslant s}\beta$, $\neg P_{\geqslant s}\beta$) from the formulas we shall examine whether there is an $LPP_{1,\text{Meas}}$-model $\mathbf{M}$ with l worlds such that for some world w from the model $w \models \alpha$. We do not try to determine probabilities precisely. Rather, we just check whether there are probabilities such that probabilistic constraints are satisfied in the corresponding world. To do that, for every world w_i, $i < l$, we consider a system of linear equalities and inequalities of the form (we write $\beta \in \delta_w$ to denote that β occurs positively in

the top conjunction of δ_w, i.e., if δ_w can be seen as $\bigwedge_i \delta_i$, then for some i, $\beta = \delta_i$):

$$
(7)\qquad
\begin{aligned}
&\sum_{j=1}^{l} \mu(w_i)(w_j) = 1\\
&\mu(w_i)(w_j) \geqslant 0, \quad \text{for every world } w_j\\
&\sum_{w_j:\beta\in\delta_{w_j}} \mu(w_i)(w_j) \geqslant s, \quad \text{for every } P_{\geqslant s}\beta \in \delta_{w_i}\\
&\sum_{w_j:\beta\in\delta_{w_j}} \mu(w_i)(w_j) < s, \quad \text{for every } \neg P_{\geqslant s}\beta \in \delta_{w_i}
\end{aligned}
$$

The first two rows correspond to the general constraints: the probability of the set of all worlds must be 1, while the probability of every measurable set of worlds must be nonnegative. The last two rows correspond to the probabilistic constraints, because

$$\sum_{w_j:\beta\in\delta_{w_j}} \mu(w_i)(w_j) = \mu(w_i)([\beta]_{w_i}).$$

Such a system is solvable iff there is a probability $\mu(w_i)$ satisfying all probabilistic constraints that appear in δ_{w_i}. Note that there are finitely many such systems that can be solved in a finite number of steps.

If the above test is positively solved there is an $LPP_{1,\mathrm{Meas}}$-model in which every world $w_i \models \delta_{w_i}$. Since α belongs to at least one of the formulas δ_{w_i}, we have that α is satisfiable. If the test fails, and there is another possibility of choosing l and/ot the set of l formulas δ_w, we continue with the procedure, otherwise we conclude that α is not satisfiable.

It is easy to see that the procedure terminates in a finite number of steps. Thus, the satisfiability problem for the class $LPP_{1,\mathrm{Meas}}$ is decidable. Since $\models \alpha$ iff $\neg\alpha$ is not satisfiable, the $LPP_{1,\mathrm{Meas}}$-validity problem is also decidable. □

The satisfiability problem for the class $LPP_{1,\mathrm{Meas}}$ is in PSPACE, while NP is the lower bound of the complexity. The former statement is a consequence of the PSPACE-completeness of a more expressive logic from [28], while the later statement follows from the fact that the logic LPP_2 can be seen as a sublogic of LPP_1.

6. Some extensions of the probabilistic language

In this section we will analyze some possible extensions of the considered probabilistic language. The first extension contains probabilistic operators of the form Q_F with the intended meaning "the probability belongs to the set F". The next extension allows reasoning about qualitative probabilities. Finally, we mention a logic introduces in [27] in which linear combinations of probabilities can be expressed. All extensions will be considered in the framework of the logic LPP_2, but analogue analysis can be performed for the other above presented logics.

6.1. Probability operators of the form Q_F. We will use $LPP_{2,P,Q,O}$ to denote a probability logic which depends on a recursive family O of recursive subsets of S in a manner which will be explained below, while P and Q in the index means that two kinds of probabilistic operators will be used. More precisely, the language of $LPP_{2,P,Q,O}$ extends the LPP_2-language with a list of unary probabilistic operators of the form Q_F, where $F \in O$. For example, the set $\mathrm{For}_{LPP_{2,P,Q,O}}$ of formulas contains $Q_F\alpha \to \neg P_{\geqslant s}\beta$. Note that every particular choice of the family O of sets produces a different probability language, a different set of probability formulas and a distinct $LPP_{2,P,Q,O}$-logic.

To give semantics to formulas, we use the class $LPP_{2,\mathrm{Meas}}$ of measurable LPP_2-models, and the corresponding satisfiability relation (from Definition 2) with additional requirement that:

- $\mathbf{M} \vDash Q_F\alpha$ iff $\mu([\alpha]) \in F$, for every $F \in O$

which covers the case of the new operators. Note that $\neg Q_F\alpha$ is not equivalent to $Q_{[0,1]\setminus F}\alpha$ because $[0,1] \setminus F \notin O$, and the later is not a well formed formula.

It is obvious, using the semantics of $P_{\geqslant s}$ and Q_F-operators, that for a set $F = \{f_1, f_2, \ldots\} \in O$, $Q_F\alpha \leftrightarrow \bigvee_{f_i \in F} P_{=f_i}\alpha$. But, if the set F is not finite, the right side of this equality is an infinitary disjunction which does not belong to the set $\mathrm{For}_{LPP_{2,P,Q,O}}$ of formulas. Similarly for the formula $P_{\geqslant s}\alpha \leftrightarrow Q_{[s,1]}\alpha$, where s is a rational number from $[0,1)$, the formula $Q_{[s,1]}\alpha \notin \mathrm{For}_{LPP_{2,P,Q,O}}$. More formally:

Definition 34. Let $\Phi, \Psi \in \mathrm{For}_{LPP_{2,P,Q,O}}$. The set $\mathrm{Mod}(\Phi) = \{\mathbf{M} \in LPP_{2,\mathrm{Meas}} : M \vDash \Phi\}$ consists of all $LPP_{2,\mathrm{Meas}}$-models of the Φ. Φ is *definable from* Ψ if $\mathrm{Mod}(\Phi) = \mathrm{Mod}(\Psi)$.

The above discussion suggests that in a general case neither the $P_{\geqslant .}$-operators are definable from the $Q_.$-operators (i.e., some formulas on the language $\{\neg, \wedge, P_{\geqslant .}\}$ are not definable from the formulas on the language $\{\neg, \wedge, Q_.\}$), nor are the $Q_.$-operators definable from the $P_{\geqslant .}$-operators. The next theorems formalize these conclusions.

Theorem 35. *Let O be a recursive family of recursive rational subsets of $[0,1]$, $F \in O$ an infinite set, and $LPP_{2,P,Q,O}$ the corresponding logic. For an arbitrary primitive proposition $p \in \phi$, there is no probabilistic formula A on the sublanguage $\{\neg, \wedge, P_{\geqslant .}\}$ such that $Q_F p$ is definable from α.*

Proof. Suppose that there is a formula A on the language $\{\neg, \wedge, P_{\geqslant .}\}$ such that $\mathrm{Mod}(Q_F p) = \{\langle W, H, \mu, v\rangle : \mu([p]) \in F\} = \mathrm{Mod}(A)$. Recall that A is satisfiable iff at least a system from the set of all linear systems that correspond to $DNF(A)$ is satisfiable. Let a_t's be the atoms of A and y_t's be the corresponding measures. The solutions of any of those systems must satisfy $\sum_{a_t \in DNF(p)} y_t \in F$. But, the solutions of the systems are of the following form: $y_t \in (r,s)$, $y_t \in [r,s)$, $y_t \in (r,s]$, and $y_t \in [r,s]$. Such sets of solutions cannot produce the infinite, but denumerable set F as it is required. Hence, $Q_F p$ is not definable over A. □

Theorem 36. *Let O be a recursive family of recursive rational subsets of $[0,1]$, $LPP_{2,P,Q,O}$ the corresponding logic, and $s \in S \setminus \{1\}$. For an arbitrary primitive*

proposition $p \in \phi$, there is no probabilistic formula A on the sublanguage $\{\neg, \wedge, Q_.\}$ such that $P_{\geqslant s}p$ is definable from A.

Proof. Suppose that there is a formula A on the language $\{\neg, \wedge, Q_.\}$ such that $\mathrm{Mod}(P_{\geqslant s}p) = \mathrm{Mod}(A)$. The models of A are exactly those that satisfy $\mu[p] \geqslant s$. But, similarly as above, the set of values for $\mu[p]$ produced by $\mathrm{Mod}(A)$ can be either denumerable, or its complement is denumerable. Hence, $P_{\geqslant s}p$ cannot be definable over A. □

Example 37. Formulas with the new probabilistic operators are suitable for reasoning about discrete sample spaces. For example, consider an experiment which consists of tossing a fair coin an arbitrary, but finite number of times. Then, $Q_F\alpha$ holds in this model, where α means that only heads (i.e., no tails) is observed in the experiment, and F denotes the set $\{\frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \ldots\}$. Since Q_F is not definable over the probability language $\{\neg, \wedge, P_{\geqslant}.\}$, this sentence cannot be described in the probability logics used so far.

6.1.1. Expressiveness of $LPP_{2,P,Q,O}$-logics. As it is noted above, every particular choice of the family of sets O produces a different $LPP_{2,P,Q,O}$-logic. In this section we describe a relation of "being more expressive" between these logics. The fact that the corresponding hierarchy has no upper bounds, is a good reason for introducing many probabilistic logics with new type of probability operators, since no single probabilistic logic covers all contexts. The choice of particular logic depends on the particular situation that we wish to formalize.

Definition 38. Let F be a rational subset of $[0,1]$. The *quasi complement* of F is a set $1 - F = \{1 - f : f \in F\}$.

Example 39. If $F = \{\frac{1}{2^i} : i = 1, 2, \ldots\}$, then, following Definition 38, $1 - F = \{\frac{2^i - 1}{2^i} : i = 1, 2, \ldots\}$.

It is easy to see that the quasi complement has the following properties:

- $1 - (F \cap G) = (1 - F) \cap (1 - G)$,
- $1 - (F \cup G) = (1 - F) \cup (1 - G)$,
- $1 - (F \setminus G) = (1 - F) \setminus (1 - G)$ and
- $1 - (1 - F) = F$.

These properties, as well as the properties of $\cup, \cap$ and $\setminus$, guarantee that an arbitrary expression on the language $\{\cup, \cap, \setminus, 1-\}$ can be rewritten in a normal form as a finite union of finite intersections of differences between sets and quasi complements of sets.

Definition 40. Let O_1 and O_2 be recursive families of recursive rational subsets of $[0,1]$. Let $F_1 \in O_1$. F_1 is *representable* in O_2 if it is equal to a finite union of finite intersections of sets, differences between sets and quasi complements of sets from O_2 and sets $[r,s]$, $[r,s)$, $(r,s]$ and (r,s), where r and s are rational numbers from $[0,1]$. The family of sets O_1 is *representable* in O_2 if each set $F_1 \in O_1$ is representable in O_2.

Example 41. Let us consider a positive integer $k > 0$, the sets

$F_1 = \{\frac{1}{2^i} : i = k, k+1, \ldots\} \cup \{\frac{3^i-1}{3^i} : i = k, k+1, \ldots\}$,
$F_2 = \{\frac{1}{2^i} : i = 1, 2, \ldots\}$,
$F_3 = \{\frac{1}{3^i} : i = 1, 2, \ldots\}$,

and the family $O_2 = \{F_2, F_3\}$. By Definition 40, F_1 is representable in O_2 because

$$F_1 = \big(F_2 \cap [0, 1/2^k]\big) \cup \big((1 - F_3) \cap \big[(3^k - 1)/3^k, 1\big]\big).$$

On the other hand, the set $F_4 = \{1/2^{2i} : i = 1, 2, \ldots\}$ is not representable in O_2.

Theorem 42. *Let O_1 and O_2 be recursive families of recursive rational subsets of $[0,1]$. Let $F_1 \in O_1$ be representable in O_2. For an arbitrary formula $\alpha \in \mathrm{For}_C$, there is a formula $\phi \in \mathrm{For}_{LPP_{2,P,Q,O_2}}$ such that $\mathrm{Mod}(Q_{F_1}\alpha) = \mathrm{Mod}(\phi)$, i.e., $Q_{F_1}\alpha$ and ϕ have the same models.*

Proof. Suppose that $F_1 = \bigcup_{i=1}^{m} \bigcap_{j=1}^{k_i} F_{i,j}$ (for the meaning of $F_{i,j}$ see below). It is easy to see that for an arbitrary formula $\alpha \in \mathrm{For}_C$ we have (in the $LPP_{2,P,Q,O_1 \cup O_2}$):

$$\vdash Q_{F_1}\alpha \leftrightarrow \bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} R_{F_{i,j}}\alpha$$

where

$$R_{F_{i,j}}\alpha = \begin{cases} P_{\geqslant s}\alpha \wedge P_{\leqslant r}\alpha, & \text{if } F_{i,j} = [s,r] \\ P_{\geqslant s}\alpha \wedge P_{<r}\alpha, & \text{if } F_{i,j} = [s,r) \\ P_{>s}\alpha \wedge P_{\leqslant r}\alpha, & \text{if } F_{i,j} = (s,r] \\ P_{>s}\alpha \wedge P_{<r}\alpha, & \text{if } F_{i,j} = (s,r) \\ Q_{F_{i,j}}\alpha, & \text{if } F_{i,j} \in O_2 \\ Q_{F'_{i,j}}\neg\alpha, & \text{if } F_{i,j} = 1 - F'_{i,j}, F'_{i,j} \in O_2 \\ Q_{F'_{i,j}}\alpha \wedge \neg Q_{F''_{i,j}}\alpha, & \text{if } F_{i,j} = F'_{i,j} \smallsetminus F''_{i,j}, F'_{i,j}, F''_{i,j} \in O_2 \\ Q_{F'_{i,j}}\alpha \wedge \neg(P_{\geqslant s}\alpha \wedge P_{\leqslant r}\alpha), & \text{if } F_{i,j} = F'_{i,j} \smallsetminus [s,r], F'_{i,j} \in O_2 \\ Q_{F'_{i,j}}\alpha \wedge \neg(P_{\geqslant s}\alpha \wedge P_{<r}\alpha), & \text{if } F_{i,j} = F'_{i,j} \smallsetminus [s,r), F'_{i,j} \in O_2 \\ Q_{F'_{i,j}}\alpha \wedge \neg(P_{>s}\alpha \wedge P_{\leqslant r}\alpha), & \text{if } F_{i,j} = F'_{i,j} \smallsetminus (s,r], F'_{i,j} \in O_2 \\ Q_{F'_{i,j}}\alpha \wedge \neg(P_{>s}\alpha \wedge P_{<r}\alpha), & \text{if } F_{i,j} = F'_{i,j} \smallsetminus (s,r), F'_{i,j} \in O_2 \end{cases}$$

Formula $\bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} R_{F_{i,j}}\alpha$ belongs to LPP_{2,P,Q,O_2}, and

$$\mathrm{Mod}(Q_{F_1}\alpha) = \mathrm{Mod}\Big(\bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} R_{F_{i,j}}\alpha\Big). \qquad \square$$

Definition 43. Let O_1 and O_2 be recursive families of recursive rational subsets of $[0,1]$, and L_1 and L_2 be the corresponding $LPP_{2,P,Q,O}$-logics. The logic L_2 is *more expressive* than the logic L_1 ($L_1 \leqslant L_2$) if for every formula $\phi \in \mathrm{For}_{LPP_{2,P,Q,O_1}}$ there is a formula $\psi \in \mathrm{For}_{LPP_{2,P,Q,O_2}}$ such that $\mathrm{Mod}(\phi) = \mathrm{Mod}(\psi)$.

Theorem 44. *Let O_1 and O_2 be recursive families of recursive rational subsets of $[0,1]$, and L_1 and L_2 be the corresponding $LPP_{2,P,Q,O}$-logics. The family O_1 is representable in the family O_2 iff $L_1 \leqslant L_2$.*

Proof. ($\Rightarrow$) Let $A \in \mathrm{For}_{LPP_{2,P,Q,O_1}}$. A is equivalent to

$$DNF(A) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} X^{i,j}(p_1, \ldots, p_n),$$

where every $X^{i,j}$ can be from the set $\{P_{\geqslant s_{i,j}}, P_{<s_{i,j}}, Q_{F_{i,j}}, \neg Q_{F_{i,j}}\}$. Furthermore, $\mathrm{Mod}(A) = \bigcup_{i=1}^{m} \bigcap_{j=1}^{k_i} \mathrm{Mod}(X^{i,j}(p_1, \ldots, p_n))$. Let us consider the case where $X^{i,j} = Q_{F_{i,j}}$. By the hypothesis the set $F_{i,j}$ is representable in O_2. Using the theorem 42 there is a formula $B_{i,j} \in \mathrm{For}_{LPP_{2,P,Q,O_2}}$ such that $\mathrm{Mod}(X^{i,j}(p_1, \ldots, p_n)) = \mathrm{Mod}(B_{i,j})$, and similarly for $X^{i,j} = \neg Q_{F_{i,j}}$, whilst the cases where $X^{i,j} = P_{\geqslant s_{i,j}}$, or $X^{i,j} = P_{<s_{i,j}}$ are both expressible in the logics L_1 and L_2. Hence, there is a formula $B \in \mathrm{For}_{LPP_{2,P,Q,O_2}}$ such that $\mathrm{Mod}(A) = \mathrm{Mod}(B)$.

($\Leftarrow$) To avoid repetition of similar arguments, in the sequel of this proof we will use $Q_{\{f\}}$ instead of $P_{=f}$. By the hypothesis, for every primitive proposition $p \in \phi$, and every $F_1 \in O_1$ there is a formula $\Phi \in \mathrm{For}_{LPP_{2,P,Q,O_2}}$ so that $\mathrm{Mod}(Q_{F_1}p) = \mathrm{Mod}(\Phi)$. If F_1 is an empty set, or a finite set, the formula $Q_{F_1}p \leftrightarrow \bigvee_{f \in F_1} Q_{\{f\}}p$ is a theorem (an empty disjunction is a contradiction), and $F_1 = \bigcup_{f \in F_1}[f, f]$ is representable in O_2.

We can show that, if $F_1 = \{f_1, f_2, \ldots\}$ is an infinite set of rational numbers from $[0, 1]$, the formula Φ cannot be propositional. Suppose that $B \in \mathrm{For}_C$. Then, the following cases must be distinguished:

- if $\Phi \rightarrow \neg p$ and $\Phi \rightarrow p$ are not theorems, consider the model $\mathbf{M} = \langle \{w_1, w_2\}, 2^{\{w_1, w_2\}}, \mu, v\rangle$ such that $\mu(\{w_1\}) = q$, $\mu(\{w_2\}) = 1 - q$, where q is an irrational number, $v(w_1)(p) = v(w_1)(\beta) = \top$, and $v(w_2)(\neg p) = v(w_2)(\beta) = \top$; since $\mu([p]) = q$, it follows that $\mathbf{M} \in \mathrm{Mod}(\Phi)$ and $\mathbf{M} \notin \mathrm{Mod}(Q_{F_1}p)$, a contradiction,
- if $\Phi \rightarrow \neg p$ is not a theorem, whilst $\Phi \rightarrow p$ is a theorem, consider an $s \in F_1 \setminus \{0\}$, and the model $\mathbf{M} = \langle \{w_1, w_2\}, 2^{\{w_1, w_2\}}, \mu, v\rangle$ such that $\mu(\{w_1\}) = s$, $\mu(\{w_2\}) = 1 - s$, $v(w_1)(p) = v(w_1)(\neg\Phi) = \top$, and $v(w_2)(\neg p) = v(w_2)(\neg\Phi) = \top$; since $\mu([p]) = s$, it follows that $\mathbf{M} \notin \mathrm{Mod}(\Phi)$ and $\mathbf{M} \in \mathrm{Mod}(Q_{F_1}p)$, a contradiction, and
- if $\Phi \rightarrow \neg p$ is a theorem, consider an $s \in F_1 \setminus \{0\}$, and the model $\mathbf{M} = \langle \{w_1, w_2\}, 2^{\{w_1, w_2\}}, \mu, v\rangle$ such that $\mu(\{w_1\}) = s$, $\mu(\{w_2\}) = 1 - s$, $v(w_1)(p) = v(w_1)(\neg\Phi) = \top$, and $v(w_2)(\neg p) = v(w_2)(\Phi) = \top$; since $\mu([p]) = s$, it follows that $\mathbf{M} \notin \mathrm{Mod}(\Phi)$ and $\mathbf{M} \in \mathrm{Mod}(Q_{F_1}p)$, a contradiction.

Hence, $\Phi \in \mathrm{For}_{LPP_{2,P,Q,O_2}} \setminus \mathrm{For}_C$. Let the disjunctive normal form of Φ be $DNF(\Phi) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} X^{i,j}(p_1, \ldots, p_n)$ such that all $\bigwedge_{j=1}^{k_i} X^{i,j}(p_1, \ldots, p_n)$ are consistent. Since $\Phi \leftrightarrow (\Phi \wedge P_{\geqslant 0}p)$ is a valid formula, we can suppose that the primitive proposition p appears in Φ. Let p be p_1, and $a_1, \ldots, a_{2^n}$ be the list of all atoms of Φ ordered such that $a_i = p \wedge \ldots$, for $i = 1, \ldots, 2^{n-1}$, and $a_i = \neg p \wedge \ldots$, for $i = 2^{n-1} + 1, \ldots, 2^n$. Let $y_1, \ldots, y_{2^n}$ denote the atoms' measures. All the $LPP_{2,\mathrm{Meas}}$-models can be seen as points $\langle s_1, s_2, \ldots, s_{2^n}\rangle$ in the 2^n-dimensional space E, such that ith coordinate corresponds to y_i, for all $i = 1, \ldots, 2^n$. Since

$\mathrm{Mod}(Q_{F_1}p_1) = \mathrm{Mod}(\Phi)$, we have for every $y \in [0,1]$:

$\langle y, 0, \ldots, 0, 1-y, 0, \ldots, 0\rangle \in \mathrm{Mod}(Q_{F_1}p_1)$ iff $\langle y, 0, \ldots, 0, 1-y, 0, \ldots, 0\rangle \in \mathrm{Mod}(\Phi)$,

where the entry $1-y$ is in the $2^{n-1}+1$'st position. Thus,

$$y \in F_1 \text{ iff } y \in \bigcup_{i=1}^{m} \bigcap_{j=1}^{k_i} \{y \mid \langle y, 0, \ldots, 0, 1-y, 0, \ldots, 0\rangle \in M(X^{i,j}(p_1, \ldots, p_n))\},$$

and by straightforward inspection of equalities, inequalities and constraints that can appear in the systems corresponding to the disjuncts from $DNF(\Phi)$, the set

$$F_1 = \{y \mid \langle y, 0, \ldots, 0, 1-y, 0, \ldots, 0\rangle \in \mathrm{Mod}(X^{i,j}(p_1, \ldots, p_n))\},$$

is representable in the family O_2.

Since every $F_1 \in O_1$ is representable in the family O_2, the family O_1 is representable in O_2. □

Theorem 44 correlates the relations of "being more expressive" between the $LPP_{2,P,Q,O}$-logics, and "being representable in" between the corresponding families of sets. In the sequel we investigate the later relation having in mind the former one. The relation "being more expressive" describes the hierarchy of expressiveness of the $LPP_{2,P,Q,O}$-logics.

Definition 45. Let O be a recursive family of rational subsets of $[0,1]$. The family of all rational subsets of $[0,1]$ that are representable in O is denoted by $\overline{O}$.

It is easy to see, using Definition 40, that a family $\overline{O}$ is closed under finite union, finite intersection, quasi complement and difference of sets. Each family $\overline{O}$ contains all finite rational subsets of $[0,1]$. Since the operations of union and intersection satisfy the commutative, associative, absorption and distributive laws, every family $\overline{O}$ with the standard set operations is a distributive lattice. Note that, if complement of a set F is understood as $[0,1] \setminus F$, $\overline{O}$ is not a Boolean algebra since $[0,1] \setminus F \notin S$. On the other hand, if $S \in \overline{O}$, and complement is understood as $S \setminus F$, $\overline{O}$ becomes a Boolean algebra.

Definition 46. Let O_1 and O_2 be recursive families of rational subsets of $[0,1]$. The binary relation $\sim$ is defined such that $O_1 \sim O_2$ iff $\overline{O_1} = \overline{O_2}$.

The relation $\sim$ is an equivalence relation on the set $\mathcal{O}$ of all recursive families of rational subsets of $[0,1]$. We use $\mathcal{O}_{/\sim}$ to denote the corresponding quotient set. Each equivalence class $o \in \mathcal{O}_{/\sim}$ contains a unique maximal family O_o such that $O_o = \overline{O}$. For such an equivalence class o and the corresponding family O_o we say that O_o represents o. Let the set $\{O_o : O_o \text{ represents } o \in \mathcal{O}_{/\sim}\}$ be denoted by $\mathcal{O}^*$. Clearly, $\mathcal{O}^*$ is countable.

Definition 47. Let O_1 and O_2 be different families from $\mathcal{O}^*$. Then $O_1 < O_2$ iff O_1 is representable in O_2.

Theorem 48. *Let O_1 and O_2 be different families from $\mathcal{O}^*$. Then $O_1 < O_2$ iff $\overline{O_1} \subset \overline{O_2}$.*

Proof. The statement is an immediate consequence of the corresponding definitions. □

Theorem 49. *The structure* $(\mathcal{O}^*, <)$ *is a lattice.*

Proof. Since $\subset$ is a partial ordering, by Theorem 48, the relation $<$ defined on $\mathcal{O}^*$ is a partial ordering, too. Moreover, any two elements of $(\mathcal{O}^*, <)$ posses both the least upper bound, and the greatest lower bound. Suppose $O_1, O_2 \in \mathcal{O}^*$. Let $O_3 = \overline{O_1 \cup O_2}$. Obviously, $O_1 < O_3$, and $O_2 < O_3$. Suppose that there is an $O_4 \in \mathcal{O}^*$, such that $O_1 < O_4$ and $O_2 < O_4$. But then, by Theorem 48, $O_1 \subset O_4$, $O_2 \subset O_4$, and $O_1 \cup O_2 \subset O_4$. It follows that $O_3 < O_4$. Hence, $\overline{O_1 \cup O_2}$ is the least upper bound of $\{O_1, O_2\}$. Similarly, the greatest lower bound of $\{O_1, O_2\}$ is $\overline{O_1 \cap O_2}$. Since $(\mathcal{O}^*, <)$ is a partially ordered set such that any two elements posses both a least upper bound, and a greatest lower bound, it is a lattice. □

The meet $(\cdot)$ and join $(+)$ operations can be defined as usual:

$$O_1 \cdot O_2 = \overline{O_1 \cap O_2}, \quad \text{and} \quad O_1 + O_2 = \overline{O_1 \cup O_2}.$$

Since every set that is representable both in O_1 and in O_2, is representable in $O_1 \cap O_2$, we have $\overline{O_1 \cap O_2} = O_1 \cap O_2$, and $O_1 \cdot O_2 = O_1 \cap O_2$. On the other hand, note that the join operation and the set union do not coincide, because for some $O_1, O_2 \in \mathcal{O}^*$, it can be $O_1 \cup O_2 \neq \overline{O_1 \cup O_2}$.

Theorem 50. *The lattice* $(\mathcal{O}^*, <)$ *is not a modular.*

Proof. We can find a counter example for the modularity law: if $O_2 < O_1$, then $(O_1 \cdot (O_2 + O_3)) = (O_2 + (O_1 \cdot O_3))$. Let $Prim = \{k_1, k_2, \ldots\}$ denote the set of all prime numbers. Then, consider the sets: $F_1 = \{\frac{1}{2^i} : i = 1, 2, \ldots\}$, $F_2 = \{\frac{1}{2^{k_i}} : i = 1, 2, \ldots\}$, and $F_3 = F_1 \setminus \{\frac{1}{2^{k_{2i-1}}} : i = 1, 2, \ldots\}$, and the families $O_1, O_2, O_3 \in \mathcal{O}^*$, such that $O_1 = \overline{\{F_1, F_2\}}$, $O_2 = \overline{\{F_2\}}$, and $O_2 = \overline{\{F_3\}}$. Obviously, $O_2 \subset O_1$, and $O_2 < O_1$. Since $F_1 = F_2 \cup F_3$, F_1 is representable in $O_2 + O_3$, and also in $O_1 \cdot (O_2 + O_3)$. On the other hand, F_1 is neither representable in O_2 nor in O_3. Thus, F_1 is not representable in $O_2 + (O_1 \cdot O_3)$, and the modularity law does not hold. □

Theorem 51. $\overline{\emptyset}$ *is the smallest element of* $(\mathcal{O}^*, <)$.

Proof. $\overline{\emptyset}$ contains all the finite rational subsets of $[0, 1]$ only. Since an arbitrary $O \in \mathcal{O}^*$ contains these sets, $\overline{\emptyset} \subset O$ and $\overline{\emptyset} < O$. □

Let $F_1 = \{r_0, r_1, \ldots\}$ be a rational subset of $[0, 1]$ with only one accumulation point. Let $O_1 = \overline{\{F_1\}}$, $O_2 \in \mathcal{O}^*$, and $O_2 < O_1$. Note that a set $F_2 \in O_2$ can be either a finite set, or an infinite set such that symmetric difference of either F_1 and F_2, $(F_1 \setminus F_2) \cup (F_2 \setminus F_1)$, or $1 - F_1$ and F_2 is finite. If all the sets from O_2 are finite, then $O_2 = 0$. Suppose that there is an infinite set $F_2 \in O_2$ that is representable in O_1. F_2 differs from F_1 (or $1 - F_1$) in finitely many elements. It follows that F_1 is representable in O_2, $O_1 < O_2$, and $O_1 = O_2$. Hence, O_1 is an atom of $(\mathcal{O}^*, <)$. Suppose that a family $O \in \mathcal{O}^*$ contains a set F with finitely

many accumulation points. For every $F_1 \subset F$ with only one accumulation point, and $O_1 = \overline{\{F_1\}}$ holds $O_1 < O$. Finally, let us consider a family which contains a set with infinitely many accumulation points. Suppose that a set F_0 is dense in $(a_0, b_0) \subset [0,1]$, and $O_0 = \overline{\{F_0\}}$. We can obtain two sequences $a_0 < a_1 < a_2 < \cdots$ and $b_0 > b_1 > b_2 > \cdots$ such that $a_i < b_j$ for every i and j, a sentence of sets $F_0 \supset F_1 \supset F_2 \supset \cdots$ that are dense in $(a_1, b_1) \subset [0,1]$, $(a_2, b_2) \subset [0,1], \cdots$, respectively, and an infinite sentence of families $O_1 = \overline{\{F_1\}}$, $O_2 = \overline{\{F_2\}}, \ldots$, such that $0 < \cdots < O_2 < O_1 < O_0$. Obviously, there is no atom in this sequence.

In particular, we have the following theorems:

Theorem 52. *A necessary and sufficient condition that an $O \in \mathcal{O}^*$ be an atom is that $O = \overline{\{F\}}$, where F is a set with only one accumulation point. The lattice $(\mathcal{O}^*, <)$ is non-atomic.*

Theorem 53. *There is no greatest element in $(\mathcal{O}^*, <)$. Consequently, the lattice $\mathcal{O}^*$ is σ-incomplete.*

Proof. Since the family of all recursive subsets of S is not recursive, for each recursive family O of recursive subsets of S there is a recursive $F \subseteq S$ non-representable by O. Hence, there is no greatest element in $\mathcal{O}^*$. Furthermore, σ-incompleteness is an immediate consequence of the fact that $\mathcal{O}^*$ is a countable ordering without upper bounds. □

Thus, we can define a hierarchy of the $LPP_{2,P,Q,O}$-logics, so that a logic L_1 is less expressive that a logic L_2 ($L_1 < L_2$) iff the corresponding families O_1 and O_2 of rational subsets of $[0,1]$ satisfy a similar requirement ($O_1 < O_2$). The hierarchy of the probability logics is isomorphic to $(\mathcal{O}^*, <)$. Thus, the probability logic LPP_2 is on the lowest level in the hierarchy of the $LPP_{2,P,Q,O}$-logics and corresponds to the 0-element of $(\mathcal{O}^*, <)$.

6.1.2. Complete axiomatization. Let us consider a fixed recursive family O of recursive subsets of S and the corresponding $LPP_{2,P,Q,O}$-logic. The axiomatic system $Ax_{LPP_{2,P,Q,O}}$ extends the system Ax_{LPP_2} with the following axiom:

(7) $P_{=s}\alpha \to Q_F\alpha$, where $F \in O$ and $s \in F$

and the inference rule:

(4) From $P_{=s}\alpha \Rightarrow \phi$, for all $s \in F$, infer $Q_F\alpha \Rightarrow \phi$.

As an illustration we give a list of useful theorems of $Ax_{LPP_{2,P,Q,O}}$:

Theorem 54. *If all the mentioned formulas belong to the set* $\mathrm{For}_{LPP_{2,P,Q,O}}$, *the following holds in the corresponding $LPP_{2,P,Q,O}$-logic:*

(1) $\vdash Q_F\alpha \to Q_G\alpha$, *for* $F \subset G$
(2) $\vdash (Q_F\alpha \wedge Q_G\alpha) \leftrightarrow Q_{F\cap G}\alpha$
(3) $\vdash (Q_F\alpha \vee Q_G\alpha) \leftrightarrow Q_{F\cup G}\alpha$
(4) $\vdash (Q_F\alpha \wedge P_{\geqslant s}\alpha) \leftrightarrow Q_{[s,1]\cap F}\alpha$, *and similar for* $P_{\geq s}\alpha$, $P_{\leq s}\alpha$, $P_{\leqslant s}\alpha$
(5) $\vdash Q_F\alpha \leftrightarrow Q_{1-F}\neg\alpha$, *where* $1 - F = \{1 - f : f \in F\}$
(6) $\vdash (Q_F\alpha \wedge \neg Q_G\alpha) \leftrightarrow Q_{F\smallsetminus G}\alpha$

Proof. Let us consider the case (1). If $F, G \in O$:

$\vdash P_{=s}\alpha \to Q_G\alpha$ for every $s \in F \subset G$, by Axiom 7
$\vdash Q_F\alpha \to Q_G\alpha$, by Rule 4.

The other statements follow similarly. □

The completeness proof for $Ax_{LPP_{2,P,Q,O}}$ follows the ideas from the corresponding proof from Section 3.4.

6.1.3. Decidability. In Section 3.5 we proved decidability of the LPP_2 logic which can be seen as an $LPP_{2,P,Q,O}$-logic with the empty family O. The proof involves a reduction of a formula to a system of linear (in)equalities. A look on this method indicates that the similar procedure might be applied for an arbitrary $LPP_{2,P,Q,O}$-logic. However, since there are also the operators of the form Q_F, instead of the system (5), we have to consider linear systems of the following form:

$$\sum_{i=1}^{2^n} y_i = 1$$

$$y_i \geqslant 0, \quad \text{for } i = 1, \ldots, 2^n$$

$$\text{(8)} \qquad \sum_{a_t \in X^1(p_1,\ldots,p_n) \in D} y_t \begin{cases} \geqslant s_1 & \text{if } X^1 = P_{\geqslant s_1} \\ < s_1 & \text{if } X^1 = P_{<s_1} \\ \in F_1 & \text{if } X^1 = Q_{F_1} \\ \notin F_1 & \text{if } X^1 = \neg Q_{F_1} \end{cases}$$

$$\ldots$$

$$\sum_{a_t \in X^k(p_1,\ldots,p_n) \in D} y_t \begin{cases} \geqslant s_k & \text{if } X^k = P_{\geqslant s_k} \\ < s_k & \text{if } X^k = P_{<s_k} \\ \in F_k & \text{if } X^1 = Q_{F_k} \\ \notin F_k & \text{if } X^1 = \neg Q_{F_k} \end{cases}$$

An obvious statement holds:

Theorem 55. *An $LPP_{2,P,Q,O}$-logic is decidable iff for every probabilistic formula $A \in \mathrm{For}_{LPP_{2,P,Q,O}} \setminus \mathrm{For}_C$ there is at least one disjunct from $DNF(A)$ such that the corresponding system* (8) *is solvable.*

The requirement from Theorem 55 is very strong. For example, consider the system

$$y_1 + y_2 = 1$$
$$y_i \geqslant 0, \text{ for } i = 1, 2$$
$$y_1 \geqslant s$$
$$y_1 \in F$$

obtained from the formula $P_{\geqslant s}p \wedge Q_F p$. The system is solvable only if $F \cap [s, 1] \neq \emptyset$ is decidable, and this depends on the set F. If F is a codomain of a suitable

rational-valued function, the system can be solved, but, in the general case, decidability of the set F does not imply that either the system is solvable or that the $LPP_{2,P,Q,O}$-logic is decidable. However, there are recursive families O such that the corresponding probabilistic logics are decidable. A trivial example of this kind is any recursive $O \subseteq [S]^{<\omega}$, where $[S]^{<\omega}$ is the family of all finite subsets of S. A nontrivial example of a decidable logics concerns the logic which is characterized by the family O such that each $F \in O$ is definable (with rational parameters) in the language of ordered groups.

6.2. Qualitative probabilities. Reasoning about qualitative probabilities is one of the most common cases of qualitative reasoning. Here we offer the first strongly complete formalization of the notion of qualitative probability within the framework of probabilistic logic. We obtain the language of the corresponding logic (denoted $LPP_{2,\preceq}$) by extending the LPP_2-language with an additional binary operator $\preceq$, such that for some For_C-formulas α and β, $\alpha \preceq \beta$ means "β is at least probable as α". Similarly as in Section 6.1, we use the class $LPP_{2,\mathrm{Meas}}$ of measurable LPP_2-models, and the corresponding satisfiability relation (from Definition 2) with another additional requirement that:

- If $\alpha, \beta \in \mathrm{For}_C$, $\mathbf{M} \models \alpha \preceq \beta$ iff $\mu([\alpha]) \leqslant \mu([\beta])$,

The axiomatic system $Ax_{LPP_{2,\preceq}}$ extends the system Ax_{LPP_2} with the following axioms:

(7) $(P_{\leqslant s}\alpha \wedge P_{\geqslant s}\beta) \rightarrow \alpha \preceq \beta$
(8) $(\alpha \preceq \beta \wedge P_{\geqslant s}\alpha) \rightarrow P_{\geqslant s}\beta$,

and the inference rule:

(4) From $A \rightarrow (P_{\geqslant s}\alpha \rightarrow P_{\geqslant s}\beta)$ for every $s \in S$, infer $A \rightarrow \alpha \preceq \beta$.

The next theorem gives us some useful properties of the probability operator $\preceq$:

Theorem 56. *Suppose that T is a set of formulas and that α, β, $\gamma \in \mathrm{For}_C$. Then the following holds:*

(1) *$T \vdash \alpha \preceq \beta$ if and only if $T \vdash P_{\geqslant s}(\alpha) \rightarrow P_{\geqslant s}(\beta)$ for all $s \in S$;*
(2) *$\vdash \alpha \preceq \beta \vee \beta \preceq \alpha$;*
(3) *$\vdash (\alpha \preceq \beta \wedge \beta \preceq \gamma) \rightarrow \alpha \preceq \gamma$;*
(4) *$\vdash \alpha \preceq \alpha$;*
(5) *If $T \vdash P_{\geqslant 1}(\alpha \rightarrow \beta)$ then $T \vdash \alpha \preceq \beta$;*
(6) *If $T \vdash \alpha \rightarrow \beta$ then $T \vdash \alpha \preceq \beta$.*

Proof. Since (5) directly follows from (1), (4) from (2), and (6) is a consequence of (5) and Rule 2, we will prove only the first three statements.

(1) Suppose that $T \vdash \alpha \preceq \beta$. By the axioms 1 and 8 we have that

$$T \vdash \alpha \preceq \beta \rightarrow (P_{\geqslant s}(\alpha) \rightarrow P_{\geqslant s}(\beta)).$$

Now applying Rule 1 we obtain that $T \vdash P_{\geqslant s}(\alpha) \rightarrow P_{\geqslant s}(\beta)$. Conversely, suppose that $T \vdash P_{\geqslant s}(\alpha) \rightarrow P_{\geqslant s}(\beta)$ for each $s \in S$. Then by Axiom 1

$$T \vdash P_{\geqslant 0}(\alpha) \rightarrow (P_{\geqslant s}(\alpha) \rightarrow P_{\geqslant s}(\beta))$$

for each $s \in S$. Applying Rule 4 we deduce that $T \vdash P_{\geqslant 0}(\alpha) \to \alpha \preceq \beta$. Finally, since $T \vdash P_{\geqslant 0}(\alpha)$ (Axiom 2), by Rule 1 we conclude that $T \vdash \alpha \preceq \beta$.

(2) First let us observe that Axiom 7 is equivalent to

$$\neg(\alpha \preceq \beta) \to (P_{\geqslant s}(\beta) \to P_{>s}(\alpha)).$$

Since $\vdash P_{>s}(\alpha) \to P_{\geqslant s}(\alpha)$, we have that $\vdash \neg(\alpha \preceq \beta) \to (P_{\geqslant s}(\beta) \to P_{\geqslant s}(\alpha))$ for every $s \in S$, so by Rule 4 we obtain $\vdash \neg(\alpha \preceq \beta) \to \beta \preceq \alpha$, which is equivalent to $\vdash \alpha \preceq \beta \vee \beta \preceq \alpha$.

(3) According to Deduction theorem, it is sufficient to prove that

$$\alpha \preceq \beta, \beta \preceq \gamma \vdash \alpha \preceq \gamma.$$

Since $\alpha \preceq \beta \vdash P_{\geqslant s}(\alpha) \to P_{\geqslant s}(\beta)$ and $\beta \preceq \gamma \vdash P_{\geqslant s}(\beta) \to P_{\geqslant s}(\gamma)$, we have that

$$\alpha \preceq \beta, \beta \preceq \gamma \vdash P_{\geqslant s}(\alpha) \to P_{\geqslant s}(\gamma).$$

This holds for all $s \in S$, so applying the statement (1) from this theorem, we obtain that $\alpha \preceq \beta, \beta \preceq \gamma \vdash \alpha \preceq \gamma$. □

The corresponding completeness proof follows the same steps as above for LPP_2. Also, decidability can be proved in the same way as in Section 3.5 since the only new type of formulas $(\alpha \preceq \beta)$ can be reduced to an inequality of the form:

$$\sum_{a_k \in \mathrm{CDNF}(\alpha)} \mu([a_k]) \leqslant \sum_{a_k \in \mathrm{CDNF}(\beta)} \mu([a_k]).$$

It is also interesting that, if we add the qualitative probability operator to the logic $LPP_2^{\mathrm{Fr}(n)}$, due to the fact that the set Range (which denotes the range of the considered probability functions) is finite, $\alpha \preceq \beta$ can be seen as an abbreviation of the formula $\bigwedge_{s \in \mathrm{Range}}(P_{\geqslant s}\alpha \to P_{\geqslant s}\beta)$. Thus, the notion of the qualitative probability is definable in $LPP_2^{\mathrm{Fr}(n)}$, and the logics $LPP_2^{\mathrm{Fr}(n)}$ and $LPP_{2,\preceq}^{\mathrm{Fr}(n)}$ coincide (in the sense that the later one is a conservative extension of the former logic).

7. First order probability logics

This section is devoted to a probabilistic extension of first order classical logic. In this case interleaving of the probabilistic operators and the classical quantifiers is important, especially when we compare first order probability logics to first order modal logics. Thus, to avoid repetition and contrary to Section 3, we will start here with the logic $LFOP_1$, the first order counterpart of the propositional probability logic LPP_1.

7.1. Syntax. The language of the $LFOP_1$-logic is an extension of the classical first order language. It is a countable set which contains for each non negative integer k, k-ary relation symbols $P_0^k, P_1^k, \ldots$, and k-ary function symbols $F_0^k, F_1^k, \ldots$, and the logical symbols $\wedge$, and $\neg$, quantifier $\forall$, a list of unary probability operators $P_{\geqslant s}$ for every rational number $s \in [0,1]$, variables $x, y, z, \ldots$, and parentheses.

The notions of existential quantifier, arity of a functional or a relational symbol, term, atomic formula, bound and free variables, sentence, and a term free for a variable in a formula can be defined as usual, while the set For_{LFOP_1} of formulas

is the smallest set containing atomic formulas and closed under formation rules: if α and β are formulas, then $\neg\alpha$, $P_{\geqslant s}\alpha$, $\alpha \wedge \beta$ and $(\forall x)\alpha$ are formulas.

Example 57. An example of a formula is:

$$P_{\geqslant s}(\forall x)P_1^1(x) \rightarrow P_3^2(y, F_0^0) \wedge P_{\geqslant r}P_{\geqslant t}P_1^1(F_1^0).$$

$\alpha(x_1, \ldots, x_m)$ indicates that free variables of the formula α form a subset of $\{x_1, \ldots, x_m\}$. If t is a term free for x in α, then $\alpha(t/x)$ denotes the result of substituting in α the term t for all free occurrences of x. We will also use the shorter form $\alpha(t)$ to denote the same substitution.

7.2. Semantics. The models we will use are similar to $LPP_{1,\text{Meas}}$-models with an important difference that worlds of models are now classical first order models. More formally:

Definition 58. An *$LFOP_1$-model* is a structure $\mathbf{M} = \langle W, H, \mu, v\rangle$ where:

- W is a non empty set of objects called worlds,
- D associates a non empty domain $D(w)$ with every world $w \in W$,
- I associates an interpretation $I(w)$ with every world $w \in W$ such that:
  - $I(w)(F_i^k)$ is a function from $D(w)^k$ to $D(w)$, for all i, and k,
  - $I(w)(P_i^k)$ is a relation over $D(w)^k$, for all i, and k.
- Prob is a probability assignment which assigns to every $w \in W$ a probability space, such that $\text{Prob}(w) = \langle W(w), H(w), \mu(w)\rangle$, where:
  - $W(w)$ is a non empty subset of W,
  - $H(w)$ is an algebra of subsets of $W(w)$ and
  - $\mu(w) : H(w) \rightarrow [0, 1]$ is a finitely additive probability measure.

The next definitions reflect the mentioned fact that worlds in $LFOP_1$-models are classical first order models.

Definition 59. Let $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$ be an $LFOP_1$-model. A *variable valuation* v assigns some element of the corresponding domain to every world w and every variable x, i.e., $v(w)(x) \in D(w)$. If $w \in W$, $d \in D(w)$, and v is a valuation, then $v_w[d/x]$ is a valuation like v except that $v_w[d/x](w)(x) = d$.

Definition 60. For an $LFOP_1$-model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$ and a valuation v the *value of a term t* (denoted by $I(w)(t)_v$) is:

- if t is a variable x, then $I(w)(x)_v = v(w)(x)$, and
- if $t = F_i^m(t_1, \ldots, t_m)$, then $I(w)(t)_v = I(w)(F_i^m)(I(w)(t_1)_v, \ldots, I(w)(t_m)_v)$.

Definition 61. The truth value of a formula α in a world $w \in W$ for a given $LFOP_1$-model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$, and a valuation v (denoted by $I(w)(\alpha)_v$) is:

- if $\alpha = P_i^m(t_1, \ldots, t_m)$, then $I(w)(\alpha)_v = \top$ if $\langle I(w)(t_1)_v, \ldots, I(w)(t_m)_v\rangle \in I(w)(P_i^m)$, otherwise $I(w)(\alpha)_v = \bot$,
- if $\alpha = \neg\beta$, then $I(w)(\alpha)_v = \top$ if $I(w)(\beta)_v = \bot$, otherwise $I(w)(\alpha)_v = \bot$,
- if $\alpha = P_{\geqslant s}\beta$, then $I(w)(\alpha)_v = \top$ if $\mu(w)\{u \in W(w) : I(u)(\beta)_v = \top\} \geqslant s$, otherwise $I(w)(\alpha)_v = \bot$,

- if $\alpha = \beta \wedge \gamma$, then $I(w)(\alpha)_v = \top$ if $I(w)(\beta)_v = \top$, and $I(w)(\gamma)_v = \top$, otherwise $I(w)(\alpha)_v = \bot$, and
- if $\alpha = (\forall x)\beta$, then $I(w)(\alpha)_v = \top$ if for every $d \in D$, $I(w)(\beta)_{v_w[d/x]} = \top$, otherwise $I(w)(\alpha)_v = \bot$.

Definition 62. A formula *holds in a world* w from an $LFOP_1$-model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$ (denoted by $(M, w) \models \alpha$) if for every valuation v, $I(w)(\alpha)_v = \top$. If $d \in D(w)$, we will use $(\mathbf{M}, w) \models \alpha(d)$ to denote that for every valuation v, $I(w)(\alpha(x))_{v_w[d/x]} = \top$.

A sentence α is*satisfiable* if there is a world w in an $LFOP_1$-model $\mathbf{M}$ such that $(\mathbf{M}, w) \models \alpha$. A set T of sentences is satisfiable if there is a world w in an $LFOP_1$-model $\mathbf{M}$ such that for every $\alpha \in T$, $(\mathbf{M}, w) \models \alpha$.

A sentence α is*valid* if for every $LFOP_1$-model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$ and every world $w \in W$, $(\mathbf{M}, w) \models \alpha$.

In the sequel we will consider a class of all $LFOP_1$-models that satisfy:

- all the worlds from a model have the same domain, i.e., for all $v, w \in W$, $D(v) = D(w)$,
- for every sentence α, and every world w from a model $\mathbf{M}$ the set $\{u \in W(w) : I(u)(\alpha)_v = \top\}$ of all worlds from $W(w)$ that satisfy α is measurable, and
- the terms are rigid, i.e., for every model their meanings are the same in all worlds.

We use $LFOP_{1,\text{Meas}}$ to denote that class of all fixed domain measurable models with rigid terms.

Example 63. Let us consider the formula $P_{\geqslant s}P_1^1(x)$, and suppose that for an $LFOP_{1,\text{Meas}}$-model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$, $w \in W$, $(\mathbf{M}, w) \models P_{\geqslant s}P_1^1(x)$. By Definition 62, this holds iff for every valuation v, $I(w)(P_{\geqslant s}P_1^1(x))_v = \top$ iff $(\mathbf{M}, w) \models (\forall x)P_{\geqslant s}P_1^1(x)$.

On the other hand, as we will show in Example 64, the satisfiability of the formula $P_{\geqslant s}P_1^1(x)$ does not imply the satisfiability of $P_{\geqslant s}(\forall x)P_1^1(x)$. The example assures an already existing impression that, although probability and modal logics are closely related, modal necessity (denoted by $\Box$) is a stronger notion than probability necessity (probability one, $P_{\geqslant 1}$).

Example 64. Let us consider, the well known Barcan formula of the first order modal logic:

$$\mathbf{BF} \quad (\forall x)\Box\alpha(x) \rightarrow \Box(\forall x)\alpha(x)$$

It is proved that **BF** holds in the class of all first order fixed domain modal models, and that it is independent from the other first order modal axioms. However, the behavior of the reminiscence of this formula:

$$\mathbf{BF}(s) \quad (\forall x)P_{\geqslant s}\alpha(x) \rightarrow P_{\geqslant s}(\forall x)\alpha(x)$$

is quite different.

If $s = 0$, $\mathbf{BF}(0)$ is valid, because $P_{\geqslant 0}(\forall x)\alpha(x)$ always holds since probability functions are nonnegative. So, suppose that $s > 0$. Let us consider the $LFOP_{1,\text{Meas}}$-model $\mathbf{M}$ such that:

- $W = \{w_1, w_2, w_3, w_4\}$
- $D = \{d_1, d_2\}$,
- $(\mathbf{M}, w_2) \models P_1^1(d_1)$, $(\mathbf{M}, w_2) \not\models P_1^1(d_2)$, $(\mathbf{M}, w_3) \models P_1^1(d_1)$, $(\mathbf{M}, w_3) \models P_1^1(d_2)$, $(\mathbf{M}, w_4) \not\models P_1^1(d_1)$, $(\mathbf{M}, w_4) \models P_1^1(d_2)$,
- $\mu(w_1)(w_2) = \frac{1}{n}$, $\mu(w_1)(w_3) = s - \frac{1}{n}$ $\mu(w_1)(w_4) = \frac{1}{n}$

It is easy to see that $(\mathbf{M}, w_1) \models (\forall x)P_{\geqslant s}P_1^1(x)$, because

$$\mu(w_1)(\{w : w \models P_1^1(d_1)\}) = \mu(w_1)(\{w_2, w_3\}) = s,$$
$$\mu(w_1)(\{w : w \models P_1^1(d_2)\}) = \mu(w_1)(\{w_3, w_4\}) = s.$$

On the other hand, $(\mathbf{M}, w_1) \not\models (\forall x)P_1^1(x)$, $(\mathbf{M}, w_2) \not\models (\forall x)P_1^1(x)$, and $(\mathbf{M}, w_4) \not\models (\forall x)P_1^1(x)$, whilst $(\mathbf{M}, w_3) \models (\forall x)P_1^1(x)$. Since $\mu(w_1)(\{w_3\}) = s - \frac{1}{n}$, $(\mathbf{M}, w_1) \not\models P_{\geqslant s}(\forall x)P_1^1(x)$, and for $s > 0$, $(\mathbf{M}, w_1) \not\models \mathbf{BF}(s)$.

7.3. A sound and complete axiomatic system. The axiomatic system Ax_{LFOP_1} is a combination of a classical first order axiomatization and the probabilistic axioms introduced in Section 3. It involves the following axiom schemas:

(1) all the axioms of the classical propositional logic
(2) $(\forall x)(\alpha \to \beta) \to (\alpha \to (\forall x)\beta)$, where x is not free in α
(3) $(\forall x)\alpha(x) \to \alpha(t/x)$, where $\alpha(t/x)$ is obtained by substituting all free occurrences of x in $\alpha(x)$ by the term t which is free for x in $\alpha(x)$
(4) $P_{\geqslant 0}\alpha$
(5) $P_{\leqslant r}\alpha \to P_{<s}\alpha$, $s > r$
(6) $P_{<s}\alpha \to P_{\leqslant s}\alpha$
(7) $(P_{\geqslant r}\alpha \wedge P_{\geqslant s}\beta \wedge P_{\geqslant 1}(\neg\alpha \vee \neg\beta)) \to P_{\geqslant \min(1, r+s)}(\alpha \vee \beta)$
(8) $(P_{\leqslant r}\alpha \wedge P_{<s}\beta) \to P_{<r+s}(\alpha \vee \beta)$, $r + s \leqslant 1$

and inference rules:

(1) From α and $\alpha \to \beta$ infer β.
(2) From α infer $(\forall x)\alpha$
(3) From α infer $P_{\geqslant 1}\alpha$.
(4) From $\beta \to P_{\geqslant s-\frac{1}{k}}\alpha$, for every integer $k \geqslant \frac{1}{s}$, infer $\beta \to P_{\geqslant s}\alpha$.

We use the notions of deducibility and consistency introduced in the definitions 30 and 31 from Section 5. The theorems 65 and 66 show that Ax_{LFOP_1} characterizes the set of all $LFOP_{1,\text{Meas}}$-valid sentences.

Theorem 65 (Soundness theorem). *The axiomatic system Ax_{LFOP_1} is sound with respect to the $LFOP_{1,\text{Meas}}$ class of models.*

Proof. Let α' be an instance of a classical propositional axiom α obtained by substituting propositional letters by formulas. Suppose that the formula α' is not valid, i.e., that for some world w from a model $\mathbf{M}$, and a valuation v, $I(w)(\alpha')_v = \bot$. It follows that we can find a classical propositional valuation ρ such that $\rho(\alpha) = \bot$, a contradiction. Let $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$ and $w \in W$ such that $(\mathbf{M}, w) \models (\forall x)\alpha(x)$.

It means that $I(w)((\forall x)\alpha(x))_v = \top$ for every valuation v. Among these valuations there must be one (denoted v') which assigns to x the value $d = I(w)(t)_v$. For this valuation $I(w)(\alpha(x))_{v'} = \top$. Since $I(w)(\alpha(x))_{v'} = I(w)(\alpha(t/x))_v$, we have $I(w)(\alpha(t/x))_v = \top$ for every valuation. Thus, every instance of Axiom 3 is valid. Note that the assumptions about fixed domains and rigidness of terms are crucial. If it is not the case, and $\alpha(t/x)$ is of the form $P_{\geqslant s}\beta(t/x)$, the term t refers to objects in other worlds (different from w). It can have a consequence that $I(w)(\alpha(t/x))_v = \bot$. The axioms 4– 8 concern the properties of measures from $LFOP_{1,\text{Meas}}$-models and obviously hold in every model. The inference rules 1 and 2 are validity-preserving for the same reason as in the classical first order logic. Consider Rule 3 and suppose that a formula α is valid. It must hold in every world from every $LFOP_{1,\text{Meas}}$-model. Thus, for every model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$, and $w \in W$, the sets $\{u \in W(w) : (\mathbf{M}, u) \models \alpha\}$ and $W(w)$ coincide. Since $\mu(w)(W(w)) = 1$, it follows that $(\mathbf{M}, w) \models P_{\geqslant 1}\alpha$. Rule 4 preserves validity because of the properties of the set of rational numbers. □

Theorem 66 (Extended completeness theorem for $LFOP_{1,\text{Meas}}$). *The axiomatic system Ax_{LFOP_1} is sound with respect to the $LFOP_{1,\text{Meas}}$ class of models.*

Proof. The completeness proof follows the same ideas as above, for example as in Section 5. The main new step is that, since we work with first-order formulas, we have a special kind of maximal consistent sets called saturated sets. A set T of formulas is saturated if it is maximal consistent and satisfies:

- if $\neg(\forall x)\alpha(x) \in T$, then for some term t, $\neg\alpha(t) \in T$.

We can prove a counterpart of Theorem 13, where the new step:

- if the set T_{i+1} is obtained by adding a formula of the form $\neg(\forall x)\beta(x)$ to the set T_i, then for some $c \in C$, $\neg\beta(c)$ is also added to T_{i+1}, so that T_{i+1} is consistent,

guarantees that every consistent set of sentences can be extended to a saturated set (C is a countably infinite set of new constant symbols). Then, the canonical model $\mathbf{M} = \langle W, D, I, \text{Prob}\rangle$ can be defined in the following way:

- W is the set of all saturated sets,
- D is the set of all variable-free terms,
- for every $w \in W$, $I(w)$ is an interpretation such that:
  - for every function symbol F_i^m, $I(w)(F_i^m)$ is a function from D^m to D such that for all variable-free terms $t_1, \ldots, t_m$ in $\overline{\mathcal{L}}$, $F_i^m : \langle t_1, \ldots, t_m\rangle \to F_i^m(t_1, \ldots, t_m)$, and
  - for every relation symbol P_i^m, $I(w)(P_i^m) = \{\langle t_1, \ldots, t_m\rangle$ for all variable-free terms $t_1, \ldots, t_m \in \overline{\mathcal{L}} : P_i^m(t_1, \ldots, t_m) \in w\}$.
- for every $w \in W$, $\text{Prob}(w) = \langle W(w), H(w), \mu(w)\rangle$ such that:
  - $W(w) = W$,
  - $H(w)$ is a class of sets $[\alpha] = \{w \in W : \alpha \in w\}$, for every sentence α, and
  - for every set $A \in H(w)$, $\mu(w)(A) = \sup_s\{P_{\geqslant s}\alpha \in w\}$.

and the rest of the proof is standard. □

The same arguments as in Section 3.4 can be used to prove completeness of Ax_{LFOP_1} with respect to the classes: $LPFOP_{1,\text{Meas,All}}$, $LFOP_{1,\text{Meas,Neat}}$ and $LFOP_{1,\text{Meas},\sigma}$, while the modifications similar to the ones from the previous sections will be appropriate for logics $LFOP_1^{\text{Fr}(n)}$, $LFOP_1^{A,\omega_1,\text{Fin}}$, $LFOP_1^S$, and the logic $LFOP_2$ (the first order probability logic without iterations) and its variants.

7.4. Decidability. The logic $LFOP_1$ and its variants contain classical first order logic. Thus, they are undecidable. The monadic fragments of the considered systems are undecidable, too. To show that, we can use the procedure due to Saul Kripke [48, 62] and consider a translation of classical first order formulas that contain only one binary relation symbol P^2 to monadic probability formulas such that a classical first order formula is valid if and only if its translation is a valid probability formula. The original translation replaces every expression of the form $P^2(t_1, t_2)$ in a classical formula by $\Diamond(P_1^1(t_1) \wedge P_2^1(t_2))$, and instead of the modal formula we can use its probabilistic counterpart $P_{\geqslant 0}(P_1^1(t_1) \wedge P_2^1(t_2))$. Since the fragment of the classical first order logic with a single binary relation symbol is not decidable, the same holds for the monadic fragments of the first order probability logics with iterations of probability operators. However, it is interesting that the monadic fragment of $LFOP_2$ is decidable.

8. Probabilistic logics with the non-classical base

Let us use the term *the basic logic* for a logic from which we start building a probability logic. So far, we have used only classical (propositional or first order) logic as the basic logic, and it might be usefull to provide some motivation for other possible choices. The most important reason to change the basic logic, from our point of view, might be the very nature of classical logic. Namely, it is basically the logic of mathematics conceived as pertaining to some outside (Platonic) reality. On this conception, statements are either true or false and forever so (truth is independent of time and place), there is no room for modality (maybe, possibly,...) or value judgment. It is not surprising that the resulting logic will have some consequences which seem rather odd in real-life situations and this issue has been debated throughout the last century, often under the heading "paradoxes of the material implication". In this section we will address those issues, and consider two logics: in the first one we will use intuitionistic logic as the basic logic, while in the second we will start from a temporal base. However, we do not argue that either "classical" or any of "non-classical" probabilistic logics is the unique logic for modelling probabilistic reasoning. Our view is more pragmatic: we believe that there are real-life situations in which the former approach could be appropriate, but the same holds for the later one.

8.1. An intuitionistic probability logic. Intuitionistic logic arises quite naturally from a conception of mathematics as a human endeavor not pertaining to some outside reality. Since the statements of mathematics are not about something which exists out there, they cannot be true or false but only proved or disproved. This leaves another category of statements, those which are as yet undetermined. Thus

intuitionistic logic may be viewed as the logic of the growth of human knowledge (as opposed to the classical logic which we may regard as the logic of the static Platonic universe of mathematical objects). Thanks to this, intuitionistic logic has less consequences which would seem rather unintuitive in a real-life situation (e.g., $(p \to q) \vee (q \to p)$ and $(p \to (q \vee r)) \to ((p \to q) \vee (p \to r))$ are not intuitionistic theorems, i.e., there are models in which they are false). In reality, there is the fact that the intuitionistic logic might be the least popular non-classical logic among the practitioners of artificial intelligence and computer science in general. However, for those comfortable with the ubiquitous S4-modal logic and uncomfortable with intuitionism, we should emphasize that these two logics are practically the same: their models are the same, while the Gödel translation enables us to interpret syntax. Furthermore, as we shall show in the Remark at the end of this section, intuitionistic logic arises naturally whenever we deal with possible worlds semantics. In any case, starting with intuitionistic logic, we naturally have, besides proved statements (probability is 1) and disproved statements (probability is 0), undetermined statements whose probability should range between 0 and 1. This is more obvious if we consider a Kripke model in which we can assign a probability to a formula on the basis of the number of possible worlds in which it is true. In our approach the probabilistic operators have the classical treatment. As a justification, we may say that once we determine the probability of an uncertain proposition α, it should be either greater or equal to some $s \in [0,1]$ or not, so it is not unreasonable to assume $P_{\geqslant s}\alpha \vee \neg P_{\geqslant s}\alpha$ (even if we reject $\alpha \vee \neg\alpha$).

We use LPP_2^I to denote the corresponding intuitionistic probability logic. At the propositional level, the language contains the connectives $\neg$, $\wedge$, $\vee$ and $\to$, while on the probabilistic level we have two lists of unary probabilistic operators $(P_{\geqslant s})_{s\in S}$, and $(P_{\leqslant s})_{s\in S}$, and the connectives $\neg$ and $\wedge$. Note that, since we have the intuitionistic base:

- at the propositional level, the propositional connectives are independent, and
- at the probabilistic level, the probabilistic operators $P_{\geqslant .}$, and $P_{\leqslant .}$ are independent, but $\vee$ and $\to$ can be defined from $\neg$ and $\wedge$.

Similarly as for the logic LPP_2, we do not allow iterations of probabilistic operators, and define the sets For_I of propositional formulas, For_P of probabilistic formulas, and $\mathrm{For}_{LPP_2^I}$ of all formulas, as in Section 3.1.

8.1.1. Semantics. Corresponding to the structure of the set $\mathrm{For}_{LPP_2^I}$, there are two levels in the definition of models. At the first level there is the notion of intuitionistic Kripke models [63], while probability comes in the picture at the second level.

Definition 67. An *intuitionistic Kripke* model for the language For_I is a structure $\langle W, \leqslant, v\rangle$ where:

- $\langle W, \leqslant\rangle$ is a partially ordered set of possible worlds which is a tree, and
- v is a valuation function, i.e., v maps the set W into the powerset $\mathcal{P}(\phi)$, which satisfies the condition: for all w, $w' \in W$, $w \leqslant w'$implies $v(w) \subseteq v(w')$.

The last requirement from Definition 67 allows that v does not determine the status of some primitive propositions from ϕ in some worlds. In each Kripke model we define the forcing relation $\Vdash \subset W \times \mathrm{For}_I$ by the following definition:

Definition 68. Let $\langle W, \leqslant, v\rangle$ be an intuitionistic Kripke model. *The forcing relation* $\Vdash$ is defined by the following conditions for every $w \in W$, $\alpha, \beta \in \mathrm{For}_I$:

- if $\alpha \in \phi$, $w \Vdash \alpha$ iff $\alpha \in v(w)$,
- $w \Vdash \alpha \wedge \beta$ iff $w \Vdash \alpha$ and $w \Vdash \beta$,
- $w \Vdash \alpha \vee \beta$ iff $w \Vdash \alpha$ or $w \Vdash \beta$,
- $w \Vdash \alpha \to \beta$ iff for every $w' \in W$ if $w \leqslant w'$ then $w' \nVdash \alpha$ or $w' \Vdash \beta$, and
- $w \Vdash \neg\alpha$ iff for every $w' \in W$ if $w \leqslant w'$ then $w' \nVdash \alpha$.

We read $w \Vdash \alpha$ as "w *forces* α" or "α is true in the world w". *Validity in the intuitionistic Kripke model* $\langle W, \leqslant, v\rangle$ is defined by $\langle W, \leqslant, v\rangle \vDash \alpha$ iff $(\forall w \in W) w \Vdash \alpha$. A formula α is *valid* ($\vDash \alpha$) if it is valid in every intuitionistic Kripke model.

Let $\mathbf{M}_I = \langle W, \leqslant, v\rangle$ be an intuitionistic Kripke model. We use $[\alpha]_{\mathbf{M}_I}$ (or shortly $[\alpha]$ if $\mathbf{M}_I$ is clear from the context) to denote $\{w \in W : w \Vdash \alpha\}$ for every $\alpha \in \mathrm{For}_I$. The family $H_I = \{[\alpha]_{\mathbf{M}_I} : \alpha \in \mathrm{For}_I\}$ is a Heyting algebra with operations:

$$[\alpha] \cup [\beta] = [\alpha \vee \beta], \quad [\alpha] \cap [\beta] = [\alpha \wedge \beta], \quad [\alpha] \Rightarrow [\beta] = [\alpha \to \beta], \quad \text{and} \quad \sim [\alpha] = [\neg\alpha].$$

Thus, H_I is a lattice on W, but it may be not closed under complementation.

Definition 69. A *measurable probabilistic model* is a structure $\mathbf{M} = \langle W, \leqslant, v, H, \mu\rangle$ where:

- $\mathbf{M}_I = \langle W, \leqslant, v\rangle$ is an intuitionistic Kripke model,
- H is an algebra on W containing $H_I = \{[\alpha]_{\mathbf{M}_I} : \alpha \in \mathrm{For}_I\}$,
- $\mu : H \to [0, 1]$ is a finitely additive probability.

Note that H contains all sets of the form $W \setminus [\alpha]_{\mathbf{M}_I}$, even if for some $\alpha \in \mathrm{For}_I$ it may be that $W \setminus [\alpha]_{\mathbf{M}_I} \neq [\neg\alpha]_{\mathbf{M}_I}$. The fact that $[\neg\alpha]$ does not have to contain the complement of $[\alpha]$ is the reason why we need both $P_{\geqslant s}$ and $P_{\leqslant s}$ operators since $P_{\leqslant s}\alpha$ will not imply $P_{\geqslant 1-s}\neg\alpha$.

We use $LPP^I_{2,\mathrm{Meas}}$ to denote the class of all measurable probabilistic models.

Definition 70. *The satisfiability relation* $\vDash$ is defined by the following conditions for every $LPP^I_{2,\mathrm{Meas}}$-model $\mathbf{M} = \langle W, \leqslant, v, H, \mu\rangle$:

- if $\alpha \in \mathrm{For}_I$, $\mathbf{M} \vDash \alpha$ if $(\forall w \in W) w \Vdash \alpha$,
- $\mathbf{M} \vDash P_{\geqslant s}\alpha$ if $\mu([\alpha]) \geqslant s$,
- $\mathbf{M} \vDash P_{\leqslant s}\alpha$ if $\mu([\alpha]) \leqslant s$,
- if $A \in \mathrm{For}_P$, $\mathbf{M} \vDash \neg A$ if $\mathbf{M} \vDash A$ does not hold, and
- if $A, B \in \mathrm{For}_P$, $\mathbf{M} \vDash A \wedge B$ if $\mathbf{M} \vDash A$, and $\mathbf{M} \vDash B$.

Definition 71. A formula $\Phi \in \mathrm{For}_{LPP^I_2}$ is *satisfiable* if there is a $LPP^I_{2,\mathrm{Meas}}$-model $\mathbf{M}$ such that $\mathbf{M} \vDash \Phi$; Φ is *valid* if for every $LPP^I_{2,\mathrm{Meas}}$-model $\mathbf{M}$, $\mathbf{M} \vDash \Phi$; a set of formulas is *satisfiable* if there is an $LPP^I_{2,\mathrm{Meas}}$-model $\mathbf{M}$ such that for every formula Φ from the set, $\mathbf{M} \vDash \Phi$.

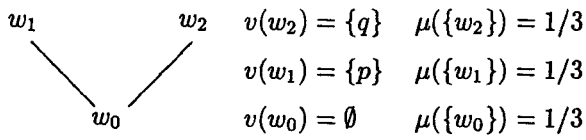


FIGURE 2. A tree-like probabilistic model

8.1.2. Examples. In this section we consider some consequences of probabilistic reasoning which is based on classical logic and which can be avoided using probabilistic logic based on intuitionistic logic.

Example 72. It is well known that $\neg(p \wedge q) \rightarrow (\neg p \vee \neg q)$ is a classical tautology, called De Morgan's law which is not an intuitionistic tautology. Still, even if we believe that it is impossible to have your cake and eat it, we do not believe that it is impossible to have your cake and we also do not believe that it is impossible to eat your cake. More formally, we would like to have $P_{\geqslant 1}\neg(p \wedge q)$, but also $P_{\leqslant \epsilon}\neg p$ and $P_{\leqslant \epsilon}\neg q$ for come small ϵ, which is impossible with classical logic.

Example 73. Consider the classical tautology $(p \rightarrow q) \vee (q \rightarrow p)$ and probability logic based on classical logic. Since tautologies have probability equal to 1, $P_{\geqslant 1}((p \rightarrow q) \vee (q \rightarrow p))$ is valid. Let us now take a real-life situation, where p and q mean "it rains" and "the sprinkler is on", respectively. It is clear that the sprinkler should not be on when it rains, i.e., that $p \rightarrow q$ should have low probability, say less than ϵ ($P_{\leqslant \epsilon}(p \rightarrow q)$). Since probability is additive, the measure of the union of two sets is less or equal than the sum of the measures of those sets. Thus, the probability of $q \rightarrow p$ has to be high. In other words, we get that it is very probable that it will rain whenever the sprinkler is on ($P_{\geqslant 1-\epsilon}(q \rightarrow p)$). If we were designing a controller for the sprinkler, this certainly would not be a desirable consequence.

On the other hand, $(p \rightarrow q) \vee (q \rightarrow p)$ is not an intuitionistic tautology. Consider the model from Figure 2. Recall that $p \rightarrow q$ being false in a Kripke model means that there is at least one possible world in which it is raining but the sprinkler is off. It is easy to see that $w_1 \Vdash q \rightarrow p$, $w_1 \nVdash p \rightarrow q$ $w_2 \Vdash p \rightarrow q$, $w_2 \nVdash q \rightarrow p$, $w_0 \nVdash (p \rightarrow q) \vee (q \rightarrow p)$, $\mu([(p \rightarrow q) \vee (q \rightarrow p)]_M) = 2/3$, and $M \nvDash P_{\geqslant 1}((p \rightarrow q) \vee (q \rightarrow p))$. Thus, the above consequence, that with high probability sprinkler causes raining, does not follow any more.

Note also that we can construct a model in which both $p \rightarrow q$ and $q \rightarrow p$ will have very low probability, say less than $1/n$, by simply adding $n-3$ linearly ordered new worlds below w_0 in M, and having $\mu(w) = 1/n$, $w \in W$. In the same model we have $P_{\geqslant 1}\neg(p \wedge q)$, $P_{\leqslant 1/n}\neg p$ and $P_{\leqslant 1/n}\neg q$, demonstrating the point of the previous example.

Example 74. Consider the classical (but not intuitionistic) tautology $(p \rightarrow (q \vee r)) \rightarrow ((p \rightarrow q) \vee (p \rightarrow r))$. Starting with classical logic makes $P_{\geqslant 1}((p \rightarrow (q \vee r)) \rightarrow ((p \rightarrow q) \vee (p \rightarrow r)))$ valid in probabilistic logic. If we take now p to be a description of our knowledge, q to be the P=NP-hypothesis, and r its negation, we obtain that $P_{\geqslant 1}((p \rightarrow (q \vee r))$ since $q \vee r$ is $q \vee \neg q$. It follows that $P_{\geqslant 1}((p \rightarrow q) \vee (p \rightarrow r))$,

which either means that our knowledge is inconsistent or that there is a considerable probability of at least one of the sentences "The P=NP-hypothesis follows from our current knowledge", "The negation of P=NP-hypothesis follows from our current knowledge", which is not very much likely. Again, since the above propositional formula is not intuitionisticaly valid, there is no such conclusion in the "intitionistic" probability logic.

8.1.3. Axiomatization, completeness, decidability. An axiomatization that characterizes the set of all $LPP^I_{2,\text{Meas}}$-valid formulas can be obtain by combining:

- any propositional intuitionistic axiomatization for For_I,
- any classical propositional axiomatization for For_P and
- probabilistic axioms and rules from Section 3.3

with the proviso that in this framework the probabilistic operators $P_{\geqslant .}$, and $P_{\leqslant .}$ are independent, so for example, Axiom 3 from the system Ax_{LPP_2} should be rewritten in the form: $P_{\geqslant 1-r}\neg\alpha \rightarrow \neg P_{\geqslant s}\alpha$ for $s > r$. We skip the corresponding completeness proof, but the proof of decidability for LPP^I_2 contains more new details and we give it in the next theorems. Let $A \in \text{For}_P$ and $\text{Subf}_I(A) = \{\alpha \in \text{For}_I : \alpha\text{is a subformula of}A\}$. Let $|A|$ and $|\text{Subf}_I(A)|$ denote the length of A, and the number of formulas in $|\text{Subf}_I(A)|$, respectively. Obviously, $|\text{Subf}_I(A)| \leqslant |A|$.

Theorem 75. *A probabilistic formula $A \in \text{For}_P$ is satisfiable iff it is satisfiable in a finite probabilistic model containing at most $2^{|A|^2}$ worlds.*

Proof. Let $\mathbf{M} = \langle W, \leqslant, v, H, \mu\rangle$, and $\mathbf{M} \vDash A$. For every $w \in W$, we use $\text{Subf}_I(w)$ to denote the set of all formulas from $\text{Subf}_I(A)$ forced in w, i.e., $\text{Subf}_I(w) = \{\alpha \in \text{Subf}_I(A) : w \Vdash \alpha\}$.

In the sequel we will follow the idea from [121, Theorem 5.3.4], and select some of the worlds from W to construct a finite model $\mathbf{M}^*$ satisfying A.

Let w_0 be the least element from W. We define the worlds of $\mathbf{M}^*$ (indexed by finite sequence) in the following way:

- $u_{\langle\rangle} = w_0$, where $\langle\rangle$ denotes the empty sequence,
- given u_σ let $u_{\sigma*\langle 1\rangle}, \ldots, u_{\sigma*\langle k\rangle}$ be the maximal set of worlds $w^{(1)}, \ldots, w^{(k)}$ from W such that for every $i, j \in \{1, \ldots, k\}$:
  - $u_\sigma \leqslant w^{(i)}$,
  - $\text{Subf}_I(u_\sigma) \neq \text{Subf}_I(w^{(i)})$,
  - if $u_\sigma \leqslant w \leqslant w^{(i)}$, then either $\text{Subf}_I(u_\sigma) = \text{Subf}_I(w)$ or $\text{Subf}_I(w) = \text{Subf}_I(w^{(i)})$, and
  - if $i \neq j$, then $\text{Subf}_I(w^{(i)}) \neq \text{Subf}_I(w^{(j)})$.

Let $W^* = \{\sigma : u_\sigma \text{ is defined}\}$, $\leqslant^*$ be the usual ordering of finite sequences, and for all $\sigma \in W^*$, and $\alpha \in \text{Var}$, $\alpha \in v^*(\sigma)$ iff $\alpha \in v(u_\sigma)$.

Using the induction on complexity of formulas we can prove that for every $\alpha \in \text{Subf}_I(A)$ and every $\sigma \in W^*$, $\sigma \Vdash \alpha$ in $\langle W^*, \leqslant^*, v^*\rangle$ iff $u_\sigma \Vdash \alpha$ in $\langle W, \leqslant, v\rangle$. If $\alpha \in \text{Var}$, the statement holds by the definition of v^*. Let $\alpha = \beta \rightarrow \gamma$. Suppose that $\sigma \nVdash \beta \rightarrow \gamma$. Then there is some $\rho \in W^*$ such that $\sigma \leqslant^* \rho$, $\rho \Vdash \beta$ and $\rho \nVdash \gamma$. By the induction hypothesis, $u_\rho \Vdash \beta$ and $u_\rho \nVdash \gamma$, $u_\sigma \leqslant u_\rho$, and $u_\sigma \nVdash \beta \rightarrow \gamma$. On the other

hand, suppose that $u_\sigma \nVdash \beta \to \gamma$. Then, there are two possibilities. First, if $u_\sigma \Vdash \beta$ it must be $u_\sigma \nVdash \gamma$, and by the induction hypothesis $\sigma \Vdash \beta$ and $\sigma \nVdash \gamma$, which means that $\sigma \nVdash \beta \to \gamma$. In the second case there is some $w \in W$ such that $u_\sigma \leqslant w$, $w \Vdash \beta$, and $w \nVdash \gamma$. Since $u_\sigma \nVdash \beta$, obviously $\mathrm{Subf}_I(u_\sigma) \neq \mathrm{Subf}_I(w)$. According to the above construction, there must be some $u_\theta \in W$ such that $u_\sigma \leqslant u_\theta \leqslant w$, $\sigma \leqslant^* \theta$, $u_\theta \Vdash \beta$, and $u_\theta \nVdash \gamma$. By the induction hypothesis, $\theta \Vdash \beta$, $\theta \nVdash \gamma$, and $\sigma \nVdash \beta \to \gamma$. The other cases follow similarly.

Let μ' be the finitely additive probability defined on $\{\{w \in W^* : w \Vdash \alpha\} : \alpha \in \mathrm{Subf}_I(A)\}$ such that $\mu'(\{w \in W^* : w \Vdash \alpha\}) = \mu([\alpha]_M)$. Since for every $\alpha \in \mathrm{Subf}_I(A)$, $[\alpha]_M \neq \emptyset$ iff $\{w \in W^* : w \Vdash \alpha\} \neq \emptyset$, is easy to see that μ' is correctly defined. Let $\mathbf{M}^* = \langle W^*, \leqslant^*, v^*, H^*, \mu^* \rangle$ be the probabilistic model such that H^* is the smallest algebra on W^* containing family $\{[\alpha]_{\mathbf{M}^*} : \alpha \in \mathrm{For}_I\}$, while μ^* is a finitely additive probability on H^* which is an extension of μ'. Note that it follows from Theorem 2 that such an extension always exists. Since probabilities of For_I-subformulas of A remain the same, $\mathbf{M}^* \vDash A$.

Finally, note that the set W^* is finite because every world has at most $2^{|\mathrm{Subf}_I(A)|}$ immediate successors and every branch contains at most $|\mathrm{Subf}_I(A)|$ worlds. Thus $|W^*| \leqslant (2^{|\mathrm{Subf}_I(A)|})^{|\mathrm{Subf}_I(A)|} \leqslant 2^{|A|^2}$. □

Theorem 76. *The satisfiability problem for probabilistic formulas is decidable.*

Proof. It follows from Theorem 75 that A is satisfiable iff it is satisfiable in a probabilistic model with at most $k_A = 2^{|A|^2}$ worlds. Thus, we can check satisfiability following ideas from Theorem 33: for every l, $1 \leqslant l \leqslant k_A$, there is only finitely many intuitionistic models with different valuations with respect to the set of propositional letters that occur in A. For every such intuitionistic model $\mathbf{M}_I = \langle W, \leqslant, v \rangle$ we can find the algebra H generated by the set $\{[\alpha]_{\mathbf{M}_I} : \alpha \in \mathrm{Subf}_I(A)\}$, and consider a linear system similar to the system (7). As there is a finite number of models and linear systems we have to check, and since linear programming problem is decidable, the same holds for the considered satisfiability problem. □

8.1.4. Remark. We will show here that even if we start with classical logic, possible-worlds semantics naturally produces intuitionistic logic. It turns out that intuitionistic implication will coincide with conditional probability when probability is equal to 1.

Let us start with a standard possible-world model $M = \langle W, H, \mu, v \rangle$. We may define a pre-order (reflexive and transitive relation) R on W by: uRw iff for every primitive proposition p, $v(u,p) = \text{true}$ implies $v(w,p) = \text{true}$. From this we may obtain a partial order in the usual way. First we introduce an equivalence relation $\sim$ defined by: $u \sim w$ iff uRw and wRu, and then we split W into equivalence classes: $C_u = \{w : u \sim w\}$. Now we may pick a selection $W' \subset W$ of representatives of equivalence classes (one for each class). So we have $(\forall u \in W)(\exists w \in W')(u \sim w)$. Obviously, R induces a partial order $\leqslant$ on W' such that $u \leqslant w$ iff uRw. Now we have a Kripke model with a partial ordering relation on worlds $\langle W', \leqslant, v \rangle$ which makes it a model for intuitionistic logic. Namely, we may define (semantically) a new propositional connective $\to$ by: $w \vDash \alpha \to \beta$ iff $(\forall w' \geqslant w)(w' \vDash \alpha$ implies

$w' \vDash \beta$). We may also define a new, intuitionistic, negation by: $-\alpha = \alpha \to \bot$. Therefore, even if we start with classical logic, when we come to models, we have an intuitionistic implication built in.

The interest in intuitionistic implication, besides the arguments proposed at the start of this section, comes from the fact that conditional probability, which is often used as the proper form of entailment in the context of probability logic, coincides in a sense with the intuitionistic implication, as can be seen from the following theorem.

Theorem 77. $\mu(\alpha \to \beta) = 1$ *iff* $\dfrac{\mu(\alpha \wedge \beta)}{\mu(\alpha)} = 1$.

However, this symmetry holds only in the case when probability is equal to 1. It is possible to construct models in which conditional probability is high while the probability of (intuitionistic) implication is low and vice versa. The reason is that, despite the fact that both operators are define globally (and not locally, in each world) the definitions are quite different. Conditional probability considers (i.e., counts) only worlds in which α is true, while intuitionistic implication takes into account also their predecessors. We may say, in a sense, that conditional probability disregards the development of events and regards only the final stages (with regard to the validity of α), i.e., the analysis starts with the worlds in which α is true and disregards the previous stages in which α may be "not yet true". Existence of long time-lines which end with worlds in which α is not true adds to the probability of $\alpha \to \beta$, while it is irrelevant for the conditional probability. On the other hand, a long sequence which has an ending in which α is true and β is not, reduces considerably the probability of $\alpha \to \beta$, while it may, in the presence of a relevant number of worlds in which both α and β are true, be insignificant for conditional probability.

8.2. A discrete linear-time probabilistic logic. In this section we describe a way in which probabilistic reasoning can be enriched with some temporal features. The temporal part of the logics is a standard discrete linear-time logic LTL [119], where the flow of time is isomorphic to natural numbers, i.e., each moment of time has a unique possible future, while the corresponding language contains the "next" operator ($\bigcirc$) and the reflexive strong "until" operator (U), (the operators "sometime" F and "always" G are definable: $\mathrm{F}\alpha = \top \mathrm{U} \alpha$ and $\mathrm{G}\alpha = \neg \mathrm{F} \neg \alpha$). Similarly as in Section 7, nesting of the probabilistic and temporal operators is important and we will start from the logic LPP_1. In our logic, denoted LPP_1^{LTL}, the probabilistic operators quantify events along a single time line. It allows us to express sentences such as "(according to the current set of information) the probability that, sometime in the future, α is true is at least s". And, as the knowledge can evolve during the time, the probability of α might change too. Note that, since the operators "sometime" and "always" can be seen as the existential and universal quantifiers over time instants, the probabilistic operators give more refined quantitative characterization of sets of time instants definable by formulas. We may try to motivate the proposed semantics in the following way.

Example 78. A suitable representation of all possible outcomes of an infinite sequence of probabilistic experiments (let us say that experiments A and B are permanently repeated resulting in a or $\neg a$, and b or $\neg b$, respectively) could be an inifinite tree, where every branch corresponds to a possible realization of the sequence of the experiments, and every time instant is described in the form $\pm a, \pm b$ depending on obtaining (or not obtaining) a and b in the corresponding experiment. We might be interested in probabilistic properties that hold for all branches. In that case we can reason about an arbitrary branch and need ability to express probabilities of events along it, for example that the probability of the event a is at least s, or some more complicated conditions, like that in every time instant, if the probability of a is less than r, then b must hold forever.

The set $\mathrm{For}_{LPP_1^{LTL}}$ of formulas is defined inductively as the smallest set containing primitive propositions and closed under formation rules: if α and β are formulas, then $\neg\alpha$, $\bigcirc\alpha$, $P_{\geqslant s}\alpha$, for every $s \in S$, $\alpha \wedge \beta$, and $\alpha \mathrm{U} \beta$ are formulas. We will use the following notational definition: $\bigcirc^0\alpha = \alpha$, and $\bigcirc^{i+1}\alpha = \bigcirc \bigcirc^i \alpha$ for $i \geqslant 0$. If $T = \{\alpha_1, \alpha_2, \ldots\}$ is a set of formulas, then $\bigcirc T$ denotes $\{\bigcirc\alpha_1, \bigcirc\alpha_2, \ldots\}$.

Example 79. An example of a formula is $(\bigcirc P_{\geqslant r} p \wedge \mathrm{F} P_{<s}(p \rightarrow q)) \rightarrow \mathrm{G} P_{=t} q$ which can be read as "if the probability of p in the next moment is at least r and sometime in the future q follows from p with the probability less than s, then the probability of q will always be equal to t."

8.2.1. Semantics. The semantics for LPP_1^{LTL} is a Kripke-style one using sequences of natural numbers as frames.

Definition 80. An LPP_1^{LTL}*-model* is a structure $\mathbf{M} = \langle W, \mathrm{Prob}, v \rangle$ where:

- $W = \{w_0, w_1, \ldots\}$ is a sequence of *time instants*,
- Prob is a probability assignment which assigns to every $w \in W$ a probability space, such that $\mathrm{Prob}(w) = \langle W(w), H(w), \mu(w) \rangle$, where:
  - $W(w) = \{w_j : j \geqslant i\}$,
  - $H(w)$ is an algebra of subsets of $W(w)$ and
  - $\mu(w) : H(w) \rightarrow [0, 1]$ is a finitely additive probability measure.
- v assigns to every $w \in W$ a two-valued evaluation of the primitive propositions, i.e., for every $w \in W$, $v(w) : \phi \rightarrow \{\mathrm{true}, \mathrm{false}\}$.

Definition 81. Let $\mathbf{M} = \langle W, \mathrm{Prob}, v \rangle$ be a LPP_1^{LTL}-model, $i \in \omega$ and α be a formula. The *satisfiability relation* $\models$ is inductively defined as follows:

- if $p \in \phi$ is a primitive proposition, $w_i \models p$ if $v(w_i)(p) = \mathrm{true}$,
- $w_i \models \neg\alpha$ if $w_i \not\models \alpha$,
- $w_i \models P_{\geqslant s}\alpha$ if $\mu(w_i)(\{w_{i+j}, j \geqslant 0 : w_{i+j} \models \alpha\}) \geqslant s$,
- $w_i \models \bigcirc\alpha$ if $w_{i+1} \models \alpha$,
- $w_i \models \alpha \wedge \beta$ if $w_i \models \alpha$ and $w_i \models \beta$.
- $w_i \models \alpha \mathrm{U} \beta$ if there is an integer $j \geqslant 0$ such that $w_{i+j} \models \beta$, and for every k such that $0 \leqslant k < j$, $w_{i+k} \models \alpha$.

We concern a reflexive, strong version of the until operator, i.e., if $\alpha U \beta$ holds in a time instant, β must eventually hold. In the above definition the future includes the present, so that:

- $w_i \vDash F\alpha$ if there is $j \geqslant 0$ such that $w_{i+j} \vDash \alpha$, and
- $w_i \vDash G\alpha$ if for every $j \geqslant 0$, $w_{i+j} \vDash \alpha$.

Also, the present time instant is included when the probability of formulas are considered. All the presented results then can be proved with essentially no change if we use the temporal and probabilistic operators referring to the strict future that does not concern the present.

Again, we will consider *measurable* models only, i.e., the class $LPP_{1,\mathrm{Meas}}^{LTL}$ of all LPP_1^{LTL}-models such that for every $w_i \in W$ the set $H(w_i)) = \{[\alpha]_{w_i} : \alpha \in \mathrm{For}_{LPP_1^{LTL}}\}$, where $[\alpha]_{w_i} = \{w_{i+j} : j \geqslant 0, w_{i+j} \vDash \alpha\}$.

The notions of ***satisfiable*** and ***valid*** formulas and ***satisfiable sets*** of formulas are defined as in Section 5.

8.2.2. Axiomatization. An axiomatization $Ax_{LPP_1^{LTL}}$ that characterizes the set of all $LPP_{1,\mathrm{Meas}}^{LTL}$-valid formulas extends the system Ax_{LPP_2} (having in mind that instances of the axiom schemas and rules must obey the syntactical rules for LPP_1^{LTL}) with the following axiom schemas:

(7) $\bigcirc(\alpha \to \beta) \to (\bigcirc\alpha \to \bigcirc\beta)$
(8) $\neg \bigcirc \alpha \leftrightarrow \bigcirc\neg\alpha$
(9) $\alpha U \beta \leftrightarrow \beta \vee (\alpha \wedge \bigcirc(\alpha U \beta))$
(10) $\alpha U \beta \to F\beta$
(11) $G\alpha \to P_{\geqslant 1}\alpha$

while the inference rules should be rewritten in the following form:

(1) from α and $\alpha \to \beta$ infer β
(2) from α infer $\bigcirc\alpha$
(3) from $\beta \to \bigcirc^i\alpha$ for all $i \geqslant 0$, infer $\beta \to G\alpha$
(4) from $\beta \to \bigcirc^m P_{\geqslant s-\frac{1}{k}}\alpha$, for any $m \geqslant 0$, and for every $k \geqslant \frac{1}{s}$, infer $\beta \to \bigcirc^m P_{\geqslant s}\alpha$.

The main novelty in $Ax_{LPP_1^{LTL}}$ concerns axioms about temporal reasoning (the axioms 7 and 8 are the usual axioms for the next operator $\bigcirc$, as well as the axioms 9 and 10 for the until operator) and mixing of probabilistic and temporal reasoning (Axiom 11). There are two infinitary inference rules: 3 and 4. The former one characterizes the always operator.

In this framework we can use the definitions 30 and 31 of deduction and consistency.

Note that, similarly to the probabilistic logics, compactness does not hold for LTL. For example, every finite subset of the set $\{F^n p : n \text{is a positive integer}\} \cup \{FG\neg p\}$ is satisfiable, while the set itself is not. So, the temporal part of $Ax_{LPP_1^{LTL}}$ offers possibility to prove extended completeness which cannot be proved using finitary means.

Modifications of $Ax_{LPP_1^{LTL}}$ according to ideas presented in the previous sections could produce the corresponding axiomatic systems for a first order logic for reasoning about discrete linear time and probability, a temporal probabilistic logic with probabilistic functions with a fixed finite range, etc. Also, we can specify additional relationships between the flow of time and the probability measures by adding new axioms:

Example 82. The formula $\neg\alpha \to (P_{\geqslant s}\alpha \to \bigcirc P_{\geqslant s}\alpha)$, considered as an additional axiom scheme, characterizes models with the property that if a formula does not hold in a time instant, then in the next time instant its probability will be not decreased.

8.2.3. Completeness and decidability. The proof of extended completeness again follows the ideas given in the previous sections, so we only outline the main new details.

Theorem 83 (Extended completeness theorem for $LPP^{LTL}_{1,\mathrm{Meas}}$). *A set T of formulas is $Ax_{LPP_1^{LTL}}$-consistent iff it is $LPP^{LTL}_{1,\mathrm{Meas}}$-satisfiable.*

Proof. We start with Deduction theorem. For example, assume that $T, \alpha \vdash \beta \to \mathrm{G}\beta'$ is obtained by Rule 3. Then:

(1) $T, \alpha \vdash \beta \to \bigcirc^i \beta'$, for $i \geqslant 0$,
(2) $T \vdash \alpha \to (\beta \to \bigcirc^i \beta')$, for $i \geqslant 0$, by the induction hypothesis,
(3) $T \vdash (\alpha \wedge \beta) \to \bigcirc^i \beta'$, for $i \geqslant 0$,
(4) $T \vdash (\alpha \wedge \beta) \to \mathrm{G}\beta'$, by Rule 3,
(5) $T \vdash \alpha \to (\beta \to \mathrm{G}\beta')$.

The axioms and rules imply some auxiliary statements (T denotes a consistent set of formulas):

(1) $\vdash \mathrm{G}\alpha \leftrightarrow \alpha \wedge \bigcirc \mathrm{G}\alpha$,
(2) $\vdash \mathrm{G} \bigcirc \alpha \leftrightarrow \bigcirc \mathrm{G}\alpha$,
(3) $\vdash (\bigcirc\alpha \to \bigcirc\beta) \to \bigcirc(\alpha \to \beta)$,
(4) $\vdash \bigcirc(\alpha \wedge \beta) \leftrightarrow (\bigcirc\alpha \wedge \bigcirc\beta)$,
(5) $\vdash \bigcirc(\alpha \vee \beta) \leftrightarrow (\bigcirc\alpha \vee \bigcirc\beta)$,
(6) $\mathrm{G}\alpha \vdash \bigcirc^i \alpha$ for every $i \geqslant 0$,
(7) if $\vdash \alpha$, then $\vdash \mathrm{G}\alpha$,
(8) if $T \vdash \alpha$, where T is a set of formulas, then $\bigcirc T \vdash \bigcirc\alpha$.
(9) for $j \geqslant 0$, $\bigcirc^j \beta, \bigcirc^0 \alpha, \ldots, \bigcirc^{j-1}\alpha \vdash \alpha \mathrm{U} \beta$,
(10) For any formula α, either $T \cup \{\alpha\}$ is consistent or $T \cup \{\neg\alpha\}$ is consistent.
(11) If $\neg(\alpha \to \mathrm{G}\beta) \in T$, then there is $j_0 \geqslant 0$ such that $T \cup \{\alpha \to \neg \bigcirc^{j_0} \beta\}$ is consistent.
(12) If $\neg(\alpha \to \bigcirc^m P_{\geqslant s}\beta) \in T$, then there is $j_0 > \frac{1}{s}$ such that $T \cup \{\alpha \to \neg \bigcirc^m P_{\geqslant s - \frac{1}{j_0}}\beta\}$ is consistent.

For example, the statement (9) follows in the following way. Assume $\vdash \alpha$. By application of Rule 2, we get $\vdash \bigcirc^k \alpha$, for every $k \in \omega$. We obtain $\vdash \mathrm{G}\alpha$ by Rule 3. From Axiom 11 and by application of Modus Ponens, we have $\vdash P_{\geqslant 1}\alpha$.

Then we can show that every consistent set T of formulas can be extended to a maximal consistent set. Let α_0, α_1,... be an enumeration of all formulas. A maximal consistent extension $\mathcal{T}$ of T can be obtained as follows:

(1) $T_0 = T$.
(2) For every $i \geqslant 0$ if $T_i \cup \{\alpha_i\}$ is consistent, then $T_{i+1} = T_i \cup \{\alpha_i\}$. Otherwise, if α_i is of the form $\gamma \to \mathrm{G}\beta$, then $T_{i+1} = T_i \cup \{\neg\alpha_i, \gamma \to \neg \bigcirc^{j_0} \beta\}$ for some $j_0 \geqslant 0$ such that T_{i+1} is consistent. Otherwise, α_i is of the form $\gamma \to \bigcirc^m P_{\geqslant s}\beta$, then $T_{i+1} = T_i \cup \{\neg\alpha_i, \gamma \to \neg \bigcirc^m P_{\geqslant s-\frac{1}{j_0}}\beta\}$ for some $j_0 > 0$ such that T_{i+1} is consistent. Otherwise, $T_{i+1} = T_i \cup \{\neg\alpha_i\}$.
(3) $\mathcal{T} = \bigcup_{i=0}^{\infty} T_i$.

For a maximal consistent extension $\mathcal{T}$ of a consistent set T of formulas we define the canonical model $\mathbf{M}_T = \langle W, \mathrm{Prob}, v\rangle$ such that:

- $W = w_0, w_1, \ldots, w_0 = \mathcal{T}$, and for $i > 0$, $w_i = \{\alpha : \bigcirc\alpha \in w_{i-1}\}$,
- for $i \geqslant 0$, $\mathrm{Prob}(w_i) = \langle W(w_i), H(w_i), \mu(w_i)\rangle$ is defined as follows:
  - $W(w_i) = \{w_{i+j} : j \geqslant 0\}$,
  - $H(w_i) = \{\{w_{i+j} : j \geqslant 0, \alpha \in w_{i+j}\}\}$,
  - for $\mu(w_i)(\{w_{i+j} : j \geqslant 0, \alpha \in w_{i+j}\}) = \sup_s\{P_{\geqslant s}\alpha \in w_i\}$,
- for every primitive proposition $p \in \phi$, and every $w_i \in W$, $v(w_i)(p) = \top$ iff $p \in w_i$.

First of all, we can prove that for every $i \geqslant 0$, w_i is a maximal consistent set. By hypothesis, w_0 is maximal and consistent. Suppose that w_{i+1} is not maximal. There is a formula α such that $\{\alpha, \neg\alpha\} \cap w_{i+1} = \emptyset$. Consequently, $\{\bigcirc\alpha, \bigcirc\neg\alpha\} \cap w_i = \emptyset$. We obtain that $\{\bigcirc\alpha, \neg\bigcirc\alpha\} \cap w_i = \emptyset$ which is in contradiction with the maximality of w_i. Suppose that w_{i+1} is not consistent, i.e., that $w_{i+1} \vdash \alpha \wedge \neg\alpha$. Then, $w_i \vdash \bigcirc(\alpha \wedge \neg\alpha)$, and $w_i \vdash \bigcirc\alpha \wedge \neg \bigcirc \alpha$ which is in contradiction with consistency of w_i.

Then, similarly as in the previous sections, we can show that $\mathbf{M}_T$ is an $LPP^{LTL}_{1,\mathrm{Meas}}$-model such that for all w_i and α, $\alpha \in w_i$ iff $w_i \vDash \alpha$. For example, if $\alpha = \bigcirc\beta$, we have $w_i \vDash \alpha$ iff $w_{i+1} \vDash \beta$ iff $\beta \in w_{i+1}$ iff $\alpha \in w_i$ (by the construction of w_{i+1}). □

For the previously presented logics as the first step in the proofs of their decidability we have used some kind of the filtration technique which helps as to show that every formula is satisfiable iff it is satisfiable in a finite model. The problem is that the filtration cannot be used here since the $LPP^{LTL}_{1,\mathrm{Meas}}$-models are (by their definition) infinite. However, we can show (following the ideas presented in [119]) that a formula is satisfiable if and only if it is satisfiable in an model such that the sequence of time instants of the model has a finite initial sequence of time instants followed by another finite sequence of time instants which permanently repeats and in that way forms the rest of the whole time-line. The lengths of both sequences are bounded by functions of the size of the considered formula. The full proof of decidability and complexity of the $LPP^{LTL}_{1,\mathrm{Meas}}$-satisfiability problem can be found in [91]. As it is rather long, we give only the corresponding main statements:

Theorem 84. *Every $LPP^{LTL}_{1,\mathrm{Meas}}$-satisfiable formula α is satisfiable in a model with the starting sequence of time instants, followed by the sequence of time instants*

which permanently repeats. The length of the former sequence is $\leqslant 2^{2|\alpha|}+1$, *and the length of the later sequence is* $\leqslant (2^{|\alpha|}+1)\times 2^{|\alpha|}$, *where* $|\alpha|$ *denotes the length of* α.

Theorem 85 (Decidability and complexity for LPP_1^{LTL}). *The* LPP_1^{LTL} *is decidable. The* $LPP_{1,\mathrm{Meas}}^{LTL}$*-satisfiability problem is* PSPACE*-hard and in non-deterministic exponential time.*

9. Logics with conditional probability

An important reason to consider conditional probability logics is given in [80]. It is argued there that conditional probability offers a more natural generalization of the rule "if α, then β" than probability of implication. Namely, if α has a low non-negative probability and $\neg\alpha\wedge\beta$ is very likely to happen, then "the probability of $\alpha\to\beta$" could be very high (since $\alpha\to\beta$ holds whenever α is false) and does not properly reflect the meaning of the rule, while on the other hand, "the conditional probability of β given α" is more appropriate.

Also, it turns out that a specific kind of conditional probability (with a nonarchimedean range) is useful in modelling default reasoning. We start this section by presenting a logic (denoted $LPCP_2^{S,\approx}$) which formalizes such conditional probability and represents approximate probabilistic knowledge, but in a similar way axiomatizations could be given to ordinary $[0,1]$-real-valued conditional probability. $LPCP_2^{S,\approx}$ can be seen as a generalization of the logic LPP_2^S from Section 4.3.

In the second part of the section we introduce another logic $LPCP_2^{\mathrm{Chr}}$ which axiomatizes so-called de Finetti's view of conditional probability [20]. In that approach conditional probability is seen as more primitive concept than unconditional probability, in contrast to Kolmogorov's definition where conditional probability is defined via unconditional probability. Conditional probability in the sense of de Finetti can be defined using a structure $\langle W,H,\mu\rangle$, where W is a non empty set, H is an algebra of subsets of W, and $\mu: H\times H^0\to[0,1]$, $H^0=H\setminus\{\emptyset\}$, is a (coherent) conditional probability satisfying:

- $\mu(A,A)=1$, for every $A\in H^0$,
- $\mu(\cdot,A)$ is a finitely additive probability on H for any given $A\in H^0$,
- $\mu(C\cap B,A)=\mu(B,A)\cdot\mu(C,B\cap A)$, for all $C\in H$ and A, B, $A\cap B\in H^0$.

Note that $\mu(A,B)$ has a meaning with the only condition that B is different from the impossible event.

9.1. A logic with approximate conditional probabilities. In this subsection, we use notions defined in Section 3, and only emphasize the main novelties. Let S be the unit interval of the Hardy field $Q[\epsilon]$. $Q[\epsilon]$ is a recursive nonarchimedean field which contains all rational functions of a fixed positive infinitesimal ϵ which belongs to a nonstandard elementary extension ${}^*R$ of the standard real numbers [56, 117]. An element ϵ of ${}^*R$ is an infinitesimal if $|\epsilon|<\frac{1}{n}$ for every natural number n. $Q[\epsilon]$ contains all rational numbers. Let $Q[0,1]$ denote the set of rational numbers from $[0,1]$.

The language of $LPCP_2^{S,\approx}$, beside the set ϕ of primitive propositions and Boolean connectives $\neg$ and $\wedge$, contains binary probabilistic operators:

$$(CP_{\leqslant s})_{s\in S}, \quad (CP_{\geqslant s})_{s\in S}, \quad \text{and} \quad (CP_{\approx r})_{r\in Q[0,1]}.$$

If $\alpha,\beta \in \mathrm{For}_C$ and $s \in S$, $r \in Q[0,1]$ then $CP_{\geqslant s}(\alpha,\beta)$, $CP_{\leqslant s}(\alpha,\beta)$ and $CP_{\approx r}(\alpha,\beta)$ are *basic probability formulas* with the intended meaning "the conditional probability of α given β is at least (at most) s and, approximately r". The set For_P of probabilistic propositional formulas is the smallest set containing all basic probability formulas and closed under Boolean formation rules (so, there are no iterations of the probabilistic operators). The set of formulas $\mathrm{For}_{LPCP_2^{S,\approx}}$ is $\mathrm{For}_C \cup \mathrm{For}_P$. Also:

- $CP_{<s}(\alpha,\beta)$ denotes $\neg CP_{\geqslant s}(\alpha,\beta)$ for $\alpha,\beta \in \mathrm{For}_C$, $s \in S$,
- $CP_{>s}(\alpha,\beta)$ denotes $\neg CP_{\leqslant s}(\alpha,\beta)$ for $\alpha,\beta \in \mathrm{For}_C$, $s \in S$,
- $CP_{=s}(\alpha,\beta)$ denotes $CP_{\geqslant s}(\alpha,\beta) \wedge CP_{\leqslant s}(\alpha,\beta)$ for $\alpha,\beta \in \mathrm{For}_C$, $s \in S$ and
- $P_{\rho s}\alpha$ denotes $CP_{\rho s}(\alpha,\top)$ for $\alpha \in \mathrm{For}_C$ and $\rho \in \{\geqslant,\leqslant,>,<,=,\approx\}$.

It should be noted that $CP_{\geqslant}$ and $CP_{\leqslant}$ are not interdefinable since the appropriate equivalence breaks down when the probability of the condition is 0.

9.1.1. Semantics. We consider the class $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$ of all measurable neat $LPCP_2^{S,\approx}$-models, which can be defined in the same way as the class $LPP_{2,\mathrm{Meas,Neat}}$ from Section 3.2, with the important difference that:

- μ is an S-valued finitely additive measure, i.e., $\mu : H \to S$.

The neatness condition is used to make our models a subclass of ${}^*R$-probabilistic models of [61, 66]. This facilitates the explanation of a possible application of $LPCP_2^{S,\approx}$ to default reasoning. All the results can be also proved for the class of measurable (but not necessarily neat) $LPCP_2^{S,\approx}$-models.

Definition 86. *The satisfiability relation* $\models \subseteq LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}} \times \mathrm{For}_{LPCP_2^{S,\approx}}$ fulfills the following conditions for every $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-model $\mathbf{M} = \langle W, H, \mu, v\rangle$:

(1) if $\alpha \in \mathrm{For}_C$, $\mathbf{M} \models \alpha$ if $(\forall w \in W)v(w)(\alpha) = \mathrm{true}$,
(2) $\mathbf{M} \models CP_{\leqslant s}(\alpha,\beta)$ if either $\mu([\beta]_{\mathbf{M}}) = 0$ and $s = 1$ or $\mu([\beta]_{\mathbf{M}}) > 0$ and $\frac{\mu([\alpha\wedge\beta]_{\mathbf{M}})}{\mu([\beta]_{\mathbf{M}})} \leqslant s$,
(3) $\mathbf{M} \models CP_{\geqslant s}(\alpha,\beta)$ if either $\mu([\beta]_{\mathbf{M}}) = 0$ or $\mu([\beta]_{\mathbf{M}}) > 0$ and $\frac{\mu([\alpha\wedge\beta]_{\mathbf{M}})}{\mu([\beta]_{\mathbf{M}})} \geqslant s$,
(4) $\mathbf{M} \models CP_{\approx r}(\alpha,\beta)$ if either $\mu([\beta]_{\mathbf{M}}) = 0$ and $r = 1$ or $\mu([\beta]_{\mathbf{M}}) > 0$ and for every positive integer n, $\frac{\mu([\alpha\wedge\beta]_{\mathbf{M}})}{\mu([\beta]_{\mathbf{M}})} \in [\max\{0, r - 1/n\}, \min\{1, r + 1/n\}]$.
(5) if $A \in \mathrm{For}_P^S$, $\mathbf{M} \models \neg A$ if $M \not\models A$,
(6) if $A, B \in \mathrm{For}_P^S$, $\mathbf{M} \models A \wedge B$ if $\mathbf{M} \models A$ and $\mathbf{M} \models B$.

Condition 3 is formulated on the useful assumption that the conditional probability is by default 1, whenever the condition has the probability 0. Also, note that the condition 4 is equivalent to saying that the conditional probability equals $r - \epsilon_i$ (or $r + \epsilon_i$) for some infinitesimal $\epsilon_i \in S$. It is easy to see that the defined operators will behave as expected, e.g., $\mathbf{M} \models P_{<s}\alpha$ iff $\mu([\alpha]_{\mathbf{M}}) < s$.

9.1.2. Axiomatization and completeness. The axiomatic system $Ax_{LPCP_2^{S,\approx}}$ which characterizes the set of all $LPCP^{S,\approx}_{2,\mathrm{Meas},\mathrm{Neat}}$-valid formulas contains the following set of axiom schemata:

(1) all For_C-instances of classical propositional tautologies
(2) all For_P-instances of classical propositional tautologies
(3) $CP_{\geqslant 0}(\alpha,\beta)$
(4) $CP_{\leqslant s}(\alpha,\beta) \to CP_{<t}(\alpha,\beta)$, $t > s$
(5) $CP_{<s}(\alpha,\beta) \to CP_{\leqslant s}(\alpha,\beta)$
(6) $P_{\geqslant 1}(\alpha \leftrightarrow \beta) \to (P_{=s}\alpha \to P_{=s}\beta)$
(7) $P_{\leqslant s}\alpha \leftrightarrow P_{\geqslant 1-s}\neg\alpha$
(8) $(P_{=s}\alpha \wedge P_{=t}\beta \wedge P_{\geqslant 1}\neg(\alpha \wedge \beta)) \to P_{=\min(1,s+t)}(\alpha \vee \beta)$
(9) $P_{=0}\beta \to CP_{=1}(\alpha,\beta)$
(10) $(P_{=t}\beta \wedge P_{=s}(\alpha \wedge \beta)) \to CP_{=s/t}(\alpha,\beta)$, $t \neq 0$
(11) $CP_{\approx r}(\alpha,\beta) \to CP_{\geqslant r_1}(\alpha,\beta)$, for every rational $r_1 \in [0,r)$
(12) $CP_{\approx r}(\alpha,\beta) \to CP_{\leqslant r_1}(\alpha,\beta)$, for every rational $r_1 \in (r,1]$

and inference rules:

(1) From φ and $\varphi \to \psi$ infer ψ.
(2) If $\alpha \in \mathrm{For}_C$, from α infer $P_{\geqslant 1}\alpha$.
(3) From $A \to P_{\neq s}\alpha$, for every $s \in S$, infer $A \to \bot$.
(4) For every $r \in Q[0,1]$, from $A \to CP_{\geqslant r-1/n}(\alpha,\beta)$, for every integer $n \geqslant 1/r$, and $A \to CP_{\leqslant r+1/n}(\alpha,\beta)$ for every integer $n \geqslant 1/(1-r)$, infer $A \to CP_{\approx r}(\alpha,\beta)$.

It is easy to see (just put $\top$ instead of β) that the axioms 3–5 generalize the corresponding axioms from the system Ax_{LPP_2}. Axiom 9 conforms with the useful practice of assuming conditional probability to be 1, whenever the condition has the probability 0. Axiom 10 expresses the standard definition of conditional probability, while the axioms 11 and 12 and Rule 4 describe the relationship between the standard conditional probability and the conditional probability infinitesimally close to some rational $r \in Q[0,1]$. The rules 3 and 4 are infinitary. Rule 3 guarantees that the probability of a formula belongs to the set S.

A useful, but straightforward theorem is:

Theorem 87. *Let $\alpha,\beta \in \mathrm{For}_C$. Then:*

(1) $\vdash CP_{\geqslant t}(\alpha,\beta) \to CP_{\geqslant s}(\alpha,\beta)$, $t > s$
(2) $\vdash CP_{\leqslant t}(\alpha,\beta) \to CP_{\leqslant s}(\alpha,\beta)$, $t < s$
(3) $\vdash CP_{=t}(\alpha,\beta) \to \neg CP_{=s}(\alpha,\beta)$, $t \neq s$
(4) $\vdash CP_{=t}(\alpha,\beta) \to \neg CP_{\geqslant s}(\alpha,\beta)$, $t < s$
(5) $\vdash CP_{=t}(\alpha,\beta) \to \neg CP_{\leqslant s}(\alpha,\beta)$, $t > s$
(6) $\vdash CP_{=r}(\alpha,\beta) \to CP_{\approx r}(\alpha,\beta)$, $r \in Q[0,1]$.
(7) $\vdash CP_{\approx r_1}(\alpha,\beta) \to \neg CP_{\approx r_2}(\alpha,\beta)$, *for* $r_1,r_2 \in Q[0,1]$, $r_1 \neq r_2$.
(8) $\vdash P_{=0}\beta \to \neg CP_{\leqslant s}(\alpha,\beta)$, *for* $s < 1$.
(9) $\vdash P_{\leqslant 1}\alpha$.

Note that, by restricting β to $\top$, we obtain analogous statements for unconditional probabilities.

The main novelty in the completeness proof follows concerns the construction of a maximal consistent extensions of a consistent set. Following notations from Theorem 13, now the construction is:

(1) $T_0 = T \cup Cn_C(T) \cup \{P_{\geqslant 1}\alpha : \alpha \in Cn_C(T)\}$
(2) for every $i \geqslant 0$,
  (a) if $T_{2i} \cup \{A_i\}$ is consistent, then $T_{2i+1} = T_{2i} \cup \{A_i\}$;
  (b) otherwise, if $T_{2i} \cup \{A_i\}$ is not consistent, we have:
    (i) if A_i is of the form $A \to CP_{\approx r}(\alpha, \beta)$, then $T_{2i+1} = T_{2i} \cup \{\neg A_i, A \to \neg CP_{\geqslant r-1/n}(\alpha, \beta)\}$, or $T_{2i+1} = T_{2i} \cup \{\neg A_i, A \to \neg CP_{\leqslant r+1/n}(\alpha, \beta)\}$, for some integer n, where n is chosen such that T_{2i+1} is consistent (we prove that this is possible below);
    (ii) otherwise, $T_{2i+1} = T_{2i} \cup \{\neg A_i\}$,
(3) for every $i \geqslant 0$, $T_{2i+2} = T_{2i+1} \cup \{P_{=s}\alpha_i\}$, where s is chosen to be an arbitrary element of S such that T_{2i+2} is consistent (we prove that this is possible below),
(4) for every $i \geqslant 0$, if T_i is enlarged by a formula of the form $P_{=0}\alpha$, add $\neg\alpha$ to $T_i \cup \{P_{=0}\alpha\}$ as well.
(5) $\mathcal{T} = \bigcup_{i=0}^{\infty} T_i$.

Let us consider the step (3) of the construction, and suppose that for every $s \in S$, $T_{2i+1} \cup \{P_{=s}\alpha_i\}$ is not consistent. Let $T_{2i+1} = T_0 \cup T^+_{2i+1}$, where T^+_{2i+1} denotes the set of all formulas $B \in \mathrm{For}_P$ that were added to T_0 in the previous steps of the construction. Then the following contradicts consistency of T_{2i+1}:

(1) $T_0, T^+_{2i+1}, P_{=s}\alpha_i \vdash \bot$, for every $s \in S$, by the hypothesis
(2) $T_0, T^+_{2i+1} \vdash \neg P_{=s}\alpha_i$, for every $s \in S$, by Deduction theorem
(3) $T_0 \vdash (\bigwedge_{B \in T^+_{2i+1}} B) \to \neg P_{=s}\alpha_i$, for every $s \in S$, by Deduction theorem
(4) $T_0 \vdash (\bigwedge_{B \in T^+_{2i+1}} B) \to \bot$, by Rule 3
(5) $T_{2i+1} \vdash \bot$,

The set $\mathcal{T}$ satisfies:

(1) There is exactly one $s \in S$ such that $P_{=s}\alpha \in \mathcal{T}$.
(2) There is exactly one $s \in S$ such that $CP_{=s}(\alpha, \beta) \in \mathcal{T}$.
(3) If $CP_{\geqslant s}(\alpha, \beta) \in \mathcal{T}$, there is $r \in S$ such that $r \geqslant s$ and $CP_{=r}(\alpha, \beta) \in \mathcal{T}$.
(4) If $CP_{\leqslant s}(\alpha, \beta) \in \mathcal{T}$, there is $r \in S$ such that $r \leqslant s$ and $CP_{=r}(\alpha, \beta) \in \mathcal{T}$.
(5) If $CP_{\approx r_1}(\alpha, \beta) \in \mathcal{T}$ and $r_2 \in Q[0,1] \setminus \{r_1\}$, then $CP_{\approx r_2}(\alpha, \beta) \notin \mathcal{T}$

As an example, let us consider the statement (1). According to Theorem 87 (3), if $P_{=s}\alpha \in \mathcal{T}$, then for every $t \neq s$, $P_{=t}\alpha \notin \mathcal{T}$. On the other hand, if for every $s \in S$, $\neg P_{=s}\alpha \in \mathcal{T}$, then $\mathcal{T} \vdash \neg P_{=s}\alpha$ for every $s \in S$. By Rule 3, $\mathcal{T} \vdash \bot$ which contradicts consistency of $\mathcal{T}$. Thus, for every $\alpha \in \mathrm{For}_C$, there is exactly one $s \in S$ such that $P_{=s}\alpha \in \mathcal{T}$. Finally, the corresponding canonical model $\mathbf{M}_{\mathcal{T}}$ can be defined as in Section 3.4, and we have:

Theorem 88 (Extended completeness theorem for $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$). *A set T of formulas is $Ax_{LPCP_2^{S,\approx}}$-consistent iff it is $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-satisfiable.*

9.1.3. Decidability. The proof of decidability of $LPCP_2^{S,\approx}$ [114] is rather long, and, similarly as in Section 8.2, we will omit it here. The proof contains a reduction of the the $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-satisfiability to linear programming problem, as in Section 3.5. However, note that Section 3.5 deals with the standard real-valued probabilities, while in $LPCP_2^{S,\approx}$ the range of probabilities is recursive and contains non-standard values, and there are operators of the form $CP_{\approx r}$ that do not appear above. Thus, in the reduction we have to eliminate the $CP_{\approx .}$-operators and to try to solve linear systems in an extension of $Q[\epsilon]$. The next example contains an illustration of the technique from [114].

Example 89. Let us consider the formula $A = C \wedge ((D \vee B) \rightarrow (D \wedge B))$, where B, C and D denote $CP_{\approx 0}(q, \top)$, $CP_{\approx 1}(\neg p \wedge \neg q, \neg q)$ and $CP_{\approx 0.4}(p \wedge q, q)$, respectively. The set of atoms, $At(A)$, contains $a_1 = p \wedge q$, $a_2 = p \wedge \neg q$, $a_3 = \neg p \wedge q$ and $a_4 = \neg p \wedge \neg q$. Let x_i denote the measure of atom a_i. The formula A is equivalent to $(B \wedge C \wedge D) \vee (\neg B \wedge C \wedge \neg D)$. We start with the first conjunct $B \wedge C \wedge D$ and suppose that the measures of q and $\neg q$ are greater than zero, i.e., that $x_1 + x_3 > 0$, and $x_2 + x_4 > 0$. $B \wedge C \wedge D$ is satisfiable iff the same holds for the following system:

$$\begin{gathered} x_1 + x_2 + x_3 + x_4 = 1, \quad x_i \geqslant 0 \text{ for } i = 1,4 \\ x_1 + x_3 > 0 \quad x_2 + x_4 > 0 \\ x_1 + x_3 \approx 0 \\ x_2/(x_2 + x_4) \approx 1 \\ x_1/(x_1 + x_3) \approx 0.4 \end{gathered}$$

which is equivalent to

$$\begin{gathered} x_1 + x_2 + x_3 + x_4 = 1, \quad x_i \geqslant 0 \text{ for } i = 1,4 \\ x_1 + x_3 > 0 \quad x_2 + x_4 > 0 \\ 0 < x_1 + x_3 < n_1\epsilon \\ x_4/(x_2 + x_4) < 1/n_2 \\ 0.4 - n_3\epsilon < x_1/(x_1 + x_3) < 0.4 + n_3\epsilon \end{gathered}$$

for some n_1, n_2, $n_3 \in N$. If we replace n_1, n_2, n_3 by their maximum denoted by n, we obtain an equivalent system. Since $\approx$ does not appear in the last system, Fourier-Motzkin elimination can be performed in the standard way. The procedure finishes with the true condition $(1 - n\epsilon)/n < 1$ which means that the considered formula is satisfiable.

9.1.4. Modelling default reasoning. The central notion in the field of default reasoning is the notion of default rules. A default rule, which can be seen as a sentence of the form "if α, then generally β", can be written as[5] $\alpha \rightarrowtail \beta$. A default base Δ is a set of default rules. Default reasoning is described in terms of the corresponding consequence relation $\mid\!\sim$, i.e., we are interested in determining the set

[5]Note that the other authors use different symbols ($\rightarrow$, $\mid\!\sim$, for example) to denote the "default implication". In the present setting those symbols may cause confusion, so we prefer to introduce a new symbol here.

of defaults that are the consequences of a default base. Then, if α is a description of our knowledge and $\Delta|\sim \alpha \rightarrowtail \beta$, we (plausibly) conclude that β is the case. There are a number of papers which describe $\vdash\!\!\sim$ in terms of classes of models and the corresponding satisfiability relations $\models$ such that $\Delta\vdash\!\!\sim \alpha \rightarrowtail \beta$ if for every model M satisfying Δ, $M \models \alpha \rightarrowtail \beta$. In [61, 66] a set of properties which form a core of default reasoning, called the system P, and the corresponding deduction relation $\vdash_P$ were proposed. The system P is based on the following axiom and rules ($\models$ denotes classical validity):

- $\alpha \rightarrowtail \alpha$ (Reflexivity)
- from $\models \alpha \leftrightarrow \alpha'$ and $\alpha \rightarrowtail \beta$, infer $\alpha' \rightarrowtail \beta$ (Left logical equivalence)
- from $\alpha \rightarrowtail \beta$ and $\alpha \rightarrowtail \gamma$, infer $\alpha \rightarrowtail \beta \wedge \gamma$ (And)
- from $\alpha \rightarrowtail \gamma$ and $\beta \rightarrowtail \gamma$, infer $\alpha \vee \beta \rightarrowtail \gamma$ (Or)
- from $\alpha \rightarrowtail \beta$ and $\alpha \rightarrowtail \gamma$, infer $\alpha \wedge \beta \rightarrowtail \gamma$ (Cautious monotonicity).

Then, for a default base Δ, $\Delta \vdash_P \alpha \rightarrowtail \beta$ if $\alpha \rightarrowtail \beta$ is deducible from Δ using the above axiom and rules. Default consequence relation was also described in terms of preferential models, and it was proved that the system P is sound and complete with respect to the class of all such models:

Theorem 90. [61, Theorem 5.18] $\Delta|\sim \alpha \rightarrowtail \beta$ *with respect to the class of all preferential models if and only if* $\Delta \vdash_P \alpha \rightarrowtail \beta$.

The same holds for a special proper subclass of the class of preferential models, the so-called rational models, also considered in [66]. These two classes are not distinguishable using the language of defaults. It turns out that many other approaches to default reasoning are characterized by P. For example, in [66] a family of nonstandard ($^*R$) probabilistic models characterizing $\vdash_P$ was proposed. An $^*R$-probabilistic model can be defined in a similar way as $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-models, with the exception that $\mu : H \rightarrow R^*$. A default $\alpha \rightarrowtail \beta$ holds in an $^*R$-probabilistic model if either the probability of α is 0 or the conditional probability of β given α is infinitesimally close to 1.

We can use $CP_{\approx 1}(\beta, \alpha)$ to syntactically represent the default $\alpha \rightarrowtail \beta$. In the sequel, we will use $\alpha \rightarrowtail \beta$ both in the original context of the system P and to denote the corresponding translation $CP_{\approx 1}(\beta, \alpha)$. In the case of a finite default base our approach produces the same result as the other mentioned approaches, namely it is equivalent to P.

Theorem 91. *For every finite default base* Δ *and for every default* $\alpha \rightarrowtail \beta$

$$\Delta \vdash_P \alpha \rightarrowtail \beta \text{ iff } \Delta \vdash_{Ax_{LPCP_2^{S,\approx}}} \alpha \rightarrowtail \beta.$$

Theorem 91 cannot be generalized to an arbitrary default base Δ, as it is illustrated by the following example.

Example 92. It is proved in [66, Lemma 2.7] that the infinite set of defaults $T = \{p_i \rightarrowtail p_{i+1}, p_{i+1} \rightarrowtail \neg p_i\}$, where p_i's are propositional letters for every integer $i \geqslant 0$, has only non well-founded preferential models (a preferential model containing an infinite descending chain of states) in which $p_0 \not\rightarrowtail \bot$, i.e., p_0 is consistent. It

means that $T \nvdash_P p_0 \rightarrowtail \bot$. On the other hand, $T \vdash_{Ax_{LPCP_2^{S,\approx}}} p_0 \rightarrowtail \bot$ since the following holds. Let an $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-model $\mathbf{M} = \langle W, H, \mu, v\rangle$ satisfy the set T. If $\mu([p_i]) = 0$, for some $i > 0$, then it must be $\mu([p_0]) = 0$, and $\mathbf{M} \models p_0 \rightarrowtail \bot$. Thus, suppose that $\mu([p_i]) \neq 0$, for every $i > 0$. Then, for every $i \geqslant 0$: $\frac{\mu([p_i \wedge p_{i+1}])}{\mu([p_i])} \approx 1$ and $\frac{\mu([\neg p_i \wedge p_{i+1}])}{\mu([p_{i+1}])} \approx 1$, i.e., $\frac{\mu([p_i \wedge p_{i+1}])}{\mu([p_i])} = 1 - \epsilon_1$ and $\frac{\mu([\neg p_i \wedge p_{i+1}])}{\mu([p_{i+1}])} = 1 - \epsilon_2$, for some infinitesimals ϵ_1 and ϵ_2. A simple calculation shows that which means that $\mu([p_i]) \leqslant \epsilon_0 \mu([p_{i+1}])$ for some infinitesimal ϵ_0. Since, for some c and k, $\epsilon_0 \leqslant c\epsilon^k$, it follows that for every $i > 0$, $0 \leqslant \mu([p_0]) \leqslant \epsilon^i$. Since $\mu([p_0]) \in S$ and there is no positive element of S with such property, it follows that $\mu([p_0]) = 0$, $[p_0] = \emptyset$ and $M \models p_0 \rightarrowtail \bot$. Since $\mathbf{M}$ is an arbitrary $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-model, $T \vdash_{LPCP_2^{S,\approx}} p_0 \rightarrowtail \bot$.

Note that the above proof of $\mu([p_0]) = 0$, does not hold in the case when the range of the probability is the unit interval of ${}^*R$ because ${}^*R$ is ω_1-saturated (which means that the intersection of any countable decreasing sequence of nonempty internal sets must be nonempty). As a consequence, thanks to the restricted ranges of probabilities that are allowed in $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-class of models, our system goes beyond the system P, when we consider infinite default bases.

$LPCP_2^{S,\approx}$ is rich enough not only to express formulas that represents defaults but also to describe more: probabilities of formulas, negations of defaults, combinations of defaults with the other (probabilistic) formulas etc. Let us now consider some situations where these possibilities allow us to obtain more conclusions than in the framework of the language of defaults.

Example 93. The translation of rational monotonicity, $((\alpha \rightarrowtail \beta) \wedge \neg(\alpha \rightarrowtail \neg\gamma)) \rightarrow ((\alpha \wedge \gamma) \rightarrowtail \beta)$, is $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-valid since rational monotonicity is satisfied in every ${}^*R$-probabilistic model, and $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$ is a subclass of that class of models. The same holds for the formula $\neg(\mathrm{true} \rightarrowtail \mathrm{false})$ corresponding to another property called normality in [31].

Note that in this example we use negated defaults that are not expressible in P.

Example 94. Let the default base consist of the following two defaults $s \rightarrowtail b$ and $s \rightarrowtail t$, where s, b and t means Swedes, blond and tall, respectively [6]. Because of the inheritance blocking problem, in some systems (for example in P) it is not possible to conclude that Swedes who are not tall are blond $((s \wedge \neg t) \rightarrowtail b)$. Since our system and P coincide if the default base is finite, the same holds in our framework. In fact, there are some $LPCP^{S,\approx}_{2,\mathrm{Meas,Neat}}$-models in which the previous formula is not satisfied. Avoiding a discussion of intuitive acceptability of the above conclusion, we point out that by adding some additional assumptions ($CP_{=1-\epsilon}(t, s)$ and $CP_{=1-\epsilon^2}(b, s)$) to the default base we can entail that conclusion too. First, note that the assumptions are compatible with defaults $s \rightarrowtail t$ and $s \rightarrowtail b$. Then, an easy calculation shows that $\frac{P(s \wedge \neg t)}{P(s)} = \frac{P(s) - P(s \wedge t)}{P(s)} = \frac{P(s) - P(s) + P(s)\epsilon}{P(s)} = \epsilon$, and similarly

$\frac{P(s\wedge\neg b)}{P(s)} = \epsilon^2$. Finally, we can estimate the conditional probability of b given $s\wedge\neg t$:

$$\frac{P(s\wedge\neg t\wedge b)}{P(s\wedge\neg t)} = \frac{P(s\wedge\neg t) - P(s\wedge\neg t\wedge\neg b)}{P(s\wedge\neg t)} \geqslant \frac{\epsilon P(s) - \epsilon^2 P(s)}{\epsilon P(s)} = 1-\epsilon.$$

It follows that $(s\wedge\neg t) \rightarrowtail b$.

9.2. A logic with coherent conditional probabilities. In this subsection S again denotes the unit interval of rational numbers. The set For_P contains *basic probability formulas* of the form $CP_{\geqslant s}(\alpha,\beta)$ and their boolean combinations.

Definition 95. An $LPCP_2^{\mathrm{Chr}}$*-model* is a structure $\mathbf{M} = \langle W, H, \mu, v\rangle$ where:

- W is a nonempty set,
- H is an algebra of subsets of W, $H^0 = H \setminus \{\emptyset\}$
- $\mu : H \times H^0 \to [0,1]$, is a conditional probability satisfying
- $\mu(A,A) = 1$, for every $A \in H^0$,
  - $\mu(\cdot, A)$ is a finitely additive probability on H for any given $A \in H^0$, and
  - $\mu(C\cap B, A) = \mu(B,A)\cdot\mu(C, B\cap A)$, for all $C \in H$ and $A, B, A\cap B \in H^0$.
- $v : W \times \phi \to \{\mathrm{true}, \mathrm{false}\}$ provides for each world $w \in W$ a two-valued evaluation of the primitive proposition.

Let $LPCP^{\mathrm{Chr}}_{2,\mathrm{Meas}}$ denote the class of all measurable $LPCP_2^{\mathrm{Chr}}$-models.

The axiomatic system $Ax_{LPCP_2^{\mathrm{Chr}}}$ which characterizes the set of all $LPCP^{\mathrm{Chr}}_{2,\mathrm{Meas}}$-valid formulas contains the following axiom schemata:

(1) all instances of the classical propositional tautologies,
(2) $CP_{\geqslant 0}(\alpha,\beta)$,
(3) $CP_{\leqslant r}(\alpha,\beta) \to CP_{<s}(\alpha,\beta)$, $s > r$,
(4) $CP_{<s}(\alpha,\beta) \to CP_{\leqslant s}(\alpha,\beta)$,
(5) $(CP_{\geqslant r}(\alpha,\gamma) \wedge CP_{\geqslant s}(\beta,\gamma) \wedge CP_{\geqslant 1}(\neg\alpha\vee\neg\beta,\gamma)) \to CP_{\geqslant \min\{1,r+s\}}(\alpha\vee\beta,\gamma)$,
(6) $(CP_{\leqslant r}(\alpha,\gamma) \wedge CP_{<s}(\beta,\gamma)) \to CP_{<r+s}(\alpha\vee\beta,\gamma)$, $r+s<1$,
(7) $CP_{\geqslant s}(\alpha,\gamma) \wedge CP_{\geqslant r}(\beta,\alpha\wedge\gamma) \to CP_{\geqslant s\cdot r}(\alpha\wedge\beta,\gamma)$,
(8) $CP_{\geqslant 1}(\beta,\gamma) \wedge CP_{\geqslant s}(\alpha\wedge\beta,\gamma) \to CP_{\geqslant s}(\alpha,\beta\wedge\gamma)$,

and inference rules:

(1) from α and $\alpha\to\beta$ infer β,
(2) from $\alpha\to\beta$ infer $CP_{\geqslant 1}(\beta,\alpha)$,
(3) from $A \to (CP_{\geqslant t}(\beta,\gamma) \to CP_{\geqslant s\cdot t}(\alpha\wedge\beta,\gamma))$, for every rational number t from $(0,1)$, infer $A \to CP_{\geqslant s}(\alpha,\gamma\wedge\beta)$.

Note that Rule 3 is the only infinitary rule in $Ax_{LPCP_2^{\mathrm{Chr}}}$. It corresponds to the last part in the definition of conditional probability. The proof of

Theorem 96 (Extended completeness theorem for $LPCP^{\mathrm{Chr}}_{2,\mathrm{Meas}}$). *A set T of formulas is $Ax_{LPCP_2^{\mathrm{Chr}}}$-consistent iff it is $LPCP^{\mathrm{Chr}}_{2,\mathrm{Meas}}$-satisfiable.*

follows the main steps from the previous sections, while by reducing the $LPCP^{\mathrm{Chr}}_{2,\mathrm{Meas}}$ satisfiability problem to the problem of checking coherence of conditional probability assessments which is decidable [20], we have that

Theorem 97. *The logic $LPCP_2^{\mathrm{Chr}}$ is decidable.*

The next example contains two formulas that illustrate some peculiarities of $LPCP_2^{\mathrm{Chr}}$.

Example 98. The formula $A = CP_{=0}(\alpha, \beta) \to CP_{>0}(\beta, \top)$ is not $LPCP_2^{\mathrm{Chr}}$-valid. Let us consider the following $LPCP_{2,\mathrm{Meas}}^{\mathrm{Chr}}$-model $\mathbf{M}$:

- $W = \{w_1, w_2\}$,
- $H = 2^W$,
- $\mu(\{w_1\}, W) = 0$, $\mu(\{w_2\}, W) = 1$, $\mu(\{w_1\}, \{w_1\}) = 1$, $\mu(\{w_2\}, \{w_1\}) = 0$, $\mu(\{w_1\}, \{w_2\}) = 0$, $\mu(\{w_2\}, \{w_2\}) = 1$,
- $v(w_1, p) = v(w_2, p) = v(w_1, q) = \text{true}$, $v(w_2, q) = \text{false}$.

In this model $\mu([p], [q]) = \mu(\{w_1, w_2\}, \{w_1\}) = 1$, $\mu([\neg p], [q]) = \mu(\emptyset, \{w_1\}) = 0$, and $\mu([q], [\top]) = \mu(\{w_1\}, W) = 0$. It means that $\mathbf{M} \models CP_{=0}(\neg p, q) \wedge CP_{=0}(q, \top)$, and A is not $LPCP_{2,\mathrm{Meas}}^{\mathrm{Chr}}$-valid.

The formula $B = CP_{=0}(\beta, \top) \wedge CP_{\geqslant \frac{1}{2}}(\alpha, \beta) \to CP_{\leqslant \frac{1}{2}}(\neg\alpha, \beta)$ is $LPCP_2^{\mathrm{Chr}}$-valid since $\mu([\cdot], [\beta])$ is finitely additive probability measure.

Note that both formulas from Example 98 have the opposite behavior when we use the Kolmogorov's approach to conditional probability (with the very often and useful assumption that the conditional probability of α given β is 1, if the probability of β is 0), i.e., A is valid, while B is not.

10. Related work

As we mentioned in Section 2, a lot of recent interest in probability logic was initiated by [79] in which Nillson gave a procedure for probabilistic entailment which, given probabilities of premises, could calculate bounds on probabilities of the derived sentences. The Nillson's approach was semantic and stimulated some authors to provide axiomatizations and decision procedures for the logic. In the same year Gaifman published a paper [35] which studied higher order probabilities and connections with modal logics.

In [27] Fagin, Halpern and Megiddo presented a propositional logic with real-valued probabilities in which higher level probabilities were not allowed (the logic was similar to LPP_2). The language of that logic allowed basic probabilistic formulas of the form $a_1 w(\alpha_1) + \cdots + a_n w(\alpha_n) \geqslant s$, where a_i's and s are rational numbers, α_i's classical propositional formulas, and $w(\alpha_i)$'s denote probabilities of α_i's. Probabilistic formulas are boolean combinations of basic probabilistic formulas. The corresponding class of models was $LPP_{2,\mathrm{Meas}}$. A finitary axiomatic system for the logic was given. Since the compactness theorem does not hold for their logic, the authors were able to prove only the simple completeness. As we mentioned above, the paper contains a proof of decidability and complexity of the logic. Models that are not measurable were also considered there. Dropping the measurability requirement made things more complicated. In that case inner and outer measures should be used since the finite additivity does not hold for the considered models. Finally, conditional probabilities were also discussed. To obtain a complete axiomatization, the authors used the machinery of the theory of

real closed fields. We note that our syntax can be extended in a straightforward manner, such that the set of well formed formulas and the related results from [27] can be exactly obtained. The papers [28, 44] of the same authors introduced a probabilistic extension of the modal logic of knowledge which is similar to LPP_1.

The papers [30, 125] presented logics with probability functions that have a fixed finite range, similar to the logic $LPP_2^{\mathrm{Fr}(n)}$.

Frisch and Haddawy presented in [32] an incomplete iteration procedure which computes increasingly narrow probability intervals. The procedure can be stopped at any time yielding partial information about the probability of sentences, and allowing one to make a tradeoff between precision and computational time. Computational aspects of probabilistic logics were also discussed in [36]. The paper [52] showed that it is possible to apply a very efficient numerical method of column generation to solve the $LPP_{2,\mathrm{Meas}}$-satisfiability problem.

First order probability logics were discussed in [1, 43]. It was shown that the set of valid formulas of the considered logic (which was similar to $LFOP_1$) is not recursively enumerable. Thus, no finitary axiomatization is possible.

In [38] a propositional logic which can be used for reasoning about probabilistic processes was presented. Besides all differences between our logic and that one, in [38] an idea to prove completeness using an infinitary rule was used similarly as in our approach.

A rule similar to Rule 3 from the axiomatic system $Ax_{LPCP_2^{s,\approx}}$ was given in [3] by Alechina. The main difference is that her rule was restricted to rationals only.

A sound first order axiomatization (which is not complete) for a logic which formalized probabilistic temporal reasoning was given in [39]. This system differs from our LPP_1^{LTL} since time intervals and a branching structure of time were considered there.

In [16, 17] Boričić and Rašković extended Heyting propositional logic by probabilistic operators. Since predicates "at lest r" and "at most r" are not mutually expressible in that context, both types of operators $P_{\geqslant r}$ and $P_{\leqslant r}$ were present in the corresponding language. Marchioni and Godo presented in [73] a modal fuzzy logic approach to model probabilistic reasoning in the sense of De Finetti. Also, in that logic, Łukasiewicz implication can be used to express comparative statements. Conditional probabilities were combined with default reasoning in a semantically based approach in [2, 70].

Uncertain reasoning is also interesting in the framework of economy. For example, an axiomatization for so-called type spaces (a notion that plays the role of probabilistic models in our paper) within the framework of probabilistic logic was given in [45]. The proposed axiomatization was simply complete with respect to the introduced semantics. A strongly complete infinitary axiomatization for type spaces is given in [77]. The main difference between that system and our approach is that infinitary formulas are allowed in [77]. As a consequence, that logic is undecidable, due to the cardinality argument.

Finally, for more comprehensive list of the papers on probability logics the reader could consult [94].

References

[1] M. Abadi and J. Y. Halpern, *Decidability and expressiveness for first-order logics of probability*, Inform. Comput. 112 (1994), 1–36.

[2] E. W. Adams, *The logic of Conditional*, Dordrecht: Reidel, 1975.

[3] N. Alechina, *Logic with probabilistic operators*; in *Proc. ACCOLADE '94*, 121–138, 1995.

[4] K. J. Barwise, *Admissible sets and structures*, Springer-Verlag, Berlin, 1975.

[5] T. Bayes, *An essay towards solving a problem in the doctrine of chances*, London, 1764.

[6] S. Benferhat, A. Saffiotti, and P. Smets, *Belief functions and default reasoning*, Artif. Intell. (122):1–69, 2000.

[7] J. Bernoulli, *Ars conjectandi*, Basel, 1713.

[8] S. Berstein, *Versuch einer axiomatischen Begründung der Wahrscheinlichkeitsrechnung*, Mitt. Math. Ges. Chrakow 209–274, (1917).

[9] K. P. S. Bhaskara Rao and M. Bhaskara Rao, *Theory of charges*, Academic Press, 1983.

[10] G. Bohlmann, *Encykl. d. math. Wiss.* vol. I, part 2, D 4b:852–917, 1901.

[11] B. Bolzano, *Wissenschaftslehre*, 1837.

[12] G. Boole, *The Mathematical Analysis of Logic*, 1847.

[13] G. Boole, *An Investigation into the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities*, 1854.

[14] E. Borel, *Traite du calcul des probabilites*, 1924.

[15] E. Borel, *Principes et formules classiques du calcul des probabilites*, 1925.

[16] B. Boričić and M. Rašković, *A probabilistic validity measure in intuitionistic propositional logic*, Math. Balkan. 10(4), 365–372, (1996).

[17] B. Boričić, *Validity measurement in some propositional logics*, Math. Log. Q. 43:4, 550–558, (1997).

[18] R. Carnap, *Logical Foundations of Probability*, University of Chicago Press, 1950.

[19] R. Carnap, *The continuum of inductive methods*, University of Chicago Press, 1952.

[20] S. Coletti and R. Scozzafava *Probabilistic logic in a coherent setting*, Kluwer, Dordrecht, 2002.

[21] A. De Morgan, *Formal logic*, London, 1847.

[22] R. Đorđević, *Barwise completeness theorems for logics with integrals*, Publ. Inst. Math., Nouv. Sér. 49(63):1–5, (1991).

[23] R. Đorđević, *Analytic completeness theorem for absolutely continuous biprobability models*, Z. Math. Logik Grundlag. Math. 38 241–246, (1992).

[24] R. Đorđević, *Analytic completeness theorem for singular biprobability logic*, Math. Log. Q. 39 228–230, (1993).

[25] R. Đorđević, *Logics with two types of integral operators*, Publ. Inst. Math., Nouv. sér. 54(68) 18–24, (1993).

[26] R. Đorđević,, M. Rašković, and Z. Ognjanović, *Completeness theorem for propositional probabilistic models whose measures have only finite ranges*, Arch. Math. Log. 43, 557–563, 2004.

[27] R. Fagin, J. Halpern and N. Megiddo, *A logic for reasoning about probabilities*, Inform. Comput. 87(1-2) 78–128, (1990).

[28] R. Fagin and J. Halpern, *Reasoning about knowledge and probability*, J. ACM 41(2) 340–367, (1994).

[29] S. Fajarado, *Probability logic with conditional expecatation*, Ann. Pure Appl. Log. 28 137–161, (1985).

[30] M. Fattorosi-Barnaba and G. Amati, *Modal operators with probabilistic interpretations I*, Stud. Log. 46(4), 383–393, (1989).

[31] N. Friedman, and J. Halpern, *Plausibility measures and default reasoning*, J. ACM 48(6) 648–685, 2001.

[32] A. Frish and P. Haddawy, *Anytime deduction for probabilistic logic*, Artif. Intell. 69, 93–122, (1994).

[33] H. Gaifman, *Concerning measures in first order calculi*, Israel J. Math. 2 1–18, (1964).

[34] H. Gaifman, M. Snir, *Probabilities over rich languages, testing and randomness*, J. Symb. Log. 47:3 495–548, (1982).
[35] H. Gaifman, *A Theory of Higher Order Probabilities*; in: J. Y. Halpern, editor, *Proc. Theoretical Aspects of Reasoning about Knowledge*, Morgan-Kaufmann, San Mateo, California, 275–292, 1986.
[36] G. Georgakopoulos, D. Kavvadias, and C. Papadimitriou, *Probabilistic satisfiability*, J. Complex. 4(1) 1–11, (1988).
[37] K. Gödel, *Über die Vollstandigkeit des Logikkalkulus*, PhD thesis, University of Viena, 1929.
[38] D. P. Guelev, *A propositional dynamic logic with qualitative probabilities*, J. Phil. Log. 28(6) 575–605, (1999).
[39] P. Haddawy, *A logic of time, chance and action for representing plans*, Artif. Intell. 80 243–308 (1996).
[40] T. Hailperin, *Probability Logic*, Notre Dame J. Formal Log. 35:3 198–212, (1984).
[41] T. Hailperin, *Boole's logic and probability*, North-Holland, 1986.
[42] T. Hailperin, *Sentential Probability Logic, Origins, Development, Current Status, and Technical Applications*, Lehigh University Press, 1996
[43] J. Y. Halpern, *An analysis of first-order logics of probability*, Artif. Intell. 46 311–350, (1990).
[44] J. Y. Halpern, *Knowledge, belief and certainty*, Ann. Math. Artif. Intell. 4 301–322, (1991).
[45] A. Heifetz, P. Mongin, *Probability logic for type spaces*, Games and Economic Behavior 35 31–53, 2001.
[46] D. N. Hoover, *Probability logic*, Ann. Math. Logic. 14 287–313, (1978).
[47] D. N. Hoover, *An analitic completeness theorem for logic with probability quantifiers*, J. Symb. Log. 52 802–816, (1987).
[48] G. E. Hughes, M. J. Cresswell, *Modal logic*, Methuen, 1968.
[49] N. Ikodinović, *Craig interpolation theorem for classical propositional logic with some probability operators*, Publ. Inst. Math., Nouv. Sér. 69(83) 27–33, 2001.
[50] N. Ikodinović and Z. Ognjanović, *A logic with coherent conditional probabilities*, in: Lluis Godo, editor, *Proc. 8th European Conference Symbolic and Quantitative Approaches to Reasoning with Uncertainty ECSQARU 2005*, Barcelona, Spain, July 6–8, 2005, Lect. Notes Artif. Intell. 3571, Springer-Verlag, 2005, 726–736.
[51] D. Jovanović, N. Mladenović and Z. Ognjanović, *Variable Neighborhood Search for the Probabilistic Satisfiability Problem*; in: K. F. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R. F. Hartl, M. Reimann, editors, *Metaheuristics: Progress in Complex Systems Optimization*, Operations Research/Computer Science Interfaces Series 39 Springer-Verlag, Berlin–New York, 2007, 173–188.
[52] B. Jaumard, P. Hansen, and M. P. de Aragao, *Column generation methods for probabilistic logic*, ORSA J. Comput. 3 135–147, (1991).
[53] H. J. Keisler, *Model Theory For Infinitary Logic*, North-Holland, Amsterdam, 1971.
[54] H. J. Keisler, *Hyperfinite model theory*; in: R. O. Gandy, J. M. E. Hyland, editors, *Logic Colloquim 76*, North-Holland, 1977, 5–110.
[55] H. J. Keisler, *Probability quantifiers*; in: J. Barwise and S. Feferman, editors, *Model Theoretic Logics*, Springer-Verlag, Berlin, 1985, 509–556.
[56] J. Keisler, *Elementary calculus. An infinitesimal approach*, 2nd ed. Prindle, Weber and Schmidt, Boston, Massachusetts, 1986.
[57] J. Keisler, *A completeness proof for adapted probability logic*, Ann. Pure Appl. Logic 31 61–70, (1986).
[58] J. Keisler, *Hyperfinite models of adapted probability logic*, Ann. Pure Appl. Logic 31 71–86, (1986).
[59] J. M. Keynes, *Treatise on Probability*, 1921.
[60] A. Kolmogorov, *Grundbegriffe der Wahrscheinlichkeitsrechnung*, Springer-Verlag, 1933.
[61] S. Kraus, D. Lehmann, and M. Magidor, *Nonmonotonic reasoning, preferential models and cumulative logics*, Artif. Intell. 44 167–207, (1990).

[62] S. A. Kripke, *The undecidability of monadic modal quantification theory*, Z. Math. Logik 8, 113–116, (1962).
[63] S. A. Kripke, *Semantical analysis of intuitionistic logic I*; in: *Formal systems and recursive functions*, North-Holland, Amsterdam, 93–130, 1965.
[64] A. Kron, *Odnos polivalentnih logika i teorije verovatnoće*, Naučna knjiga, Beograd, 1967.
[65] J. H. Lamber, In *Neues organon*, Leipzig, 1764.
[66] D. Lehmann, and M. Magidor, *What does a conditional knowledge base entail?*, Artif. Intell. 55 1–60, (1992).
[67] G. W. Leibnitz, *De conditionibus*, 1665.
[68] G. W. Leibnitz, *Specimen juris*, 1669.
[69] G. W. Leibnitz, *De Nouveaux essais*, 1765.
[70] T. Lukasiewicz, *Probabilistic Default Reasoning with Conditional Constraints*, Ann. Math. Artif. Intell. 34, 35–88, 2002.
[71] H. MacColl, *The calculus of equivalent statements and integration limits*, Proc. London Math. Soc. 9 9–20, 177–186, 1877, 1878.
[72] H. MacColl, *Symbolic reasoning*, Mind 6 5–60, 1897.
[73] E. Marchioni and L. Godo, *A Logic for Reasoning about Coherent Conditional Probability: A Modal Fuzzy Logic Approach*, Lect. Notes Artif. Intell. 3229, 213–225, 2004.
[74] Z. Marković, Z. Ognjanović, and M. Rašković, *A probabilistic extension of intuitionistic logic*, Math. Log. Q. 49 415–424, 2003.
[75] Z. Marković, Z. Ognjanović, and M. Rašković, *An intuitionistic logic with probabilistic operators*, Publ. Inst. Math., Nouv. Sér. 73(87), 31–38, 2003.
[76] Z. Marković, Z. Ognjanović and M. Rašković, *What is the Proper Propositional Base for Probabilistic Logic?*; in: L. A. Zadeh, editor, *Proc. Information Processing and Management of Uncertainty in Knowledge-Based Systems Conf. IPMU 2004*, Perugia, Italy, July, 4–9, 2004, 443–450, 2004.
[77] M. Meier, *An infinitary probability logic for type spaces*, Israel J. Math., to appear
[78] G. H. Moore, *A house divided against itself: the emergence of first-order logic as the basis for mathematics*; in: E. R. Phillips, editor, *MAA Studies In Mathematics. Studies in the history of mathematics*, Mathematical Association of America, Washington, 1987, 98–136,
[79] N. Nilsson, *Probabilistic logic*, Artif. Intell. 28 71–87, (1986).
[80] N. Nilsson, *Probabilistic logic revisited*, Artif. Intell. 59 39–42, (1993).
[81] Z. Ognjanović, M. Rašković, *A logic with higher order probabilities*, Publ. Inst. Math., Nouv. Sér. 60(74), 1–4, (1996).
[82] Z. Ognjanović, *A logic for temporal and probabilistic reasoning, Workshop on Probabilistic Logic and Randomised Computation ESSLLI '98*, Saarbruecken, Germany, 1998.
[83] Z. Ognjanović, *Neke verovatnosne logike i njihove primene u računarstvu*, PhD thesis, PMF Kragujevac, 1999.
[84] Z. Ognjanović, and M. Rašković, *Some probability logics with new types of probability operators*, J. Log. Comput. 9(2) 181–195, (1999).
[85] Z. Ognjanović, and M. Rašković, *Some first-order probability logics*, Theor. Comput. Sci. 247(1-2) 191–212, 2000.
[86] Z. Ognjanović, J. Kratica and M. Milovanović, *A genetic algorithm for satisfiability problem in a probabilistic logic: A first report*, Lect. Notes Comput. Sci. 2143, 805–816, Springer-Verlag, 2001.
[87] Z. Ognjanović, U. Midić and J. Kratica, *A genetic algorithm for probabilistic SAT problem*, in: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L. A. Zadeh, *Artif. Intell. and Soft Computing ICAISC 2004*, Zakopane, Poland, June 7–11, 2004, Lect. Notes Artif. Intell. 3070, Springer-Verlag, 2004, 462–467.
[88] Z. Ognjanović, Z. Marković and M. Rašković, *Completeness theorem for a Logic with imprecise and conditional probabilities*, Publ. Inst. Math., Nouv. Sér. 78(92):35–49, 2005.
[89] Z. Ognjanović, U. Midić and N. Mladenović, *A Hybrid Genetic and Variable Neighborhood Descent for Probabilistic SAT Problem*; in: M. J. Blesa, editor, *Proc. Second Internat.*

Workshop on Hybrid Metaheuristics HM2005, Barcelona, Spain, August 29–31, 2005, Lect. Notes Comput. Sci. 3636, Springer-Verlag, 2005, 42–53.

[90] Z. Ognjanović, N. Ikodinović and Z. Marković, *A logic with Kolmogorov style conditional probabilities*; in: *Proc. 5th Panhellenic logic symposium*, Athens, Greece, July 25–28, 2005, 111–116, 2005.

[91] Z. Ognjanović, *Discrete Linear-time Probabilistic Logics: Completeness, Decidability and Complexity*, J. Log. Comput. 16(2), 257–285, 2006.

[92] Z. Ognjanović, Z. Marković, M. Rašković, *Completeness Theorem for a Logic with Imprecise and Conditional Probabilities*, in: *Logic Colloquium '05*, Athens, Greece, July 28–August 3, 2005, Bull. Symb. Log. 12(2), 357, 2006.

[93] Z. Ognjanović, A. Perović, M. Rašković, *Logics with the Qualitative Probability Operator*, Logic J. IGPL 16(2) 105–120, 2008.

[94] Z. Ognjanović, T. Timotijević, and A. Stanojević, *Database of papers about probability logics*, Mathematical institute, Belgrade, page http://problog.mi.sanu.ac.yu/, 2005.

[95] C. S. Pierce, *On a improvement in Boole's calculus of logic*, 1867.

[96] P. S. Poretskiy, *Solution of the general problem of the theory of probability by using mathematical logic*, 1887.

[97] M. Rašković, *Model theory for L_{AM} logic*, Publ. Inst. Math., Nouv. Sér. 37(51) 17–22, (1985).

[98] M. Rašković, *Completeness theorem for biprobability models*, J. Symb. Log. 51(3) 586–590, (1986).

[99] M. Rašković and. RŽivaljević, *Barwise completeness theorem for some biprobability logics*, Z. Math. Logik Grundlagen Math. 31 133–135, (1986).

[100] M. Rašković, *Weak completeness theorem for $L_{AP\forall}$*, Zb. Rad. PMF Kragujevac 8 69–72, (1987).

[101] M. Rašković, *Completeness theorem for singular biprobability models*, Proc. Am. Math. Soc. 102 389–392, (1988).

[102] M. Rašković, *Completeness theorem for a monadic logic with both ordinary first-order and probability quantifiers*, Publ. Inst. Math., Nouv. Sér. 47(61):1–4, (1990).

[103] M. Rašković, *A completeness theorem for an infinitary intuitionistic logic with both ordinary and probability quantifiers*, Publ. Inst. Math., Nouv. Sér. 50(64):7–13, (1991).

[104] M. Rašković and R. Đorđević, *Finite completeness theorem for biprobability logics*, Math. Balkan. 5 12–14, (1991).

[105] M. Rašković and R. Đorđević, *Second order probability logic*, Math. Balkan. 6 105–108, (1992).

[106] M. Rašković, *Classical logic with some probability operators*, Publ. Inst. Math., Nouv. Sér. 53(67), 1–3, (1993).

[107] M. Rašković and R. Đorđević, *Continuous time probability logic*, Publ. Inst. Math., Nouv. Sér. 57(71)143–146, (1995).

[108] M. Rašković and R. Đorđević, *Probability Quantifiers and Operators*, VESTA, Beograd, 1996.

[109] M. Rašković, R. Đorđević and M. Bradić, *Weak cylindric probability algebras*, Publ. Inst. Math., Nouv. Sér. 61(75):6–16, (1997).

[110] M. Rašković and Z. Ognjanović, *A first order probability logic, LP_Q*, Publ. Inst. Math., Nouv. Sér. 65(79), 1–7, (1999).

[111] M. Rašković and R.Đorđević, *Cylindric probability algebras*, Publ. Inst. Math., Nouv. Sér. 68(82):20–36, 2000.

[112] M. Rašković, Z. Ognjanović, and Z. Marković, *A Probabilistic Approach to Default Reasoning*; in: *Proc. NMR '04*, 335–341, 2004.

[113] M. Rašković, Z. Ognjanović, and Z. Marković, *A Logic with Conditional Probabilities*; in: *Proc. JELIA'04*, Lect. Notes Comput. Sci. 3229, Springer-Verlag, 2004, 226–238.

[114] M. Rašković, Z. Marković and Z. Ognjanović, *A logic with approximate conditional probabilities that can model default reasoning*, Internat. J. Approx. Reason. 49(1) 52-66, 2008.

[115] H. Reichenbach, *Wahrscheinlichkeitslehre*, 1935.
[116] H. Reichenbach, *The theory of probability*, University of California Press, 1949.
[117] A. Robinson, *Non-standard analysis*, North-Holland, Amsterdam, 1966.
[118] H. Rodenhausen, *The completeness theorem for adapted probability logic*, PhD thesis Heidelberg, 1982.
[119] A. Sistla and E. Clarke, *The complexity of propositional linear temporal logic*, J. ACM 32(3) 733–749, (1985).
[120] D. Scott and P. Krauss, *Assigning probabilities to logical formulas*; in: J. Hintikka, P. Suppes, editors, *Aspects of inductive logic*, North-Holland, 1966, 219–264.
[121] C.A. Smorynski, *Applications of Kripke models*; in: A.S. Troelstra, editor, *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, Lect. Notes Math. 344, Springer-Verlag, Berlin, 1973, [124].
[122] N.I. Styazhkin, *History of Mathematical Logic from Leibniz to Peano*, MIT Press, 1969.
[123] P. Suppes, *Probabilistic inference and the concept of total evidence*; in: J. Hintikka, P. Suppes, editors, *Aspects of inductive logic*, North-Holland, 1966, 49–65.
[124] A.S. Troelstra, editor, *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, Lect. Notes Math. 344, Springer-Verlag, Berlin, 1973.
[125] W. van der Hoeck, *Some consideration on the logics P_FD*, J. Appl. Non-Classical Log. 7(3), 287–307, (1997).

Milenko Mosurović, Tatjana Stojanović
and Ana Kaplarević-Mališić

REASONING IN BASIC DESCRIPTION LOGICS AND DESCRIPTION LOGICS WITH MODAL OPERATORS

Abstract. Description logics are a family of knowledge representation languages constructed for a wide area of application domains. They are based on the notion of concepts and roles, and are mainly characterized by constructors that allow complex concepts and roles to be built from atomic ones. Constructor selection depends on the application domain the representation formalism is designed for. This paper concerns a special type of DLs extended with temporal operators. After an overview of basic description logics, original results related to the temporal extensions of DLs, precisely $\mathcal{DLR}_{US}$, which has been modeled with an aim to overcome problems of reasoning over conceptual schemas and queries in temporal databases are given.

Mathematics Subject Classification (2000): 03B70, 68T30, 68T27, 68T35, 03B44, 03B45, 03B35, 03B60, 03B80, 03F20

CONTENTS

1. Introduction

Description logics (DL) are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formal way [StaStu04]. Besides representation enabling, they have the task to provide tools for reasoning about the knowledge described by them. They lie on the tracks of the research in the field of knowledge representation.

DL were established with a motivation of providing a formal foundation on network-based knowledge representation systems. In the 1970's research in the field of knowledge representation was very intensive. It gave a wide spectrum of ideas and solutions which were more or less usable or GENERAL. Roughly speaking, there were two types of knowledge representation approaches [Baa et al. 02]: logic-based formalisms as more general and formal and non-logic-based representations, as specialized and, often, ad hoc approaches.

Among these specialized non-logic-based representations there were semantic networks and frames, broadly used in practice. Although they were significantly different, they could both be regarded as network structures, where the structure of the network aims at representing domain knowledge as a set of individuals and their relationships [Baa et al. 02]. Hence, they were often referred to as network-based structures (see [Leh92]). Owing to their more human-centered origins, the network-based systems were often considered as more usable in practice than the logical systems. On the other hand, their less precise semantic characterization

[Baa et al. 02], i.e., concepts as general classes and individuals as instances of concepts were mixed in one vocabulary, which resulted in the absence of general reasoning functionalities.

Attempt on making the system better was representing basic elements of network-based systems by relaying on the first-order logic. It turned out that some constraints were not describable. Moreover, in many cases first-order theorem provers were a too big machinery. However, using only fragments of the first-order logic, depending on features of representation language, was good enough. These conclusions were made in big part owing to development of KL-ONE system [BraSch85], the first realized system of the so-called "structured inheritance networks" [Bra77, Bra78, SchSmo91]. KL-ONE family of languages are considered as DL ancestors.

The following three ideas, induced by work on KL-ONE systems, have largely shaped the subsequent development of DLs [Baa et al. 02]:

- The basic syntactic building blocks are atomic concepts (unary predicates), atomic roles (binary predicates), and individuals (constants).
- The expressive power of the language is restricted in using a rather small set of (epistemologically adequate) constructors for building complex concepts and roles.
- Implicit knowledge about concepts and individuals can be inferred automatically with help of inference procedures. In particular, subsumption relationships between concepts and instance relationships between individuals and concepts play an important role: unlike IS-A links in Semantic Networks, which are explicitly introduced by the user, subsumption relationships and instance relationships are inferred from the definition of the concepts and the properties of the individuals.

Having above in mind it is clear why the research in the area of Description Logics began under the label of terminological systems. "Later, the emphasis was on the set of concept-forming constructs admitted in the language, giving rise to the name concept languages. In more recent years, after attention was further moved towards the properties of the underlying logical systems, the term Description Logics became popular" [Baa et al. 02].

Major characteristics of description logics are:

- emphasis on reasoning;
- formal logic-based semantics;
- inference patterns;
- subsumption relations between concepts of a terminology;
- hierarchy of concepts derived from subsumption relations.

Reasoning procedures in DL must be decidable and their complexity depends on expressiveness.

All improvements that were brought by DL were the consequences of the fact that they were, in most cases, developed with formal background and with a concrete area of application in mind. Today, there are various implemented systems based on Description Logics which are used in various application domains. Depending on domains and system requirements necessary description formalisms differ

by expressive power and, consequently, by formal and computational properties of reasoning. With the same motivation different extensions of DL were investigated, too. Although semantics of extensions could be interesting for studying, most of the researches associated with language extensions are focused on finding reasoning procedures for the extended languages. Within these, extensions depending on application domain constructs for non-monotonic, epistemic, and temporal reasoning, and constructs for representing belief and uncertain and vague knowledge could be interesting.

The conventional description logics were designed to represent knowledge only about static application domains. To capture various dynamic features, for instance, intensional knowledge (in multi-agent systems), dependence on time or actions (in distributed systems), description logics are combined with suitable "modal" (propositional) logics, say epistemic, temporal, or dynamic. Again, there is a variety of possible combinations (see e.g. [Sch93, Laux94, BaaLau95, BaaOhl93]). Some of them are rather simple and do not increase substantially the complexity of the combined logics (for example, the temporal description logic of Schild [Sch93] is ExpTime-complete); others are too expressive and undecidable (e.g. the multi-dimensional description logic of Baader and Ohlbach [BaaOhl93]).

An optimal compromise between the expressive power and decidability was found in the series of papers [WolZakh98, WolZakh99c, WolZakh99b, MosZakh99], where various expressive and yet decidable description logics with epistemic, temporal, and dynamic operators were constructed.

This paper gives an overview of basic description logics as well as original results, which concern the temporal extensions of Description Logics. The paper is organized as follows. Section 2 is based on [Baa et al. 02] and gives an introduction to description logics as a formal language for representing knowledge and reasoning based on that knowledge. It gives bases of syntax and semantics, and the typical reasoning tasks are described. At the end of the section some extensions of basic language are given. Section 3 mainly refers to modal extensions of description logics with emphasis on temporal extensions of description logics, precisely $\mathcal{DLR}_{US}$ as temporal extension of non-temporal description logic $\mathcal{DLR}_-$. It also gives an example of how the presented logics can be applied in temporal databases.

2. Basic description logics

Description logic (DL) is a common name[1] for a family of knowledge representation formalisms applied on a domain (the "world") by defining relevant domain terminology. They are based on a common family of languages, called description languages, which provide a set of constructors to build class (concept) and properties (role) descriptions. Such description can be used in axioms and assertions of DL knowledge bases and can be reasoned about with respect to DL knowledge bases by DL systems.

[1]Previously used names where *terminological knowledge representation languages, concept languages, term subsumption languages, KL-ONE-based knowledge representation languages*

2.1. Definition of the basic formalism. Knowledge base generated from a knowledge representation system based on DL has two components: TBox and ABox. TBox introduces terminology, i.e., vocabulary of an application domain, while ABox gives assertions about named individuals of concepts from introduced terminology.

The terminology consists of concepts and roles. Concepts denote sets of individuals, while roles denote binary relations between individuals. Complex descriptions of concepts and roles can be built by users in all DL systems. Description language for building these descriptions has model-theoretic semantics. Statements in the TBox and the ABox can be translated into first-order logic or an extension of it.

DL system also offers to reason about terminologies, individuals and assertions. Typical tasks for reasoning on a TBox level are

- determining satisfiability of terminology and
- subsumption relations of concepts.

Important reasoning problems an a ABox level are:

- determining consistency of sets of assertions (i.e., if ABox has a model) and
- whether a set of assertions entails that an individual is an instance of a given concept.

These checks can help to determine whether a knowledge base is meaningful or to organize concepts into a hierarchy according to their generality.

Knowledge representation (KR) system is integrated into a wider environment of an application. Other components interact with KR system by querying and modifying the knowledge base by adding and retracting concepts, roles and assertions. Rules present unlimited mechanism for adding assertions. They represent an extension of a logic core of formalism that can be logically interpreted.

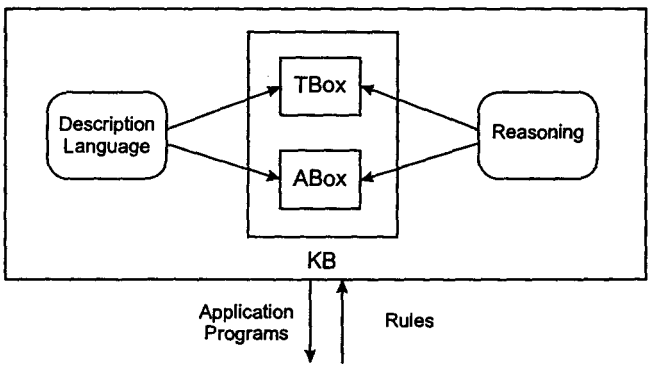


FIGURE 1. Architecture of KR system based on DL

2.1.1. The basic description language $\mathcal{AL}$. Elementary descriptions are atomic concepts and atomic roles. Complex descriptions can be built from them inductively with concept constructors. In abstract notation, we use the letter A for

atomic concepts, the letter R for atomic roles, and the letters C and D for concept descriptions. We shall discuss various languages from the family of $\mathcal{AL}$-languages[2].

Concept descriptions in $\mathcal{AL}$ are formed according to the following syntax rule:

$$\begin{array}{rcll} C, D & \to & A \mid & \text{(atomic concept)} \\ & & \top \mid & \text{(universal concept)} \\ & & \bot \mid & \text{(bottom concept)} \\ & & \neg A \mid & \text{(atomic negation)} \\ & & C \sqcap D \mid & \text{(intersection)} \\ & & \forall R.C \mid & \text{(value restriction)} \\ & & \exists R.\top & \text{(limited existential quantification)} \end{array}$$

In $\mathcal{AL}$ only the $\top$ is allowed in the scope in an existential quantification over a role and negation can only be applied to atomic concept. The sublanguage of $\mathcal{AL}$ obtained by disallowing atomic negation is $\mathcal{FL}^-$. The sublanguage of $\mathcal{FL}^-$ obtained by disallowing limited existential quantification is $\mathcal{FL}_0$.

Example 1. Let Person and Female be atomic concepts. Then Person $\sqcap$ Female and Person $\sqcap \neg$Female are $\mathcal{AL}$-concepts describing those persons that are female and those that are not. If hasChild is an atomic role, then the concept Person $\sqcap$ $\exists$hasChild.$\top$ denotes those persons that have a child, and the concept Person $\sqcap$ $\forall$hasChild.Female denotes those persons all of whose children are female. Using the bottom concept we describe persons without a child by the concept Person $\sqcap$ $\forall$hasChild.$\bot$.

In order to define a formal semantics of $\mathcal{AL}$-concepts, we introduce interpretations $\mathcal{I}$ that consist of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concept descriptions by the following inductive definitions:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ \bot^{\mathcal{I}} &= \emptyset \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid (\forall b)(a,b) \in R^{\mathcal{I}} \to b \in C^{\mathcal{I}}\} \\ (\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid (\exists b)(a,b) \in R^{\mathcal{I}}\} \end{aligned}$$

Two concepts C and D are equivalent ($C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$. For example, it is easy to verify that concepts $\forall$hasChild.Female$\sqcap$$\forall$hasChild.Student and $\forall$hasChild.(Female $\sqcap$ Student) are equivalent.

2.1.2. The family of $\mathcal{AL}$-languages. More expressive languages are obtained by adding further constructors to $\mathcal{AL}$.

[2]The language $\mathcal{AL}$=(*attributive language*) has been introduced in [SchSmo91] as a minimal language that is of practical interest.

The union of concepts (indicated by the letter $\mathcal{U}$) is written as $C \sqcup D$, and interpreted as $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$.

Full existential quantification (indicated by the letter $\mathcal{E}$) is written by $\exists R.C$, and interpreted as $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\exists b)(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$. Note that $\exists R.C$ differs from $\exists R.\top$ in that arbitrary concepts are allowed to occur in the scope of the existential quantifier.

Number restrictions (indicated by the letter $\mathcal{N}$) are written as $\geqslant nR$(at-least restriction) and as $\leqslant nR$ (at-most restriction), where n represents a nonnegative integer. They are interpreted as

$$(\geqslant nR)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \geqslant n\right\},$$
$$(\leqslant nR)^{\mathcal{I}} = \left\{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \leqslant n\right\}$$

respectively, where "$|\cdot|$" denotes the cardinality of a set.

The negation of arbitrary concepts (indicated by the letter $\mathcal{C}$, for "complement") is written by $\neg C$, and interpreted as $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$.

With the additional constructors concept:

$$\mathsf{Person} \sqcap (\leqslant 1\mathsf{hasChild} \sqcup (\geqslant 3\mathsf{hasChild} \sqcap \exists \mathsf{hasChild.Female}))$$

describes those persons that have either not more than one child or at least three children, one of which is female.

By extension by $\mathcal{AL}$ any subset of the above constructors generates a particular $\mathcal{AL}$-language. Each $\mathcal{AL}$-language is named by a string of the form $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$, where each letter represents the corresponding constructor.

From the semantic point of view, not all of these languages are distinct. The semantics enforces the equivalences $(C \sqcup D) \equiv \neg(\neg C \sqcap \neg D)$ and $\exists R.C \equiv \neg\forall R.\neg C$ (union and full existential quantification can be expressed using negation). We assume that union and full existential quantification are available in every language that contains negation and vice versa ($\mathcal{ALC}$ is used instead of $\mathcal{ALUE}$ and $\mathcal{ALCN}$ instead of $\mathcal{ALUEN}$).

2.1.3. Description languages as fragments of predicate logic. Since an interpretation $\mathcal{I}$ assigns to every atomic concept (role) a unary (binary) relation over $\Delta^{\mathcal{I}}$, we can view atomic concepts (roles) as unary (binary) predicates. Then:

- any concept C can be translated into a predicate logic formula $\phi_C(x)$ with one free variable x such that for every interpretation $\mathcal{I}$ the set of elements of $\Delta^{\mathcal{I}}$ satisfying $\phi_C(x)$ is exactly $C^{\mathcal{I}}$
- an atomic concept A is translated into the formula $A(x)$
- the constructors intersection, union, and negation are translated into logical conjunction, disjunction, and negation, respectively
- if C is already translated into $\phi_C(x)$ and R is an atomic role, then value restriction and existential quantification are captured by the formulae

$$\phi_{\forall R.C}(y) = (\forall x)(R(y,x) \rightarrow \phi_C(x))$$
$$\phi_{\exists R.C}(y) = (\exists x)(R(y,x) \wedge \phi_C(x))$$

where y is a new variable

- number restrictions are expressed by the formulae

$$\phi_{\geqslant nR}(x) = (\exists y_1, \ldots, y_n) R(x, y_1) \wedge \ldots \wedge R(x, y_n) \wedge \bigwedge_{i<j} y_i \neq y_j$$

$$\phi_{\leqslant nR}(x) = (\forall y_1, \ldots, y_{n+1}) R(x, y_1) \wedge \ldots \wedge R(x, y_{n+1}) \rightarrow \bigvee_{i<j} y_i = y_j$$

2.1.4. Terminologies. In the sequel, we will introduce:

- terminological axioms, which make statements about relations between concepts or roles,
- definitions as specific axioms and
- terminologies as sets of definitions by which we introduce atomic concepts as abbreviations or names for complex concepts.

Terminological axioms. In the most general case, terminological axioms have the form

$$C \sqsubseteq D \quad (R \sqsubseteq S) \qquad \text{or} \qquad C \equiv D \quad (R \equiv S)$$

where C, D are concepts (and R, S are roles). Axioms of the first kind are called inclusions, while axioms of the second kind are called equalities.

An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies an equality $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. If $\mathcal{T}$ is a set of axioms, then $\mathcal{I}$ satisfies $\mathcal{T}$ iff $\mathcal{I}$ satisfies each element of $\mathcal{T}$. If $\mathcal{I}$ satisfies an axiom (resp. a set of axioms), then it is a model of this axiom (resp. set of axioms). Two axioms or two sets of axioms are equivalent if they have the same models.

Definitions. An equality whose left-hand side is an atomic concept is a definition. Definitions are used to introduce symbolic names for complex descriptions.

A set of definitions should be unequivocal. A finite set of definitions $\mathcal{T}$ is a terminology or TBox if no symbolic name is defined more than once, that is, if for every atomic concept A there is at most one axiom in $\mathcal{T}$ whose left-hand side is A.

Example 2. A terminology (TBox) with concepts about family relationships can be introduced as follows:

$$\begin{aligned}
\mathsf{Woman} &\equiv \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} &\equiv \mathsf{Person} \sqcap \neg\mathsf{Woman} \\
\mathsf{Mother} &\equiv \mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Father} &\equiv \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\
\mathsf{Parent} &\equiv \mathsf{Father} \sqcup \mathsf{Mother} \\
\mathsf{Grandmother} &\equiv \mathsf{Mother} \sqcap \exists \mathsf{hasChild}.\mathsf{Parent} \\
\mathsf{MotherWithManyChildren} &\equiv \mathsf{Mother} \sqcap \geqslant 3\mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} &\equiv \mathsf{Mother} \sqcap \forall \mathsf{hasChild}.\neg\mathsf{Woman} \\
\mathsf{Wife} &\equiv \mathsf{Woman} \sqcap \exists \mathsf{hasHusband}.\mathsf{Man} \quad \square
\end{aligned}$$

Suppose, that $\mathcal{T}$ is a terminology. We divide the atomic concepts occurring in $\mathcal{T}$ into two sets, the name symbols $\mathcal{N}_{\mathcal{T}}$ (defined concepts) that occur on the left-hand side of some axiom and the base symbols $\mathcal{B}_{\mathcal{T}}$ (primitive concepts) that occur only

on the right-hand side of axioms. Based on this, terminologies define name symbols using base symbols.

A base interpretation for $\mathcal{T}$ is an interpretation that interprets only the base symbols. Let $\mathcal{J}$ be a base interpretation. An interpretation $\mathcal{I}$ that interprets also the name symbols is an extension of $\mathcal{J}$ if it has the same domain as $\mathcal{J}$, i.e., $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, and if it agrees with $\mathcal{J}$ for the base symbols. We say that $\mathcal{T}$ is definitorial if every base interpretation has exactly one extension that is a model of $\mathcal{T}$. In other words, if we know what the base symbols stand for, and $\mathcal{T}$ is definitorial, then the meaning of the name symbols is completely determined. If a terminology is definitorial, then every equivalent terminology is also definitorial.

The question whether a terminology is definitorial or not is related to the question whether or not its definitions are cyclic.

(1) $$\mathsf{Human}' \equiv \mathsf{Animal} \sqcap \mathsf{hasParent}.\mathsf{Human}'$$

Let A, B be atomic concepts occurring in $\mathcal{T}$. We say that A directly uses B in $\mathcal{T}$ if B appears on the right-hand side of the definition of A. The transitive closure of the relation "directly uses" is called "uses". Then $\mathcal{T}$ contains a cycle iff there exists an atomic concept in $\mathcal{T}$ that uses itself. Otherwise, $\mathcal{T}$ is called acyclic.

If a terminology $\mathcal{T}$ is acyclic, then it is definitorial. Definitions in terminology $\mathcal{T}$ can be expanded by replacing each occurrence of a name on the right-hand side of a definition with the concepts that it represents. If $\mathcal{T}$ is a acyclic this process eventually stops giving a terminology $\mathcal{T}'$ containing solely definitions of the form $A \equiv C'$, where C' contains only base symbols and no name symbols. $\mathcal{T}'$ is the expansion of $\mathcal{T}$. Size of the expansion can be exponential in the size of the original terminology.

Example 3. The expansion of the Family TBox previously introduced is:

$$\begin{aligned}
\mathsf{Woman} &\equiv \mathsf{Person} \sqcap \mathsf{Female}\\
\mathsf{Man} &\equiv \mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})\\
\mathsf{Mother} &\equiv (\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}\\
\mathsf{Father} &\equiv (\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}\\
\mathsf{Parent} &\equiv ((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female})) \sqcap \exists \mathsf{hasChild}.\mathsf{Person})\\
&\quad \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person})\\
\mathsf{Grandmother} &\equiv ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person})\\
&\quad \sqcap \exists \mathsf{hasChild}.(((\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}))\\
&\qquad \sqcap \exists \mathsf{hasChild}.\mathsf{Person})\\
&\quad \sqcup ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}))\\
\mathsf{MotherWithManyChildren} &\equiv ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person}) \sqcap \geqslant 3\mathsf{hasChild}\\
\mathsf{MotherWithoutDaughter} &\equiv ((\mathsf{Person} \sqcap \mathsf{Female}) \sqcap \exists \mathsf{hasChild}.\mathsf{Person})\\
&\quad \sqcap \forall \mathsf{hasChild}.(\neg(\mathsf{Person} \sqcap \mathsf{Female}))\\
\mathsf{Wife} &\equiv (\mathsf{Person} \sqcap \mathsf{Female})\\
&\quad \sqcap \exists \mathsf{hasHusband}.(\mathsf{Person} \sqcap \neg(\mathsf{Person} \sqcap \mathsf{Female}))
\end{aligned}$$

Lemma 1. *Let $\mathcal{T}$ be an acyclic terminology and $\mathcal{T}'$ be its expansion. Then*

(1) *$\mathcal{T}$ and $\mathcal{T}'$ have the same name and base symbols;*
(2) *$\mathcal{T}$ and $\mathcal{T}'$ are equivalent;*
(3) *$\mathcal{T}$ and $\mathcal{T}'$ are definitorial.*

Proof. Let $\mathcal{T}_1$ be a terminology. Suppose $A \equiv C$ and $B \equiv D$ are definitions in $\mathcal{T}_1$ such that B occurs in C. Let C' be the concept obtained from C by replacing each occurrence of B in C with D, and let $\mathcal{T}_2$ be the terminology obtained from $\mathcal{T}_1$ by replacing the definition $A \equiv C$ with $A \equiv C'$. Then both terminologies have the same name and base symbols. Moreover, since $\mathcal{T}_2$ has been obtained from $\mathcal{T}_1$ by replacing equals by equals, both terminologies have the same models. Since $\mathcal{T}'$ is obtained from $\mathcal{T}$ by a sequence of replacement steps like the ones above, this proves statements (1) and (2).

Suppose now that $\mathcal{J}$ is an interpretation of the base symbols. We extend it to an interpretation $\mathcal{I}$ that covers also the name symbols by setting $A^{\mathcal{I}} = C'^{\mathcal{J}}$, if $A \equiv C'$ is the definition of A in $\mathcal{T}'$. Clearly, $\mathcal{I}$ is a model of $\mathcal{T}'$, and it is the only extension of $\mathcal{J}$ that is a model of $\mathcal{T}'$. This shows that $\mathcal{T}'$ is definitorial. Moreover, $\mathcal{T}$ is definitorial as well, since it is equivalent to $\mathcal{T}'$. □

Of course, there are also terminologies with cycles that are definitorial, but:

Theorem 1. *Every definitorial $\mathcal{ALC}$-terminology is equivalent to an acyclic terminology.*

The theorem is a reformulation of Beths Definability Theorem [Gab72] for the modal propositional logic K_n.

2.1.5. Terminologies with inclusion axioms. In the case of concepts that cannot be defined completely necessary conditions for concept membership are still stated using an inclusion. Inclusion whose left-hand side is atomic is a specialization.

For example, concept "Women" from TBox in Example 2 can be described in less detail with the specialization

$$\mathsf{Woman} \sqsubseteq \mathsf{Person} \tag{2}$$

If specialization is allowed in a terminology, then the terminology looses its definitorial impact, even if it is acyclic. A set of axioms $\mathcal{T}$ is a generalized terminology if the left-hand side of each axiom is an atomic concept and for every atomic concept there is at most one axiom where it occurs on the left-hand side.

Generalized terminology $\mathcal{T}$ can be transformed into a regular terminology $\bar{\mathcal{T}}$, containing definitions only, such that $\bar{\mathcal{T}}$ is equivalent to $\mathcal{T}$ in a sense specified below. $\bar{\mathcal{T}}$ is obtained from $\mathcal{T}$ by choosing a new base symbol $\bar{A}$ for every specialization $A \sqsubseteq C$ in $\mathcal{T}$ and by replacing the specialization $A \sqsubseteq C$ with the definition $A \equiv \bar{A} \sqcap C$. The terminology $\bar{\mathcal{T}}$ is the normalization of $\mathcal{T}$.

If a TBox contains the specialization (2), then the normalization contains the definition $\mathsf{Woman} \equiv \overline{\mathsf{Woman}} \sqcap \mathsf{Person}$. The additional base symbol $\overline{\mathsf{Woman}}$ stands for the qualities that distinguish a woman among persons.

Lemma 2. *Let $\mathcal{T}$ be a generalized terminology and $\bar{\mathcal{T}}$ its normalization. Then*

(1) *Every model of $\bar{\mathcal{T}}$ is a model of $\mathcal{T}$.*
(2) *For every model $\mathcal{I}$ of $\mathcal{T}$ there is a model $\bar{\mathcal{I}}$ of $\bar{\mathcal{T}}$ that has the same domain as $\mathcal{I}$ and agrees with $\mathcal{I}$ on the atomic concepts and roles in $\mathcal{T}$.*

Proof. The first statement holds because a model $\bar{\mathcal{I}}$ of $\bar{\mathcal{T}}$ satisfies $A^{\bar{\mathcal{I}}} = (\bar{A} \sqcap C)^{\bar{\mathcal{I}}} = \bar{A}^{\bar{\mathcal{I}}} \cap C^{\bar{\mathcal{I}}}$, which implies $A^{\bar{\mathcal{I}}} \subseteq C^{\bar{\mathcal{I}}}$. Conversely, if $\mathcal{I}$ is a model of $\mathcal{T}$, then the extension $\bar{\mathcal{I}}$ of $\mathcal{I}$, defined by $\bar{A}^{\bar{\mathcal{I}}} = A^{\mathcal{I}}$, is a model of $\bar{\mathcal{T}}$, because $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ implies $A^{\mathcal{I}} = A^{\mathcal{I}} \cap C^{\mathcal{I}} = \bar{A}^{\mathcal{I}} \cap C^{\mathcal{I}}$, and therefore $\bar{\mathcal{I}}$ satisfies $A \equiv \bar{A} \sqcap C$. □

In theory, inclusion axioms do not extend the expressivity of terminologies, while, in practice, they are a convenient means to introduce terms into a terminology that cannot be defined completely.

2.1.6. World Descriptions. The second component of a knowledge base, in addition to the terminology or TBox, is the world description or ABox.
Assertions about individuals. In the ABox, individuals are introduced, by giving them names, and properties of these individuals are asserted. We denote individual names by a, b, c. Using concepts C and roles R, one can make assertions of the following two kinds in an ABox: $C(a)$, and $R(b, c)$. The first kind are concept assertions, and they state that a belongs to (the interpretation of) C. The second kind are role assertions, and they state that c is a filler of the role R for b. An ABox, denoted as $\mathcal{A}$, is a finite set of such assertions.

Example 4. If JOHN, PAUL, and MARY are individual names, then Father(JOHN) means that John is a father, and hasChild(MARY, PAUL) means that Paul is a child of Mary. An example of an ABox for TBox from Example 2:

MotherWithoutDaughter(MARY)	Father(JOHN)
hasChild(MARY, JOHN)	hasChild(JOHN, HARRY)
hasChild(MARY, PAUL)	

In a simplified view, an ABox can be seen as an instance of a relational database with only unary and binary relations. Contrary to the "closed-world semantics" of classical databases, the semantics of ABoxes is an "open-world semantics", since normally knowledge representation systems can be applied in situations where it cannot be assumed that the knowledge in the KB is complete. The TBox also imposes semantic relations between the concepts and roles in the ABox that do not have counterparts in database semantics.

ABoxes are given semantics by extending interpretations to individual names. From this point on, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ not only maps atomic concepts and roles to sets and relations, but also maps each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. This mapping is constructed with respect to the unique name assumption (UNA), that is, if a, b are distinct names, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The interpretation $\mathcal{I}$ satisfies the concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it satisfies the role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation satisfies the ABox $\mathcal{A}$ if it satisfies each assertion in $\mathcal{A}$. In this case we say that $\mathcal{I}$ is a model of the assertion or of the

ABox. Finally, $\mathcal{I}$ satisfies an assertion α or an ABox $\mathcal{A}$ with respect to a TBox $\mathcal{T}$ if in addition to being a model of α or of $\mathcal{A}$, it is a model of $\mathcal{T}$. Thus, a model of $\mathcal{A}$ and $\mathcal{T}$ is an abstraction of a concrete world where the concepts are interpreted as subsets of the domain as required by the TBox and where the membership of the individuals to concepts and their relationships with one another in terms of roles respect the assertions in the ABox.

Individual names in the description language. Sometimes, it is convenient to allow individual names (also called nominals) not only in the ABox, but in the description language as well. The most basic constructor employing individual is the "set" (or one-of), written by $\{a_1, \ldots, a_n\}$, where $a_1, \ldots, a_n$ are individual names. As one could expect, such a set concept is interpreted as

$$\{a_1, \ldots, a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\} \tag{3}$$

With sets in the description language one can, for instance, define the concept of permanent members of the UN security council as $\{\mathsf{CHINA}, \mathsf{FRANCE}, \mathsf{RUSSIA}, \mathsf{UK}, \mathsf{US}\}$.

Another constructor involving individual names is the "fills" constructor $R : a$ for a role R. The semantics of this constructor is:

$$(R : a)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (d, a^{\mathcal{I}}) \in R^{\mathcal{I}}\} \tag{4}$$

that is, $R : a$ stands for the set of those objects that have a as a filler of the role R.

2.2. Inferences.

A knowledge representation system can perform specific types of reasoning. Knowledge base, containing TBox and ABox has semantics that makes it equivalent to a set of axioms in first-order predicate logic. Like any other set of axioms, it contains implicit knowledge that can be made explicit through inferences.

Further discussion shows that the main problem with inference is consistency check for ABox, to which all other inferences can be reduced.

2.2.1. Reasoning tasks for concepts.

During the modeling of a domain terminology $\mathcal{T}$ is constructed by defining new concepts. It is important to check if new concepts are contradictory or not. A concept is meaningful if there is an interpretation that satisfies the axioms of $\mathcal{T}$, such that the concept denotes a nonempty set in that interpretation. Such a concept is satisfiable with respect to $\mathcal{T}$, otherwise it is unsatisfiable.

To check whether a domain model is contradictory or not, or to optimize queries that are formulated as concepts, it might be needed to know whether a concept is more general than another one (the subsumption problem). A concept C is subsumed by a concept D if in every model of $\mathcal{T}$ the set denoted by C is a subset of the set denoted by D. The algorithms that check the subsumption may also be used for organizing concepts of a TBox in a taxonomy according to their generality.

Two more relationships between concepts are equivalence and disjointness. These properties are formally defined as follows. Let $\mathcal{T}$ be a TBox.

Satisfiability: A concept C is satisfiable with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty – $\mathcal{I}$ is a model of C.

Subsumption: A concept C is subsumed by a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$ – $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Equivalence: Two concepts C and D are equivalent with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$ - $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

Disjointness: Two concepts C and D are disjoint with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.

If the TBox is empty, we simply write $\models C \sqsubseteq D$ if C is subsumed by D, and $\models C \equiv D$ if C and D are equivalent.

Lemma 3 (Reduction to Subsumption). *For concepts C, D we have*

(1) *C is unsatisfiable $\Leftrightarrow$ C is subsumed by $\bot$;*
(2) *C and D are equivalent $\Leftrightarrow$ C is subsumed by D and D is subsumed by C;*
(3) *C and D are disjoint $\Leftrightarrow$ $C \sqcap D$ is subsumed by $\bot$.*

The statements also hold with respect to a TBox.

Most DL systems that can check subsumption can perform all four inferences defined above, because almost all of description languages implemented in actual DL systems contain an unsatisfiable concept and all of them include the intersection operator "$\sqcap$".

Subsumption, equivalence, and disjointness of concepts can be reduced to the satisfiability problem if in addition to intersection, a system allows forming of the negation of a description [Smo88].

Lemma 4 (Reduction to Unsatisfiability). *For concepts C, D we have*

(1) *C is subsumed by D $\Leftrightarrow$ $C \sqcap \neg D$ is unsatisfiable;*
(2) *C and D are equivalent $\Leftrightarrow$ both $(C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable;*
(3) *C and D are disjoint $\Leftrightarrow$ $C \sqcap D$ is unsatisfiable.*

The statements also hold with respect to a TBox.

Since, for sets M, N we have $M \subseteq N$ iff $M \setminus N = \emptyset$, then the reduction of subsumption becomes apparent and easy to understand. The reduction of equivalence is correct because C and D are equivalent, if and only if C is subsumed by D and D is subsumed by C. Finally, the reduction of disjointness is just a rephrasing of the definition.

In an $\mathcal{AL}$-language without full negation, subsumption and equivalence cannot be reduced to unsatisfiability in the way shown in Lemma 4. The complexity of such inferences is somewhat different.

As seen in Lemma 3, from the viewpoint of worst-case complexity, subsumption is the most general inference for any $\mathcal{AL}$-language. Lemma 5 shows that unsatisfiability is a special case of each of the other problems. Lemma 3 and 5 show that, in order to obtain complexity bounds for inferences on concepts in $\mathcal{AL}$-languages (more precisely, for the complexity of the unsatisfiability, the equivalence, and the disjointness problem), it suffices to assess lower bounds for unsatisfiability and upper bounds for subsumption.

Lemma 5 (Reducing Unsatisfiability). *Let C be a concept. Then the following statements are equivalent:*

(1) *C is unsatisfiable;* (3) *C and $\bot$ are equivalent;*

(2) *C is subsumed by $\bot$;* (4) *C and $\top$ are disjoint.*

The statements also hold with respect to a TBox.

2.2.2. Eliminating the TBox. This section shows that, if $\mathcal{T}$ is an acyclic TBox, it is always possible to reduce reasoning problems with respect to $\mathcal{T}$ to problems with respect to the empty TBox. As seen in Lemma 1, $\mathcal{T}$ is equivalent to its expansion $\mathcal{T}'$. Recall that in the expansion every definition is of the form $A \equiv D$ such that D contains only base symbols, but no name symbols. For each concept C we define the expansion of C with respect to $\mathcal{T}$ as the concept C' that is obtained from C by replacing each occurrence of a name symbol A in C by the concept D, where $A \equiv D$ is the definition of A in $\mathcal{T}'$, the expansion of $\mathcal{T}$.

Since the expansion C' is derived from C by replacing names with descriptions in such a way that both are interpreted in the same way in any model of $\mathcal{T}'$, it follows that

- $C \equiv_{\mathcal{T}} C'$.

Thus, C is satisfiable with respect to $\mathcal{T}$ iff C' is satisfiable with respect to $\mathcal{T}$. However, C' contains no defined names, and thus C' is satisfiable with respect to $\mathcal{T}$ iff it is satisfiable. This yields that

- C is satisfiable with respect to $\mathcal{T}$ iff C' is satisfiable.

If D is another concept, then $D \equiv_{\mathcal{T}} D'$, and this yields that $C \sqsubseteq_{\mathcal{T}} D$ iff $C' \sqsubseteq_{\mathcal{T}} D'$ and $C \equiv_{\mathcal{T}} D$ iff $C' \equiv_{\mathcal{T}} D'$. Since C' and D' contain only base symbols, this implies

- $\mathcal{T} \models C \sqsubseteq D$ iff $\models C' \sqsubseteq D'$
- $\mathcal{T} \models C \equiv D$ iff $\models C' \equiv D'$.

With similar arguments we can show that

- C and D are disjoint with respect to $\mathcal{T}$ iff C' and D' are disjoint.

Expanding concepts with respect to an acyclic TBox allows removing the TBox from reasoning problems.

Expanding concepts may substantially increase computational complexity, since in the worst case the size of $\mathcal{T}'$ is exponential in the size of $\mathcal{T}$. A complexity analysis of the difficulty of reasoning with respect to TBoxes shows that the expansion of definitions is a source of complexity that cannot always be avoided.

2.2.3. Reasoning tasks for ABoxes. After designed a terminology and using the reasoning services of DL system to check that all concepts are satisfiable and that the expected subsumption relations hold, the ABox can be filled with assertions about individuals. An ABox contains two types of assertions: concept assertions of the form $C(a)$ and role assertions of the form $R(a, b)$. It is understandable that the representation of such knowledge has to be consistent.

An ABox $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$, if there is an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$. It is simply said that $\mathcal{A}$ is consistent if it is consistent with respect to the empty TBox.

For example, the set of assertions {Mother(MERY), Father(MERY)} is consistent with respect to the empty TBox, however, the assertions are not consistent with respect to the Family TBox.

Similarly as for concepts, checking the consistency of an ABox with respect to an acyclic TBox can be reduced to checking an expanded ABox. The expansion of $\mathcal{A}$ with respect to $\mathcal{T}$ is defined as the ABox $\mathcal{A}'$ that is obtained from $\mathcal{A}$ by replacing each concept assertion $C(a)$ in $\mathcal{A}$ with the assertion $C'(a)$, where C' is the expansion of C with respect to $\mathcal{T}$. In every model of $\mathcal{T}$, a concept C and its expansion C' are interpreted in the same way. Therefore, $\mathcal{A}'$ is consistent with respect to $\mathcal{T}$ iff $\mathcal{A}$ is consistent with respect to $\mathcal{T}$. However, since $\mathcal{A}'$ does not contain a name symbol defined in $\mathcal{T}$ it is consistent with respect to $\mathcal{T}$ iff it is consistent. The conclusion is:

- $\mathcal{A}$ is consistent with respect to $\mathcal{T}$ iff its expansion $\mathcal{A}'$ is consistent.

Other inferences that are going to be introduced can also be defined with respect to a TBox or for an ABox alone. As in the case of consistency, reasoning tasks for ABoxes with respect to acyclic TBoxes can be reduced to reasoning on expanded ABoxes.

Over an ABox $\mathcal{A}$, queries can be posed about the relationships between concepts, roles and individuals. The prototypical ABox inference on which such queries are based is the instance check, or the check whether an assertion is entailed by an ABox. An assertion α is entailed by $\mathcal{A}$ and we write $\mathcal{A} \models \alpha$ if every interpretation that satisfies $\mathcal{A}$, that is, every model of $\mathcal{A}$, also satisfies α. If α is a role assertion, the instance check is easy, since description language does not contain constructors to form complex roles. If α is of the form $C(a)$, the instance check can be reduced to the consistency problem for ABoxes because there is the following connection:

- $\mathcal{A} \models C(a)$ iff $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent.

Reasoning about concepts can also be reduced to consistency checking. We have seen in Lemma 4 that the important reasoning problems for concepts can be reduced to the one to decide whether a concept is satisfiable or not. Similarly, concept satisfiability can be reduced to ABox consistency because for every concept C it holds:

- C is satisfiable iff $\{C(a)\}$ is consistent,

where a is an arbitrarily chosen individual name.

Conversely, in [Sch94] it has been shown that ABox consistency can be reduced to the concept satisfiability in languages with the "set" and the "fills" constructors.

If knowledge bases are considered as means to store information about individuals, it may be needed to know all individuals that are instances of a given concept description C, that is, the description language is used to formulate queries. Given an ABox $\mathcal{A}$ and a concept C, the retrieval problem is to find all individuals a such that $\mathcal{A} \models C(a)$. A non-optimized algorithm for a retrieval query can be realized by testing whether each individual occurring in the ABox is an instance of the query concept C.

The dual inference to retrieval is the realization problem: given an individual a and a set of concepts, find the most specific concepts C from the set such that $\mathcal{A} \models C(a)$. Here, the most specific concepts are those that are minimal with respect to the subsumption ordering $\sqsubseteq$.

2.3. Reasoning algorithms. As it was shown in the previous section, if conjunction and negation are allowed in certain DL, then all relevant inference problems can be reduced to consistency problem for ABoxes. If negation is not allowed, then subsumption of concepts can be computed by so-called structural subsumption algorithms, i.e., algorithms that compare the syntactic structure of (possibly normalized) concept descriptions. Such algorithms are, usually, very efficient, but they are only complete for rather simple languages with little expressivity. In particular, DLs with (full) negation and disjunction cannot be handled by structural subsumption algorithms. For such languages, tableau-based algorithms are often used.

Designing new algorithms for reasoning in DLs can be unnecessary in many cases. Trying to reduce the problem to a known inference problem in logics is a good way. For example, decidability of the inference problems for $\mathcal{ALC}$ and many other DLs can be obtained as a consequence of the known decidability result for the two variable fragment of the first-order predicate logic. The language $\mathcal{L}^2$ consists of all formulae of the first-order predicate logic that can be built with the help of predicate symbols (including equality) and constant symbols (but without function symbols) using only the variables x, y [Mor75]. By appropriately reusing variable names, any concept description of the language $\mathcal{ALC}$ can be translated into an $\mathcal{L}^2$-formula with one free variable [Bor96]. This connection between $\mathcal{ALC}$ and $\mathcal{L}^2$ shows that any extension of $\mathcal{ALC}$ by constructors that can be expressed with the help of only two variables yields a decidable DL. Number restrictions and composition of roles are examples of constructors that cannot be expressed within $\mathcal{L}^2$, but number restrictions can be expressed in $\mathcal{C}^2$, the extension of $\mathcal{L}^2$ by counting quantifiers, which has recently been shown to be decidable [Grä et al. 97, Pac97]. However, the complexity of the decision procedures obtained in this way is usually higher than necessary: for example, the satisfiability problem for $\mathcal{L}^2$ is NEXPTIME-complete, whereas satisfiability of $\mathcal{ALC}$-concept descriptions is "only" PSPACE-complete.

Lower complexity decision procedures can be obtained by using the connection between DLs and propositional modal logics. $\mathcal{ALC}$ is a syntactic variant of the propositional multi-modal logic **K** [Sch91] and the extension of $\mathcal{ALC}$ by transitive closure of roles corresponds to Propositional Dynamic Logic (PDL) [Baa91]. Some of the algorithms used in propositional modal logics for deciding satisfiability are very similar to the tableau-based algorithms newly developed for DLs. Instead of using tableau-based algorithms, decidability of certain propositional modal logics (and thus of the corresponding DLs), can also be shown by establishing the finite model property [Fit93] of the logic (i.e., showing that a formula/concept is satisfiable iff it is satisfiable in a finite interpretation) or by using tree automata [VarWol86].

2.3.1. Structural subsumption algorithms. These algorithms usually proceed in two phases. First, the descriptions to be tested for subsumption are normalized, and then the syntactic structure of the normal forms is compared. Ideas underlying this approach will be shown for the language $\mathcal{FL}_0$, which allows for conjunction ($C \sqcap D$) and value restrictions ($\forall R.C$). Then the bottom concept ($\bot$), atomic negation ($\neg A$) and number restrictions ($\leqslant nR$ and $\geqslant nR$) handling will be presented.

An $\mathcal{FL}_0$-concept description is in a normal form iff it is of the form

$$A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$$

where $A_1, \ldots, A_m$ are distinct concept names, $R_1, \ldots, R_n$ are distinct role names, and $C_1, \ldots, C_n$ are $\mathcal{FL}_0$-concept descriptions in normal form. Using associativity, commutativity and idempotence of $\sqcap$, and the fact that the descriptions $\forall R.(C \sqcap D)$ and $(\forall R.C) \sqcap (\forall R.D)$ are equivalent, it is easy to see that any description can be transformed into an equivalent one in the normal form.

Lemma 6. *Let $A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$ be the normal form of the $\mathcal{FL}_0$-concept description C, and $B_1 \sqcap \ldots \sqcap B_k \sqcap \forall S_1.D_1 \sqcap \ldots \sqcap \forall S_l.D_l$ the normal form of the $\mathcal{FL}_0$-concept description D. Then $C \sqsubseteq D$ iff the following two conditions hold:*

(1) *for all i, $1 \leqslant i \leqslant k$ there exists j, $1 \leqslant j \leqslant m$ such that $B_i = A_j$*
(2) *for all i, $1 \leqslant i \leqslant l$ there exists j, $1 \leqslant j \leqslant n$ such that $S_i = R_j$ and $C_i \sqsubseteq D_j$*

Having this lemma in mind, it is easy to construct recursive algorithm for computing subsumption. That algorithm has a polynomial time complexity [LevBra87].

If $\mathcal{FL}_0$ is extended by language constructors that can express unsatisfiable concepts, then the definition of the normal form must be changed. On the other hand, the structural comparison of the normal forms must take into account that an unsatisfiable concept is subsumed by every concept. The simplest DL where this occurs is $\mathcal{FL}_\bot$ the extension of $\mathcal{FL}_0$ by the bottom concept $\bot$.

An $\mathcal{FL}_\bot$-concept description is of the normal form iff it is $\bot$ or of the form

$$A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$$

where $A_1, \ldots, A_m$ are distinct concept names different from $\bot$, $R_1, \ldots, R_n$ are distinct role names, and $C_1, \ldots, C_n$ and $\mathcal{FL}_\bot$-concept descriptions in the normal form. Such a normal form can easily be computed. In principle, one just computes the $\mathcal{FL}_0$-normal form of the description (where $\bot$ is treated as an ordinary concept name): $B_1 \sqcap \ldots \sqcap B_k \sqcap \forall R_1.D_1 \sqcap \ldots \sqcap \forall R_n.D_n$. If one of the B_is is $\bot$ then replace the whole description by $\bot$. Otherwise, apply the same procedure recursively to the D_js. The structural subsumption algorithm for $\mathcal{FL}_\bot$ works just like the one for $\mathcal{FL}_0$ with the only difference that $\bot$ is subsumed by any description.

Extension of $\mathcal{FL}_\bot$ by atomic negation can be treated similarly. During the computation of the normal form, negated concept names are treated like concept names. If a name and its negation occur on the same level of the normal form, then $\bot$ is added, which can then be treated as described above. The structural comparison of the normal forms treats negated concept names just like concept names.

Finally, if we consider the language $\mathcal{ALN}$, the additional presence of number restrictions leads to a new type of conflict. On one hand, as in the case of atomic negation, number restrictions may be conflicting with each other (e.g. $\geqslant 2R$ and $\leqslant 1R$). On the other hand, at-least restrictions $\geqslant nR$ for $n \geqslant 1$ are in conflict with value restrictions $\forall R.\bot$. When computing the normal form, number restrictions can be treated like concept names. The next step is taking care of the new types of

conflicts by introducing $\bot$ and using it for normalization as described above. During the structural comparison of normal forms, inherent subsumption relationships between number restrictions (e.g. $\geqslant nR \sqsubseteq \geqslant mR$ iff $n \geqslant m$) must also be taken into account [BorPat94].

Structural subsumption algorithms, like described above, usually fail to be complete for larger DLs. In particular, they cannot treat disjunction, full negation, and full existential restriction $\exists R.C$. These constructors can be more efficiently treated in subsumption algorithms that are constructed in a tableau-based style.

2.3.2. Tableau algorithms. Tableau algorithms use another idea to examine subsumption of concept descriptions. Precisely, as it was shown in Subsection 2.2, they use negation to reduce subsumption to (un)satisfiability of concept descriptions using: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable.

Before describing a tableau-based satisfiability algorithm for $\mathcal{ALCN}$ in more detail, we illustrate the underlying ideas using a few basic rules:

- For any existential restriction the algorithm introduces a new individual as role filler, and this individual must satisfy the constraints expressed by the restriction.
- The algorithm uses value restrictions in interaction with already defined role relations to impose new constraints on individuals.
- For disjunctive constraints, the algorithm tries both possibilities in successive attempts. It must backtrack if it reaches an obvious contradiction, i.e., if the same individual must satisfy constraints that are obviously conflicting.
- If an at-most number restriction is violated, then the algorithm must identify different role fillers.

2.3.3. A tableau-based satisfiability algorithm for $\mathcal{ALCN}$. Describing the algorithm needs introducing an appropriate data structure which will be used for representing constraints like "a belongs to (the interpretation of) C" and "b is an R-filler of a". Although many papers on tableau algorithms for DLs introduce the new notion of a constraint system for this purpose, considering the types of constraints that must be expressed, ABox assertions can be used for their representations. Since the presence of at-most number restrictions may lead to the identification of different individual names, the unique name assumption (UNA) will not be imposed on the ABoxes considered by the algorithm. Instead, explicit inequality assertions of the form $x \neq y$ for individual names x, y, with the obvious semantics that an interpretation $\mathcal{I}$ satisfies $x \neq y$ iff $x^{\mathcal{I}} \neq y^{\mathcal{I}}$ will be allowed. These assertions are assumed to be symmetric, i.e., saying that $x \neq y$ belongs to an ABox $\mathcal{A}$ is the same as saying that $y \neq x$ belongs to $\mathcal{A}$.

Let C_0 be an $\mathcal{ALCN}$-concept. In order to test satisfiability of C_0, the algorithm starts with the ABox $\mathcal{A}_0 = \{C_0(x_0)\}$, and applies consistency preserving transformation rules (see Figure 2) to do ABox until no more rules apply. If the "complete" ABox obtained in this way does not contain an obvious contradiction (called clash), then $\mathcal{A}_0$ is consistent (and thus C_0 is satisfiable), and inconsistent

(unsatisfiable) otherwise. The transformation rules that handle disjunction and at-most restrictions are non-deterministic in the sense that a given ABox is transformed into finitely many new ABoxes such that the original ABox is consistent iff one of the new ABoxes is so. For this reason instead of single ABoxes finite sets of ABoxes $\mathcal{S} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ are considered. Such a set is consistent iff there is some i, $1 \leqslant i \leqslant k$ such that $\mathcal{A}_i$ is consistent. A rule in Figure 2 is applied to a given finite set of ABoxes $\mathcal{S}$ as follows: it takes an element $\mathcal{A}$ of $\mathcal{S}$ and replaces it by one ABox $\mathcal{A}'$, by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$, or by finitely many ABoxes $\mathcal{A}_{i,j}$.

The $\rightarrow_{\sqcap}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$
Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_{\sqcup}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$
Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' = \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_{\exists}$-rule
Condition: $\mathcal{A}$ contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x,z)$ are in $\mathcal{A}$
Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y), R(x,y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_{\forall}$-rule
Condition: $\mathcal{A}$ contains $(\forall R.C)(x)$ and $R(x,y)$, but it does not contain $C(y)$
Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.

The $\rightarrow_{\geqslant}$-rule
Condition: $\mathcal{A}$ contains $(\geqslant nR)(x)$ and there are no individual names $z_1, \ldots, z_n$ such that $R(x,z_i)$ $(1 \leqslant i \leqslant n)$ and $z_i \neq z_j$ $(1 \leqslant i < j \leqslant n)$ are contained in $\mathcal{A}$
Action: $\mathcal{A}' = \mathcal{A} \cup \{R(x,y_i) \mid 1 \leqslant i \leqslant n\} \cup \{y_i \neq y_i \mid 1 \leqslant i < j \leqslant n\}$ where $y_1 \ldots, y_n$ are distinct individual names not occurring in $\mathcal{A}$.

The $\rightarrow_{\leqslant}$-rule
Condition: $\mathcal{A}$ contains distinct individual names $y_1, \ldots, y_{n+1}$ such that $(\leqslant nR)(x)$ and $R(x,y_1), \ldots, R(x,y_{n+1})$ are in $\mathcal{A}$ and $y_i \neq y_j$ is not in $\mathcal{A}$ for some $i \neq j$
Action: For each pair y_i, y_j such that $i > j$ and $y_i \neq y_j$ is not in $\mathcal{A}$, the ABox $\mathcal{A}_{i,j} = [y_i/y_j]\mathcal{A}$ is obtained from $\mathcal{A}$ by replacing each occurrence of y_i by y_j.

FIGURE 2. Transformation rules of the satisfiability algorithm.

Consequent to the definition of the transformation rules the following lemma is valid:

Lemma 7 (Soundness). *Assume that $\mathcal{S}'$ is obtained from the finite set of ABoxes $\mathcal{S}$ by application of a transformation rule. Then $\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent.*

The second important property of the set of transformation rules is that the transformation process always terminates:

Lemma 8 (Termination [BaaSat99, Don et al. 97]). *Let C_0 be an $\mathcal{ALCN}$-concept description. There cannot be an infinite sequence of rule applications*

$$\{\{C_0(x_0)\}\} \to \mathcal{S}_1 \to \mathcal{S}_2 \to \dots$$

Lemma 9. *Let $\mathcal{A}$ be an ABox contained in $\mathcal{S}_i$ for some $i \geqslant 1$. Then:*

- *For every individual $x \neq x_0$ occurring in $\mathcal{A}$, there is a unique sequence $R_1, \dots, R_l$ $(l \geqslant 1)$ of role names and a unique sequence $x_1, \dots, x_{l-1}$ of individual names such that $\{R_1(x_0, x_1), R_2(x_1, x_2), \dots, R_l(x_{l-1}, x)\} \subseteq \mathcal{A}$. In this case, we say that x occurs on level l in $\mathcal{A}$.*
- *If $C(x) \in \mathcal{A}$ for an individual name x on level l, then the maximal role depth of C (i.e., the maximal nesting of constructors involving roles) is bounded by the maximal role depth of C_0 minus l. Consequently, the level of any individual in $\mathcal{A}$ is bounded by the maximal role depth of C_0.*
- *If $C(x) \in \mathcal{A}$, then C is a subdescription of C_0. Consequently, the number of different concept assertions on x is bounded by the size of C_0.*
- *The number of different role successors of x in $\mathcal{A}$ (i.e., individuals y such that $R(x, y) \in \mathcal{A}$ for a role name R) is bounded by the sum of the numbers occurring in at-least restrictions in C_0 plus the number of different existential restrictions in C_0.*

Starting with $\{\{C_0(x_0)\}\}$, we thus obtain after a finite number of rule applications a set of ABoxes $\hat{\mathcal{S}}$, to which no more rules apply. An ABox $\mathcal{A}$ is called complete iff none of the transformation rules applies to it. Consistency of a set of complete ABoxes can be determined by looking for clashes. The ABox $\mathcal{A}$ contains a clash iff one of the following three situations occurs:

(i) $\{\bot(x)\} \subseteq \mathcal{A}$ for some individual name x;
(ii) $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name A;
(iii) $\{(\leqslant nR)(x)\} \cup \{R(x, y_i) \mid 1 \leqslant i \leqslant n+1\} \cup \{y_i \neq y_j \mid 1 \leqslant i < j \leqslant n+1\} \subseteq \mathcal{A}$ for individual names $x, y_1, \dots, y_{n+1}$, a nonnegative integer n, and a role name R.

Obviously, an ABox that contains a clash cannot be consistent. Hence, if all the ABoxes in $\hat{\mathcal{S}}$ contain a clash, then $\hat{\mathcal{S}}$ is inconsistent, and thus by the soundness lemma $\{C_0(x_0)\}$ is inconsistent as well. Consequently, C_0 is unsatisfiable. If, however, one of the complete ABoxes in $\hat{\mathcal{S}}$ is clash-free, then $\hat{\mathcal{S}}$ is consistent. By soundness of the rules, this implies consistency of $\{C_0(x_0)\}$, and thus satisfiability of C_0.

Lemma 10 (Completeness). *Any complete and clash-free ABox $\mathcal{A}$ has a model.*

This lemma can be proved by defining the canonical interpretation $\mathcal{I}_\mathcal{A}$ induced by $\mathcal{A}$:

(i) the domain $\Delta^{\mathcal{I}_\mathcal{A}}$ of $\mathcal{I}_\mathcal{A}$ consists of all the individual names occurring in $\mathcal{A}$;
(ii) for all atomic concepts A we define $A^{\mathcal{I}_\mathcal{A}} = \{x \mid A(x) \in \mathcal{A}\}$;

(iii) for all atomic roles R we define $R^{\mathcal{I}_\mathcal{A}} = \{(x, y) \mid R(x, y) \in \mathcal{A}\}$.

$\mathcal{I}_\mathcal{A}$ satisfies all the role assertions in $\mathcal{A}$, by definition, and, by induction on the structure of concept descriptions, it is easy to show that it satisfies the concept assertions as well. The inequality assertions are satisfied since $x \neq y \in \mathcal{A}$ only if x, y are different individual names.

The facts stated in Lemma 9 imply that the canonical interpretation has the shape of a finite tree whose depth is linearly bounded by the size of C_0 and whose branching factor is bounded by the sum of the numbers occurring in at-least restrictions in C_0 plus the number of different existential restrictions in C_0. Consequently, $\mathcal{ALCN}$ has the finite tree model property, i.e., any satisfiable concept C_0 is satisfiable in a finite interpretation $\mathcal{I}$ that has the shape of a tree whose root belongs to C_0.

Theorem 2. *It is decidable whether or not an $\mathcal{ALCN}$-concept is satisfiable.*

Theorem 3. *Satisfiability of $\mathcal{ALCN}$-concept descriptions is* PSPACE*-complete.*

2.3.4. Extension to the consistency problem for ABoxes. Algorithm that decides consistency of $\mathcal{ALCN}$-ABoxes can be constructed as an extension of described tableau-based satisfiability algorithm. Let $\mathcal{A}$ be an $\mathcal{ALCN}$-ABox. To test $\mathcal{A}$ for consistency, we first add inequality assertions $a \neq b$ for every pair of distinct individual names a, b occurring in $\mathcal{A}$. Let $\mathcal{A}_0$ be the ABox obtained in this way. The consistency algorithm applies the rules of Figure 2 to the singleton set $\{\mathcal{A}_0\}$. Soundness and completeness of the rule set can be shown as before.

Termination can be enabled by requiring that generating rules $\rightarrow_\exists$ and $\rightarrow_\geqslant$ may only be applied if none of the other rules are applicable.

Following a similar idea, the consistency problem for $\mathcal{ALCN}$-ABoxes can be reduced to satisfiability of $\mathcal{ALCN}$-concept descriptions [Hol96]. Roughly speaking, this reduction works as follows: In a preprocessing step, one applies the transformation rules only to old individuals (i.e., individuals present in the original ABox). Subsequently, one can forget about the role assertions, i.e., for each individual name in the preprocessed ABox, the satisfiability algorithm is applied to the conjunction of its concept assertions.

Theorem 4. *Consistency of $\mathcal{ALCN}$-ABoxes is* PSPACE*-complete.*

2.3.5. Extension to general inclusion axioms. In the above subsections, we have considered the satisfiability problem for concept descriptions and the consistency problem for ABoxes without an underlying TBox. In fact, for acyclic TBoxes one can simply expand the definitions. Expansion is, however, no longer possible if general inclusion axioms of the form $C \sqsubseteq D$, where C and D may be complex descriptions, are allowed. Instead of considering finitely many such axioms $C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n$, it is sufficient to consider the single axiom $\top \sqsubseteq \widehat{C}$, where

$$\widehat{C} = (\neg C_1 \sqcup D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n).$$

The axiom $\top \sqsubseteq \widehat{C}$ simply claims that any individual must belong to the concept $\widehat{C}$. The tableau algorithm introduced above can easily be modified in such a manner

that it takes the following axiom into account: all individuals (both the original individuals and the ones newly generated by the $\rightarrow_{\exists}$- and $\rightarrow_{\geqslant}$-rule) are simply asserted to belong to $\widehat{C}$. However, this may produce nonterminating algorithm.

Termination can be regained by detecting cyclic computations, and then blocking the application of generating rules: the application of the rules $\rightarrow_{\exists}$ to an individual x is blocked by an individual y in an ABox $\mathcal{A}$ iff $\{D \mid D(x) \in \mathcal{A}\} \subseteq \{D' \mid D'(y) \in \mathcal{A}\}$. The main idea underlying blocking is that the blocked individual x can use the role successors of y instead of generating new ones.

To avoid cyclic blocking (of x by y and vice versa), we consider an enumeration of all individual names, and define that an individual x may only be blocked by individuals y that occur before x in this enumeration. This notion of blocking, together with some other technical assumptions, enables soundness, completeness as well as termination of algorithm [Buc et al. 93, Baa96]. Thus, consistency of $\mathcal{ALCN}$-ABoxes with respect to general inclusion axioms is decidable. Since an algorithm may generate role paths of exponential length before blocking, it is no longer in PSPACE. In fact, even for the language $\mathcal{ALC}$, satisfiability with respect to a single general inclusion axiom is known to be EXPTIME [DonMas00]. The tableau-based algorithm sketched above is a NEXPTIME algorithm. However, using the translation technique mentioned at the beginning of this section, it can be shown [DeG95] that $\mathcal{ALCN}$-ABoxes and general inclusion axioms can be translated into PDL(Propositional Dynamic Logic), which satisfiability can be decided in exponential time.

Theorem 5. *Consistency of $\mathcal{ALCN}$-ABoxes with respect to general inclusion axioms is* EXPTIME*-complete.*

2.3.6. Extension to other language constructors. The tableau-based algorithms for checking concept satisfiability and ABox consistency can also be employed for languages with other concept and/or role constructors. Each new constructor requires a new rule, and this rule can usually be obtained by simply considering the semantics of the constructor. Soundness of such a rule is often very easy to show. Completeness and termination are more difficult to control, since they must also take into account interactions between different rules. As it was shown above, termination can sometimes only be obtained if the application of rules is restricted by an appropriate strategy. Of course, one may only impose such a strategy if one can show that it does not perturb completeness.

2.3.7. Reasoning with respect to terminologies. As it was said before, terminologies (TBoxes) are sets of concept definitions (i.e., equalities of the form $A \equiv C$ where A is atomic) such that every atomic concept occurs at most once as a left-hand side.

Acyclic terminologies. As shown in Section 2.2, reasoning with respect to acyclic terminologies can be reduced to reasoning without terminologies by expanding the TBox, followed by replacing name symbols by their definitions in the terminology. Unfortunately, this increases the complexity of reasoning, since the expanded TBox may be exponentially larger than the original one [Neb90].

For more expressive languages, the presence of acyclic TBoxes does not necessarily increase the complexity of the subsumption problem. For example, subsumption of concept descriptions in the language $\mathcal{ALC}$ is PSPACE-complete, and so is subsumption with respect to acyclic terminologies. Of course, in order to obtain a PSPACE-algorithm for subsumption in $\mathcal{ALC}$ with respect to acyclic TBoxes, one cannot first expand the TBox completely since this might need exponential space. The main idea is that one uses a tableau-based algorithm like the one described, with the difference that it receives concept descriptions containing name symbols as input. Expansion is then done by the following rule: if the tableau-based algorithm encounters an assertion of the form $A(x)$, where A is a name occurring on the left-hand side of a definition $A \equiv C$ in the TBox, then it adds the assertion $C(x)$. However, it does not further expand C at this stage. It is not difficult to show that this yields a PSPACE-algorithm for satisfiability (and thus also for subsumption) of concepts with respect to acyclic TBoxes in $\mathcal{ALC}$ [Lut99].

There are, however, extensions of $\mathcal{ALC}$ for which this technique is not proper. One such example is the language $\mathcal{ALCF}$, i.e., $\mathcal{ALC}$ extended by functional roles as well as agreements and disagreements on chains of functional roles (see Section 2.4 below). Satisfiability of concepts is PSPACE-complete for this language [HolNut90], but satisfiability of concepts with respect to acyclic terminologies is NEXPTIME-complete [Lut99].

Cyclic terminologies. For cyclic terminologies, expansion would not terminate. If we use descriptive semantics, then cyclic terminologies are a special case of terminologies with general inclusion axioms. Thus, the tableau-based algorithm for handling general inclusion axioms previously introduced can also be used for cyclic $\mathcal{ALCN}$-TBoxes with descriptive semantics.

For less expressive DLs, more efficient algorithms can, however, be obtained with the help of techniques based on finite automata.

2.4. Language extensions. In Section 2.1 we have introduced the language $\mathcal{ALCN}$ as a Description Logic prototype. For many applications, the expressive power of $\mathcal{ALCN}$ is not sufficient. For this reason, various other language constructors have been introduced in the literature and are employed by systems. In [Baa et al. 02] these language extensions were roughly classified into two categories, "classical" and "nonclassical" extensions. Intuitively, a classical extension is one whose semantics can easily be defined within the model-theoretic framework introduced in Section 2.1, whereas defining the semantics of a nonclassical constructor is more problematic and requires an extension of the model-theoretic framework. Hereafter, the most important classical extensions of Description Logics will be briefly introduced.

2.4.1. Role constructors. Since roles are interpreted as binary relations, it is quite natural to employ the usual operations on binary relations (such as Boolean operators, composition, inverse, and transitive closure) as role forming constructors.

Definition 1 (Role constructors). Every role name is a role description (atomic role), and if R, S are role descriptions, then $R \sqcap S$ (intersection), $R \sqcup S$ (union), $\neg R$

(complement), $R \circ S$ (composition), R^+ (transitive closure), R^- (inverse), $id(C)$ (role identity) are also role descriptions.

Given an interpretation $\mathcal{I}$ is extended to (complex) role descriptions as follows:

(i) $(R \sqcap S)^{\mathcal{I}} = R^{\mathcal{I}} \cap S^{\mathcal{I}}$, $(R \sqcup S)^{\mathcal{I}} = R^{\mathcal{I}} \cup S^{\mathcal{I}}$, $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$;
(ii) $(R \circ S)^{\mathcal{I}} = \{(a,c) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (\exists b)(a,b) \in R^{\mathcal{I}} \wedge (b,c) \in S^{\mathcal{I}}\}$;
(iii) $(R^+)^{\mathcal{I}} = \bigcup_{i \geqslant 1} (R^{\mathcal{I}})^i$, i.e., $(R^+)^{\mathcal{I}}$ is the transitive closure of $(R^{\mathcal{I}})$;
(iv) $(R^-)^{\mathcal{I}} = \{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}$
(v) $id(C)^{\mathcal{I}} = \{(a,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid a \in C^{\mathcal{I}}\}$, i.e., each instance of concept to itself.

For example, the union of the roles hasSon and hasDaughter can be used to define the role hasChild, and the transitive closure of hasChild expresses the role hasOffspring. The inverse of hasChild yields the role hasParent.

Example 5. The following $\mathcal{ALCI}_{reg}$ TBox $\mathcal{T}_{file}$ models a file-system constituted by file-system elements:

$$\begin{aligned} \mathsf{FSelem} &\sqsubseteq \exists \mathsf{name.String} \\ \mathsf{FSelem} &\equiv \mathsf{Directory} \sqcup \mathsf{File} \\ \mathsf{Directory} &\sqsubseteq \neg \mathsf{File} \\ \mathsf{Directory} &\sqsubseteq \forall \mathsf{child.FSelem} \\ \mathsf{File} &\sqsubseteq \forall \mathsf{child}.\bot \\ \mathsf{Root} &\sqsubseteq \mathsf{Directory} \\ \mathsf{Root} &\sqsubseteq \forall \mathsf{child}^-.\bot \end{aligned}$$

The axioms in $\mathcal{T}_{file}$ imply that in a model every object connected by a chain of role child to an instance of Root is an instance of FSelem. Formally,

$$\mathcal{T}_{file} \models \exists (\mathsf{child}^-)^+.\mathsf{Root} \sqsubseteq \mathsf{FSelem}$$

It is shown that the complexity of satisfiability and subsumption of concepts in the language $\mathcal{ALCN}^{\sqcap}$ (also called $\mathcal{ALCNR}$ in the literature and which extends $\mathcal{ALCN}$ by intersection of roles) are still PSPACE-complete [Don et al. 97, Tob01]. Decidability of the extension of $\mathcal{ALCN}$ by the three Boolean operators and the inverse operator is a direct consequence of the fact that concepts of the extended language can be expressed in $\mathcal{C}^2$, i.e., first-order predicate logic with two variables and counting quantifiers, which is known to be decidable in NEXPTIME [Grä et al. 97, Pac97]. It is also shown [LutSat00] that $\mathcal{ALC}$ extended by role complement is EXPTIME-complete, whereas $\mathcal{ALC}$ extended by role intersection and atomic role complement is NEXPTIME-complete.

For $\mathcal{ALC}_{trans}$ (which extends $\mathcal{ALC}$ by transitive-closure, composition, and union of roles) subsumption and satisfiability problem have been shown to be decidable [Baa91] and EXPTIME-complete [FisLad79, Pra79, Pra80]. The extension of $\mathcal{ALC}_{trans}$ by the inverse constructor corresponds to converse PDL [FisLad79], which can also be shown to be decidable in deterministic exponential time [Var85]. $\mathcal{ALC}_{trans}$ extended by inverse and number restrictions does not have the finite

model property. Nevertheless, this DL still has an EXPTIME-complete subsumption and satisfiability problem.

2.4.2. Expressive number restrictions. First, we will consider the so-called qualified number restrictions, where the number restrictions are concerned with role-fillers belonging to a certain concept.

Example 6. Given the role hasChild, the simple number restrictions introduced above can only state that the number of all children is within certain limits, such as in the concept $\geqslant$ 2hasChild$\sqcap \leqslant$ 5hasChild. Qualified number restrictions can also express that there are at least 2 sons and at most 5 daughters:

$$\geqslant \text{2hasChild.Male}\sqcap \leqslant \text{5hasChild.Female}$$

Adding qualified number restrictions to $\mathcal{ALC}$ leaves the important inference problems (like subsumption and satisfiability of concepts, and consistency of ABoxes) decidable: the worst-case complexity is still PSPACE-complete. The language is decidable if general sets of inclusion axioms are allowed [Buc et al. 93].

The second group of extensions are those which allow for complex role expressions inside number restrictions. The extension of $\mathcal{ALCN}$ by number restrictions involving composition has a decidable satisfiability and subsumption problem. On the other hand, if any number restrictions involving composition, union and inverse, or number restrictions involving composition and intersection are added, then satisfiability and subsumption become undecidable [BaaSat96, BaaSat99]. For $\mathcal{ALC}_{trans}$ the extension by number restrictions involving compositionis already undecidable [BaaSat99].

Third, if the explicit numbers n in number restrictions are replaced by variables α that stand for arbitrary nonnegative integers, the expressive power of language can further be increased by introducing explicit quantification of the numeric variables.

It is shown that $\mathcal{ALCN}$ extended by such symbolic number restrictions with universal and existential quantification of numerical variables has an undecidable satisfiability and subsumption problem. If one restricts this language to existential quantification of numerical variables and negation on atomic concepts, then satisfiability becomes decidable, but subsumption remains undecidable.

2.4.3. Role-value-maps. Role-value-maps are a family of very expressive concept constructors, which were, however, available in the original KL-One-system.

Definition 2 (Role-value-maps). A role chain is a composition $R_1 \circ \cdots \circ R_n$ of role names. If R, S are role chains, then $R \subseteq S$ and $R = S$ are concepts.

A given interpretation $\mathcal{I}$ is extended to role-value-maps as follows:

$$\text{(i)} \quad (R \subseteq S)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\forall b)((a,b) \in R^{\mathcal{I}} \rightarrow (a,b) \in S^{\mathcal{I}})\}$$

$$\text{(ii)} \quad (R = S)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\forall b)((a,b) \in R^{\mathcal{I}} \leftrightarrow (a,b) \in S^{\mathcal{I}})\}$$

Example 7. The concept

$$\text{Person} \sqcap (\text{hasChild} \circ \text{hasFriend} \subseteq \text{knows})$$

describes the persons knowing all friends of their children, and

$$\mathsf{Person} \sqcap (\mathsf{marriedTo} \circ \mathsf{likesToEat} = \mathsf{likesToEat})$$

describes persons having the same favorite foods as their spouse.

Unfortunately, in the presence of role-value-maps, the subsumption problem is undecidable, even if the language allows only for conjunction and value restriction as additional constructors.

Solution to this problem is restricting the attention to role chains of functional roles, also called attributes or features in the literature. An interpretation $\mathcal{I}$ interprets the role R as a functional role iff $\{(a,b),(a,c)\} \subseteq R^{\mathcal{I}}$ implies $b = c$. In the following, it will be assumed that the set of role names is partitioned into the set of functional roles and the set of ordinary roles. Any interpretation must interpret the functional roles as such. Functional roles will be denoted with small letters f, g, possibly with index.

Definition 3 (Agreements). If f, g are role chains of functional roles, then $f \doteq g$ and $f \not\doteq g$ are concepts (agreement and disagreement).

A given interpretation $\mathcal{I}$ is extended to agreements and disagreements as follows:

(i) $(f \doteq g)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\exists b)((a,b) \in f^{\mathcal{I}} \wedge (a,b) \in g^{\mathcal{I}})\}$

(ii) $(f \not\doteq g)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\exists b_1, b_2)(b_1 \neq b_2 \wedge (a,b_1) \in f^{\mathcal{I}} \wedge (a,b_2) \in g^{\mathcal{I}})\}$

In the literature, the agreement constructor is sometimes also called the "same-as" constructor. Since f, g are the role chains between the functional roles, there can be at most one role filler for a with respect to the respective role chain. The semantics of agreements and disagreements requires these role fillers to exist (and be equal or distinct) for a to belong to the concept.

Example 8. Roles such as hasMother, hasFather and hasLastName with their usual interpretation are functional roles, whereas hasParent and hasChild are not. The concept

$$\mathsf{Person} \sqcap (\mathsf{hasLastName} \doteq \mathsf{hasMother} \circ \mathsf{hasLastName})$$
$$\sqcap (\mathsf{hasLastName} \not\doteq \mathsf{hasFather} \circ \mathsf{hasLastName})$$

describes persons whose last name coincides with the last name of their mother, but not with the last name of their father.

The restriction to functional roles makes reasoning in $\mathcal{ALC}$ extended by agreements and disagreements decidable [HolNut90]. However, if general inclusion axioms (or transitive closure of functional roles or cyclic definitions) are allowed, then agreements and disagreements between chains of functional roles again cause subsumption to become undecidable.

2.4.4. Functional restrictions ($\mathcal{F}$). Functional restrictions are the simplest form of number restrictions considered in description logics, and allow for specifying local functionality of roles, i.e., that instances of certain concepts have unique role-fillers for a given role. By adding functional restrictions on atomic roles and their inverse

to $\mathcal{ALCI}_{reg}$ we obtain the description logic $\mathcal{ALCFI}_{reg}$. Functional restrictions has a form $\leqslant 1Q$, where Q is a basic role, i.e., either an atomic role or the inverse of an atomic role. Such a functional restriction is interpreted as follows:

$$(\leqslant 1Q)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in Q^{\mathcal{I}}\}| \leqslant 1\}$$

Reasoning in $\mathcal{ALCFI}_{reg}$ is EXPTIME-complete. Also, $\mathcal{ALCFI}_{reg}$ has the tree model property, which states that if a $\mathcal{ALCFI}_{reg}$-concept C is satisfiable then it is satisfied in an interpretation which has the structure of a (possibly infinite) tree with bounded branching degree. This makes the space for using of techniques based on automata on infinite trees.

2.4.5. Qualified number restrictions ($\mathcal{Q}$). Qualified number restrictions is the most general form of number restrictions, and allow for specifying arbitrary cardinality constraints on roles with role-fillers belonging to a certain concept. In particular we will consider qualified number restrictions on basic roles, i.e., atomic roles and their inverse. By adding such constructs to $\mathcal{ALCI}_{reg}$ we obtain the description logic $\mathcal{ALCQI}_{reg}$.

Qualified number restrictions has a form $\leqslant nQ.C$ and $\geqslant nQ.C$, where n is a nonnegative integer, Q is a basic role, and C is an $\mathcal{ALCQI}_{reg}$-concept. Such constructs are interpreted as follows:

$$(\leqslant nQ.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in Q^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \leqslant n\}$$
$$(\geqslant nQ.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in Q^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \geqslant n\}$$

Reasoning in $\mathcal{ALCQI}_{reg}$ is still EXPTIME-complete.

2.4.6. Relations of arbitrary arity. A limitation of traditional description logics is that only binary relationships between instances of concepts can be represented, which is a quite limitation in a process of modeling relationships among more than two objects in some real world situations. Such relationships can be described by making use of relations of arbitrary arity instead of (binary) roles.

Let us consider the description logic $\mathcal{DLR}$, which represents a natural generalization of traditional description logics towards n-ary relations. The basic elements of $\mathcal{DLR}$ are atomic relations and atomic concepts, denoted by $\mathbf{P}$ and A, respectively. Arbitrary relations, of given arity between 2 and n_{max}, and arbitrary concepts are formed according to the following syntax

$$\mathbf{R} \rightarrow \top_n \mid \mathbf{P} \mid (i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2$$
$$C \rightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[i]\mathbf{R} \mid \leqslant k[i]\mathbf{R}$$

where i denotes a component of a relation, i.e., an integer between 1 and $n_{\max}$, n denotes the arity of a relation, i.e., an integer between 2 and $n_{\max}$, and k denotes a nonnegative integer.

For $\mathcal{DLR}$ interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is introduced as follows:

$$\top_n^{\mathcal{I}} \subseteq (\Delta)^n$$
$$\mathbf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$$

$$
\begin{aligned}
(\neg \mathbf{R})^{\mathcal{I}} &= \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}} \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} &= \mathbf{R}_n^{\mathcal{I}} \cap \mathbf{R}_n^{\mathcal{I}} \\
(i/n : C)^{\mathcal{I}} &= \{(d_1, \ldots, d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \\
\top_1^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists [i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid (\exists (d_1, \ldots, d_n) \in \mathbf{R}^{\mathcal{I}}) d_i = d\} \\
(\leqslant k[i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \left|\{(d_1, \ldots, d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\}\right| \leqslant k\}
\end{aligned}
$$

Theorem 6. *Logical implication in $\mathcal{DLR}$ is* EXPTIME*-complete.*

$\mathcal{DLR}$ can be extended to include regular expressions built over projections of relations on two of their components, thus obtaining $\mathcal{DLR}_{reg}$ (decidability is also EXPTIME-complete).

$\mathcal{DLR}$ and $\mathcal{DLR}_{reg}$ are generalizations of $\mathcal{ALCQI}$ and $\mathcal{ALCQI}_{reg}$, and they can be extended by Boolean constructs on roles and role inclusion axioms. Obtained languages have EXPTIME-complete logical implication.

Reasoning in $\mathcal{SHIQ}$, which is $\mathcal{ALCQI}$ extended with roles that are transitive, and with role inclusion axioms on arbitrary roles (direct, inverse, and transitive), is still EXPTIME-complete.

3. Description logics with modal operators

3.1. Preliminaries. We begin by defining the modal concept description language $\mathcal{ALC}_{\mathcal{M}}$ and its semantics.

The primitive symbols of $\mathcal{ALC}_{\mathcal{M}}$ are:

- *concept names* $C_0, C_1, \ldots$,
- *role names* $R_0, R_1, \ldots$, and
- *object names* $a_0, a_1, \ldots$.

Starting from these we can form compound concepts and formulas using the following constructs. Suppose R is a role name and C, D are concepts (for the basis of our inductive definition we assume concept names to be atomic concepts). Then $\top$, $C \sqcap D$, $\neg C$, $\exists R.C$, and $\Diamond C$ (or $C\mathcal{U}D$, $C\mathcal{S}D$ for a strict linear order) are *concepts.*

Atomic formulas are expressions of the form $\top$, $C = D$, $a : C$, and $a\,R\,b$, where a, b are object names. If φ and ψ are formulas then so are $\varphi \wedge \psi$, $\neg\varphi$, and $\Diamond\varphi$ (or $\psi\mathcal{U}\varphi$, $\psi\mathcal{S}\varphi$ for a strict linear order).

The pure description part of this language is $\mathcal{ALC}$. By adding the constructs for the formation of the union $R \sqcup S$, composition $R \circ S$, transitive reflexive closure R^* and test $C?$, we can extend it to $\mathcal{C}$, and to $\mathcal{CI}$ ($\mathcal{CIQ}$) by adding still inversion R^- (inversion R^- and number restrictions $\exists^{\geqslant n} B.C$, where B is a role name or its converse). The corresponding modal description language is denoted then by $\mathcal{C}_{\mathcal{M}}$, $\mathcal{CI}_{\mathcal{M}}$ and $\mathcal{CIQ}_{\mathcal{M}}$.

A *model* of $\mathcal{ALC}_{\mathcal{M}}$ based on a frame $\mathfrak{F} = \langle W, \lhd \rangle$ is a pair $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ in which I is a function associating with each $w \in W$ a structure

$$I(w) = \left\langle \Delta^{I,w}, R_0^{I,w}, \ldots, C_0^{I,w}, \ldots, a_0^{I,w}, \ldots \right\rangle,$$

where $\Delta^{I,w}$ is a nonempty set of objects, the *domain* of w, $R_i^{I,w}$ are binary relations on $\Delta^{I,w}$, $C_i^{I,w}$ subsets of $\Delta^{I,w}$, and $a_i^{I,w}$ are objects in $\Delta^{I,w}$ such that $a_i^{I,w} = a_i^{I,v}$, for any $v, w \in W$.

One can distinguish between three types of models: those with *constant*, *expanding*, and *varying domains*. In models with constant domains $\Delta^{I,v} = \Delta^{I,w}$, for all $v, w \in W$. In models with expanding domains $\Delta^{I,v} \subseteq \Delta^{I,w}$ whenever $v \lhd w$. And models with varying domains are just arbitrary models.

Given a model $\mathfrak{M}$ and a world w in it, we define the *value* $C^{I,w}$ of a concept C in w and the *truth-relation* $(\mathfrak{M}, w) \models \varphi$ (or simply $w \models \varphi$, if $\mathfrak{M}$ is understood) by taking:

$$\top^{I,w} = \Delta, \quad \text{and} \quad C^{I,w} = C_i^{I,w}, \text{ for } C = C_i;$$
$$(C \sqcap D)^{I,w} = C^{I,w} \cap D^{I,w}; \qquad (\neg C)^{I,w} = \Delta \setminus C^{I,w};$$
$$x \in (\Diamond C)^{I,w} \text{ iff } \exists v \rhd w \; x \in C^{I,v};$$
$$x \in (\exists R.C)^{I,w} \text{ iff } \exists y \in C^{I,w} \; x R^{I,w} y;$$

$$\begin{aligned}
w \models C = D \quad & \text{iff} \quad C^{I,w} = D^{I,w};\\
w \models a : C \quad & \text{iff} \quad a^{I,w} \in C^{I,w};\\
w \models aRb \quad & \text{iff} \quad a^{I,w} R^{I,w} b^{I,w};\\
w \models \Diamond\varphi \quad & \text{iff} \quad \exists v \rhd w \; v \models \varphi;\\
w \models \varphi \wedge \psi \quad & \text{iff} \quad w \models \varphi \text{ and } w \models \psi;\\
w \models \neg\varphi \quad & \text{iff} \quad w \not\models \varphi.
\end{aligned}$$

If $\mathfrak{F} = \langle W, < \rangle$ is a strict linear order with modal operators $\mathcal{U}$ and $\mathcal{S}$, than we have
$x \in (C \,\mathcal{U}\, D)^{I,w}$ iff there is $u > w$ such that $x \in D^{I,u}$ and $x \in C^{I,v}$ for all $v \in (w, u)$;
$x \in (C \,\mathcal{S}\, D)^{I,w}$ iff there is $u < w$ such that $x \in D^{I,u}$ and $x \in C^{I,v}$ for all $v \in (u, w)$;
$w \models \psi \,\mathcal{U}\, \chi$ iff there is $u > w$ such that $u \models \chi$ and $v \models \psi$ for all $v \in (w, u)$; and
$w \models \psi \,\mathcal{S}\, \chi$ iff there is $u < w$ such that $u \models \chi$ and $v \models \psi$ for all $v \in (u, w)$.

A formula φ is *satisfiable* in a class of models $\mathcal{M}$ if there is a model $\mathfrak{M} \in \mathcal{M}$ and a world w in $\mathfrak{M}$ such that $w \models \varphi$. We will use special names for certain classes of models with one accessibility relation. Namely,

- $\mathcal{K}$ the class of all models;
- $\mathcal{S}5$ the class of models based on frames with the universal relations, i.e., $u \lhd v$ for all u and v;
- $\mathcal{KD}45$ the class of transitive, serial ($\forall u \exists v \; u \lhd v$) and Euclidean ($u \lhd v \wedge u \lhd w \to v \lhd w$) models;
- $\mathcal{S}4$ the class of all quasi-ordered models;
- $\mathcal{K}4$ the class of transitive models;

$\mathcal{GL}$ the class of transitive Noetherian models
(i.e., containing no infinite ascending chains); and
$\mathcal{N}$ the class of models based on $\langle \mathbb{N}, < \rangle$.

We are in a position now to present known decidability and complexity results concerning formula-satisfiability problems [MosZakh99, Mos2000].

Theorem 7. (1) *The formula-satisfiability problem, when we adopt expanding domain assumption, for the language $\mathcal{ALC}_{\mathcal{M}}$ in each of the classes $\mathcal{K}$, $\mathcal{N}$, $\mathcal{GL}$, $\mathcal{S}4$, and $\mathcal{K}4$ is* NEXPTIME*-hard.*

(2) *The formula-satisfiability problem for the language $\mathcal{ALC}_{\mathcal{M}}$ and $\mathcal{CI}_{\mathcal{M}}$ in the classe $\mathcal{K}$ is* NEXPTIME*-complete (no matter whether the models have constant or expanding domains).*

(3) *The formula-satisfiability problem for the language $\mathcal{ALC}_{\mathcal{M}}$ and $\mathcal{CIQ}_{\mathcal{M}}$ in the classe $\mathcal{S}5$ is* NEXPTIME*-complete.*

(4) *The formula-satisfiability problem for the language $\mathcal{ALC}_{\mathcal{M}}$ and $\mathcal{CIQ}_{\mathcal{M}}$ in the class $\mathcal{N}$ is* EXPSPACE*-complete.*

For these logics, tableau algorithms were developed [Lutz et al. 01, Lutz et al. 02]. Further on, we will continue with presenting of one temporal extension of description logics [Arta et al. 01, Arta et al. 02], as a special case of modal extension of description logics.

3.2. The Temporal Description Logic. Here, we adopt the *snapshot* representation of abstract temporal databases (and temporal conceptual models); see for example [ChoSaa98]. The flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where $\mathcal{T}_p$ is a set of time points (or chronons) and $<$ a binary precedence relation on $\mathcal{T}_p$, is assumed to be isomorphic to $\langle \mathbb{Z}, < \rangle$. Thus, a temporal database can be regarded as a map from time points in $\mathcal{T}$ to standard (relational) databases with the same domain of attributes and the same interpretation of constants.

As a language of temporal database conceptual schemas we use a natural combination of the propositional linear temporal logic with *Since* and *Until* [SisCl85, Gab et al. 94] and the (non-temporal) description logic $\mathcal{DLR}$ [Cal et al. 98]. The resulting *temporal description logic* will be denoted by $\mathcal{DLR}_{\mathcal{US}}$.

The basic syntactical types of $\mathcal{DLR}_{\mathcal{US}}$ are *entities* (i.e., unary predicates, also known as *concepts*) and *nary relations* of arity $\geqslant 2$. Starting from a set EN of *atomic entities* and a set RN of *atomic relations* we define inductively (complex) entity and relation expressions as is shown in the upper part of Fig. 3, where the binary constructs $(\sqcap, \sqcup, \mathcal{U}, \mathcal{S})$ are applied to relations of the same arity, i, j, k, n are natural numbers, $i \leqslant n$, and j does not exceed the arity of R.

A *temporal conceptual database schema* (or a *knowledge base*) is a finite set Σ of $\mathcal{DLR}_{\mathcal{US}}$-formulas. Atomic formulas are formulas of the form $E_1 \sqsubseteq E_2$ and $R_1 \sqsubseteq R_2$, with R_1 and R_2 being relations of the same arity. If φ and ψ are $\mathcal{DLR}_{\mathcal{US}}$formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \mathcal{U} \psi$, $\varphi \mathcal{S} \psi$. $E_1 \doteq E2$ is used as an abbreviation for $(E_1 \sqsubseteq E_2) \wedge (E_2 \sqsubseteq E_1)$, for both entities and relations. Temporal conceptual database schemas will serve as constraints for temporal databases.

$$
\begin{array}{rcl}
R & \rightarrow & \top_n | RN | \neg R | R_1 \sqcap R_2 | R_1 \sqcup R_2 | i/n : E | \\
 & & \Diamond^+ R | \Diamond^- R | \Box^+ R | \Box^- R | \oplus R | \ominus R | R_1 \mathcal{U} R_2 | R_1 \mathcal{S} R_2 \\
 & & \\
E & \rightarrow & \top | EN | \neg E | E_1 \sqcap E_2 | E_1 \sqcup E_2 | \exists^{\gtrless k}[j] R | \\
 & & \Diamond^+ E | \Diamond^- E | \Box^+ E | \Box^- E | \oplus E | \ominus E | E_1 \mathcal{U} E_2 | E_1 \mathcal{S} E_2
\end{array}
$$

$$
\begin{array}{lcl}
(\top_n)^{I(t)} & \subseteq & (\Delta^I)^n \\
RN^{I(t)} & \subseteq & (\top_n)^{I(t)} \\
(\neg R)^{I(t)} & = & (\top_n)^{I(t)} \setminus R^{I(t)} \\
(R_1 \sqcap R_2)^{I(t)} & = & R_1^{I(t)} \cap R_2^{I(t)} \\
(i/n : E)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid d_i \in E^{I(t)}\} \\
(R_1 \mathcal{U} R_2)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid \exists v > t. (\langle d_1, \ldots, d_n \rangle \in R_2^{I(v)} \\
 & & \qquad \wedge \forall w \in (t, v). \langle d_1, \ldots, d_n \rangle \in R_1^{I(w)})\} \\
(R_1 \mathcal{S} R_2)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid \exists v < t. (\langle d_1, \ldots, d_n \rangle \in R_2^{I(v)} \\
 & & \qquad \wedge \forall w \in (v, y). \langle d_1, \ldots, d_n \rangle \in R_1^{I(w)})\} \\
(\Diamond^+ R)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid \exists v > t. \langle d_1, \ldots, d_n \rangle \in R^{I(v)}\} \\
(\oplus R)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid \langle d_1, \ldots, d_n \rangle \in R^{I(t+1)}\} \\
(\Diamond^- R)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid \exists v < t. \langle d_1, \ldots, d_n \rangle \in R^{I(v)}\} \\
(\ominus R)^{I(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{I(t)} \mid \langle d_1, \ldots, d_n \rangle \in R^{I(t-1)}\} \\
\top^{I(t)} & = & \Delta^I \\
EN^{I(t)} & \subseteq & \top^{I(t)} \\
(\neg E)^{I(t)} & = & \top^{I(t)} \setminus E^{I(t)} \\
(E_1 \sqcap E_2)^{I(t)} & = & E_1^{I(t)} \cap E_2^{I(t)} \\
(\exists^{\gtrless k}[j] R)^{I(t)} & = & \{d \in \top^{I(t)} \mid \sharp\{\langle d_1, \ldots, d_n \rangle \in R^{I(t)} | d_j = d\} \gtrless k\} \\
(E_1 \mathcal{U} E_2)^{I(t)} & = & \{d \in \top^{I(t)} \mid \exists v > t. (d \in E_2^{I(v)} \wedge \forall w \in (t, v). d \in E_1^{I(w)})\} \\
(E_1 \mathcal{S} E_2)^{I(t)} & = & \{d \in \top^{I(t)} \mid \exists v < t. (d \in E_2^{I(v)} \wedge \forall w \in (v, t). d \in E_1^{I(w)})\}
\end{array}
$$

FIGURE 3. Syntax and semantics of $\mathcal{DLR}_{\mathcal{US}}$.

The language of $\mathcal{DLR}_{\mathcal{US}}$ is interpreted in *temporal models* over $\mathcal{T}$, which are triples of the form $I \doteq \langle \mathcal{T}, \Delta^I, \cdot^{I(t)} \rangle$, where Δ^I is nonempty set of objects (the *domain* of I) and $\cdot^{I(t)}$ an *interpretation function* such that, for every $t \in \mathcal{T}$, every entity E, and every n-ary relation R, we have $E^{I(t)} \subseteq \Delta^I$ and $R^{I(t)} \subseteq (\Delta^I)^n$. The semantics of entity and relation expressions is defined in the lower part of Fig. 3, where $(u, v) = \{w \in \mathcal{T} \mid u < w < v\}$ and the operators $\Box^+$ (always in the future) and $\Box^-$ (always in the past) are the duals of $\Diamond^+$ (some time in the future) and $\Diamond^-$ (some time in the past), respectively, i.e., $\Box^+ E \equiv \neg\Diamond^+\neg E$ and $\Box^- E \equiv \neg\Diamond^-\neg E$, for both entities and relations. For entities, the temporal operators $\Diamond^+$, $\oplus$ (at the next moment), and their past counterparts can be defined via $\mathcal{U}$ and $\mathcal{S}$: $\Diamond^+ E \equiv \top \mathcal{U} E$, $\oplus E \equiv \bot \mathcal{U} E$, etc. However, this is not possible for relations of arity > 1, since $\top_n$—the top $n$ary relation—can be interpreted by different subsets of the n-ary

cross product $\top \times \cdots \times \top$ at different time points.[3] The operators $\Diamond^*$ (at some moment) and its dual $\Box^*$ (at all moments) can be defined for both entities and relations as $\Diamond^* E \equiv E \sqcup \Diamond^+ E \sqcup \Diamond^- E$ and $\Box^* E \equiv E \sqcup \Box^+ E \sqcup \Box^- E$, respectively.

The nontemporal fragment of $\mathcal{DLR}_{\mathcal{US}}$ coincides with $\mathcal{DLR}$. For both entity and relation expressions all the Boolean constructs are available. The selection expression $i/n : E$ denotes an $n$ary relation whose i-th argument ($i \leqslant n$) is of type E; if it is clear from the context, we omit n and write $(i : E)$. The projection expression $\exists^{\geqslant k}[i]R$ is a generalization with cardinalities of the projection operator over the ith argument of the relation R (which coincides with $\exists^{\geqslant 1}[i]R$). It is also possible to use the named attribute version of the model by replacing argument position numbers with *role names*.

Given a formula φ, an interpretation I, and a time point $t \in \mathcal{T}$, the truthrelation $I, t \vDash \varphi$ (φ holds in I at moment t) is defined inductively as follows:

$$
\begin{array}{lll}
I, t \vDash E_1 \sqsubseteq E_2 & \text{iff} & E_1^{I(t)} \subseteq E_2^{I(t)} \\
I, t \vDash R_1 \sqsubseteq R_2 & \text{iff} & R_1^{I(t)} \subseteq R_2^{I(t)} \\
I, t \vDash \varphi \wedge \psi & \text{iff} & I, t \vDash \varphi \text{ and } I, t \vDash \psi \\
I, t \vDash \neg\varphi & \text{iff} & I, t \nvDash \varphi \\
I, t \vDash \varphi \mathcal{U} \psi & \text{iff} & \exists v > t.(I, v \vDash \psi \wedge \forall w \in (t, v).I, w \vDash \varphi) \\
I, t \vDash \varphi \mathcal{S} \psi & \text{iff} & \exists v < t.(I, v \vDash \psi \wedge \forall w \in (t, v).I, w \vDash \varphi)
\end{array}
$$

A formula φ is called *satisfiable* if there is a temporal model I such that $I, t \vDash \varphi$, for some time point t. A conceptual schema Σ is *satisfiable* if the conjunction $\bigwedge \Sigma$ of all formulas in Σ is satisfiable (we write $I, t \vDash \Sigma$ instead of $I, t \vDash \bigwedge \Sigma$); in this case I is called a *model* of Σ. We say that Σ is *globally satisfiable* if there is I such that $I, t \vDash \Sigma$ for every t ($I, t \vDash \Sigma$, in symbols). An entity E (or relation R) is *satisfiable* if there is I such that $E^{I(t)} \neq \emptyset$ (respectively, $R^{I(t)} \neq \emptyset$), for some time point t. Finally, we say that Σ (*globally*) *implies* φ and write $\Sigma \vDash \varphi$ if we have $I \vDash \varphi$ whenever $I \vDash \Sigma$.

Note that an entity E is satisfiable iff $\neg(E \sqsubseteq \bot)$ is satisfiable. An n-ary relation R is satisfiable iff $\neg(\exists^{\geqslant 1}[i]R \sqsubseteq \bot)$ is satisfiable for some $i \leqslant n$. A conceptual schema Σ is globally satisfiable iff $\Box^*(\bigwedge \Sigma)$ is satisfiable. And $\Sigma \vDash \varphi$ iff $\Box^*(\bigwedge \Sigma) \wedge \neg\varphi$ is not satisfiable. Thus, all reasoning tasks connected with the notions introduced above reduce to satisfiability of formulas.

The logic $\mathcal{DLR}_{\mathcal{US}}$ can be regarded as a rather expressive fragment of the first -order temporal logic $L^{\{\text{since,until}\}}$; cf. [ChoSaa98, Hod et al. 2000].

3.3. Temporal queries. One more important reasoning task is known as the problem of query containment (see, e.g., [ChoSaa98, Cho94, Abi et al. 96] for a survey and a discussion about temporal queries). A *non-recursive Datalog query* (i.e., a disjunction of conjunctive queries or SPJqueries) over a $\mathcal{DLR}_{\mathcal{US}}$ schema Σ is an

[3] For instance, we may have $\langle d_1, d_2 \rangle \in (\Diamond^+ R)^{I(t)}$ because $\langle d_1, d_2 \rangle \in (\Diamond^+ R)^{I(t+2)}$, but $\langle d_1, d_2 \rangle \notin (\top_2)^{I(t+1)}$.

expression of the form

$$Q(\vec{x}) :- \bigvee_j Q_j(\vec{x}, \vec{y_j}, \vec{c_j}),$$

where each Q_j is a conjunction of atoms

$$Q_j(\vec{x}, \vec{y_j}, \vec{c_j}) \equiv \bigwedge_i P_j^i(\vec{x_j^i}, \vec{y_j^i}, \vec{c_j^i}),$$

P_j^i are $\mathcal{DLR}_{\mathcal{US}}$ entity or relation expressions, $\vec{x_j^i}$, $\vec{y_j^i}$, and $\vec{c_j^i}$ are sequences of distinguished variables, existential variables, and constants, respectively, the number of which is in agreement with the arity of P_j^i. The variables $\vec{x}$ in the head are the union of all the distinguished variables in each Q_j; the existential variables are used to make coreferences in the query, and constants are fixed values. The arity of Q is the number of variables in $\vec{x}$.

It is to be noted that we allow entities and relations in the query to occur in the conceptual schema Σ. This approach is similar to that of [Cal et al. 98], where atoms in a query can be constrained by means of schema formulas.

The semantics of queries is defined as follows. Let I be a temporal model and t a time point in $\mathcal{T}$ such that I satisfies Σ at t, i.e., $I, t \models \Sigma$. The snapshot interpretation

$$I(t) = \left\langle \Delta^I, \{E^{I(t)} \mid E \in EN\}, \{R^{I(t)} \mid R \in RN\} \right\rangle$$

can be regarded as a usual firstorder structure (i.e., a snapshot nontemporal database at time t conforming in a sense to the conceptual schema), and so the whole I as a first-order temporal model (with constant domain Δ^I in which some values of the query constants are specified). The *evaluation* of a query Q of arity n under the constraints Σ in the model I at moment t is the set

$$\mathsf{ans}(Q, I(t)) = \left\{ \vec{\sigma} \in (\Delta^I)^n \mid I, t \models \bigvee_j \exists \vec{y_j}.Q_j(\vec{\sigma}, \vec{y_j}, \vec{c_j}) \right\}$$

Given two queries (of the same arity) Q_1 and Q_2 over Σ, we say that Q_1 is *contained* in Q_2 under the constraints Σ and write $\Sigma \models Q_1 \subseteq Q_2$ if, for every temporal model I and every time point t, we have $\mathsf{ans}(Q_1, I(t)) \subseteq \mathsf{ans}(Q_2, I(t))$ whenever $I, t \models \Sigma$. Note that the *query satisfiability problem*—given a query Q over a schema Σ, to determine whether there are I and t such that $I, t \models \Sigma$ and $\mathsf{ans}(Q, I(t)) \neq \emptyset$—is reducible to query containment: Q is satisfiable iff $\Sigma \not\models Q(\vec{x}) \subseteq P(\vec{x}) \wedge \neg P(\vec{x})$, where P is a $\mathcal{DLR}_{\mathcal{US}}$relation of the same arity as Q.

3.4. Conceptual Schema and Query Examples. As an example, let us consider the following conceptual schema Σ, where we introduce a shortcut for global atomic formulas $E_1 \sqsubseteq^* E_2 \equiv \Box^*(E_1 \sqsubseteq E_2)$, for both entities and relations:

$$\mathsf{Works\text{-}for} \sqsubseteq^* \mathsf{emp/2 : Employee} \sqcap \mathsf{act/2 : Project}$$

$$\mathsf{Manages} \sqsubseteq^* \mathsf{man/2 : TopManager} \sqcap \mathsf{prj/2 : Project}$$

$$\mathsf{Employee} \sqsubseteq^* \exists^{=1}[\mathsf{from}]\mathsf{PaySlipNumber}$$

$$\sqcap\ \exists^{=1}[\mathsf{from}](\mathsf{PaySlipNumber} \sqcap \mathsf{to/2 : Integer})$$

$$\sqcap\ \exists^{=1}[\mathsf{from}]\mathsf{Salary} \sqcap \exists^{=1}[\mathsf{from}](\mathsf{Salary} \sqcap \mathsf{to}/2 : \mathsf{Integer})$$

$$\top \sqsubseteq^* \exists^{\leqslant 1}[\mathsf{to}](\mathsf{PaySlipNumber} \sqcap \mathsf{from}/2 : \mathsf{Employee})$$

$$\mathsf{Managerv} \sqsubseteq^* \mathsf{Employee} \sqcap (\mathsf{AreaManager} \sqcup \mathsf{TopManager})$$

$$\mathsf{AreaManager} \sqsubseteq^* \mathsf{Manager} \sqcap \neg\mathsf{TopManager}$$

$$\mathsf{TopManager} \sqsubseteq^* \mathsf{Manager} \sqcap \exists^{=1}[\mathsf{man}]\mathsf{Manages}$$

$$\mathsf{Project} \sqsubseteq^* \exists^{\geqslant 1}[\mathsf{act}]\mathsf{Works\text{-}for} \sqcap \exists^{=1}[\mathsf{prj}]\mathsf{Manages}$$

$$\mathsf{Employee} \sqcap \neg(\exists^{\geqslant 1}[\mathsf{emp}]\mathsf{Works\text{-}for}) \sqsubseteq^* \mathsf{Manager}$$

$$\mathsf{Managerv} \sqsubseteq^* \neg(\exists^{\geqslant 1}[\mathsf{emp}]\mathsf{Works\text{-}for}) \sqcap (\mathsf{Qualified}\ \mathcal{S}\ (\mathsf{Employee} \sqcap \neg\mathsf{Manager}))$$

The theory introduces Works-for as a binary relation between Projects and employees, and Manages as a binary relation between managers and projects. Employees have exactly one pay slip number and one salary each, which are represented as binary relations (with from and to roles) with an integer domain; moreover, a pay slip number uniquely identifies an employee (it acts as a key). It is stated that managers are employees, and are partitioned into area managers and top managers. Top Managers participate exactly once in the relation Manages, i.e., every top manager manages exactly one project. Projects participate at least once to the relation Works-for and exactly once in the relation Manages. Finally, employees not working for a project are exactly the managers, and managers should be qualified, i.e., should have passed a period of being employees. The meaning of the above conceptual schema (with the exception of the last two formulas) is illustrated by the left-hand part of the diagram in Fig. 4.

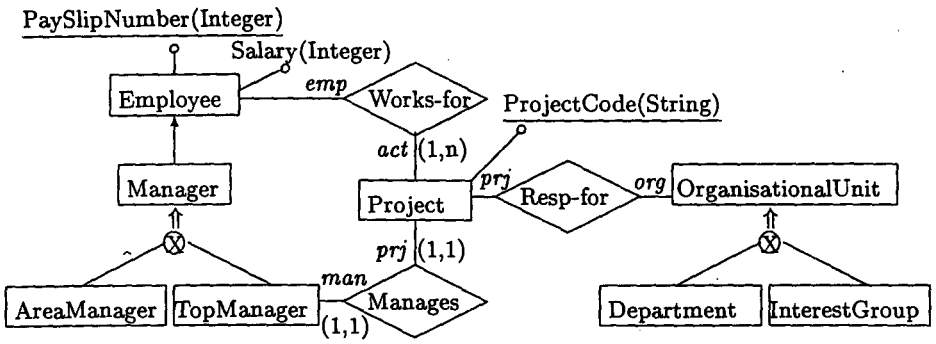


FIGURE 4. The example EER diagram.

The conceptual schema Σ globally logically implies that, for every project, there is at least one employee who is not a manager, and that a top manager worked in a project before managing some (possibly different) project:

$$\Sigma \models \mathsf{Project} \sqsubseteq^* \exists^{\geqslant 1}[\mathsf{act}](\mathsf{Works\text{-}for} \sqcap \mathsf{emp} : \neg\mathsf{Manage})$$

$$\Sigma \models \mathsf{TopManager} \sqsubseteq^* \Diamond^- \exists^{\geqslant 1}[\mathsf{emp}](\mathsf{Works\text{-}for} \sqcap \mathsf{act} : \mathsf{Project})$$

Note also that if we add to Σ the formula

$$\mathsf{Employee} \sqsubseteq^* \exists^{\geqslant 1}[\mathsf{emp}]\mathsf{Works\text{-}for}$$

saying that every employee should work for at least one project, then all the entities and the relations mentioned in the conceptual schema are interpreted as the empty set in every model of Σ, i.e., they are not satisfiable relative to Σ.

The expressivity of the query language can be understood with the following examples:

"Find all people who have worked for only one project"

$$Q(x) : - (\exists^{=1}[\mathsf{emp}](\Diamond^*\mathsf{Works\text{-}for}))(x)$$

"Find all managers whose terminal project has code prj342*"*

$$Q(x) : - \mathsf{Manager(x)} \wedge \mathsf{Manages(x, prj342)} \wedge (\Box^+\neg\mathsf{Manages})(x, y)$$

"Find all projecthoppers—people who never spent more than two consecutive years in a project"

$$Q(x) : - \left(\Box^*\neg\exists^{\geqslant 1}[\mathsf{emp}](\mathsf{Works\text{-}for} \sqcap \oplus\mathsf{Works\text{-}for} \sqcap \oplus \oplus \mathsf{Worksfor})\right)(x)$$

"Find all people who did not work between two projects"

$$Q(x) : - \left(\Diamond^-\exists^{\geqslant 1}[\mathsf{emp}]\mathsf{Works\text{-}for}\right)\mathsf{(x)} \wedge \left(: \exists^{\geqslant 1}[\mathsf{emp}]\mathsf{Works\text{-}for}\right)\mathsf{(x)} \wedge \left(\Diamond^+\exists^{\geqslant 1}[\mathsf{emp}]\mathsf{Works\text{-}for}\right)\mathsf{(x)}$$

We now consider the problem of query containment under constraints, where the constraints are expressed by the above exemplified schema Σ. Consider the following queries

$$\begin{aligned} Q_1(x,y) \;&:-\; \neg\mathsf{AreaManager}(x) \wedge \mathsf{Manages}(x,z) \wedge \mathsf{Project}(z) \wedge \\ &\qquad \mathsf{Resp\text{-}for}(y,z) \wedge \mathsf{Department}(y) \\ Q_2(x,y) \;&:-\; (\Diamond^-\exists^{\geqslant 1}[1]\mathsf{Works\text{-}for})(x) \wedge \mathsf{Manages}(x,z) \wedge \\ &\qquad \mathsf{Resp\text{-}for}(y,z) \wedge \neg\mathsf{InterestGroup}(y) \end{aligned}$$

It is not difficult to see that these two queries are equivalent under the constraints in Σ, i.e., $\Sigma \models Q_1 \subseteq Q_2$ and $\Sigma \models Q_2 \subseteq Q_1$.

3.5. Decidability and complexity. In this section we only summarise the computational behaviour of $\mathcal{DLR}_{\mathcal{US}}$ and its fragments over the flow of time $\langle \mathbb{Z}, < \rangle$. Unfortunately, full $\mathcal{DLR}_{\mathcal{US}}$, even restricted to *atomic* formulas, turns out to be undecidable.

Theorem 8. *The global satisfiability problem for $\mathcal{DLR}_{\mathcal{US}}$ conceptual schemas containing only atomic formulas is undecidable.*

Proof. The proof is by reduction of the well-known undecidable tiling problem [Rob71]: given a finite set of square tiles of fixed orientation and with coloured edges, decide whether it can tile the grid $\mathbb{Z} \times \mathbb{N}$. Suppose $T = \{T_1, \ldots, T_k\}$ is a

set of tiles with colours $left(T_i)$, $right(T_i)$, $up(T_i)$, and $down(T_i)$. Consider the following schema Σ, where $D_1, \dots, D_k$ are concepts and R is a binary relation:

$$
\begin{array}{l}
R \doteq \Box^+ R, \quad R \doteq \Diamond^+ R, \quad \top \doteq \exists R.\top, \\
D_i \sqsubseteq \neg D_j, \quad \top \doteq D_1 \sqcup \dots \sqcup D_k, \text{ for } i \neq j, \\
D_i \sqsubseteq \bigsqcup_{right(T_i)=left(T_j)} \forall R.D_j, \text{ for } i \leqslant k, \\
D_i \sqsubseteq \bigsqcup_{up(T_i)=down(T_j)} \bigcirc D_j, \text{ for } i \leqslant k.
\end{array}
$$

(Here $\exists R.C = \exists^{\geqslant 1}[1](R \sqcap 2/2 : C)$, $\forall R.C = \neg\exists R.\neg C$.) It is readily checked that Σ is globally satisfiable iff T tiles $\mathbb{Z} \times \mathbb{N}$. □

The main technical reason for undecidability is the possibility of temporalising binary relations. The proof uses a very small fragment of $\mathcal{DLR}_{\mathcal{US}}$: even $\mathcal{ALC}$ with $\Box^+$ or one global role is enough to get undecidability. This gives us some grounds to conjecture that already the basic temporal EER model with just snapshot relations is undecidable.

The fragment $\mathcal{DLR}^-_{\mathcal{US}}$, in which the temporal operators can be applied only to entities and formulas, exhibits a much better computational behaviour. In this case we have the following hierarchy:

Theorem 9. *Let the flow of time be $\langle \mathbb{Z}, < \rangle$. Then*

(1) *the problem of logical implication in $\mathcal{DLR}^-_{\mathcal{US}}$ involving only atomic formulas is* ExpTime*-complete;*

(2) *the formula satisfiability problem (and so the problem of logical implication) in $\mathcal{DLR}^-_{\mathcal{US}}$ is* ExpSpace*-complete;*

(3) *the querycontainment problem for nonrecursive Datalog queries under $\mathcal{DLR}^-_{\mathcal{US}}$-constraints is decidable in* 2ExpTime *and is* ExpSpace*hard.*

In the remainder of the section we sketch a proof of this theorem. To make it more transparent, we confine ourselves to considering only the future fragment $\mathcal{DLR}^-_{\mathcal{U}}$ of $\mathcal{DLR}_{\mathcal{US}}$. (From now on $\Box$ stands for $\Box^+$ and $\bigcirc$ for $\oplus$.) The main technical tool in the proof is the method of quasimodels developed in [WolZakh98, WolZakh99b]. The idea behind the notion of a quasimodel is to represent the state of the (in general, infinite) domain of a temporal model at a each moment of time by finitely many types of the domain objects at this moment (modulo a given finite set of formulas); the evolution of types in time is described by special functions called runs.

Suppose that Γ consists of a finite set $f(\Gamma)$ of $\mathcal{DLR}^-_{\mathcal{U}}$-formulas and a finite set $c(\Gamma)$ of concepts, $f(\Gamma)$ is closed under sub-formulas, $c(\Gamma)$ under subconcepts, both are closed under (single) negation, and each concept occurring in $f(\Gamma)$ belongs $c(\Gamma)$. A *concept type* for Γ is a subset t of $c(\Gamma)$ such that

$$
\begin{array}{l}
C \sqcap D \in t \text{ iff } C, D \in t, \text{ for all } C \sqcap D \in c(\Gamma); \\
\neg C \in t \text{ iff } C \notin t, \text{ for all } C \in c(\Gamma).
\end{array}
$$

A *formula type* for Γ is a subset Φ of $f(\Gamma)$ such that

$$\psi \wedge \chi \in \Phi \text{ iff } \psi, \chi \in \Phi, \text{ for all } \psi \wedge \chi \in f(\Gamma);$$
$$\neg\psi \in \Phi \text{ iff } \psi \notin \Phi, \text{ for all } \psi \in f(\Gamma).$$

A pair $\langle T, \Phi\rangle$, where T is a set of concept types and Φ a formula type for Γ, is called a *quasistate candidate* for Γ. We say that the quasistate candidate $\mathfrak{C} = \langle T, \Phi\rangle$ is a *quasistate* for Γ if the following (non-temporal) $\mathcal{DLR}$-formula $\alpha_{\mathfrak{C}}$

$$\Big(\bigsqcup_{t\in T} c(t) \doteq \top\Big) \wedge \bigwedge_{t\in T} \neg(c(t) \doteq \bot) \wedge \bigwedge \Phi$$

is satisfiable. Here $c(t)$ denotes the conjunction of all concepts in t, concepts of the form $C\mathcal{U}D$ are regarded as atomic concepts $A_{C\mathcal{U}D}$, and formulas of the form $\varphi\mathcal{U}\psi$ in Φ are regarded as atomic formulas $A_{\varphi\mathcal{U}\psi} = \top$.

Consider now a sequence of quasistates $Q = \langle Q(n) : n \in \mathbb{Z}\rangle$, where $Q(n) = \langle T_n, \Phi_n\rangle$. A *run* in Q is a sequence $r = \langle r(n) : n \in \mathbb{Z}\rangle$ such that

1. $r(n) \in T_n$ for every $n \in \mathbb{Z}$;
2. for every $C\mathcal{U}D \in c(\Gamma)$ and every $n \in \mathbb{Z}$, we have $C\mathcal{U}D \in r(n)$ iff there is $l > n$ such that $D \in r(l)$ and $C \in r(k)$ for all $k \in (n, l)$.

Finally, Q is called a *quasimodel for* Γ if the following conditions hold:

3. for every $n \in \mathbb{Z}$ and every $t \in T_n$ there is a run r in Q such that $r(n) = t$;
4. for every $\psi\mathcal{U}\chi \in f(\Gamma)$ and every $n \in \mathbb{Z}$, we have $\psi\mathcal{U}\chi \in \Phi_n$ iff there is $l > n$ such that $\chi \in \Phi_l$ and $\psi \in \Phi_k$ for all $k \in (n, l)$.

Given a $\mathcal{DLR}_{\mathcal{U}}^{-}$-formula φ, we denote by $cl(\varphi)$ the closure under (single) negation of the set of subformulas and subconcepts of φ.

Theorem 10. *$\mathcal{DLR}_{\mathcal{U}}^{-}$-formula φ is satisfiable iff there is a quasimodel for $cl(\varphi)$ such that $\varphi \in \Phi_0$.*

Proof. Suppose φ is satisfied in a model I with domain Δ. For every $n \in \mathbb{Z}$, define $Q(n) = \langle T_n, \Phi_n\rangle$ by taking $T_n = \{t^n(x) : x \in \Delta\}$, $\Phi_n = \{\psi \in cl(\varphi) : I, n \models \psi\}$, where $t^n(x) = \{C \in cl(\varphi) : x \in C^{I(n)}\}$. It is easy to see that $\langle Q(n) : n \in \mathbb{Z}\rangle$ is a quasimodel for φ. (Note that the sequence $\langle t^n(x) : n \in \mathbb{Z}\rangle$ is a run through $t^n(x)$, for every $n \in \mathbb{Z}$ and every $x \in \Delta$). To show the converse we require the following lemma.

Lemma 11. *For any cardinal $\kappa \geqslant \aleph_0$ and any quasistate $\mathfrak{C}$ for φ, the formula $\alpha_{\mathfrak{C}}$ is satisfied in a (non-temporal) $\mathcal{DLR}$-model J in which $|[x]^J| = \kappa$ for all x in the domain Δ of J, where $[x]^J = \{y \in \Delta : \forall C \in cl(\varphi)(x \in C^J \Leftrightarrow y \in C^J)\}$.*

Proof. As $\mathcal{DLR}$ is a fragment of first-order logic, we have a countable $\mathcal{DLR}$-model I satisfying $\alpha_{\mathfrak{C}}$. Define J as the disjoint union of κ copies of I; more precisely, let

$$\Delta^J = \{\langle x, \xi\rangle : x \in \Delta^I, \xi < \kappa\},$$
$$P_i^J = \{\langle\langle x_0, \xi\rangle, \ldots, \langle x_n, \xi\rangle\rangle : \langle x_0, \ldots, x_n\rangle \in P_i^I, \xi < \kappa\},$$
$$(\mathsf{T}_n)^J = \{\langle\langle x_0, \xi\rangle, \ldots, \langle x_n, \xi\rangle\rangle : \langle x_0, \ldots, x_n\rangle \in (\mathsf{T}_n)^I, \xi < \kappa\}.$$

It is easy to see that J is as required. □

Suppose now that $\varphi \in \Phi_0$, for a quasimodel Q. Let κ be a cardinal exceeding the cardinality of the set Ω of all runs in Q and $\aleph_0$, and let $\Delta = \{\langle r, \xi\rangle : r \in \Omega, \xi < \kappa\}$.

Note that $|\{\langle r, \xi\rangle \in \Delta : r(n) = t\}| = \kappa$, for every $n \in \mathbb{Z}$ and every $t \in T_n$. By Lemma 11, for every $n \in \mathbb{Z}$ there is a $\mathcal{DLR}$-model $J(n)$ with domain Δ satisfying $\alpha_Q(n)$ and such that $\{C \in cl(\varphi) : \langle r, \xi\rangle \in C^{J(n)}\} = r(n)$, for all $r \in \Omega$ and $\xi < \kappa$. It is easy to see that the temporal $\mathcal{DLR}$-model $I = \langle \mathbb{Z}, \Delta, \cdot^{I(n)}\rangle$ defined by taking $I(n) = J(n)$, for every $n \in \mathbb{Z}$, satisfies φ at moment 0. □

Thus, the satisfiability problem for $\mathcal{DLR}_{\mathcal{U}}^-$-formulas reduces to checking satisfiability in quasimodels. Consider now a $\mathcal{DLR}_{\mathcal{U}}^-$-schema Σ and two queries

$$Q_i(\vec{x}) :- \bigvee_j Q_j^i(\vec{x}, \vec{y_{ij}}, \vec{\omega_{ij}}), \quad i = 1, 2.$$

Denote by $cl(\Sigma, Q_1, Q_2)$ the closure under (single) negation of the set of all formulas and concepts occurring in Σ, Q_1 and Q_2. Given a formula or a concept χ, denote by $\overline{\chi}$ the result of replacing all subformulas (subconcepts) in χ of the form $\chi_1 \mathcal{U} \chi_2$ with $A_{\chi_1 \mathcal{U} \chi_2} = \top$ (respectively, $A_{\chi_1 \mathcal{U} \chi_2}$). Thus, $\overline{\chi}$ is a $\mathcal{DLR}$ formula or concept, and the Q_i are non-temporal $\mathcal{DLR}$-queries.

Theorem 11. *Q_1 is not contained in Q_2 relative to Σ iff there is a quasimodel Q for $cl(\Sigma, Q_1, Q_2)$ such that $\overline{Q_1}$ is not contained in $\overline{Q_2}$ relative to $\overline{\Sigma} \cup \{\overline{\alpha}_{Q(0)}\}$.*

Proof. ($\Rightarrow$) Without loss of generality we may assume that we have a model I such that $I(0) \models \Sigma$ and $\mathsf{ans}(Q_1, I(0)) \not\subseteq \mathsf{ans}(Q_2, I(0))$. Construct a quasimodel Q for $cl(\Sigma, Q_1, Q_2)$ as in the proof of Theorem 10. To show that $\overline{Q_1}$ is not contained in $\overline{Q_2}$ relative to $\overline{\Sigma} \cup \{\overline{\alpha}_{Q(0)}\}$, it is enough to extend the (non-temporal) model $I(0)$ to the new surrogate atoms of the form $A_{C_1 \mathcal{U} C_2}$ and $A_{\chi_1 \mathcal{U} \chi_2}$ in accordance with their behaviour in I at time point 0:

$$A_{C_1 \mathcal{U} C_2}^{I(0)} = (C_1 \mathcal{U} C_2)^{I(0)} \quad \text{and} \quad A_{\chi_1 \mathcal{U} \chi_2}^{I(0)} = \begin{cases} \top, & \text{if } I(0) \models \chi_1 \mathcal{U} \chi_2 \\ \bot, & \text{otherwise.} \end{cases}$$

($\Leftarrow$) is also proved similarly to Theorem 10. The only difference is that now we select $J(0)$ such that $J(0) \models \overline{\Sigma} \wedge \{\overline{\alpha}_{Q(0)}\}$ and $\mathsf{ans}(\overline{Q_1}, J(0)) \not\subseteq \mathsf{ans}(Q_2, J(0))$. □

So, the query-containment problem for $\mathcal{DLR}_{\mathcal{U}}^-$ reduces to satisfiability in quasimodels and the query-containment problem for (non-temporal) $\mathcal{DLR}$. The latter problem was shown to be decidable in 2ExpTime time in [Cal et al. 98]. But how to check satisfiability in quasimodels? First of all, we need a procedure deciding whether a quasistate candidate is a quasistate for a given set of formulas and concepts. The following proposition can be proved using the reduction in [Cal et al. 98].

Proposition 1. (i) *Given a $\mathcal{DLR}_{\mathcal{U}}^-$-formula φ, it is decidable in* NExpTime *whether a quasistate candidate for $cl(\varphi)$ is a quasistate.*

(ii) *Given a $\mathcal{DLR}_{\mathcal{U}}^-$-schema Σ and queries Q_1, Q_2, it is decidable in* 2ExpTime *whether $\overline{Q_1}$ is contained in $\overline{Q_2}$ relative to $\overline{\Sigma} \cup \{\overline{\alpha}_{\mathfrak{C}}\}$ for a quasistate candidate $\mathfrak{C}$ for $cl(\Sigma, Q_1, Q_2)$.*

Now, given a set Γ as defined above, we have at most $O(2^{2^{|\Gamma|}})$ distinct quasistates for Γ. The problem then is whether they can be properly arranged to form a quasimodel for Γ. As we have no past temporal operators, it is enough to consider the flow of time $\langle \mathbb{N}, < \rangle$ and quasimodels of the form $Q = \langle Q(n) : n \in \mathbb{N} \rangle$.

Let Q be a sequence of quasistates $Q(i) = \langle T_i, \Phi_i \rangle$, $i \in \mathbb{N}$, and r a sequence of elements from T_i such that $r(i) \in T_i$. Say that r *realises* $C\mathcal{U}D \in r(n)$ *in* m *steps* if there is $l \leqslant m$ such that $D \in r(n+l)$ and $C \in r(n+k)$ for all $k \in (0, l)$. A formula $\psi \mathcal{U} \chi \in \Phi_n$ *is realised in* m *steps* if there is $l \leqslant m$ such that $\chi \in \Phi_{n+l}$ and $\psi \in \Phi_{n+k}$ for all $k \in (0, l)$. We also say that a pair t, t' of concept types is *suitable* if for every $C\mathcal{U}D \in \Gamma$, $C\mathcal{U}D \in t$ iff either $D \in t'$ or $C \in t'$ and $C\mathcal{U}D \in t'$.

Suppose Q_1 and Q_2 are finite sequences of quasistates for Γ of length l_1 and l_2, respectively, and let $Q = Q_1 * Q_2^*$ (i.e., $Q = Q_1 * Q_2 * Q_2 * Q_2 * \ldots$) with $Q(n) = \langle T_n, \Phi_n \rangle$. One can check that Q is a quasimodel for Γ if the following conditions hold:

(a) for every $i \leqslant l_1 + l_2$ and every $t' \in T_{i+1}$, there is $t \in T_i$ such that the pair t, t' is suitable;
(b) for every $i \leqslant l_1 + 1$ and every $t_i \in T_i$, all concepts of the form $C\mathcal{U}D \in t_i$ are realised in $l_1 + l_2 - i$ steps in some sequence $t_i, t_{i+1}, \ldots, t_{l_1+l_2}$ in which $t_{i+j} \subsetneq T_{i+j}$ and every pair of adjacent elements is suitable;
(c) for every $i \leqslant l_1 + l_2$, and every $\psi \mathcal{U} \chi \in \Gamma$, $\psi \mathcal{U} \chi \in \Phi_i$ iff either $\chi \in \Phi_{i+1}$ or $\psi \in \Phi_{i+1}$ and $\psi \mathcal{U} \chi \in \Phi_{i+1}$;
(d) for every $i \leqslant l_1 + 1$, all formulas of the form $\psi \mathcal{U} \chi \in \Phi_i$ are realised in $l_1 + l_2 - i$ steps.

Moreover, given a quasimodel for Γ, one can always extract from it a subquasimodel $Q = Q_1 * Q_2^*$ which satisfies (a)-(d) above, all quasistates in Q_1 are distinct and $|Q_2| = O(2^{2^{|\Gamma|}})$.

Using this observation together with Proposition 1 one can construct an EXPSPACE formula-satisfiability checking algorithm and a 2EXPTIME query-containment checking algorithm.

A proof of EXPSPACE-hardness of the formula-satisfiability problem we show for a much weaker logic $\mathcal{ALC}_{\mathcal{U}}$.

The primitive symbols of $\mathcal{ALC}_{\mathcal{U}}$ are: *concept names* $C_0, C_1, \ldots$ and *role names* $R_0, R_1, \ldots$. Starting from these we can form compound concepts and formulas using the following constructs. Suppose R is a role name and C, D are concepts (for the basis of our inductive definition we assume concept names to be atomic concepts). Then $\top$, $C \sqcap D$, $\neg C$, $\exists R.C$, and $C\mathcal{U}D$ are *concepts*. *Atomic formulas* are expressions of the form $\top$, and $C = D$. If φ and ψ are formulas then so are $\varphi \wedge \psi$, $\neg \varphi$, and $\psi \mathcal{U} \varphi$.[4]

Lemma 12. *Let the flow of time be* $\mathcal{N} = \langle \mathbb{N}, < \rangle$[5]. *Then the formula satisfiability problem in* $\mathcal{ALC}_{\mathcal{U}}$ *is* EXPSPACE*-hard.*

---

[4]Language $\mathcal{ALC}_{\mathcal{U}}$ is a fragment of $\mathcal{DLR}_{\mathcal{U}}^{-}$ since $\exists R.C$ is abbreviation for $\exists^{\geqslant 1}[1](R \sqcap 2/2 : C)$.

[5]The class of models based on $\langle \mathbb{N}, < \rangle$

In the proof we will use abbreviations (having in mind that we do not use S) $\Box^*\psi = \psi \wedge \Box\psi$, $\Diamond^*\psi = \psi \vee \Diamond\psi$ for formula ψ, and for a concept C abbreviations $\Box^*C = C \sqcap \Box C$, $\Diamond^*C = C \sqcup \Diamond C$; $\forall R.C = \neg\exists R.\neg C$.

Proof. We will show here the lower bound for the satisfaction problem in $\mathcal{N}$ by reducing to it the n-CORIDOR tiling problem, n given in binary, which is known to be EXPSPACE-complete [Boas96]. Namely, for a set $\mathcal{T} = \{t_1, \ldots, t_s\}$ of tiles and $n < \omega$, we construct an $\mathcal{ALC}_U$-formula φ of length $O(n^2 + s^2)$ such that φ is satisfied iff $\mathcal{T}$ tiles $2^n \times m$ rectangle, for some $m < \omega$ in such a way that sides of this rectangle are, say, white.

To encode the 2^n column, we define 2^n concepts B_j, $0 \leqslant j < 2^n$, using n concept names $C_0, \ldots, C_{n-1}$ and a role name R. Let ψ_1 be the conjunction of the following formulas:

$$\exists R.\top = \top, \quad \neg((\neg C_0 \sqcap \cdots \sqcap \neg C_{n-1}) = \bot),$$

$$\bigwedge_{i=0}^{n-1}\Big(\prod_{j=0}^{i-1} C_j \to (C_i \to \forall R.\neg C_i) \sqcap (\neg C_i \to \forall R.C_i) = \top\Big),$$

$$\bigwedge_{i=0}^{n-1}\Big(\bigsqcup_{j=0}^{i-1} \neg C_j \to (C_i \to \forall R.C_i) \sqcap (\neg C_i \to \forall R.\neg C_i) = \top\Big).$$

For any $j \in \{0, \ldots, 2^n - 1\}$ written in binary as $(d_{n-1}, \ldots, d_0)$, we put $B_j = C_0^{d_0} \sqcap \cdots \sqcap C_{n-1}^{d_{n-1}}$, where C^d is C if $d = 1$ and $\neg C$ otherwise. If ψ_1 is satisfied in an $\mathcal{ALC}$-model, then the sets B_j in this model are nonempty, pairwise disjoint and cover the domain of the model.

Let $B, Q_0, \ldots, Q_{n-1}$ be $(n+1)$ new concept names. We use them to encode 2^n sets $[j]$ of worlds containing all w_j for which $w_j \vDash Q_0^{d_0} \sqcap \cdots \sqcap Q_{n-1}^{d_{n-1}} = \top$ and $(d_{n-1}, \ldots, d_0)$ is binary representations of j. B will coincide with B_j in the all worlds from $[j]$. This is ensured by the formula ψ_2:

$$\bigwedge_{i=0}^{n-1}(\Diamond^* C_i = \Box^* C_i) \wedge \Big(B = \prod_{i=0}^{n-1}((C_i \sqcap Q_i) \sqcup (\neg C_i \sqcap \neg Q_i))\Big) \wedge \Box^*(\Diamond^* B = \top)$$

$$\wedge \Box^*\Big(\bigwedge_{i=0}^{n-1}((Q_i = \top) \vee (Q_i = \bot))\Big) \wedge (Z = \top) \wedge \Box^*(Z = \neg Q_0 \sqcap \cdots \sqcap \neg Q_{n-1}).$$

Let $w_0^0 < w_0^1 < \cdots$ be the ordering of worlds in $[0]$. Worlds $w \in [j]$ such that $w_0^i < w < w_0^{i+1}$ will be denoted by w_j^i. For example, we may have:

$$\mathbf{w_0^0} < w_3^0 < w_1^0 < w_2^0 < w_1^0 < \mathbf{w_0^1} < w_2^1 < w_2^1 < w_1^1 < \mathbf{w_0^2} < w_1^2 < w_2^2 < \mathbf{w_0^3} < \mathbf{w_0^4} < \cdots$$

To encode the $2^n \times m$ grid, we will use new concept names D, F, F^u, A, A^u to construct the conjunction ψ_3 of the formulas

$$\Box^*(\neg(D = \bot) \wedge (D \sqcap \Diamond D = \bot))$$

$$\Box^*(F = \Diamond(Z \sqcap \Diamond^* D) \sqcap \neg\Diamond(Z \sqcap \Diamond(Z \sqcap \Diamond^* D)))$$

$$\Box^*(F^u = \Diamond(Z \sqcap \Diamond^* F) \sqcap \neg\Diamond(Z \sqcap \Diamond(Z \sqcap \Diamond^* F)))$$

$$\Box^*\big((A = B \sqcap F) \wedge (A^u = B \sqcap F^u) \wedge \neg(A = \bot)\big)$$
$$\Box^*\big((Z \sqcap F \subseteq \Diamond^* A) \wedge (Z \sqcap F^u \subseteq \Diamond^* A^u)\big).$$

Let

$$F_i = F^{I(w_i^j)} = \bigcup_{w_0^{i+1} \leqslant w < w_0^{i+2}} D^{I(w)} \quad \text{and} \quad A_{ij} = F_i \sqcap B_j.$$

Then $(F^u)^{I(w_j^i)} = F_{i+1}$, $A^{I(w_j^i)} = A_{ij}$ and $(A^u)^{I(w_j^i)} = A_{i+1,j}$. If $\psi_1 \wedge \psi_2 \wedge \psi_3$ is satisfied in a model $\mathcal{N}$, then for every $w \in \mathbb{N}$, there is a unique pair (i,j), $i \in \mathbb{N}$, $j < 2^n$ such that $w = w_j^i$. Conversely, for any such pair (i,j), there is a $w \in \mathbb{N}$ such that $w = w_j^i$.

Let ψ_4 be the conjunction of the following formulas

$$\Box^*((P = \top) \vee (P = \bot)) \wedge \Diamond(Z \sqcap P = \top) \wedge \Box^*((P = \top) \to \Box(P = \bot)),$$
$$\Box^*(S = \Diamond P) \wedge (\Diamond(Z \sqcap \Diamond P) = \bot),$$
$$\Box^*((K = \top) \vee (K = \bot)) \wedge \Diamond(Z \sqcap K = \top) \wedge \Box^*((K = \top) \to \Box(K = \bot)),$$
$$\Box^*(E = \Diamond(Z \sqcap K) \sqcap \neg\Diamond(Z \sqcap \Diamond(Z \sqcap K)), \quad \Box^*(W = \Diamond K \sqcap \neg E).$$

For each tile $t_i \in \mathcal{T}$ we take a concept name T_i. Its intended meaning is as follows: we say that t_k covers an element (i,j) in the grid iff $A_{ij} \subseteq T_k$. We are now in position to guarantee that every element of the grid is covered by precisely one tile and that the colours on adjacent edge of adjacent tiles match.

$$\psi_5 = \bigwedge_{i=1}^{s} (\Diamond^* T_i = \Box^* T_i) \wedge \Big(\bigsqcup_{i=1}^{s} T_i = \top\Big) \wedge \bigwedge_{i \neq j} (T_i \sqcap T_j = \bot) \wedge \Box^* \bigwedge_{i=1}^{s} \big((A \sqcap T_i = \bot) \vee (A \sqcap T_i = A)\big),$$

$$\psi_6 = \Box^* \bigwedge_{k=0}^{n-1} \Big(\Big(\neg Q_k \sqcap \prod_{j=0}^{k-1} Q_j\Big) = \top \to A^r = F \sqcap \prod_{i=0}^{k-1} \neg C_i \sqcap C_k \sqcap \prod_{i=k+1}^{n-1} \big((C_i \sqcap Q_i) \sqcup (\neg C_i \sqcap \neg Q_i)\big)\Big),$$

(If $A = A_{ij}$ in some world, then $A^r = A_{i,j+1}$ in this world.)

$$\psi_7 = \Box^*\Big(Z \sqcap A \subseteq \bigsqcup_{\text{left}(l)=\text{white}} T_l\Big) \wedge \Box^*\Big(\Big(\prod_{i=0}^{n-1} Q_i\Big) \sqcap A \subseteq \bigsqcup_{\text{right}(l)=\text{white}} T_l\Big),$$

$$\psi_8 = \Box^*\Big(S \sqcap A \subseteq \bigsqcup_{\text{down}(l)=\text{white}} T_l\Big) \wedge \Box^*\Big(E \sqcap A \subseteq \bigsqcup_{\text{up}(l)=\text{white}} T_l\Big),$$

$$\psi_9 = \Box^*\Big(\neg\Big(\Big(\prod_{i=0}^{n-1} Q_i\Big) = \top\Big) \to \bigwedge_{j=1}^{s} \Big(W \sqcap A \subseteq T_j \to W \sqcap A^r \subseteq \bigsqcup_{\text{right}(j)=\text{left}(l)} T_l\Big)\Big),$$

$$\psi_{10} = \Box^*\Big(\bigwedge_{j=1}^{s}\Big(W \sqcap A \sqsubseteq T_j \to W \sqcap A^u \sqsubseteq \bigsqcup_{\mathrm{up}(j)=\mathrm{down}(l)} T_l\Big)\Big).$$

One can show that $\varphi = \psi_1 \wedge \cdots \wedge \psi_{10}$ is as required. □

It follows, in particular, that the query-containment problem is EXPSPACE-hard as well. It is an open problem, however, whether there exists an EXPSPACE algorithm deciding this problem.

Finally, we show EXPTIME-completeness of the logical implication for atomic formulas in $\mathcal{DLR}_{\mathcal{U}}^-$ by means of a polynomial reduction of $\mathcal{DLR}_{\mathcal{U}}^-$ to the logic $\mathcal{DLR}_{reg}$ of [Cal et al. 98]. For our purposes, it is enough to know that $\mathcal{DLR}_{reg}$ allows one to form the transitive closure R^* of every binary relation R, and that the satisfiability problem in $\mathcal{DLR}_{reg}$ is in EXPTIME. To simplify the presentation, we reduce here the fragment $\mathcal{DLR}_{\Box\bigcirc}^-$ of $\mathcal{DLR}_{\mathcal{U}}^-$ with the temporal operators $\Box$ and $\bigcirc$ only (the reader should not have problems to extend this reduction to the language with $\mathcal{U}$).

Fix a binary relation R and define a translation $*$ from $\mathcal{DLR}_{\Box\bigcirc}^-$ to $\mathcal{DLR}_{reg}$ as follows: $P^* = P$ for every atom P of $\mathcal{DLR}$, $(\bigcirc C)^* = \forall R.C^*$ and $(\Box C)^* = \forall R^*.C^*$; $*$ commutes with the remaining constructs, and $(P_1 \sqsubseteq P_2)^* = P_1^* \sqsubseteq P_2^*$.

Lemma 13. *Suppose that $\Gamma \cup \{\varphi\}$ is a set of atomic $\mathcal{DLR}_{\Box\bigcirc}^-$-formulas and that R does not occur in $\Gamma \cup \{\varphi\}$. Then $\Gamma \models \varphi$ iff φ^* is a logical consequence of the following set Ξ of $\mathcal{DLR}_{reg}$-formulas*

$$\Gamma^*,\ \exists^{=1}R.\top \doteq \top,\ \exists^{=1}R^-.\top \doteq \top,$$

where $\exists^{=1}R^-.C = \exists^{=1}[2](R \sqcap 1/2 : C)$.

Proof. Suppose $\Gamma \not\models \varphi$. Then there is a model I such that $I, 0 \not\models \varphi$, but $I, n \models \Gamma$ for all $n \in \mathbb{Z}$. Define a $\mathcal{DLR}$-model $J = \langle \Delta', P_1^J, \ldots, R^J\rangle$ by taking $\Delta' = \Delta^I \times \mathbb{Z}$,

$$\langle\langle x_1, n_1\rangle, \ldots, \langle x_l, n_l\rangle\rangle \in P_i^J \text{ iff } n_i = n_j, \text{ for } i, j \leqslant l, \text{ and } \langle x_1, \ldots, x_n\rangle \in P_i^{I(n_1)}$$

$$\langle\langle x_1, n_1\rangle, \langle x_2, n_2\rangle\rangle \in R^J \text{ iff } x_1 = x_2 \text{ and } n_2 = n_1 + 1.$$

It is readily checked that $J \models \Xi$ and $J \not\models \varphi^*$.

Conversely, suppose that $J = \langle \Delta, P_1^J, \ldots, R^J\rangle$ is a model such that $J \models \Xi$ but $J \not\models \varphi^*$. Let $\Sigma = \bigcup\{cl(\chi) : \chi \in \Gamma \cup \{\varphi\}\}$ and, for every $x \in \Delta$,

$$t(x) = \{C \in c(\Sigma) : x \in (C^*)^J\}.$$

Then the pair $\langle T, \Phi\rangle$, where $T = \{t(x) : x \in \Delta\}$ and $\Phi = \{\chi \in f(\Sigma) : J \models \chi^*\}$, is a quasistate for Σ. Define a map Q by taking $Q(n) = \langle T, \Phi\rangle$ for all $n \in \mathbb{Z}$. It is easy to see that Q is a quasimodel. Hence, by Theorem 10, we have a model I such that $I \models \Gamma$ but $I, 0 \not\models \varphi$. □

4. Conclusion

DLs are a family of knowledge representation languages constructed for a wide area of application domains. This paper presents one type of expressive description logic $\mathcal{DLR}_{\mathcal{US}}$, which has been modeled with an aim to overcome problems

of reasoning over conceptual schemas and queries in temporal databases. It is a special type of description logic extended with modal operators. $\mathcal{DLR}_{US}$ is a $\mathcal{DLR}$ description logic with a temporal dimension.

$\mathcal{DLR}$ have been used in the area of non-temporal databases to characterize in a uniform way both conceptual modeling and queries [LevRous98, Cal et al. 98]. Some of interesting properties of $\mathcal{DLR}$ logic are [ArtaFra01]:

- allows the logical reconstruction and the extension of data and knowledge representational tools,
- has an ability to completely define entities and relations as $\mathcal{DLR}$ views over other entities and relation over conceptual schemas
- can express a large class of integrity constraints
- enables a view-based query answering.

Its combination with the propositional temporal logic, enabled with operators *Since* and *Until* [SisCl85, Gab et al. 94] resulted in a $\mathcal{DLR}_{US}$. $\mathcal{DLR}_{US}$ allowed using temporal operators to all syntactical terms of $\mathcal{DLR}$: entities, relations and schemas.

In this paper we presented the syntax and the semantics of $\mathcal{DLR}_{US}$ as well as the solution of the query containment task problem. An example of conceptual schema and query is given. At the end we summarize the computational behavior of $\mathcal{DLR}_{US}$ and its fragments over the flow of time.

References

[Abi et al. 96] S. Abiteboul, L. Herr, J. Van den Bussche, *Temporal versus firstorder logic to query temporal databases*; in: *Proc. 15th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'96)*, 1996, pp 49–57 .

[ArtaFra01] A. Artale, E. Franconi, *A survey of temporal extensions of description logics*, Ann. Math. Art. Intell. **30**(14) (2001).

[Arta et al. 01] A. Artale, E. Franconi, M. Mosurović, F. Wolter, M. Zakharyaschev, *The DLR(US) temporal description logic*; in: Stanford, D. McGuinness, P. Patel-Schneider, C. Goble, R. Moeller (eds), *Proc. Internat. Workshop on Description Logic* (DL 2001), pp. 96–105.

[Arta et al. 02] A. Artale, E. Franconi, F. Wolter, M. Zakharyaschev, *A temporal description logic for reasoning over conceptual schemas and queries*; in: S. Flesca, S. Greco, N. Leone, G. Ianni (eds), *Proc. JELIA '02*, Lect. Notes Comput. Sci. 2424, Springer-Verlag, 2002, pp. 98–110.

[Baa91] F. Baader, *Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles*; in: *Proc. 12th Int. Joint Conf. Artif. Intell. (IJCAI91)*, 1991.

[BaaLau95] F. Baader, A. Laux, *Terminological logics with modal operators*; in: *Proc. 14th Int. Joint Conf. Artif. Intell.*, Montreal, Canada (1995), Morgan Kaufman, 808–814,

[Baa et al. 02] F. Baader, D.L. McGuinness, D.N. Peter, F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation, And Applications*, Cambridge Univ. Press, 2002.

[Baa96] F. Baader, M. Buchheit, B. Hollunder, *Cardinality restrictions on concepts*, Artif. Intell. **88**(12) (1996), 195-213.

[BaaOhl93] F. Baader and H. Ohlbach. *A multi-dimensional terminological knowledge representation language*; in: *Proc. 13th Int. Joint Conf. Artif. Intell.*, 1993, pp. 690–695.

[BaaSat96] F. Baader, U. Sattler, *Number restrictions on complex roles in description logics: A preliminary report*; in: *Proc. 5th Int. Conf. Principles of Knowledge Representation and Reasoning* (KR96), 1996, pp. 328-338.

[BaaSat99] F. Baader, U. Sattler, *Expressive number restrictions in description logics*, J. Logic Comput. **9**(3) (1999), 319-350.

[Boas96] van Emde Boas, *The convenience of tilings*, Tehnical Report CT-96-01, Institute for Logic, Language and Computation, University of Amsterdam.

[BorPat94] R. J. Brachman, H. J. Levesque, *The tractability of subsumption in frame-based description languages*; in: *Proc. 4th Nat. Conf. Artif. Intell. (AAAI84)*, 1984, pp. 34-37.

[Bor96] A. Borgida, *On the relative expressiveness of description logics and predicate logics*, Artif. Intell. **82**(12) (1996), 353-367.

[Bra77] R. J. Brachman, *Whats in a concept: Structural foundations for semantic networks*, Int. J. Man-Machine Studies **9**(2) (1977), 127-152.

[Bra78] R. J. Brachman, *Structured inheritance networks*; in: W. A. Woods and R. J. Brachman, editors, *Research in Natural Language Understanding*, Quarterly Progress Report No. 1, BBN Report No. 3742, pp. 36-78. Bolt, Beranek and Newman, Cambridge, Masssachussets, 1978.

[BraSch85] R. J. Brachman, J. G. Schmolze, *An overview of the KL-ONE knowledge representation system*, Cognitive Sci. **9**(2) (1985), 171-216.

[Buc et al. 93] M. Buchheit, F. M. Donini, A. Schaerf, *Decidable reasoning in terminological knowledge representation systems*, J. Artif. Intell. Research **1** (1993), 109-138.

[Cal et al. 98] D. Calvanese, G. De Giacomo, M. Lenzerini, *On the decidability of query containment under constraints*; in: *Proc. 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems* (PODS'98), pp. 149–158.

[Cho94] J. Chomicki, *Temporal query languages: a survey*; in: *Proc. 1st Internat. Conf. Temporal Logic (ICTL'94)*, pp. 506–534 (1994).

[ChoSaa98] J. Chomicki, G. Saake, editors, *Logics for Databases and Information Systems*, Kluwer, 1998.

[DeG95] G. De Giacomo, *Decidability of Class-Based Knowledge Representation Formalisms*, PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza" (1995).

[Don et al. 97] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, *The complexity of concept languages*, Inform. Comput. **134** (1997), 1-58.

[DonMas00] F. M. Donini, F. Massacci, *EXPtime tableaux for ALC*, Artif. Intell. **124**(1) (2000), 87-138.

[FisLad79] M. J. Fischer, R. E. Ladner, *Propositional dynamic logic of regular programs*, J. Comput. System Sci. **18** (1979), 194-211.

[Fit93] M. Fitting, *Basic modal logic*; in: *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pp. 365-448, Oxford Science Publications (1993).

[Gab72] D. M. Gabbay, *Craigs interpolation theorem for modal logics*; in: *Conference in Mathematical Logic, London* 70, Lect. Notes Math. 255, Springer-Verlag, 1972.

[Gab et al. 94] D. M. Gabbay, I. Hodkinson, M. Reynolds, *Temporal Logic: mathematical Foundations and Computational Aspects*, Oxford University Press, 1994.

[Grä et al. 97] E. Grädel, M. Otto, E. Rosen, *Two-variable logic with counting is decidable*; in: *Proc. 12th IEEE Symp. Logic in Computer Science* (LICS97), pp. 306-317, IEEE Computer Society Press, 1997.

[Hod et al. 2000] I. Hodkinson, F. Wolter, M. Zakharyaschev, *Decidable fragments of firstorder temporal logics*, Ann. Pure Appl. Log. **106** (2000), 85–134.

[Hol96] B. Hollunder, *Consistency checking reduced to satisfiability of concepts in terminological systems*, Ann. Math. Artif. Intell. **18**(24) (1996), 133-157.

[HolNut90] B. Hollunder, W. Nutt, *Subsumption algorithms for concept languages*, Technical Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1990.

[Laux94] A. Laux. *Beliefs in multi-agent worlds: a terminological approach*; in: *Proc. 11th Europ. Conf. Artificial Intelligence*, pp. 299–303, Amsterdam, 1994.

[Leh92] F. Lehmann, *Semantic Networks in Artificial Intelligence*, Pergamon Press, Oxford, 1992.

[LevBra87] H. J. Levesque, R. J. Brachman, *Expressiveness and tractability in knowledge representation and reasoning*, Comput. Intell. **3** (1987), 78-93.

[LevRous98] A. Y. Levy, M-C. Rousset, *Combining horn rules and description logics in CARIN*, Artif. Intell. **104**(1–2) (1998), 165–209.

[Lut99] C. Lutz, *Complexity of terminological reasoning revisited*; in: *Proc. 6th Int. Conf. on Logic for Programming and Automated Reasoning* (LPAR99), Lect. Notes Artif. Intell. 1705, pp. 181-200, Springer-Verlag, 1999.

[LutSat00] C. Lutz, U. Sattler, *The complexity of reasoning with boolean modal logic*; in: *Proc. Advances in Modal Logic (AiML 2000).*

[Lutz et al. 01] C. Lutz, H. Sturm, F. Wolter, M. Zakharyaschev, *Tableaux for temporal description logic with constant domains*; in: *Automated Reasoning*, Lect. Notes. Artif. Intell. 2083, Springer-Verlag, 121–136 (2001).

[Lutz et al. 02] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev, *A tableau decision algorithm for modalized ALC with constant domains*, Studia Logica **72** (2002), 1990–232.

[Mor75] M. Mortimer, *On languages with two variables*, Z. Math. Logik Grundlagen Math. **21** (1975), 135-140.

[Mos2000] M. Mosurović, *Složenost opisnih logika s modalnim operatorima*, Doktorska disertacija, Univerzitet u Beogradu (2000).

[MosZakh99] M. Mosurović, M. Zakharyaschev. *On the complexity of description logics with modal operators*; in: P. Kolaitos and G. Koletos, editors, *Proc. 2nd Panhellenic Logic Symp.*, pp. 166–171, Delphi (1999).

[Neb90] B. Nebel, *Terminological reasoning is inherently intractable*, Artif. Intell. **43** (1990), 235-249.

[Pac97] L. Pacholski, W. Szwast, L. Tendera, *Complexity of two-variable logic with counting*; in: *Proc. 12th IEEE Symp. Logic in Computer Science* (LICS97), pp. 318-327. IEEE Computer Society Press (1997).

[Pra79] V. R. Pratt, *Models of program logic*; in: *Proc. 20th Annual Symp. Foundations of Computer Science* (FOCS79), pp. 115-122 (1979).

[Pra80] V. R. Pratt, *A near-optimal method for reasoning about action*, J. Comput. System Sci. **20** (1980), 231-255.

[Rob71] R. Robinson, *Undecidability and non-periodicity for tilings of the plan*, Invent. Math. **12** (1971), 177-209.

[Sch91] K. Schild, *A correspondence theory for terminological logics: Preliminary report*; in: *Proc. 12th Int. Joint Conf. on Artif. Intell.* (IJCAI91), pp. 466-471 (1991).

[Sch93] K. Schild, *Combining terminological logics with tense logic*; in: *Proc. 6th Portug. Conf. Artif. Intell.*, pp. 105–120, Porto (1993).

[SchSmo91] M. Schmidt-Schauß, G. Smolka, *Attributive concept descriptions with complements*, Artif. Intell. **48**(1) (1991), 1–26.

[Sch94] A. Schaerf, *Reasoning with individuals in concept languages*, Data and Knowledge Engineering **13**(2) (1994), 141-176.

[SisCl85] A. S. Sistla, E. M. Clarke. *The complexity of propositional linear temporal logics*, J. Assoc. Comput. Mach. **32**(3) (1985), 733–749.

[Smo88] G. Smolka, *A feature logic with subsorts*, LILOG report 33, IBM Deutschland, Stuttgart, West Germany (1988).

[StaStu04] S. Staab, R. Studer, *Handbook on Ontologies*, International Handbooks on Information Systems, Springer-Verlag (2004).

[Tob01] S. Tobies, PSPACE *reasoning for graded modal logics*, J. Logic Comput. **11**(1) (2001), 85-106.

[Var85] M. Y. Vardi, *The taming of converse: Reasoning about two-way computations*; in: R. Parikh, editor, *Proc. 4th Workshop on Logics of Programs*, Lect. Notes Comput. Sci. 193, pp. 413-424, Springer-Verlag (1985).

[VarWol86] M. Y. Vardi, P. Wolper, *Automata-theoretic techniques for modal logics of programs*, J. Comput. System Sci. **32** (1986), 183-221; a preliminary version appeared in Proc. 16th ACM SIGACT Symp. on Theory of Computing (STOC84).

[WolZakh98] F. Wolter, M. Zakharyaschev, *Satisfiability problem in description logics with modal operators*; in: *Proc. 6th Internat. Conf. Principles of Knowledge Representation and Reasoning (KR'98)*, pp. 512–523, Trento, Italy (June 1998).

[WolZakh99a] F. Wolter, M. Zakharyaschev. *Modal description logics: Modalizing roles*, Fundam. Inform. **39**(4) (1999), 411–438.

[WolZakh99b] F. Wolter, M. Zakharyaschev, *Multidimensional description logics*; in: *Proc. IJCAI'99*, pp. 104–109 (1999).

[WolZakh99c] F. Wolter, M. Zakharyaschev. *Temporalizing description logics*; in: D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems*, Studies Press–Wiley (1999).

Silvia Ghilezan and Silvia Likavec

COMPUTATIONAL INTERPRETATIONS OF LOGICS

Abstract. The fundamental connection between logic and computation, known as the Curry–Howard correspondence or formulae-as-types and proofs-as-programs paradigm, relates logical and computational systems. We present an overview of computational interpretations of intuitionistic and classical logic:

- intuitionistic natural deduction - λ-calculus
- intuitionistic sequent calculus - λ^{Gtz}-calculus
- classical natural deduction - $\lambda\mu$-calculus
- classical sequent calculus - $\overline{\lambda}\mu\tilde{\mu}$-calculus.

In this work we summarise the authors' contributions in this field. Fundamental properties of these calculi, such as confluence, normalisation properties, reduction strategies call-by-value and call-by-name, separability, reducibility method, λ-models are in focus. These fundamental properties and their counterparts in logics, via the Curry–Howard correspondence, are discussed.

CONTENTS

Introduction

Gentzen's natural deduction is a well established formalism for expressing proofs. The simply typed λ-calculus of Church is a core formalism for writing programs. According to Curry–Howard correspondence, first formulated in 1969 by Howard [101], simply typed λ-calculus represents a computational interpretation of intuitionistic logic in natural deduction style and simplifying a proof corresponds to executing a program.

Griffin extended the Curry–Howard correspondence to classical logic in his seminal 1990 paper [94], by observing that classical tautologies suggest typings for certain control operators. The $\lambda\mu$-calculus of Parigot [121] expresses the content of classical natural deduction and has been the basis of a number of investigations into the relationship between classical logic and theories of control in programming languages [122, 40, 120, 19, 3]. At the same time proof-term calculi expressing a computational interpretation of classical logic serve as tools for extracting the constructive content of classical proofs [118, 6]. The recent interest in the Curry–Howard correspondence for sequent calculus [96, 12, 60, 56] made it clear that the computational content of sequent derivations and cut-elimination can be expressed through various extensions of the λ-calculus. There are several term calculi based on sequent calculus, in which reduction corresponds to cut elimination [97, 147, 34, 151, 107]. In contrast to natural deduction proof systems, sequent calculi exhibit inherent symmetries in proof structures which create technical difficulties in analyzing the reduction properties of these calculi.

In this work we summarise the authors' contributions in this field.

- **Part 1 – Background** gives a brief account on different formulations of intuitionistic and classical propositional logic as well as on λ-calculus and other proof-term calculi which express computational interpretations of logics.
  - Section 1 presents natural deduction and sequent calculus formulations of intuitionistic and classical propositional logic;
  - Section 2-5 present different term calculi that embody proofs in logics: the well-known λ-calculus of Church, $\lambda\mu$-calculus of Parigot [121], $\bar{\lambda}$-

calculus of Herbelin [96], λ^{Gtz}-calculus of Espírito Santo [56] and $\overline{\lambda}\mu\tilde{\mu}$-calculus of Curien and Herbelin [34];
- Section 6 presents the fundamental relation between logic and computation, the Curry–Howard correspondence, which links formulae with types and proofs with terms and programs.

- **Part 2 – Contributions** has five sections and is the main part of this work, concentrating on the authors' contributions in each of the following fields:
  - Section 7 – *Intuitionistic natural deduction and λ-calculus*: summarises the results of Barendregt and Ghilezan [12], Ghilezan [73, 75, 74, 76, 78, 77, 79, 80, 81], Dezani, Ghilezan and Venneri [45], Ghilezan and Likavec [88, 89], Ghilezan and Kunčak [84, 85], Ghilezan, Kunčak and Likavec [86], Likavec [109], Dezani and Ghilezan [41, 43, 42], and Dezani, Ghilezan and Likavec [44];
  - Section 8 – *Intuitionistic sequent calculus and λ^{Gtz}-calculus*: summarises the results of Espírito Santo, Ghilezan and Ivetić [57], and Ghilezan and Ivetić [83];
  - Section 9 – *Classical natural deduction and $\lambda\mu$-calculus*: summarises the results of Herbelin and Ghilezan [98];
  - Section 10 – *Classical sequent calculus and $\overline{\lambda}\mu\tilde{\mu}$-calculus*: summarises the results of Dougherty, Ghilezan and Lescanne [48, 49, 50, 51], Dougherty, Ghilezan, Lescanne and Likavec [52], and Likavec and Lescanne [111];
  - Section 11 – *Application to functional and object-oriented programming languages*: summarises the results of Herbelin and Ghilezan [98], Likavec [110], and Bettini, Bono and Likavec [13, 14, 15, 16, 17, 18].
- **Part 3 – Related work** gives some pointers to the related work in the literature.

Part 1 – Background

1. Natural deduction and sequent calculus

In 1879 Gottlob Frege wrote his *Begriffsschrift* [68] paving a path for modern logic. Frege wanted to show that logic gave birth to mathematics. He invented axiomatic predicate logic, including quantified variables, adding iterations to the previous world of the logical constants *and, or, if... then..., not, some* and *all*. With Frege's "conceptual notation" inferences involving very complex mathematical statements could be represented. He formalised the rule of *modus ponens* using two kinds of judgements: premises and conclusions. Over time, Frege's pictorial notation (see [151] for an example of the original notation) evolved into the notation similar to the one we use today, namely $A \to B$ meaning "A implies B" and $\vdash A$ asserting "A is true". Here is the *modus ponens* rule using this notation

$$\frac{\vdash A \to B \qquad \vdash A}{\vdash B}$$

Axiomatic systems in the Hilbert tradition consist of axioms, modus ponens, and a few other inference rules. Another perspective to capture mathematical reasoning was to describe deduction through inference rules which explain the meaning of the logical connectives and quantifiers.

This giant step in formalizing logic was Gerhard Gentzen's *Unterschungen über das logische Schliessen* [71] written in 1935. In this work, Gentzen introduced the systems of *natural deduction* and *sequent calculus* for propositional and predicate logic, in both *intuitionistic* and *classical* variants. These two systems have the same set of derivable statements. In his work, Gentzen introduced assumptions, so his judgement had the following form:

$$A_1, \ldots, A_n \vdash B$$

meaning "Under the assumption that $A_1, \ldots, A_n$ are true we can conclude that B is true". Using this notation, the modus ponens rule is written as follows

$$\frac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}$$

where Γ and Δ denote sequences of formulae.

1.1. Natural deduction: intuitionistic logic NJ and classical logic NK. We now present the two systems of Gentzen: *natural deduction* for intuitionistic logic, denoted by NJ, and classical logic, denoted by NK, as well as *sequent calculus* for intuitionistic logic, denoted by LJ, and classical logic, denoted by LK. For comprehensive account of the subject we refer the reader to [128].

The set of formulae of propositional logic is given by the following abstract syntax:

$$A, B \;= X \mid A \to B \mid A \wedge B \mid A \vee B \mid \neg A$$

where X denotes an atomic formula and capital Latin letters $A, B, C, \ldots$ denote formulae or single propositions. We will mostly deal with implicational formulae only and sometimes comment on other connectives. Hence, a formula can be one of the following: atomic formula X or implication $A \to B$. Capital Greek letters $\Gamma, \Delta, \ldots$ are used to denote sequences of formulae called antecedents and succedents. Γ, A stands for $\Gamma \cup \{A\}$.

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (\text{axiom})$$

$$\frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\to \text{elim}) \qquad \frac{\Gamma,\ A \vdash B}{\Gamma \vdash A \to B}\ (\to \text{intro})$$

FIGURE 1. NJ: intuitionistic natural deduction

The rules of Gentzen's natural deduction intuitionistic logic NJ and classical logic NK are given in Figures 1 and 2, respectively. Gentzen's system consists of structural and logical rules. The only structural rule is the axiom, whereas each of

$$\frac{}{\Gamma, A \vdash A, \Delta}\ (\text{axiom})$$

$$\frac{\Gamma \vdash A \to B, \Delta \quad \Gamma \vdash A, \Delta}{\Gamma \vdash B, \Delta}\ (\to \text{elim}) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\ (\to \text{intro})$$

FIGURE 2. NK: classical natural deduction

the connectives has introduction and elimination logical rules. Each introduction rule has the connective in the conclusion but not in the premises, whereas each elimination rule has the connective in the premises but not in the conclusion.

The following formulae are provable in classical logic, but not in intuitionistic:

- Pierce's law: $(A \to B) \to A \to A$
- Excluded middle: $A \vee \neg A$
- Double negation: $\neg\neg A \to A$.

The connection between logical connectives in classical logic and their dependencies is well known. As opposed to classical logic, connectives in intuitionistic logic are independent.

1.2. Sequent calculus: intuitionistic logic LJ and classical logic LK. Gentzen introduced the sequent calculus primarily as a tool to prove the consistency of predicate logic. In sequent calculus, a sequent has the form

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_m \quad \text{or shorter} \quad \Gamma \vdash \Delta$$

which corresponds to the formula

$$A_1 \wedge \ldots \wedge A_n \to B_1 \vee \ldots \vee B_m.$$

For each connective, there are left and right logical rules, depending on whether the connective is introduced in antecedent or succedent. The rules of Gentzen's sequent calculus intuitionistic logic LJ and classical logic LK are given in Figures 3 and 4, respectively. Right rules in sequent calculus correspond to introduction rules in natural deduction, whereas left rules correspond to elimination rules. Both natural deduction and sequent calculus can be extended to incorporate other connectives, as well as quantifiers.

The cut rule simplifies and shortens deductions, but at the same time makes it impossible to reconstruct the proofs, since we cannot know which formula was eliminated using the cut rule. Therefore, it is of uttermost importance to know that it is possible to leave out the cut rule and still obtain the system with the same set of derivable statements. This is exactly what Gentzen's *Cut elimination property (Hauptsatz)* proves.

Gentzen also formulated the *subformula property*: given a judgement $\Gamma \vdash A$, its proof can be simplified in such a way that only the propositions appearing in Γ and A and their subformulae appear in the proof of $\Gamma \vdash A$.

$$\frac{A \in \Gamma}{\Gamma \vdash A}\,(\text{axiom})$$

$$\frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \to B \vdash C}\,(\to \text{left}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\,(\to \text{right})$$

$$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\,(\text{cut})$$

FIGURE 3. LJ: intuitionistic sequent calculus

$$\frac{}{\Gamma, A \vdash A, \Delta}\,(\text{axiom})$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\,(\to \text{left}) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\,(\to \text{right})$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\,(\text{cut})$$

FIGURE 4. LK: classical sequent calculus

Theorem (Equivalence).
A formula is derivable in NJ if and only if it is derivable in LJ.
A formula is derivable in NK if and only if it is derivable in LK.

2. λ-calculus

2.1. Untyped λ-calculus. The λ-calculus was originally formalised by Alonzo Church in 1932 [27] as a part of a general theory of functions and logic, in order to establish the limits of what was computable. Later on, it was shown that the full system was inconsistent. But the subsystem dealing with functions only proved to be a successful model for the computable functions and is called the *λ-calculus*.

The λ-calculus is a formal system that is meant to deal with functions and constructions of new functions. Expressions in this theory are called *λ-terms* and each such expression denotes a function. We denote the set of λ-terms by Λ.

Church developed a formalism for defining computable functions using three basic constructions: variables, λ-abstraction, and application, with one reduction rule. The formal syntax of λ-calculus is given by the following:

$$t ::= x \mid \lambda x.t \mid tt$$

where x is a variable, $\lambda x.t$ is a λ-abstraction (which represents a mapping $x \mapsto t$), and tt is the application (which represents application of a function to its argument). For comprehensive account of the subject we refer the reader to [10].

The set $Fv(t)$ of free variables of a λ-term t is defined inductively.

1. $Fv(x) = \{x\}$ 2. $Fv(t_1t_2) = Fv(t_1) \cup Fv(t_2)$ 3. $Fv(\lambda x.t) = Fv(t)\backslash\{x\}$.

The set Λ° of closed lambda terms is the set of lambda terms with no free variables

$$\Lambda^\circ = \{t \in \Lambda \mid Fv(t) = \emptyset\}.$$

The following reduction rule is called the *α-reduction*

$$\lambda x.t \to \lambda y.t[x := y],$$

where all the free occurrences of the variable x in t are replaced with a fresh variable y not occurring in t. The substitution $t_1[x := t_2]$ is not part of the syntax and it is defined so that all the free occurrences of the variable x in t_1 are replaced by t_2, taking into account that the free variables in t_2 remain free in the term obtained.

The main reduction rule of the λ-calculus is the *β-reduction*

$$(\lambda x.t_1)t_2 \to_\beta t_1[x := t_2].$$

A λ-term of the form $(\lambda x.t_1)t_2$ is called a *redex*. The transitive reflexive contextual closure of $\to_\beta$ is denoted by $\twoheadrightarrow_\beta$. The β-equality $=_{(\beta)}$ (β-conversion) is the symmetric transitive closure of $\twoheadrightarrow_\beta$.

The *η-reduction* is given by

$$\lambda x.tx \to_\eta t, \quad x \notin Fv(t),$$

where $\lambda x.tx$ is called an *η-redex*, provided that $x \notin Fv(t)$. The transitive reflexive closure of $\to_\eta$ is denoted by $\twoheadrightarrow_\eta$. The η-equality $=_{(\eta)}$ is the symmetric transitive closure of $\twoheadrightarrow_\eta$. The reductions β and η together generate a reduction denoted by $\twoheadrightarrow$.

This simple syntax equipped with simple reduction rules gives rise to a powerful formal system which is Turing complete. The functions representable in λ-calculus coincide with Turing computable functions and recursive functions.

We give now some of the basic notions that we will use later.

- If $t \equiv \lambda x_1 \dots x_n.(\lambda x.t_0)t_1 \dots t_m$, $n \geqslant 0$, $m \geqslant 1$, then $(\lambda x.t_0)t_1$ is called the *head-redex* of t (Barendregt [10, p. 173]). We write $t \to_h t'$ if t' is obtained from t by reducing the head redex of t (head reduction). We write $t \to_i t'$ if t' is obtained from t by reducing a redex other than the head redex (internal reduction). We also use the transitive closures of these relations, notation $\twoheadrightarrow_h$ and $\twoheadrightarrow_i$, respectively.
- A term is a *normal form* if it does not contain any redex. A term is *normalising* (has a normal form) if it reduces to a normal form. The set of all λ-terms that have a normal form will be denoted by $\mathcal{N}$. All normal forms are of the form:

  $$\lambda y_1 \dots y_n.zt_1 \dots t_k,$$

  where t_i, $1 \leqslant i \leqslant k$, $0 \leqslant k$, are again normal forms, and z can be one of y_j, $1 \leqslant j \leqslant n$, $0 \leqslant n$.

- A term is *strongly normalising* if all its reduction paths end in a normal form (are finite). $\mathcal{SN}$ will denote the set of strongly normalising terms, i.e.,

$$\mathcal{SN} = \{t \in \Lambda \mid \neg(\exists t_1, t_2, \ldots \in \Lambda)\, t \to_\beta t_1 \to_\beta t_2 \to_\beta \ldots\}.$$

- A *head normal form* is a term of the form

$$\lambda x_1 \ldots x_n . y t_1 \ldots t_l,$$

where y can be one of x_i, $1 \leqslant i \leqslant n, 0 \leqslant n$ and $t_j \in \Lambda$, $1 \leqslant j \leqslant l, 0 \leqslant l$.
- A term t is *solvable* (*has a head normal form*) if there exists $t' \in \Lambda$ such that $t \to t'$ and t' is a head normal form. The set of all solvable λ-terms is denoted by $\mathcal{HN}$. A term is *unsolvable* if it is not solvable.
- A term is a *weak head normal form* if it starts with an abstraction, or with a variable. A term is *weakly head normalising* (has a weak head normal form) if it reduces to a weak head normal form. The set of all λ-terms that have a weak head normal form will be denoted by $\mathcal{WN}$.

$$\mathcal{WN} = \{t \in \Lambda \mid (\exists t', t_1, \ldots, t_n \in \Lambda)\, t \twoheadrightarrow_\beta \lambda x.t' \text{ or } t \twoheadrightarrow_\beta x t_1 \ldots t_n\}.$$

- *Church-Rosser theorem (Confluence)*: If $t_1 \twoheadleftarrow t \twoheadrightarrow t_2$, then there exists a λ-term $t_3 \in \Lambda$ such that $t_1 \twoheadrightarrow t_3 \twoheadleftarrow t_2$.

2.2. Typed λ-calculus. In 1940 Church formulated *typed λ-calculus* [28] as a way to avoid the paradoxes existing in other logics. Types are syntactical objects assigned to λ-terms in order to specify the properties of these λ-terms.

The basic type assignment system is the *simply typed λ-calculus* $\lambda \to$, or Curry's type inference system. The types in this system are formed using only the arrow operator $\to$. The application of λ-terms yields the arrow elimination on types, while the abstraction yields the arrow introduction.

The set Type of types is defined as follows.

$$A, B ::= X \mid A \to B$$

where X ranges over a denumerable set $TVar$ of type atoms.

The following notions will be used in our work:

- A *type assignment* is an expression of the form $t : A$, where $t \in \Lambda$ and $A \in \mathsf{Type}$.
- A *context* (*basis*) Γ is a set $\{x_1 : A_1, \ldots, x_n : A_n\}$ of type assignments with different term variables, $Dom\,\Gamma = \{x_1, \ldots, x_n\}$ and $\Gamma \setminus \mathbf{x} = \{A_1, \ldots, A_n\}$. We use capital Greek letters $\Gamma, \Delta, \Gamma_1, \ldots$ to denote contexts.
- A context extension $\Gamma, x : A$ denotes the set $\Gamma \cup \{x : A\}$, where $x \notin Dom\Gamma$.

The type assignment $t : B$ is derivable from the context Γ in the type system $\lambda \to$, notation $\Gamma \vdash t : B$, if $\Gamma \vdash t : B$ can be generated by the axiom and rules given in Figure 5.

We list some of the most important properties of $\lambda \to$. The property of preservation of types under reduction is referred to as *Subject reduction.*

Theorem (Subject reduction). *If $\Gamma \vdash t : A$ and $t \twoheadrightarrow u$, then $\Gamma \vdash u : A$.*

$$\frac{}{\Gamma, x : A \vdash x : A}\,(ax)$$

$$\frac{\Gamma \vdash t_1 : A \to B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1 t_2 : B}\,(\to E) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\,(\to I)$$

FIGURE 5. $\lambda \to$: simply typed λ-calculus

An important property, which might be a reason for introducing types in λ-calculus, is the *strong normalisation* of all typeable terms.

Theorem (Strong normalisation). *If a term is typeable in $\lambda \to$, then it is strongly normalising.*

The correspondence between formulae of intuitionistic logic NJ and types of $\lambda \to$, together with the correspondence of proofs in NJ and terms of $\lambda \to$, was given by Howard [101] based on earlier work of Curry. It is nowadays referred to as the *Curry–Howard correspondence* between logic and computation.

Theorem (Curry–Howard correspondence). $\Gamma \vdash t : A$ *if and only if* $\Gamma \smallsetminus \mathbf{x} \vdash A$ *is derivable in NJ.*

There are many known extensions of $\lambda \to$. Extensions with polymorphic types and dependent types fit perfectly in the so called Barendregt's cube. For comprehensive account of the subject we refer the reader to [9].

2.3. Intersection types for λ-calculus. The extension of $\lambda \to$ which characterises exactly the strongly normalising terms is with intersection types, which are also suitable for analysing λ models and various normalisation properties of λ-terms. The intersection type assignment systems are originated by Coppo and Dezani [29, 30], Barendregt et al. [11], Copo et al. [31], Pottinger [127], and Sallé [134]. In this system, the new type-forming operator is introduced, the intersection $\cap$, whose properties are in accordance with its interpretation as intersection of types. Consequently, it is possible to assign two types A and B to a certain λ-term at the same time. Another outstanding feature of this system is the universal type Ω which can be assigned to all λ-terms. Therefore the question of typability is trivial in these systems.

We focus on the intersection type assignment system $\lambda\cap^{\Omega}$ with the type Ω. The set Type of types in $\lambda\cap^{\Omega}$ is defined as follows

$$A, B ::= X \mid \Omega \mid A \to B \mid A \cap B$$

where X ranges over a denumerable set $TVar$ of type atoms. A type assignment, a context, and a context extension are defined as usual.

The *preorder* on Type is defined in the following way:

(i) The relation $\leqslant$ is defined on Type by the following axioms and rules:

1. $A \leqslant A$ 5. $A \leqslant B,\ A \leqslant C \Rightarrow A \leqslant B \cap C$

$$\frac{}{\Gamma, x : A \vdash x : A}\,(ax) \qquad \frac{}{\Gamma \vdash t : \Omega}\,(\Omega)$$

$$\frac{\Gamma \vdash t_1 : A \to B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1 t_2 : B}\,(\to E) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\,(\to I)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash t : B}{\Gamma \vdash t : A \cap B}\,(\cap I) \qquad \frac{\Gamma \vdash t : A, \quad A \leqslant B}{\Gamma \vdash t : B}\,(\leqslant)$$

FIGURE 6. $\lambda\cap^{\Omega}$: intersection type assignment system

2. $A \leqslant B,\ B \leqslant C \Rightarrow A \leqslant C$
3. $A \cap B \leqslant A,\ A \cap B \leqslant B$
4. $(A \to B) \cap (A \to C) \leqslant A \to B \cap C$
6. $A \leqslant A',\ B \leqslant B' \Rightarrow A \cap B \leqslant A' \cap B'$
7. $A \leqslant A',\ B \leqslant B' \Rightarrow A' \to B \leqslant A \to B'$
8. $A \leqslant \Omega$
9. $A \to \Omega \leqslant \Omega \to \Omega$.

(ii) The induced equivalence relation is defined by:

$$A \sim B \Leftrightarrow A \leqslant B \,\&\, B \leqslant A.$$

The usual axiom of the preorder on intersection types is $\Omega \leqslant \Omega \to \Omega$ (Barendregt et al. [11]). Having this axiom one can distinguish head normalising terms from unsolvable terms by their typeability, but cannot distinguish weakly head normalising terms from unsolvable terms. Instead we adopt the axiom $A \to \Omega \leqslant \Omega \to \Omega$, which allows us to distinguish weakly head normalising from unsolvable terms (Dezani et al. [47]).

The type assignment $t : B$ is derivable from the context Γ in the type system $\lambda\cap^{\Omega}$, notation $\Gamma \vdash t : B$, if $\Gamma \vdash t : B$ can be generated by the axioms and rules given in Figure 6.

The following rule is derivable from the rules given in Figure 6:

$$\frac{\Gamma \vdash t : A \cap B}{\Gamma \vdash t : A\ (B)}\,(\cap E).$$

Some of the type assignment systems that can be obtained by combining the rules above and can be regarded as restrictions of $\lambda\cap^{\Omega}$ are given by the following axioms and rules:

- $\lambda\cap$: (ax), $(\to E)$, $(\to I)$, $(\cap E)$, $(\cap I)$, and $(\leqslant)$.
- $\mathcal{D}$: (ax), $(\to E)$, $(\to I)$, $(\cap E)$, and $(\cap I)$.
- $\mathcal{D}\Omega$: (ax), $(\to E)$, $(\to I)$, $(\cap E)$, $(\cap I)$, and (Ω).

All the eight typed calculi of Barendregt's cube satisfy the strong normalisation property, meaning that typeability in the system implies strong normalisation. A unique property of the two intersection type systems without Ω, namely $\lambda\cap$ and $\mathcal{D}$, is the inverse of strong normalisation property. In these systems all strongly normalising terms are typeable. Thus terms typeable in these systems coincide with strongly normalising terms. This outstanding property of intersection type

systems has merited a lot of attention and has been proven by different authors and different means in [127, 31, 149, 75, 1], the list is not complete.

Theorem (Strong normalisation). *The calculi $\lambda\cap$ and $\mathcal{D}$ satisfy the following*

$$t \text{ is typable } \Leftrightarrow t \text{ is strongly normalising.}$$

There are many known extensions of the λ-calculus with intersection types: Lengrand's et al. calculus with explicit substitutions [108], Matthes's calculus with generalised applications [113], Dougherty's et al. calculus for classical logic [51], Carlier and Wells's and Kfoury and Wells's calculi with expansion variables for type inference [26, 103], Dunfield and Pfenning's calculus with intersection, union, indexed, and universal and existential dependent types [54], to name just a few.

3. λ^{Gtz}-calculus

3.1. Syntax and reduction rules. There were several attempts, over the years, to design a term calculus which would embody the Curry–Howard correspondence for intuitionistic sequent calculus. The first calculus accomplishing this task is Herbelin's $\bar{\lambda}$-calculus given in [96]. Recent interest in the Curry–Howard correspondence for sequent calculus [96, 12, 60, 56] made it clear that the computational content of sequent derivations and cut-elimination can be expressed through an extension of the λ-calculus. The λ^{Gtz}-calculus was proposed by Espírito Santo [56] as a modification of Herbelin's $\bar{\lambda}$-calculus. Its simply typed version corresponds to the sequent calculus for implicational fragment of intuitionistic logic.

The abstract syntax of λ^{Gtz} is given by:

$$\begin{array}{lrcl} \text{(Terms)} & t,u,v & ::= & x \mid \lambda x.t \mid tk \\ \text{(Contexts)} & k & ::= & \widehat{x}.t \mid u :: k. \end{array}$$

Terms are either variables, abstractions or *cuts* tk. A context is either a *selection* or a *context cons(tructor)*. According to the form of k, a cut may be an explicit substitution $t(\widehat{x}.v)$ or a multiary generalised application $t(u_1 :: \cdots u_m :: \widehat{x}..v)$, $m \geqslant 1$. In the last case, if $m = 1$, we get a generalised application $t(u :: \widehat{x}.v)$; if $v = x$, we get a multiary application $t[u_1, \cdots, u_m]$ ($\widehat{x}.x$ can be seen as the empty list of arguments).

In $\lambda x.t$ and $\widehat{x}.t$, t is the scope of the binders λx and $\widehat{x}$, respectively. The scope of binders extends to the right as much as possible.

Reduction rules of λ^{Gtz} are the following:

$$\begin{array}{lrcl} (\beta) & (\lambda x.t)(u :: k) & \to & u(\widehat{x}.tk) \\ (\pi) & (tk)k' & \to & t(k@k') \\ (\sigma) & t\widehat{x}.v & \to & v[x := t] \\ (\mu) & \widehat{x}.xk & \to & k, \text{ if } x \notin k \end{array}$$

where $v[x := t]$ denotes meta-substitution defined as usual, and $k@k'$ is defined by

$$(u :: k)@k' = u :: (k@k') \qquad (\widehat{x}.t)@k' = \widehat{x}.tk'.$$

The rules β, π, and σ reduce cuts to the trivial form $y(u_1 :: \cdots u_m :: \widehat{x}.v)$, for some $m \geqslant 1$, which represents a sequence of left introductions. Rule β generates

a substitution, and rule σ executes a substitution on the meta-level. Rule π generalises the permutative conversion of the λ-calculus with generalised applications. Rule μ has a structural character, and it either performs a trivial substitution in the reduction $t(\widehat{x}.xk) \to tk$, or it minimises the use of the generality feature in the reduction $t(u_1 \cdots u_m :: \widehat{x}.xk) \to t(u_1 \cdots u_m :: k)$.

3.2. Simply typed λ^{Gtz}-calculus. The set Type of types, ranged over by $A, B, C, \ldots, A_1, \ldots$, is defined inductively:

$$A, B \ ::= \ X \mid A \to B$$

where X ranges over a denumerable set $TVar$ of type atoms.

There are two kinds of type assignment:

- $\Gamma \vdash t : A$ for typing terms;
- $\Gamma; B \vdash k : A$ for typing contexts.

The special place between the symbols ; and $\vdash$ is called the *stoup* and was proposed by Girard [93]. Stoup contains a selected formula, the one with which we continue computation.

The type assignment system $\lambda^{\mathsf{Gtz}} \to$ is given in Figure 7.

$$\frac{}{\Gamma, x : A \vdash x : A}\ (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_R) \qquad \frac{\Gamma \vdash t : A \quad \Gamma; B \vdash k : C}{\Gamma; A \to B \vdash t :: k : C}\ (\to_L)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma; A \vdash k : B}{\Gamma \vdash tk : B}\ (Cut) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma; A \vdash \widehat{x}.t : B}\ (Sel)$$

FIGURE 7. $\lambda^{\mathsf{Gtz}} \to$: simply typed λ^{Gtz}-calculus

4. $\lambda\mu$-calculus

4.1. Syntax and reduction rules. The original version of the typed $\lambda\mu$-calculus was formulated by Parigot [121] as the extension of λ-calculus with certain sequential operators and was meant to provide a proof term assignment for classical logic in natural deduction style. As said in [19], "$\lambda\mu$-calculus is a typed λ-calculus which is able to save and restore the runtime environment."

The $\lambda\mu$-calculus was introduced as a call-by-name language, but it received a call-by-value interpretation by Ong and Stewart in [120].

The syntax of the $\lambda\mu$-calculus is given by the following:

$$\begin{array}{lrcl} \text{unnamed terms:} & t & ::= & x \mid \lambda x.t \mid tu \mid \mu\beta.c \\ \text{named terms:} & c & ::= & [\alpha]t. \end{array}$$

We distinguish two kinds of variables: λ-variables $(x, y, \ldots x_1, \ldots)$ and μ-variables $(\alpha, \beta, \ldots \alpha_1, \ldots)$. We also distinguish two kinds of terms: *named* and *unnamed*

terms. Named terms enable us to name arbitrary subterms by μ-variables and refer to them later.

The reduction rules of the $\lambda\mu$-calculus are:

$$\begin{array}{rcl}(\lambda x.u)t & \to & u[x := t] \\ (\mu\beta.c)t & \to & \mu\beta.c[[\beta]w := [\beta](wt)] \\ [\alpha](\mu\beta.c) & \to & c[\beta := \alpha].\end{array}$$

In the second rule, every subterm of c of the form $[\beta]w$ is replaced by a term $[\beta](wt)$.

4.2. Simply typed $\lambda\mu$-calculus. The original version of the $\lambda\mu$-calculus is typed.

A type assignment $t : A$ is derivable from the contexts Γ and Δ in the system $\lambda\mu$, notation

$$\Gamma \vdash t : A, \Delta$$

if $\Gamma \vdash t : A, \Delta$ can be generated by the following axiom and rules given in Figure 8.

$$\frac{}{\Gamma, y : A \vdash y : A, \Delta}\ (\text{axiom})$$

$$\frac{\Gamma \vdash u : A \to B, \Delta \quad \Gamma \vdash t : A, \Delta}{\Gamma \vdash ut : B, \Delta}\ (\to \text{elim}) \qquad \frac{\Gamma, y : A \vdash u : B, \Delta}{\Gamma \vdash \lambda y.u : A \to B, \Delta}\ (\to \text{intro})$$

$$\frac{\Gamma \vdash u : A, \Delta, \beta : A, \alpha : B}{\Gamma \vdash \mu\alpha.[\beta]u : B, \Delta, \beta : A}\ (\mu)$$

FIGURE 8. Simply typed $\lambda\mu$-calculus

The typed calculus is both, strongly normalising and confluent and the types are preserved by the reduction.

5. $\bar{\lambda}\mu\tilde{\mu}$-calculus

5.1. Syntax and reduction rules. The $\bar{\lambda}\mu\tilde{\mu}$-calculus was introduced by Curien and Herbelin in [34].

The untyped version of the calculus can be seen as the foundation of a functional programming language with explicit notion of control and was further studied by Dougherty, Ghilezan, and Lescanne in [87, 48, 49, 51].

The syntax of $\bar{\lambda}\mu\tilde{\mu}$ is given by the following, where v ranges over the set Term of terms, e ranges over the set Coterm of coterms and c ranges over the set Command of commands:

$$t ::= x \mid \lambda x\,.\,t \mid \mu\alpha\,.\,c \qquad e ::= \alpha \mid t \bullet e \mid \tilde{\mu}x\,.\,c \qquad c ::= \langle t \parallel e \rangle.$$

There are two kinds of variables in the calculus: the set Var_v of variables (denoted by Latin letters $x, y, \ldots$) and the set Var_e of covariables (denoted by Greek letters $\alpha, \beta, \ldots$). The variables can be bound by λ-abstraction or by $\tilde{\mu}$-abstraction, whereas the covariables can be bound by μ-abstraction. The sets of free variables

and covariables, Fv_t and Fv_e, are defined as usual, respecting Barendregt's convention [10] that no variable can be both, bound and free, in the expression.

Terms yield values, while coterms consume values. A command is a cut of a term against a coterm. Commands are the place where terms and coterms interact. The components can be nested and more processes can be active at the same time.

There are only three rules that characterise the reduction in $\overline{\lambda}\mu\widetilde{\mu}$:

$$\begin{array}{lrcl} (\rightarrow') & \langle \lambda x\,.\,t_1 \parallel t_2 \bullet e\rangle & \rightarrow & \langle t_2 \parallel \widetilde{\mu}x.\langle t_1 \parallel e\rangle\rangle \\ (\mu) & \langle \mu\alpha\,.\,c \parallel e\rangle & \rightarrow & c[\alpha := e] \\ (\widetilde{\mu}) & \langle t \parallel \widetilde{\mu}x\,.\,c\rangle & \rightarrow & c[x := t]. \end{array}$$

The above substitutions are defined as to avoid variable capture [10].

As a rewriting calculus $\overline{\lambda}\mu\widetilde{\mu}$ has a critical pair $\langle \mu\alpha\,.\,c_1 \parallel \widetilde{\mu}x\,.\,c_2\rangle$ where both, (μ) and $(\widetilde{\mu})$ rule can be applied non-deterministically, producing two different results. For example,

$$\langle \mu\alpha.\langle y \parallel \beta\rangle \parallel \widetilde{\mu}x.\langle z \parallel \gamma\rangle\rangle \rightarrow_{\mu} \langle y \parallel \beta\rangle \quad \text{and} \quad \langle \mu\alpha.\langle y \parallel \beta\rangle \parallel \widetilde{\mu}x.\langle z \parallel \gamma\rangle\rangle \rightarrow_{\widetilde{\mu}} \langle z \parallel \gamma\rangle,$$

where α and β denote syntactically different covariables.

Hence, the calculus is not confluent. But if the priority is given either to (μ) or to $(\widetilde{\mu})$ rule, we obtain two confluent subcalculi $\overline{\lambda}\mu\widetilde{\mu}_T$ and $\overline{\lambda}\mu\widetilde{\mu}_Q$. There are two possible reduction strategies in the calculus that depend on the orientation of the critical pair. If the priority is given to (μ) redexes, call-by-value reduction is obtained ($\overline{\lambda}\mu\widetilde{\mu}_Q$-calculus), whereas giving the priority to $(\widetilde{\mu})$ redexes, simulates call-by-name reduction ($\overline{\lambda}\mu\widetilde{\mu}_T$-calculus).

This is more than simply a reflection of the well-known fact that the equational theories of call-by-name and call-by-value differ. It is a reflection of the great expressive power of the language: a single expression containing several commands can encompass several complete computational processes, and the μ and $\widetilde{\mu}$ reductions allow free transfer of control between them.

We first give the syntactic constructs of $\overline{\lambda}\mu\widetilde{\mu}_T$ and $\overline{\lambda}\mu\widetilde{\mu}_Q$, respectively:

$\overline{\lambda}\mu\widetilde{\mu}_T$		$\overline{\lambda}\mu\widetilde{\mu}_Q$	
$c ::=$	$\langle t \parallel e\rangle$	$c ::=$	$\langle t \parallel e\rangle$
$t ::=$	$x \mid \lambda x\,.\,t \mid \mu\alpha\,.\,c$	$V ::=$	$x \mid \lambda x\,.\,t$
$E ::=$	$\alpha \mid t \bullet E$	$t ::=$	$\mu\alpha\,.\,c \mid V$
$e ::=$	$\widetilde{\mu}x\,.\,c \mid E$	$e ::=$	$\alpha \mid \widetilde{\mu}x\,.\,c \mid V \bullet e.$

In $\overline{\lambda}\mu\widetilde{\mu}_T$ the new syntactic subcategory E of coterms, called *applicative contexts*, is introduced, in order to model call-by-name reduction. In $\overline{\lambda}\mu\widetilde{\mu}_Q$, notice the presence of the new syntactic construct V that models the *values*.

The reduction rules for $\overline{\lambda}\mu\tilde{\mu}_T$ and $\overline{\lambda}\mu\tilde{\mu}_Q$ are the following:

$$\begin{array}{lrcl}
\overline{\lambda}\mu\tilde{\mu}_T & & & \\
(\rightarrow) & \langle \lambda x\,.\,t_1 \parallel t_2 \bullet E\rangle & \rightarrow & \langle t_1[x \leftarrow t_2] \parallel E\rangle \\
(\mu) & \langle \mu\alpha\,.\,c \parallel E\rangle & \rightarrow & c[\alpha := E] \\
(\tilde{\mu}) & \langle t \parallel \tilde{\mu}x\,.\,c\rangle & \rightarrow & c[x := t]
\end{array}$$

$$\begin{array}{lrcl}
\overline{\lambda}\mu\tilde{\mu}_Q & & & \\
(\rightarrow') & \langle \lambda x\,.\,t_1 \parallel V_2 \bullet e\rangle & \rightarrow & \langle V_2 \parallel \tilde{\mu}x.\langle t_1 \parallel e\rangle\rangle \\
(\mu) & \langle \mu\alpha\,.\,c \parallel e\rangle & \rightarrow & c[\alpha := e] \\
(\tilde{\mu}) & \langle V \parallel \tilde{\mu}x\,.\,c\rangle & \rightarrow & c[x := V].
\end{array}$$

Notice that in [34] only the rule $(\rightarrow')$ is considered for both subcalculi. In [87, 48, 49, 51] only the rule $(\rightarrow)$ is used. In [111, 110] $(\rightarrow)$ reduction is used rather than $(\rightarrow')$ reduction in the case of $\overline{\lambda}\mu\tilde{\mu}_T$, since the application of the $(\rightarrow')$ rule will always be immediately followed by the application of the $(\tilde{\mu})$ rule and that is exactly the rule $(\rightarrow)$. This choice makes explicit the priorities of the rules in each subcalculus.

In their original work on the $\overline{\lambda}\mu\tilde{\mu}$-calculus [34], Curien and Herbelin defined a call-by-name and a call-by-value cps-translations of the complete *typed* $\overline{\lambda}\mu\tilde{\mu}$-calculus into simply typed λ-calculus. The important point to notice is that they also interpret the types of the form $A-B$, which are dual to the arrow types $A \rightarrow B$. The translations validate call-by-name and call-by-value discipline, respectively.

In addition, as described in [34], the sequent calculus basis for $\overline{\lambda}\mu\tilde{\mu}$ supports the interpretation of the reduction rules of the system as operations of an abstract machine. In particular, the right- and left-hand sides of a sequent directly represent the *code* and *environment* components of the machine. This perspective is elaborated more fully in [32]. See [33] for a discussion of the importance of symmetries in computation.

5.2. Simply typed $\overline{\lambda}\mu\tilde{\mu}$-calculus. The set Type of types for the $\overline{\lambda}\mu\tilde{\mu}$-calculus is obtained by closing a set of base types X under implication

$$A, B ::= X \mid A \rightarrow B.$$

Type bases have two components, the *antecedent* a set of bindings of the form $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, and the *succedent* of the form $\Delta = \alpha_1 : B_1, \ldots, \alpha_k : B_k$, where x_i, α_j are distinct for all $i = 1, \ldots, n$ and $j = 1, \ldots, k$. The judgements of the type system are given by the following:

$$\Gamma \vdash r : A \mid \Delta \qquad \Gamma \mid e : A \vdash \Delta \qquad c : (\Gamma \vdash \Delta)$$

where Γ is the antecedent and Δ is the succedent. The first judgement is a typing for a term, the second one is a typing for a coterm and the third one is a typing for a command. The box denotes a distinguished output or input, i.e., a place where the computation will continue or where it happened before. The type assignment system for the $\overline{\lambda}\mu\tilde{\mu}$-calculus, introduced by Curien and Herbelin in [34], is given in Figure 9.

$$\frac{}{\Gamma, x:A \vdash x:A \mid \Delta}\,(axR) \qquad \frac{}{\Gamma \mid \alpha:A \vdash \alpha:A, \Delta}\,(axL)$$

$$\frac{\Gamma \vdash r:A \mid \Delta \quad \Gamma \mid e:B \vdash \Delta}{\Gamma \mid r \bullet e:A \to B \vdash \Delta}\,(\to L) \qquad \frac{\Gamma, x:A \vdash r:B \mid \Delta}{\Gamma \vdash \lambda x.r:A \to B \mid \Delta}\,(\to R)$$

$$\frac{c:(\Gamma \vdash \alpha:A, \Delta)}{\Gamma \vdash \mu\alpha.c:A \mid \Delta}\,(\mu) \qquad \frac{c:(\Gamma, x:A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c:A \vdash \Delta}\,(\tilde{\mu})$$

$$\frac{\Gamma \vdash r:A \mid \Delta \quad \Gamma \mid e:A \vdash \Delta}{\langle r \parallel e \rangle : (\Gamma \vdash \Delta)}\,(cut)$$

FIGURE 9. Simply typed $\overline{\lambda}\mu\tilde{\mu}$-calculus

6. Curry–Howard correspondence

The fundamental connection between logic and computation is given by Curry–Howard correspondence or formulae-as-types, proofs-as-terms, proofs-as-programs interpretation. It relates many computational and logical systems and can be applied to intuitionistic and classical logic, to sequent calculus and natural deduction.

Under the traditional Curry–Howard correspondence formulae provable in intuitionistic natural deduction coincide with types inhabited in simply typed λ-calculus. This was observed already by Curry, first formulated by Howard in 1969 [101], used extensively by de Bruijin in the Automath project and by Lambek in category theory. This correspondence extends to all eight calculi of Barendregt's cube and corresponding logical systems. We refer the reader to [139] for an extensive account of this topic.

Only in 1990 Griffin [94] showed that this correspondence can be extended to classical logic, pointing out that classical tautologies suggest typings for certain control operators: the Pierce's Law corresponds to the type of `call-cc` operator in Scheme (introduced by Sussman and Steele [143]) and the Law of Double Negation corresponds to the type of C operator (introduced by Felleisen et al. [63, 64]).

Extensive research in both natural deduction and sequent calculus formulations of classical logic followed. One of the cornerstones is the $\lambda\mu$-calculus of Parigot [121] which corresponds to classical natural deduction. It was followed by term calculi designed to incorporate classical sequent calculus: the Symmetric Lambda Calculus of Barabanera and Berardi [6], the $\overline{\lambda}\mu\tilde{\mu}$-calculus of Curien and Herbelin [34], and the Dual calculus of Wadler [151, 152].

Part 2 – Contributions

In this part we give an overview of the work done by the authors in the field of computational interpretations of logics. In Section 7 we focus on intuitionistic

natural deduction and the λ-calculus. In Section 8 we deal with intuitionistic sequent calculus and the λ^{Gtz}-calculus of [56]. In Sections 9 and 10 we concentrate on classical logic: the $\lambda\mu$-calculus [121], proof term assignment for classical natural deduction; and three proof term calculi for classical sequent calculus: the $\overline{\lambda}\mu\tilde{\mu}$-calculus [34], the dual calculus [151, 152] and the Symmetric Lambda Calculus [6]. Finally, in Section 11 we turn to application to programming language theory.

7. Intuitionistic natural deduction and λ-calculus

7.1. Terms for natural deduction and sequent calculus intuitionistic logic.

The correspondence between sequent calculus derivations and natural deduction derivations is not a one-to-one map: several cut-free derivations correspond to one normal derivation. In Barendregt and Ghilezan [12] this is explained by two extensionally equivalent type assignment systems for untyped λ-terms, namely λN and λL, one corresponding to intuitionistic natural deduction NJ and the other to intuitionistic sequent calculus LJ. These two systems constitute different grammars for generating the same (type assignment relation for untyped) λ-terms. Moreover, the second type assignment system has a 'cut-free' fragment (λL^{cf}) which generates exactly the typeable λ-terms in normal form. The cut elimination theorem becomes a simple consequence of the fact that typed λ-terms posses a normal form.

There are three type systems that assign types to untyped λ-terms:

- λN is the simply typed λ-calculus, $\lambda\!\!\rightarrow$, given in Figure 5;
- λL given in Figure 10;
- λL^{cf}, the cut-free fragment of λL (rules of Figure 10 without the (cut) rule).

The last two systems have been described by Gallier [70], Barbanera et al. [8], and Mints [114]. The three systems λN, λL, and λL^{cf} correspond exactly to the intuitionistic natural deduction NJ (Figure 1), the intuitionistic sequent calculus LJ (Figure 3), and the cut-free fragment of LJ. We denote NJ, LJ, and cut-free LJ by N, L and L^{cf} respectively.

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x:A}\ (\text{axiom})$$

$$\frac{\Gamma\vdash s:A \quad \Gamma,x:B\vdash t:C}{\Gamma,y:A\to B\vdash t[x:=ys]:C}\ (\to \text{left}) \qquad \frac{\Gamma,x:A\vdash t:B}{\Gamma\vdash \lambda x.t:A\to B}\ (\to \text{right})$$

$$\frac{\Gamma,\vdash s:A \quad \Gamma,x:A\vdash t:B}{\Gamma\vdash t[x:=s]:B}\ (\text{cut})$$

FIGURE 10. λL-calculus

First we show the known relation between derivation in N and L: for all Γ and A the following holds

$$\Gamma\vdash_N A \Longleftrightarrow \Gamma\vdash_L A.$$

The following result was observed for N and λN by Curry, Howard, de Bruijn and Lambek. It is referred to as the Curry–Howard, formulae-as-types, proofs-as-terms and proofs-as-programs correspondence (interpretation, paradigm).

Theorem (Curry–Howard correspondence). *Let S be one of the logical systems N, L or L^{cf} and let λS be the corresponding type assignment system. Then*

$$\Gamma \setminus \mathbf{x} \vdash_S A \iff \exists t \in \Lambda^\circ(\mathbf{x})\ \Gamma \vdash_{\lambda S} t : A.$$

where $\Lambda^\circ(\mathbf{x}) = \{t \in \Lambda \mid Fv(t) \subseteq \mathbf{x}\}$.

The proof of the equivalence between systems N and L can be 'lifted' to that of λN and λL, i.e.,

$$\Gamma \vdash_{\lambda L} t : A \iff \Gamma \vdash_{\lambda N} t : A.$$

Finally, using the cut-free system we get as bonus the Hauptsatz of [71] for minimal implicational sequent calculus, i.e.,

$$\Gamma \vdash_L A \iff \Gamma \vdash_{L^{cf}} A.$$

The main contribution of this work is expository, since it deals with well known results. In this work, the emphasis is on λ-terms rather than on derivations, since λ-terms are easier to reason about than two dimensional derivations.

7.2. Logical interpretation of intersection types. In Ghilezan [72] we consider the inhabitation in intersection and union type assignment system versus provability in intuitionistic (Heyting's) natural deduction propositional logic NJ with conjunction and disjunction (as given in Section 1.1, where the language of NJ contains also the constant $\top$).

$$\frac{\Gamma, x : A \vdash t_1 : C \quad \Gamma, x : B \vdash t_1 : C \quad \Gamma \vdash t_2 : A \cup B}{\Gamma \vdash t_1[x := t_2] : C}\ (\cup E)$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : A \cup B} \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash t : A \cup B}\ (\cup I)$$

FIGURE 11. $\lambda \cap \cup$: intersection and union type assignment system

The intersection and union type assignment system $\lambda\cap\cup$ is obtained by extending the system $\lambda\cap^\Omega$ with the rules given in Figure 11 where a pre-order $\leqslant$ on $\lambda \cap \cup$ is the extension of the pre-order $\leqslant$ on $\lambda\cap^\Omega$ obtained by adding the following rules: (i) $A \leqslant A \cup B, B \leqslant A \cup B$, (ii) $A \cup A \leqslant A$, (iii) $A \leqslant C, B \leqslant C \Rightarrow A \cup B \leqslant C$, and (iv) $A \cap (B \cup C) \leqslant (A \cap B) \cup (A \cap C)$.

The Curry–Howard correspondence between types inhabited in the intersection and union type assignment system and formulae provable in intuitionistic propositional logic with implication, conjunction, disjunction, and truth does not hold. Inhabitation implies provability, but there are provable formulae which are not inhabited. This is shown in Hindley [99] in a syntactical way. We give a semantical proof of this fact by giving the appropriate type interpretations in $\mathcal{P}(D)$, starting

from any lambda model $\mathcal{M} = \langle D, \cdot, [\![\ ]\!] \rangle$ (see [11]) and by mapping the set of intersection and union types into the set of propositional formulae that replaces each occurrence of $\cap$, $\cup$, and Ω in a type by $\wedge$, $\vee$, and $\top$ respectively.

The fact that types inhabited in $\lambda\cap^{\Omega}$ do not correspond to the provable formulae in intuitionistic propositional logic with $\rightarrow$ and $\wedge$, was shown in Hindley [99] by showing that the type

$$(A \rightarrow A) \cap ((A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C)$$

is not inhabited in $\lambda\cap^{\Omega}$ although it is provable in intuitionistic logic.

To show that some provable formulae are not inhabited we construct a model of $\lambda\cap^{\Omega}$ which is not a model of some provable formula, i.e., its interpretation in this model is empty.

In order to obtain the Curry–Howard correspondence for intuitionistic propositional logic LJ with conjunction and disjunction, we consider the extension of the simply typed lambda calculus with conjunction and disjunction types and the corresponding elimination and introduction rules, given in Figure 12. For this purpose, the set Type of types is given by the following

$$A, B = X \mid A \rightarrow B \mid A \wedge B \mid A \vee B$$

and the set of lambda terms Λ_c is obtained by expanding the set Λ with new constants c, c_1, and c_2 for conjunction and d, d_1, and d_2 for disjunction. $\lambda^{\rightarrow}_{\wedge}$ denotes the type assignment system obtained from $\lambda \rightarrow$ by adding the rules considering conjunction and λ_c denotes the type assignment system obtained from $\lambda \rightarrow$ by adding the rules considering conjunction and disjunction.

$$\frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash c_1 t : A} \qquad \frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash c_2 t : B} \ (\wedge E)$$

$$\frac{\Gamma \vdash t_1 : A \quad \Gamma \vdash t_2 : B}{\Gamma \vdash c t_1 t_2 : A \wedge B} \ (\wedge I)$$

$$\frac{\Gamma, x : A \vdash t_1 : C \quad \Gamma, x : B \vdash t_2 : C \quad \Gamma \vdash t_3 : A \cup B}{\Gamma \vdash d x t_1 t_2 t_3 : C} \ (\vee E)$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash d_1 t : A \vee B} \qquad \frac{\Gamma \vdash t : B}{\Gamma \vdash d_2 t : A \vee B} \ (\vee I)$$

FIGURE 12. λ_c: type assignment system with conjunction and disjunction

We link the inhabitation in the intersection and union type assignment system with the inhabitation in this extension of the simply typed lambda calculus. We prove that inhabitation is decidable in $\lambda^{\rightarrow}_{\wedge}$ and λ_c by linking them to the question of decidability of provability in logics.

The difference between the special conjunction $\cap$ (called intersection) and the arbitrary propositional conjunction $\wedge$ is in the rule $(\cap I)$. In order to show that

the term t has the intersection type it is necessary to show that t has both types in the same basis. t is the same in the conclusion as in both premises of the rule $(\cap I)$. The same holds for the rule $(\cap E)$. Thus in these two steps t remains the same although the deduction grows and the λ-terms do not correspond to the deductions. With the usual propositional conjunction $\wedge$ the lambda terms correspond to the deductions since it is possible to obtain a term of conjunction type from two terms with different types. Something similar happens with the special disjunction $\cup$ (called union).

In Dezani, Ghilezan and Venneri [45] we consider intersection and union types in Combinatory logic, which is a formal system equivalent to λ-calculus. In [45] we investigate the Curry–Howard correspondence between Hilbert (axiomatic) style intuitionistic logic and Combinatory logic. We propose a typed version of Combinatory logic with intersection and union types. This was a novelty, since all the existing systems with intersection types up to 1990s were type assignment systems. For the difference between typed systems (typeability *à la Church*) and type assignment systems (typeability *à la Curry*) we refer the reader to Barendregt [9]. Different typed lambda calculi with intersection types were further proposed by Liquori and Ronchi Della Rocca [23] and Bono et al. [112].

7.3. Intersection types and topologies in λ-calculus. In Ghilezan [80] typeability of terms in the full intersection type assignment system $\lambda\cap^{\Omega}$ is used to introduce topologies on the set of lambda terms Λ. We consider sets of lambda terms that can be typeable by a given type in a given environment:

$$\mathcal{V}_{\Gamma,A} = \{t \in \Lambda \mid \Gamma \vdash t : A\}.$$

For a fixed Γ the family of sets $\{\mathcal{V}_{\Gamma,A}\}_{A\in\mathsf{Type}}$ forms the basis of a topology on Λ, called the Γ-*fit topology*. Open sets in the Γ-fit topology are unions of basic open sets.

These topologies lead to simple proofs of some fundamental results of the lambda calculus such as the continuity theorem and the genericity lemma. We show that application is continuous, unsolvable terms are bottoms, and $\beta\eta$-normal forms are isolated points with respect to these topologies.

The restriction of these topologies to the set of closed lambda terms Λ°, called the *fit topology*, appears to be unique. It is defined by considering the set of all closed lambda terms that can be typed by a given type:

$$\mathcal{V}_{A} = \{t \in \Lambda^{\circ} \vdash t : A\}.$$

The family $\{\mathcal{V}_{A}\}_{A\in\mathsf{Type}}$ forms a basis for a topology on Λ°.

We compare the fit topology and the filter topology [11] and show that: (i) they coincide on the set Λ° of closed λ-terms, (ii) for every Γ-fit topology on the set Λ there is a coincident topology on Λ and vice versa.

The fit topology is a simpler description of the filter topology since the main difference between these topologies is that the former is a topology introduced on the set of types and then traced on terms by the inverse map, whereas the latter is introduced directly on the set of terms.

7.4. Reducibility method. The *reducibility method* is a well known framework for proving reduction properties of λ-terms typeable in different type systems. It was introduced by Tait [144] for proving the strong normalisation property of simply typed λ-calculus. Later it was used to prove *strong normalisation property* of various type systems in [145, 92, 105, 75], *the Church-Rosser property* (confluence) of $\beta\eta$-reduction in [104, 141, 115, 116] and to characterise some special classes of λ-terms such as strongly normalising terms, normalising terms, head normalising terms, and weak head normalising terms by their typeability in various intersection type systems in [69, 47, 41].

In Ghilezan and Likavec [88] we develop a general reducibility method for proving reduction properties of λ-terms typeable in intersection type systems with and without the universal type Ω, whereas in [89] we focus only on the intersection type assignment system $\lambda\cap^{\Omega}$ with the type Ω. Sufficient conditions for its application are derived. This method leads to uniform proofs of confluence, standardization, and weak head normalisation of terms typeable in the system with the type Ω. In this system the reducibility method can be extended to a proof method suitable to prove reduction properties of untyped λ-terms with certain invariance.

The general idea of the reducibility method is to provide a link between terms typeable in a type system and terms satisfying certain reduction properties (e.g., strong normalisation, confluence). For that reason types are interpreted by suitable sets of λ-terms: saturated and stable sets in Tait [144] and Krivine [105] and admissible relations in Mitchell [115] and [116]. These interpretations are based on the sets of terms considered (e.g., strong normalisation, confluence). Then the soundness of type assignment with respect to these interpretations is obtained. A consequence of soundness is that every term typeable in the type system belongs to the interpretation of its type. This is an intermediate step between the terms typeable in a type system and terms satisfying the reduction property considered.

Necessary notions for the reducibility method are (as presented in [89]): 1. type interpretation; 2. term valuations; 3. closure conditions; 4. soundness of the type assignment.

1. Type interpretation. We consider the set of all λ-terms Λ as the *applicative structure* whose domain are λ-terms and where the application is the application of terms. If $\mathcal{P} \subseteq \Lambda$ is a fixed set, the type interpretation $[\![-]\!] : \mathrm{Type} \to 2^{\Lambda}$ is defined by:

$(I1)$ $[\![X]\!] = \mathcal{P}$, X is an atom;
$(I2)$ $[\![A \cap B]\!] = [\![A]\!] \cap [\![B]\!]$;
$(I3)$ $[\![A \to B]\!] = ([\![A]\!] \Rightarrow [\![B]\!]) \cap \mathcal{P} = \{t \in \mathcal{P} \mid \forall s \in [\![A]\!] \quad ts \in [\![B]\!]\}$;
$(I4)$ $[\![\Omega]\!] = \Lambda$.

An important property of the type interpretation is that $[\![A]\!] \subseteq \mathcal{P}$ for all types $A \not\sim \Omega$.

2. Term valuations. Let $\rho : \mathrm{var} \to \Lambda$ be a valuation of term variables in Λ. Then $[\![-]\!]_\rho : \Lambda \to \Lambda$ is defined as follows

$$[\![t]\!]_\rho = t[x_1 := \rho(x_1), \ldots, x_n := \rho(x_n)], \quad \text{where } Fv(t) = \{x_1, \ldots, x_n\}.$$

The *semantic satisfiability relation* $\models$ connects the type interpretation with the term valuation.

(i) $\rho \models t : A$ iff $[\![t]\!]_\rho \in [\![A]\!]$;
(ii) $\rho \models \Gamma$ iff $(\forall (x : A) \in \Gamma)\ \rho(x) \in [\![A]\!]$;
(iii) $\Gamma \models t : A$ iff $(\forall \rho \models \Gamma)\ \rho \models t : A$.

3. Closure conditions. Let us impose some conditions on $\mathcal{P} \subseteq \Lambda$.

- $\mathcal{X} \subseteq \Lambda$ satisfies the *$\mathcal{P}$-variable property*, notation $VAR(\mathcal{P}, \mathcal{X})$, if

$$(\forall x \in \mathrm{var})\ (\forall n \geqslant 0)\ (\forall t_1, \ldots, t_n \in \mathcal{P})\ \ x t_1 \ldots t_n \in \mathcal{X}.$$

- $\mathcal{X} \subseteq \Lambda$ is *$\mathcal{P}$-saturated*, notation $SAT(\mathcal{P}, \mathcal{X})$, if

$$(\forall t, s \in \Lambda)\ (\forall n \geqslant 0)\ (\forall t_1, \ldots, t_n \in \mathcal{P})$$
$$t[x := s] t_1 \ldots t_n \in \mathcal{X} \ \Rightarrow\ (\lambda x.t) s t_1 \ldots t_n \in \mathcal{X}.$$

- $\mathcal{X} \subseteq \Lambda$ is *$\mathcal{P}$-closed*, notation $CLO(\mathcal{P}, \mathcal{X})$, if $t \in \mathcal{X} \Rightarrow \lambda x.t \in \mathcal{P}$.

The preorder on types is interpreted as the set theoretic inclusion. We prove the following realizability property, which is referred to as the *soundness property* or the *adequacy property*.

Theorem (Soundness of the type assignment). *If $VAR(\mathcal{P}, \mathcal{P})$, $SAT(\mathcal{P}, \mathcal{P})$, and $CLO(\mathcal{P}, \mathcal{P})$ are satisfied, then $\Gamma \vdash t : A \Rightarrow \Gamma \models t : A$.*

An immediate consequence of soundness is the following statement.

Theorem (Reducibility method). *If $VAR(\mathcal{P}, \mathcal{P})$, $SAT(\mathcal{P}, \mathcal{P})$, and $CLO(\mathcal{P}, \mathcal{P})$, then for all types $A \not\sim \Omega$ and $A \not\sim \Omega \to B$, where $B \not\sim \Omega$*

$$\Gamma \vdash t : A \Rightarrow t \in \mathcal{P}.$$

Proof method for Λ. To establish a proof method for untyped λ-terms it is necessary that a set $\mathcal{P} \subseteq \Lambda$ is invariant under abstraction, i.e.,

$$t \in \mathcal{P} \ \Leftrightarrow\ \lambda x.t \in \mathcal{P}. \qquad \square$$

If $\mathcal{P}$ is invariant under abstraction and satisfies $VAR(\mathcal{P}, \mathcal{P})$ and $SAT(\mathcal{P}, \mathcal{P})$, then $\mathcal{P} = \Lambda$. This method is applicable when:

- $\mathcal{P} = \mathcal{C} = \{t \in \Lambda \mid \beta\text{-reduction is confluent on } t\}$;
- $\mathcal{P} = \mathcal{S} = \{t \mid \text{every reduction of } t \text{ can be done in a standard way}\}$;
- $\mathcal{P} = \mathcal{WN} = \{t \mid t \text{ is weakly head normalising}\}$.

In [88] we distinguish the following two different kinds of type interpretation with respect to a given set $\mathcal{P} \subseteq \Lambda$.

(i) The type interpretation $[\![-]\!] : \mathrm{Type} \to 2^\Lambda$ is defined by:
  ($I1$) $[\![X]\!] = \mathcal{P}$, X is an atom;
  ($I2$) $[\![A \cap B]\!] = [\![A]\!] \cap [\![B]\!]$;
  ($I3$) $[\![A \to B]\!] = [\![A]\!] \Rightarrow [\![B]\!] = \{t \in \Lambda \mid \forall s \in [\![A]\!] \quad ts \in [\![B]\!]\}$.

(ii) The Ω-type interpretation $[\![-]\!]^\Omega : \mathrm{Type}^\Omega \to 2^\Lambda$ is defined by
  ($\Omega 1$) $[\![X]\!]^\Omega = \mathcal{P}$, X is an atom;
  ($\Omega 2$) $[\![A \cap B]\!]^\Omega = [\![A]\!]^\Omega \cap [\![B]\!]^\Omega$;

$(\Omega 3)$ $[\![A \to B]\!]^\Omega = [\![A]\!]^\Omega \Rightarrow_\Omega [\![B]\!]^\Omega = ([\![A]\!]^\Omega \Rightarrow [\![B]\!]^\Omega) \cap \mathcal{P} =$
$= \{t \in \mathcal{WN} \mid \forall s \in [\![A]\!]^\Omega \,.\, ts \in [\![B]\!]^\Omega\};$

$(\Omega 4)$ $[\![\Omega]\!]^\Omega = \Lambda$.

Also, we distinguish two different closure conditions which a given set $\mathcal{P} \subseteq \Lambda$ has to satisfy. By combining different type interpretations with appropriate closure conditions on $\mathcal{P} \subseteq \Lambda$ we prove the soundness of the type assignment in both cases. In this way a method for proving properties of λ-terms typeable with intersection types is obtained.

Preliminary version of the work presented in [89, 88] is [86].

The problem of typability in a type system is whether there exists a type for a given term. The typability in the full intersection type assignment system $\lambda\cap^\Omega$ is trivial since there exists a universal type Ω which can be assigned to every term in this system.

But without the rule (Ω), the situation changes. In Likavec [109] we focus on typability of terms in the intersection type assignment systems without the type Ω. We show that all the strongly normalising terms are typable in these systems. They are the only terms typable in these systems. We also present detailed proofs for [88, 89].

7.5. Behavioural inverse limit models. In Dezani et al. [44] we construct two inverse limit λ-models which completely characterise sets of terms with similar computational behaviours:

Normalisation properties

(1) A term t *has a normal form*, $t \in \mathcal{N}$, if t reduces to a normal form.
(2) A term t *has a head normal form*, $t \in \mathcal{HN}$, if t reduces to a term of the form $\lambda\vec{x}.y\vec{t}$ (where possibly y appears in $\vec{x}$).
(3) A term t *has a weak head normal form*, $t \in \mathcal{WN}$, if t reduces to an abstraction or to a term starting with a free variable.

Persistent normalisation properties

(1) A term t is *persistently normalising*, $t \in \mathcal{PN}$, if $t\vec{u} \in \mathcal{N}$ for all $\vec{u} \in \mathcal{N}$.
(2) A term t is *persistently head normalising*, $t \in \mathcal{PHN}$, if $t\vec{u} \in \mathcal{HN}$ for all $\vec{u} \in \Lambda$.
(3) A term t is *persistently weak head normalising*, $t \in \mathcal{PWN}$, if $t\vec{u} \in \mathcal{WN}$ for all $\vec{u} \in \Lambda$.

Closability properties

(1) A term t is *closable*, $t \in \mathcal{C}$, if t reduces to a closed term.
(2) A term t is *closable normalising*, $t \in \mathcal{CN}$, if t reduces to a closed normal form.
(3) A term t is *closable head normalising*, $t \in \mathcal{CHN}$, if t reduces to a closed head normal form.

We build two *inverse limit* λ*-models* $\mathcal{D}_\infty$ and $\mathcal{E}_\infty$, according to Scott [136], which completely characterise each of the mentioned sets of terms. For that we need to discuss the functional behaviours of the terms belonging to these classes with respect

to the step functions. Given compact elements a and b in the Scott domains $\mathcal{A}$ and $\mathcal{B}$ respectively, the *step function* $\mathsf{a} \Rightarrow \mathsf{b}$ is defined by $\lambda \mathsf{c}.$ **if** $\mathsf{a} \sqsubseteq \mathsf{c}$ **then** b **else** $\bot$

Definition of models

(1) Let $\mathcal{D}_\infty$ be the inverse limit λ-model obtained by taking as $\mathcal{D}_0$ the lattice in Figure 13, as $\mathcal{D}_1$ the lattice $[\mathcal{D}_0 \to \mathcal{D}_0]_\bot$, and by defining the embedding $\mathsf{i}_0^{\mathcal{D}} : \mathcal{D}_0 \to [\mathcal{D}_0 \to \mathcal{D}_0]_\bot$ as follows:

$$\mathsf{i}_0^{\mathcal{D}}(\hat{\mathsf{n}}) = (\bot \Rightarrow \hat{\mathsf{h}}) \sqcup (\mathsf{n} \Rightarrow \hat{\mathsf{n}}), \qquad \mathsf{i}_0^{\mathcal{D}}(\mathsf{n}) = (\hat{\mathsf{h}} \Rightarrow \mathsf{h}) \sqcup (\hat{\mathsf{n}} \Rightarrow \mathsf{n}),$$
$$\mathsf{i}_0^{\mathcal{D}}(\hat{\mathsf{h}}) = \bot \Rightarrow \hat{\mathsf{h}}, \qquad \mathsf{i}_0^{\mathcal{D}}(\mathsf{h}) = \hat{\mathsf{h}} \Rightarrow \mathsf{h}, \qquad \mathsf{i}_0^{\mathcal{D}}(\bot) = \bot.$$

(2) Let $\mathcal{E}_\infty$ be the inverse limit λ-model obtained by taking as $\mathcal{E}_0$ the cpo in Figure 13, as $\mathcal{E}_1$ the cpo $[\mathcal{E}_0 \to \mathcal{E}_0]$, and by defining the embedding $\mathsf{i}_0^{\mathcal{E}} : \mathcal{E}_0 \to [\mathcal{E}_0 \to \mathcal{E}_0]$ as follows:

$$\mathsf{i}_0^{\mathcal{E}}(\hat{\mathsf{n}}) = (\bot \Rightarrow \hat{\mathsf{h}}) \sqcup (\mathsf{n} \Rightarrow \hat{\mathsf{n}}), \qquad \mathsf{i}_0^{\mathcal{E}}(\mathsf{n}) = (\hat{\mathsf{h}} \Rightarrow \mathsf{h}) \sqcup (\hat{\mathsf{n}} \Rightarrow \mathsf{n}),$$
$$\mathsf{i}_0^{\mathcal{E}}(\hat{\mathsf{h}}) = \bot \Rightarrow \hat{\mathsf{h}}, \qquad \mathsf{i}_0^{\mathcal{E}}(\mathsf{h}) = \hat{\mathsf{h}} \Rightarrow \mathsf{h},$$
$$\mathsf{i}_0^{\mathcal{E}}(\mathsf{c}) = \mathsf{c} \Rightarrow \mathsf{c}, \qquad \mathsf{i}_0^{\mathcal{E}}(\bot) = \bot \Rightarrow \bot.$$

(3) We will denote the partial orders on $\mathcal{D}_\infty$ and $\mathcal{E}_\infty$ by $\sqsubseteq^{\mathcal{D}}$ and $\sqsubseteq^{\mathcal{E}}$, respectively.

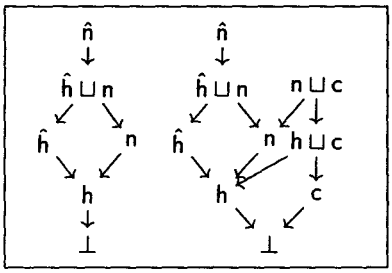


FIGURE 13. The lattice $\mathcal{D}_0$ and the cpo $\mathcal{E}_0$

More precisely, for each of these sets of terms there is a corresponding element in at least one of the two models such that a term belongs to the set if and only if its interpretation (in a suitable environment) is greater than or equal to that element. This is the result of the following theorem.

Theorem (Main Theorem, Version I). *Let $\mathcal{D}_\infty$ and $\mathcal{E}_\infty$ be the inverse limit λ-models defined above and $\rho_{\hat{\mathsf{n}}}$ the environment defined by $\rho_{\hat{\mathsf{n}}}(x) = \hat{\mathsf{n}}$ for all $x \in$ var (since each variable is in $\mathcal{PN}$). Then:*

(1) $t \in \mathcal{PN}$ *iff* $[\![t]\!]^{\mathcal{D}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{D}} \hat{\mathsf{n}}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{E}} \hat{\mathsf{n}}$;
(2) $t \in \mathcal{N}$ *iff* $[\![t]\!]^{\mathcal{D}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{D}} \mathsf{n}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{E}} \mathsf{n}$;
(3) $t \in \mathcal{PHN}$ *iff* $[\![t]\!]^{\mathcal{D}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{D}} \hat{\mathsf{h}}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{E}} \hat{\mathsf{h}}$;
(4) $t \in \mathcal{HN}$ *iff* $[\![t]\!]^{\mathcal{D}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{D}} \mathsf{h}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{E}} \mathsf{h}$;
(5) $t \in \mathcal{PWN}$ *iff* $[\![t]\!]^{\mathcal{D}_\infty}_{\rho_{\hat{\mathsf{n}}}} \sqsupseteq^{\mathcal{D}} \bigsqcup_{n \in \mathbf{N}} (\underbrace{\bot \Rightarrow \ldots \Rightarrow \bot}_{n} \Rightarrow \bot)$;

(6) $t \in \mathcal{WN}$ *iff* $[\![t]\!]^{\mathcal{D}_\infty}_{\rho_{\hat{n}}} \sqsupseteq^{\mathcal{D}} \bot \Rightarrow \bot$;
(7) $t \in \mathcal{CN}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{n}}} \sqsupseteq^{\mathcal{E}} \mathsf{c} \sqcup \mathsf{n}$;
(8) $t \in \mathcal{CHN}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{n}}} \sqsupseteq^{\mathcal{E}} \mathsf{c} \sqcup \mathsf{h}$;
(9) $t \in \mathcal{C}$ *iff* $[\![t]\!]^{\mathcal{E}_\infty}_{\rho_{\hat{n}}} \sqsupseteq^{\mathcal{E}} \mathsf{c}$.

This is proved by using the finitary logical descriptions of the models $\mathcal{D}_\infty$ and $\mathcal{E}_\infty$, obtained by defining two *intersection type assignment systems* in the following way. Starting from atomic types corresponding to the elements of $\mathcal{D}_0$ and $\mathcal{E}_0$, we construct the sets $\mathbb{T}^{\mathcal{D}}$ and $\mathbb{T}^{\mathcal{E}}$ of types using the *function type* constructor $\to$ and the *intersection type* constructor $\cap$ between *compatible* types, where two types are compatible if the corresponding elements have a join. Types are denoted by $A, B, A_1, \ldots$. $A^n \to B$ is short for $\underbrace{A \to \cdots \to A}_{n} \to B$ $(n \geqslant 0)$. The preorder between types is induced by reversing the order in the initial cpo and by encoding the initial embedding, according to the correspondence: (i) function type constructor corresponds to step function and (ii) intersection type constructor corresponds to join.

Then, we define the sets $\mathcal{F}^{\mathcal{D}}$ and $\mathcal{F}^{\mathcal{E}}$ of filters on the sets $\mathbb{T}^{\mathcal{D}}$ and $\mathbb{T}^{\mathcal{E}}$, respectively. Both $\mathcal{F}^{\mathcal{D}}$ and $\mathcal{F}^{\mathcal{E}}$, ordered by subset inclusion, are Scott domains. The compact elements are precisely the principal filters, and the bottom element is $\uparrow \Omega$. $\mathcal{F}^{\mathcal{D}}$ is an ω-algebraic complete lattice, since it has the top element $\mathbb{T}^{\mathcal{D}}$.

We can show that $\mathcal{F}^{\mathcal{D}}$ and $\mathcal{D}_\infty$ are isomorphic as ω-algebraic complete lattices, and that $\mathcal{F}^{\mathcal{E}}$ and $\mathcal{E}_\infty$ are isomorphic as Scott domains. This isomorphism falls in the general framework of *Stone dualities*. The interest of the above isomorphism lies in the fact that the interpretations of λ-terms in $\mathcal{D}_\infty$ and $\mathcal{E}_\infty$ are isomorphic to the filters of types one can derive in the corresponding type assignment systems. This gives the desired finitary logical descriptions of the models.

Theorem (Finitary logical descriptions).

(1) *For any* $t \in \Lambda$ *and* $\rho : var \mapsto \mathcal{F}^{\mathcal{D}}$, $[\![t]\!]^{\mathcal{F}^{\mathcal{D}}}_{\rho} = \{A \in \mathbb{T}^{\mathcal{D}} \mid \exists \Gamma.\, \Gamma \rhd \rho \,\&\, \Gamma \vdash^{\mathcal{D}} t : A\}$;
(2) *For any* $t \in \Lambda$ *and* $\rho : var \mapsto \mathcal{F}^{\mathcal{E}}$, $[\![t]\!]^{\mathcal{F}^{\mathcal{E}}}_{\rho} = \{A \in \mathbb{T}^{\mathcal{E}} \mid \exists \Gamma.\, \Gamma \rhd \rho \,\&\, \Gamma \vdash^{\mathcal{E}} t : A\}$,
where $\Gamma \rhd \rho$ *means that for* $(x : B) \in \Gamma$ *one has that* $B \in \rho(x)$.

Therefore, the primary complete characterisation can be stated equivalently as follows: a term belongs to one of the nine sets mentioned if and only if it has a certain type (in a suitable basis) in one of the obtained type assignment systems. This is the result of the following theorem.

Theorem (Main Theorem, Version II).

(1) $t \in \mathcal{PN}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{D}} t : \hat{\nu}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{E}} t : \hat{\nu}$;
(2) $t \in \mathcal{N}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{D}} t : \nu$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{E}} t : \nu$;
(3) $t \in \mathcal{PHN}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{D}} t : \hat{\mu}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{E}} t : \hat{\mu}$;
(4) $t \in \mathcal{HN}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{D}} t : \mu$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{E}} t : \mu$;
(5) $t \in \mathcal{PWN}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{D}} t : \Omega^n \to \Omega$ *for all* $n \in \mathbb{N}$;
(6) $t \in \mathcal{WN}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{D}} t : \Omega \to \Omega$;
(7) $t \in \mathcal{CN}$ *iff* $\Gamma_{\hat{\nu}} \vdash^{\mathcal{E}} t : \gamma \cap \nu$;

(8) $t \in \mathcal{CHN}$ *iff* $\Gamma_{\mathcal{D}} \vdash^{\mathcal{E}} t : \gamma \cap \mu$;
(9) $t \in \mathcal{C}$ *iff* $\Gamma_{\mathcal{D}} \vdash^{\mathcal{E}} t : \gamma$.

The proofs of the ($\Rightarrow$) parts are mainly straightforward inductions and case split, with the exception of the case of persistently normalising terms, which are treated using the notions of safe and unsafe suberms (see [21]). The proofs of the ($\Leftarrow$) parts require the set-theoretic semantics of intersection types and saturated sets, which is referred to as the reducibility method. To that purpose we define the *interpretations of types* in $\mathbb{T}^{\mathcal{D}}$ and in $\mathbb{T}^{\mathcal{E}}$ as follows:

Interpretation of types

(1) The map $[\![-]\!]^{\mathcal{D}} : \mathbb{T}^{\mathcal{D}} \to \mathcal{P}(\Lambda)$ is defined by:
  (i) $[\![\nu]\!]^{\mathcal{D}} = \mathcal{N}$, $[\![\hat{\nu}]\!]^{\mathcal{D}} = \mathcal{PN}$, $[\![\mu]\!]^{\mathcal{D}} = \mathcal{HN}$, $[\![\hat{\mu}]\!]^{\mathcal{D}} = \mathcal{PHN}$, $[\![\Omega]\!]^{\mathcal{D}} = \Lambda$;
  (ii) $[\![A \cap B]\!]^{\mathcal{D}} = [\![A]\!]^{\mathcal{D}} \cap [\![B]\!]^{\mathcal{D}}$;
  (iii) $[\![A \to B]\!]^{\mathcal{D}} = [\![A]\!]^{\mathcal{D}} \xrightarrow{\mathcal{D}} [\![B]\!]^{\mathcal{D}} = \{t \in \mathcal{WN} \mid \forall u \in [\![A]\!]^{\mathcal{D}} \quad tu \in [\![B]\!]^{\mathcal{D}}\}$.
(2) The map $[\![-]\!]^{\mathcal{E}} : \mathbb{T}^{\mathcal{E}} \to \mathcal{P}(\Lambda)$ is defined by:
  (i) $[\![\nu]\!]^{\mathcal{E}} = \mathcal{N}$, $[\![\hat{\nu}]\!]^{\mathcal{E}} = \mathcal{PN}$, $[\![\mu]\!]^{\mathcal{E}} = \mathcal{HN}$ $[\![\hat{\mu}]\!]^{\mathcal{E}} = \mathcal{PHN}$, $[\![\gamma]\!]^{\mathcal{E}} = \mathcal{C}$, $[\![\Omega]\!]^{\mathcal{E}} = \Lambda$;
  (ii) $[\![A \cap B]\!]^{\mathcal{E}} = [\![A]\!]^{\mathcal{E}} \cap [\![B]\!]^{\mathcal{E}}$;
  (iii) $[\![A \to B]\!]^{\mathcal{E}} = [\![A]\!]^{\mathcal{E}} \xrightarrow{\mathcal{E}} [\![B]\!]^{\mathcal{E}} = \{t \in \Lambda \mid \forall u \in [\![A]\!]^{\mathcal{E}} \quad tu \in [\![B]\!]^{\mathcal{E}}\}$.

The main contribution of the present paper is to show that *only two* models can characterise many different sets of terms. On the one hand it seems that we cannot find elements representing weak head normalisability and closability in the same model, since the first property requires the lifting of the space of functions and this does not agree with the second one. On the other hand, there are properties which appear strongly connected, like each normalisation property with its persistent version. It is not clear if these properties can be characterised separately, i.e., if one can build models in which only one of these properties is characterised.

A preliminary version of the present paper (dealing only with the first six sets of terms) is [41]. An extended abstract of the present paper is [43].

8. Intuitionistic sequent calculus and λ^{Gtz}-calculus

8.1. Intersection types for λ^{Gtz}-calculus. In Espírito Santo et al. [57], we introduce intersection types for the λ^{Gtz}-calculus. The set Type of types, ranged over by $A, B, C, \ldots, A_1, \ldots$, is defined inductively:

$$A, B \; ;::= \; X \mid A \to B \mid A \cap B$$

where X ranges over a denumerable set $TVar$ of type atoms.

The type assignment system $\lambda^{\mathsf{Gtz}}\cap$ is given in Figure 14.

The following rules are admissible in $\lambda^{\mathsf{Gtz}}\cap$:

1. If $\Gamma, x : A_i \vdash t : C$ then $\Gamma, x : \cap A_i \vdash t : C$.
2. If $\Gamma, x : A_i; D \vdash k : C$ then $\Gamma, x : \cap A_i; D \vdash k : C$.

Basis expansion and bases intersection are defined in an obvious way. Standard form of generation lemma holds for λ^{Gtz}.

$$\frac{}{\Gamma, x : \cap A_i \vdash x : A_i \;\; i = 1, \ldots, n, \;\; n \geqslant 1} \; (Ax)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \; (\to_R) \qquad \frac{\Gamma \vdash u : A_i \;\; \forall i \;\; \Gamma; B \vdash k : C}{\Gamma; \cap A_i \to B \vdash u :: k : C} \; (\to_L)$$

$$\frac{\Gamma \vdash t : A_i, \;\; \forall i \;\; \Gamma; \cap A_i \vdash k : B}{\Gamma \vdash tk : B} \; (Cut) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma; A \vdash \widehat{x}.t : B} \; (Sel)$$

FIGURE 14. $\lambda^{\mathsf{Gtz}}\cap$: type assignment system for λ^{Gtz}-calculus

Example: In λ-calculus with intersection types, the term $\lambda x.xx$ has the type $(A \cap (A \to B)) \to B$. The corresponding term in λ^{Gtz}-calculus is $\lambda x.x(x :: \widehat{y}.y)$. Although being a normal form this term is not typeable in the simply typed λ^{Gtz}-calculus. It is typeable in $\lambda^{\mathsf{Gtz}}\cap$ in the following way:

$$\dfrac{\dfrac{}{x : A \cap (A \to B) \vdash x : A \to B}\,(Ax) \qquad \dfrac{\dfrac{}{x : A \cap (A \to B) \vdash x : A}\,(Ax) \qquad \dfrac{\dfrac{}{x : A \cap (A \to B), y : B \vdash y : B}\,(Ax)}{x : A \cap (A \to B); B \vdash \widehat{y}.y : B}\,(Sel)}{x : A \cap (A \to B); A \to B \vdash (x :: \widehat{y}.y) : B}\,(\to_L)}{x : A \cap (A \to B) \vdash x(x :: \widehat{y}.y) : B}\,(Cut)$$
$$\vdash \lambda x.x(x :: \widehat{y}.y) : (A \cap (A \to B)) \to B \;\; (\to_R).$$

In [58] we describe our quest for the intersection type assignment system $\lambda^{\mathsf{Gtz}}\cap$ and offered a new, equivalent system $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$. Both systems $\lambda^{\mathsf{Gtz}}\cap$ and $\lambda^{\mathsf{Gtz}}\cap_{\mathsf{I}}$ successfully characterize strongly normalizing terms of the λ^{Gtz}-calculus. We also report on various alternative formulations of the system. Two of them are not successful and we explain why they fail and how they lead us to the system$\lambda^{\mathsf{Gtz}}\cap$.

8.2. Subject reduction and strong normalisation. Basic properties of this system are analysed and the Subject reduction property is proved i.e.,

$$\text{If } \Gamma \vdash t : A \text{ and } t \to t', \text{ then } \Gamma \vdash t' : A.$$

The reduction μ is of different nature, since it reduces contexts instead of terms. A similar result for this reduction rule is given, i.e.,

$$\text{If } \Gamma; \cap B_i \vdash \widehat{x}.xk : A, \text{ then } \Gamma; B_i \vdash k : A, \text{ for some } i.$$

In [83], a slightly modified type assignment system $\lambda^{\mathsf{Gtz}}\cap$ with respect to the one given in 8.1 is considered. Subject reduction holds for this system as well.

We use intersection types in [57] to give a characterisation of the strongly normalising terms of an intuitionistic sequent calculus (where LJ easily embeds). The sequent term calculus presented in this paper integrates smoothly the λ-terms with generalised application or explicit substitution.

In order to prove that typeability in $\lambda^{\mathsf{Gtz}}\cap$ implies strong normalisation for the $\lambda^{\mathsf{Gtz}}\cap$, we connect it with the well-known system $\mathcal{D}$ for the λ-calculus (given in Section 2.2) via an appropriate mapping, and then use strong normalisation theorem for λ-terms typeable in system $\mathcal{D}$.

Terms in $\mathcal{D}$ are ordinary λ-terms equipped with the following two reduction relations, in addition to standard β reduction:

$$(\pi_1)\quad (\lambda x.M)NP \to (\lambda x.MP)N \qquad (\pi_2)\quad M((\lambda x.P)N) \to (\lambda x.MP)N.$$

We let $\pi = \pi_1 \cup \pi_2$. We use capital letters here to denote the terms in $\mathcal{D}$ to differentiate them from the terms in $\lambda^{\mathsf{Gtz}}\cap$.

We define a mapping F from λ^{Gtz} to λ. The idea is the following. If $F(t) = M$, $F(u_i) = N_i$ and $F(v) = P$, then $t(u_1 :: u_2 :: (x)v)$, say, is mapped to $(\lambda x.P)(MN_1N_2)$. Formally, a mapping $F : \lambda^{\mathsf{Gtz}}Terms \longrightarrow \lambda Terms$ is defined simultaneously with an auxiliary mapping $F' : \lambda Terms \times \lambda^{\mathsf{Gtz}}Contexts \longrightarrow \lambda Terms$ as follows:

$$\begin{array}{ll} F(x) = x & F'(N, \widehat{x}.t) = (\lambda x.F(t))N \\ F(\lambda x.t) = \lambda x.F(t) & F'(N, u :: k) = F'(NF(u), k). \\ F(tk) = F'(F(t), k) & \end{array}$$

We prove the following theorems:

- **Soundness of F:** If $\lambda^{\mathsf{Gtz}}\cap$ proves $\Gamma \vdash t : A$, then $\mathcal{D}$ proves $\Gamma \vdash F(t) : A$.
- **Reduction of SN:** For all $t \in \lambda^{\mathsf{Gtz}}$, if $F(t)$ is $\beta\pi$-SN, then t is $\beta\pi\sigma\mu$-SN.

The main theorem is the following:

Theorem (Typeability $\Rightarrow$ SN). *If a λ^{Gtz}-term t is typeable in $\lambda^{\mathsf{Gtz}}\cap$, then t is $\beta\pi\sigma\mu$-SN.*

In order to prove that SN implies typeability we prove the following:

- $\beta\pi\sigma$-normal forms and $\beta\pi\sigma\mu$-normal forms of the λ^{Gtz}-calculus are typeable in the $\lambda^{\mathsf{Gtz}}\cap$ system.
- **Subject expansion property:** If $t \to t'$, t is the redex and t' is typeable in $\lambda^{\mathsf{Gtz}}\cap$, then t is typeable in $\lambda^{\mathsf{Gtz}}\cap$.

The main theorem is the following:

Theorem (SN $\Rightarrow$ typeability). *All strongly normalising ($\beta\sigma\pi\mu - SN$) terms are typeable in the $\lambda^{\mathsf{Gtz}}\cap$ system.*

Finally, in order to deal with generalised applications and explicit substitutions, we consider two extensions of the λ-calculus: the ΛJ-calculus, where application $M(N, x.P)$ is *generalised* [102]; and the λx-calculus, where substitution $M[x := N]$ is *explicit* [132]. Intersection types have been used to characterise the strongly normalising terms of both ΛJ-calculus [113] and λx-calculus [108]. But in both [113] and [108] the "natural" typing rules for generalised application or substitution had to be suplemented with extra rules in order to secure that every strongly normalising term is typeable. Hence, the "natural" rules failed to capture the strongly normalising terms. We prove that λ^{Gtz} and $\lambda^{\mathsf{Gtz}}\cap$ are useful for resolving these issues.

Let t be a λ^{Gtz}-term.

(1) t is a λJ-*term* if every cut occurring in t is of the form $t(u :: \widehat{x}.v)$.
(2) t is a $\lambda\mathrm{x}$-*term* if every cut occurring in t has one of the forms $t(u :: \widehat{x}.x)$ or $t(\widehat{x}.v)$.

We define appropriate type assignment systems $\lambda J\cap$ and $\lambda\mathrm{x}\cap$ for these calculi and prove the following:

(1) Let t be a λJ-term. t is $\beta\pi\sigma\mu - SN$ iff t is typeable in $\lambda J\cap$.
(2) Let t be a $\lambda\mathrm{x}$-term. t is $\beta\pi\sigma\mu - SN$ iff t is typeable in $\lambda\mathrm{x}\cap$.

The extended version of [57] can be found in [59].

9. Classical natural deduction and $\lambda\mu$-calculus

9.1. Terms for natural deduction and sequent calculus classical logic. In Ghilezan [82], the work of Barendregt and Ghilezan [12] is further elaborated and its results are generalised for classical logic. Two extensionally equivalent type assignment systems for the $\lambda\mu$-calculus are considered. The type assignment system $\lambda\mu N$ is actually the simply typed $\lambda\mu$-calculus, given in Figure 8. It corresponds to implicational fragment of classical natural deduction NK (given in Figure 2), whereas the type assignment system $\lambda\mu L$ given in Figure 15 corresponds to implicational fragment of classical sequent calculus LK (given in Figure 4). In addition, a cut free variant of $\lambda\mu L$, denoted by $\lambda\mu L^{\mathrm{cf}}$, is introduced and used to give a short proof of Cut elimination theorem for classical logic.

$$\frac{}{\Gamma, y : A \vdash y : A, \Delta}\ (\text{axiom})$$

$$\frac{\Gamma \vdash u : A, \Delta \quad \Gamma, x : B \vdash t : C, \Delta}{\Gamma, y : A \to B \vdash t[x := yu] : C, \Delta}\ (\to \text{left}) \qquad \frac{\Gamma, y : A \vdash t : B, \Delta}{\Gamma \vdash \lambda y.t : A \to B, \Delta}\ (\to \text{right})$$

$$\frac{\Gamma \vdash t : A, \Delta, \beta : A, \alpha : B}{\Gamma \vdash \mu\alpha.[\beta]t : B, \Delta, \beta : A}\ (\mu)$$

$$\frac{\Gamma \vdash u : B, \Delta \quad \Gamma, x : B \vdash t : A, \Delta}{\Gamma \vdash t[x := u] : A, \Delta}\ (\text{cut})$$

FIGURE 15. $\lambda\mu L$-calculus

In Figure 15 a *term context* $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$ is a set of variable declarations such that for every variable x_i there is at most one declaration $x_i : A_i$ in Γ and a *co-term context* $\Delta = \{\alpha_1 : B_1, \ldots, \alpha_k : B_k\}$ is a set of co-variable declarations such that for every co-variable α_l there is at most one declaration $\alpha_l : B_l$ in Δ. In this setup $\Gamma \setminus \mathbf{x} = \{A_1, \ldots A_n\}$ and $\Delta \setminus \alpha = \{B_1, \ldots B_k\}$.

It is shown that the statement A is derivable from assumptions in Γ in NK if and only if it is derivable from the same assumptions in LK, i.e., for all Γ and A

$$\Gamma \vdash_{NK} A, \Delta \Longleftrightarrow \Gamma \vdash_{LK} A, \Delta.$$

The following result was given by Parigot [121] as an extension of the well-known proposition-as-types interpretation of intuitionistic logic.

Theorem (Curry–Howard correspondence for classical logic). *If SK is one of the logical systems NK, LK or LK^{cf} and if $\lambda\mu S$ is the corresponding type assignment system, then*

$$\Gamma \setminus \mathbf{x} \vdash_{SK} A, \Delta \setminus \alpha \iff \exists t \in \Lambda^{\circ}(\mathbf{x}) \cup \Lambda^{\circ}_{\mu}(\alpha)\ \Gamma \vdash_{\lambda\mu S} t : A, \Delta.$$

where $\Lambda^{\circ}(\vec{x}) = \{t \in \Lambda \mid Fv(t) \subseteq \mathbf{x}\}$, $\Lambda^{\circ}_{\mu}(\alpha) = \{t \in \Lambda \mid Fv_{\mu}(t) \subseteq \alpha\}$

It is also proved that

$$\Gamma \vdash_{\lambda\mu L} t : C, \Delta \Longleftrightarrow \Gamma \vdash_{\lambda\mu N} t : C, \Delta.$$

Finally, using the type assignment system $\lambda\mu L^{\mathrm{cf}}$, Cut elimination theorem of Gentzen [71] for classical implicational sequent calculus is proved, i.e.,

$$\Gamma \vdash_{LK} A \Longleftrightarrow \Gamma \vdash_{LK^{\mathrm{cf}}} A.$$

The type assignment system $\lambda\mu L$ is a novel system for encoding proofs in classical sequent logic. The main focus of this paper is on $\lambda\mu$-terms, rather than on derivations.

9.2. Separability in $\lambda\mu$-calculus. In Herbelin and Ghilezan [98], we investigate the separability property of $\lambda\mu$-calculus. In the untyped λ-calculus Böhm's theorem deals with the separability property of λ-terms [20, 35, 10, 105]. For two different normal forms there is a context such that one of these terms converges in this context, whereas the other one diverges in the same context. A consequence of this theorem is that $\beta\eta$ equality is the *maximal consistent equality* between λ-terms having normal forms. Hence, if t and u are two λ–terms having different $\beta\eta$ normal forms, meaning that $t = u$ cannot be proved in λ-calculus, and if this calculus is extended with $t = u$, then according to Böhm's theorem every equality of λ-terms can be proved in the extended calculus. In other words such an extended calculus is inconsistent.

Two terms are *observationally equivalent* if, whenever put in the same context, either they both make it reducible to a normal form or they both make it diverge. More generally, two terms may be considered as equivalent if, when observed from outside, they exhibit the same behaviour. Therefore another important consequence of Böhm's separability in the λ-calculus setting is that observational equivalence for normalisable terms coincides with $\beta\eta$-equivalence. The proof of Böhm's theorem can be considered as a refutation procedure for observational equivalence. An overview of the relation between Böhm's theorem and observational equivalence is given by Dezani-Ciancaglini and Giovannetti [46].

Regarding computational interpretations of classical logic Böhm's separability property has been investigated in Parigot's $\lambda\mu$-calculus, so far. David and Py [38]

showed that Parigot's $\lambda\mu$-calculus does not satisfy Böhm's separability property. This means that the equality of Parigot's $\lambda\mu$-calculus is not the maximal consistent equality between $\lambda\mu$-terms having normal forms.

Saurin [135] studied the Böhm's separability property in a syntactic modification of the $\lambda\mu$-calculus by de Groote [39] which is denoted here by $\Lambda\mu$ following Saurin. The syntax of the $\Lambda\mu$-calculus is given by the following:

$$t \quad ::= \quad x \mid \lambda x.t \mid tu \mid \mu\beta.t \mid [\alpha]t.$$

The reduction rules are the same as of the $\lambda\mu$-calculus (see Subsection 4).

Saurin showed that the $\Lambda\mu$-calculus is a strict extension of Parigot's $\lambda\mu$-calculus and that it enjoys Böhm's separability property. Therefore, the equality in $\Lambda\mu$-calculus is the maximal consistent equality of $\lambda\mu$-terms having normal forms. The two syntax were up to now considered as almost the same. Obviously this subtle move in the syntax has significant consequences. In [98] we restored Böhm separability in $\lambda\mu$ by extending the syntax of $\lambda\mu$ with a dynamically bound continuation variable $\widehat{\mathsf{tp}}$ and the reduction rules with two rules

$$\begin{array}{lcl} [\widehat{\mathsf{tp}}]\mu\widehat{\mathsf{tp}}.c & \to & c \\ \mu\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}]t & \to & t. \end{array}$$

In this way we obtained $\lambda\mu\widehat{\mathsf{tp}}$, actually its call-by-name variant. It is possible to establish then a mutual embedding of $\Lambda\mu$ and $\lambda\mu\widehat{\mathsf{tp}}$. Embedding of $\Lambda\mu$ into the extended $\lambda\mu$, actually $\Pi : \Lambda\mu \longrightarrow \lambda\mu\widehat{\mathsf{tp}}$ is given by the following:

$$\begin{array}{lcl} \Pi(x) & \triangleq & x \\ \Pi(\lambda x.t) & \triangleq & \lambda x.\Pi(t) \\ \Pi(t\,s) & \triangleq & \Pi(t)\,\Pi(s) \\ \Pi(\mu\alpha.t) & \triangleq & \mu\alpha.[\widehat{\mathsf{tp}}]\Pi(t) \\ \Pi([\alpha]t) & \triangleq & \mu\widehat{\mathsf{tp}}.[\alpha]\Pi(t). \end{array}$$

Embedding of the extended $\lambda\mu$ into $\Lambda\mu$, actually $\Sigma : \lambda\mu\widehat{\mathsf{tp}} \longrightarrow \Lambda\mu$ is the following:

$$\begin{array}{lcll} \Sigma(x) & \triangleq & x & \\ \Sigma(\lambda x.t) & \triangleq & \lambda x.\Sigma(t) & \\ \Sigma(t\,s) & \triangleq & \Sigma(t)\,\Sigma(s) & \\ \Sigma(\mu\alpha.[\beta]t) & \triangleq & \mu\alpha.([\beta]\Sigma(t)) & \text{if } \beta \text{ and } \widehat{\mathsf{tp}} \text{ are distinct} \\ \Sigma(\mu\alpha.[\widehat{\mathsf{tp}}]t) & \triangleq & \mu\alpha.(\Sigma(t)) & \\ \Sigma(\mu\widehat{\mathsf{tp}}.[\alpha]t) & \triangleq & [\alpha]\Sigma(t) & \text{if } \alpha \text{ and } \widehat{\mathsf{tp}} \text{ are distinct} \\ \Sigma(\mu\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}]t) & \triangleq & \Sigma(t). & \end{array}$$

¿From the desired properties:

- $t = u$ in $\Lambda\mu$ implies $\Pi(t) = \Pi(u)$ in $\lambda\mu\widehat{\mathsf{tp}}$,
- $t = u$ in $\lambda\mu\widehat{\mathsf{tp}}$ implies $\Sigma(t) = \Sigma(u)$ in $\Lambda\mu$,

we conclude the separability of the extended $\lambda\mu$-calculus.

Theorem (Separability). *$\lambda\mu\widehat{\mathsf{tp}}$, the extended $\lambda\mu$-calculus is observationally complete for normal forms, i.e., for any two normal forms there exists an evaluation*

$$
\begin{array}{rcl}
X \in \mathit{TypeConstants} & & \\
A, B & ::= & X \mid A_\Sigma \to B \\
\Gamma & ::= & \emptyset \mid \Gamma, x : A_\Sigma \\
\Delta & ::= & \emptyset \mid \Delta, \alpha : A \\
\Sigma, \Xi & ::= & \bot \mid A \cdot \Sigma
\end{array}
$$

$$\frac{}{\Gamma, x : A_\Sigma \vdash_\Sigma x : A; \Delta}\ Ax$$

$$\frac{\Gamma, x : A_\Sigma \vdash_\Xi t : B; \Delta}{\Gamma \vdash_\Xi \lambda x.t : (A_\Sigma \to B); \Delta}\ (\to_i) \qquad \frac{\Gamma \vdash_\Xi t : (A_\Sigma \to B); \Delta \quad \Gamma \vdash_\Sigma s : A; \Delta}{\Gamma \vdash_\Xi t\, s : B; \Delta}\ (\to_e)$$

$$\frac{\Gamma \vdash_\Sigma c : \bot; \Delta, \alpha : A}{\Gamma \vdash_\Sigma \mu\alpha.c : A; \Delta} \qquad \frac{\Gamma \vdash_{A\cdot\Sigma} c : \bot; \Delta}{\Gamma \vdash_\Sigma \mu\widehat{\mathrm{tp}}.c : A; \Delta} \qquad \frac{\Gamma \vdash_\Sigma t : A; \Delta, \alpha : A}{\Gamma \vdash_\Sigma [\alpha]t : \bot; \Delta, \alpha : A} \qquad \frac{\Gamma \vdash_\Sigma t : A; \Delta}{\Gamma \vdash_{A\cdot\Sigma} [\widehat{\mathrm{tp}}]t : \bot; \Delta}$$

FIGURE 16. Simple typing of $\lambda\mu\widehat{\mathrm{tp}}$-calculus

$\lambda\mu\widehat{\mathrm{tp}}$-context $C[]$, such that, in $\lambda\mu\widehat{\mathrm{tp}}$, $C[t] = x$ and $C[s] = y$ for x and y being arbitrary fresh variables.

Separability in simply typed $\lambda\mu$-calculus is an open question. It was shown in [140, 138, 53] that separability in simply typed λ-calculus needs different treatment from Böhm's method for the untyped λ-calculus. There is ongoing research along the lines of the approach by Došen and Petrić [53].

9.3. Simple types for extended $\lambda\mu$-calculus. In Herbelin and Ghilezan [98] we propose a system of simple types for call-by-name $\lambda\mu\widehat{\mathrm{tp}}$, the $\lambda\mu$-calculus extended by a dynamically bound continuation variable, which is introduced in the previous subsection. Like for typing $\lambda\mu$, we have two kinds of sequents, one for each category of expressions:

$$\Gamma \vdash_\Sigma t : A; \Delta \quad \text{(for terms)}$$
$$\Gamma \vdash_\Sigma c : \bot; \Delta \quad \text{(for commands)}.$$

Like for $\lambda\mu$, we have a context of *hypotheses* Γ that assigns types to term variables and a context of *conclusions* Δ that assigns types to continuation variables. But we have also to take care of the $\mu\widehat{\mathrm{tp}}$ dynamic binder.

There is an extra data to type the dynamic effects. Each use of $\mu\widehat{\mathrm{tp}}$ pushes the current continuation on a stack of dynamically bound continuations. Each call to $\widehat{\mathrm{tp}}$ pops the top continuation from this stack. The extra information needed to type the dynamic binding is not a single formula but the ordered list Σ of the types of the continuations present in the stack.

The type system, given in Figure 16 enjoys preservation of types under reduction.

Theorem (Subject reduction).

(i) *If* $\Gamma \vdash_\Sigma t : A; \Delta$ *and* $t \to s$, *then* $\Gamma \vdash_\Sigma s : A; \Delta$.
(ii) *If* $\Gamma \vdash_\Sigma c : \bot; \Delta$ *and* $c \to c'$, *then* $\Gamma \vdash_\Sigma c' : \bot; \Delta$.

10. Classical sequent calculus and $\overline{\lambda}\mu\tilde{\mu}$-calculus

10.1. Confluence of call-by-name and call-by-value disciplines. In Likavec and Lescanne [111], we deal with untyped $\overline{\lambda}\mu\tilde{\mu}$-calculus and its semantics, with complete proofs given in [110].

This work investigates some properties of $\overline{\lambda}\mu\tilde{\mu}_T$ and $\overline{\lambda}\mu\tilde{\mu}_Q$, the two subcalculi of untyped $\overline{\lambda}\mu\tilde{\mu}$-calculus of Curien and Herbelin [34], closed under the call-by-name and the call-by-value reduction, respectively. The syntax and reduction rules of $\overline{\lambda}\mu\tilde{\mu}$ were given in Section 5.

First of all, the proof of confluence for both versions of the $\overline{\lambda}\mu\tilde{\mu}$-calculus is given, adopting the method of parallel reductions given by Takahashi [146]. This approach consists of simultaneously reducing all the redexes existing in a term.

We present the proof for $\overline{\lambda}\mu\tilde{\mu}_T$, the proof for $\overline{\lambda}\mu\tilde{\mu}_Q$ being a straightforward modification of the proof for $\overline{\lambda}\mu\tilde{\mu}_T$. The complete proofs can be found in [110]. We denote the reduction defined by the three reduction rules for $\overline{\lambda}\mu\tilde{\mu}_T$ by $\to_{\mathrm{n}}$ and its reflexive, transitive, and closure by congruence by $\twoheadrightarrow_{\mathrm{n}}$.

First, we define the notion of parallel reduction $\Rightarrow_{\mathrm{n}}$ for $\overline{\lambda}\mu\tilde{\mu}_T$. We prove that $\twoheadrightarrow_{\mathrm{n}}$ is reflexive and transitive closure of $\Rightarrow_{\mathrm{n}}$, so in order to prove the confluence of $\twoheadrightarrow_{\mathrm{n}}$, it is enough to prove the diamond property for $\Rightarrow_{\mathrm{n}}$. The diamond property for $\Rightarrow_{\mathrm{n}}$, follows from the stronger "Star property" for $\Rightarrow_{\mathrm{n}}$ that we prove.

The parallel reduction, denoted by $\Rightarrow_{\mathrm{n}}$ is defined inductively, as follows:

$$\frac{}{x \Rightarrow_{\mathrm{n}} x}\ (g1_n) \qquad \frac{v \Rightarrow_{\mathrm{n}} v'}{\lambda x\,.\,t \Rightarrow_{\mathrm{n}} \lambda x\,.\,t'}\ (g2_n) \qquad \frac{c \Rightarrow_{\mathrm{n}} c'}{\mu\alpha\,.\,c \Rightarrow_{\mathrm{n}} \mu\alpha\,.\,c'}\ (g3_n)$$

$$\frac{}{\alpha \Rightarrow_{\mathrm{n}} \alpha}\ (g4_n) \qquad \frac{v \Rightarrow_{\mathrm{n}} v', E \Rightarrow_{\mathrm{n}} E'}{v \bullet E \Rightarrow_{\mathrm{n}} v' \bullet E'}\ (g5_n) \qquad \frac{c \Rightarrow_{\mathrm{n}} c'}{\tilde{\mu}x\,.\,c \Rightarrow_{\mathrm{n}} \tilde{\mu}x\,.\,c'}\ (g6_n)$$

$$\frac{v \Rightarrow_{\mathrm{n}} v', e \Rightarrow_{\mathrm{n}} e'}{\langle v \parallel e\rangle \Rightarrow_{\mathrm{n}} \langle v' \parallel e'\rangle}\ (g7_n) \qquad \frac{v_1 \Rightarrow_{\mathrm{n}} v_1', v_2 \Rightarrow_{\mathrm{n}} v_2', E \Rightarrow_{\mathrm{n}} E'}{\langle \lambda x\,.\,t_1 \parallel v_2 \bullet E\rangle \Rightarrow_{\mathrm{n}} \langle v_1'[x := v_2'] \parallel E'\rangle}\ (g8_n)$$

$$\frac{c \Rightarrow_{\mathrm{n}} c', E \Rightarrow_{\mathrm{n}} E'}{\langle \mu\alpha\,.\,c \parallel E\rangle \Rightarrow_{\mathrm{n}} c'[\alpha := E']}\ (g9_n) \qquad \frac{v \Rightarrow_{\mathrm{n}} v', c \Rightarrow_{\mathrm{n}} c'}{\langle v \parallel \tilde{\mu}x\,.\,c\rangle \Rightarrow_{\mathrm{n}} c'[x := v']}\ (g10_n).$$

It is easy to prove that for every term G:

1. $G \Rightarrow_{\mathrm{n}} G$;
2. If $G \to_{\mathrm{n}} G'$ then $G \Rightarrow_{\mathrm{n}} G'$;
3. If $G \Rightarrow_{\mathrm{n}} G'$ then $G \twoheadrightarrow_{\mathrm{n}} G'$;
4. If $G \Rightarrow_{\mathrm{n}} G'$ and $H \Rightarrow_{\mathrm{n}} H'$,
   then $G[x := H] \Rightarrow_{\mathrm{n}} G'[x := H']$ and $G[\alpha := H] \Rightarrow_{\mathrm{n}} G'[\alpha := H']$.

¿From 2. and 3. we conclude that $\twoheadrightarrow_{\mathrm{n}}$ is the reflexive and transitive closure of $\Rightarrow_{\mathrm{n}}$.

Next, we define the term G^* which is obtained from G by simultaneously reducing all the existing redexes of the term G.

$(*1_n)$ $x^* \equiv x$ $(*2_n)$ $(\lambda x\,.\,t)^* \equiv \lambda x\,.\,t^*$ $(*3_n)$ $(\mu\alpha\,.\,c)^* \equiv \mu\alpha\,.\,c^*$
$(*4_n)$ $\alpha^* \equiv \alpha$ $(*5_n)$ $(v \bullet E)^* \equiv v^* \bullet E^*$ $(*6_n)$ $(\widetilde{\mu}x\,.\,c)^* \equiv \widetilde{\mu}x\,.\,c^*$
$(*7_n)$ $(\langle v \parallel e\rangle)^* \equiv \langle v^* \parallel e^*\rangle$ if $\langle v \parallel e\rangle \neq \langle \lambda x\,.\,t_1 \parallel v_2 \bullet E\rangle$, $\langle v \parallel e\rangle \neq \langle \mu\alpha\,.\,c \parallel E\rangle$ and $\langle v \parallel e\rangle \neq \langle v \parallel \widetilde{\mu}x\,.\,c\rangle$
$(*8_n)$ $(\langle \lambda x\,.\,t_1 \parallel v_2 \bullet E\rangle)^* \equiv \langle v_1^*[x := v_2^*] \parallel E^*\rangle$
$(*9_n)$ $(\langle \mu\alpha\,.\,c \parallel E\rangle)^* \equiv c^*[\alpha := E^*]$
$(*10_n)$ $(\langle v \parallel \widetilde{\mu}x\,.\,c\rangle)^* \equiv c^*[x := v^*]$

We prove that if $G \Rightarrow_{\mathrm{n}} G'$ then $G' \Rightarrow_{\mathrm{n}} G^*$. Then it is easy to deduce the diamond property for $\Rightarrow_{\mathrm{n}}$: if $G_1 \,{}_{\mathrm{n}}\!\Leftarrow G \Rightarrow_{\mathrm{n}} G_2$ then $G_1 \Rightarrow_{\mathrm{n}} G' \,{}_{\mathrm{n}}\!\Leftarrow G_2$ for some G'. Finally, from the previous, it follows that $\overline{\lambda}\mu\widetilde{\mu}_T$ is confluent, i.e., if $G_1 \,{}_{\mathrm{n}}\!\twoheadleftarrow G \twoheadrightarrow_{\mathrm{n}} G_2$ then $G_1 \twoheadrightarrow_{\mathrm{n}} G' \,{}_{\mathrm{n}}\!\twoheadleftarrow G_2$ for some G'.

As a step towards a better understanding of denotational semantics of $\overline{\lambda}\mu\widetilde{\mu}$-calculus, its *untyped* call-by-value ($\overline{\lambda}\mu\widetilde{\mu}_Q$) and call-by-name ($\overline{\lambda}\mu\widetilde{\mu}_T$) versions are interpreted. Untyped $\overline{\lambda}\mu\widetilde{\mu}$-calculus is Turing-complete, hence a naive set-theoretic approach would not be enough. Continuation semantics of $\overline{\lambda}\mu\widetilde{\mu}_Q$ and $\overline{\lambda}\mu\widetilde{\mu}_T$ is given using the category of negated domains of [142], and Moggi's Kleisli category over predomains for the continuation monad [117]. Soundness theorems are given for both, call-by-value and call-by-name subcalculi, thus relating operational and denotational semantics. A detailed account on the literature on continuation semantics is also given. Lack of space forbids us to give a detailed account on the semantics here.

10.2. Strong normalisation in unrestricted $\overline{\lambda}\mu\widetilde{\mu}$-calculus. In Dougherty et al. [51], we develop a new intersection type system for the $\overline{\lambda}\mu\widetilde{\mu}$-calculus of Curien and Herbelin [34]. The system in this work improves on earlier type disciplines for $\overline{\lambda}\mu\widetilde{\mu}$ (including the current authors' [48, 49]): in addition to characterising the $\overline{\lambda}\mu\widetilde{\mu}$ expressions that are strongly normalising under free (unrestricted) reduction, the system enjoys the Subject reduction and the Subject expansion properties.

The set Type of *raw types* is generated from an infinite set $TVar$ of type-variables as follows

$$A, B ::= TVar \mid A \to B \mid A^\circ \mid A \cap B$$

where A° is the *dual* type of type A. We consider raw types modulo the equality generated by saying that (i) intersection is associative and commutative and (ii) for all raw types A, $A^{\circ\circ} = A$,

A *type* is either a *term-type* or a *coterm-type* or the special constant $\bot$. A raw type is a *term-type* if it is either a type variable, or of the form $(A_1 \to A_2)$ or $(A_1 \cap \cdots \cap A_k)$, $i \geqslant 2$ for term-types A_i, or of the form D° for a coterm-type D. A raw type is a *coterm-type* if it is either a coterm variable, or of the form A° for a term-type A or of the form $(D_1 \cap \cdots \cap D_k)$, $i \geqslant 2$ for coterm-types D_i. Note that every coterm-type is a type of the form A°, where A is a term-type, or an intersection of such types.

Each type other than $\bot$ is uniquely—up to the equivalences mentioned above—of one of the forms in the table below. Furthermore, for each type T there is a unique type which is T°. If T is a term-type [resp., coterm-type] then T° is a coterm-type

[resp., term-type].

	term-types	coterm-types
	τ	τ
	$(A_1 \to A_2)$	$(A_1 \to A_2)^\circ$
for $n \geqslant 2$:	$(A_1 \cap A_2 \cap \cdots \cap A_n)$	$(A_1 \cap A_2 \cap \cdots \cap A_n)^\circ$
for $n \geqslant 2$:	$(A_1^\circ \cap A_2^\circ \cap \cdots \cap A_n^\circ)^\circ$	$(A_1^\circ \cap A_2^\circ \cap \cdots \cap A_n^\circ)$.

The characterisation of the two columns as being "term-types" or "coterm-types" holds under the convention that the A_i displayed are all term-types.

We refer to types of the form $(A \to B)$ and $(A_1 \cap \cdots \cap A_k) \to B$ uniformly using the notation $(\bigcap A_i \to B)$, with the understanding that the $\bigcap A_i$ might refer to a single non-intersection type.

The type assignment system $\mathcal{M}^\cap$ is given by the typing rules in Figure 17, where v is any (co)variable.

$$\frac{}{\Sigma,\ v : (T_1 \cap \cdots \cap T_k) \vdash v : T_i}\ (\text{ax})$$

$$\frac{\Sigma,\ x : A \vdash r : B}{\Sigma \vdash \lambda x.r\ :\ A \to B}\ (\to \text{r}) \qquad \frac{\Sigma \vdash r : A_i \quad i = 1, \ldots, k \qquad \Sigma \vdash e : B^\circ}{\Sigma \vdash r \bullet e : ((A_1 \cap \cdots \cap A_k) \to B)^\circ}\ (\to \text{e})$$

$$\frac{\Sigma,\ \alpha : A^\circ \vdash c : \bot}{\Sigma \vdash \mu\alpha.c : A}\ (\mu) \qquad \frac{\Sigma,\ x : A \vdash c : \bot}{\Sigma \vdash \tilde{\mu}x.c : A^\circ}\ (\tilde{\mu})$$

$$\frac{\Sigma \vdash r : A \qquad \Sigma \vdash e : A^\circ}{\Sigma \vdash \langle r \parallel e \rangle : \bot}\ (\text{cut})$$

FIGURE 17. The typing system $\mathcal{M}^\cap$

In the system presented here there is no unrestricted $\cap$-introduction rule which is significant for the treatments of Subject reduction and Type soundness. Intersection types can be generated for redexes by the (μ) or $(\tilde{\mu})$ rules only. The rationale behind the new type system is to accept the introduction of an intersection only at specific positions and specific times when typing an expression, namely when an arrow is introduced on the left; then a type intersection is only introduced at the parameter position. Still, the new system types exactly all the strongly normalising expressions.

Example: The normal form $'lx.\mu\alpha.\langle x \parallel x \bullet \alpha\rangle$, which corresponds to the normal form $'lx.xx$ in $'l$-calculus, is not typable in $\overline{\lambda}\mu\tilde{\mu}$ with simple types. It is typable in the currently introduced system $\mathcal{M}^\cap$ by $'lx.\mu\alpha.\langle x \parallel x \bullet \alpha\rangle : A \cap (A \to B) \to B$.

Theorem (Subject expansion). *Let t and s be arbitrary terms or coterms and let v be a variable or covariable. Suppose $\Sigma \vdash t[v := s] : T$ and suppose that s is*

typable in context Σ. *Then there is a type* $D = (D_1 \cap \cdots \cap D_k)$, $k \geqslant 1$, *such that*

$$\Sigma \vdash s : D_i \quad \textit{for each } i \qquad \textit{and} \qquad \Sigma, v : D \vdash t : T.$$

Theorem (Main result). *A* $\overline{\lambda}\mu\tilde{\mu}$ *term is strongly normalising if and only if it is typable in* $\mathcal{M}^{\cap}$.

It is straightforward to prove that strong normalisation implies typeability using the fact that normal forms are typeable.

To prove strong normalisation under free reduction for typable expressions is more challenging. The difficulty using a traditional reducibility (or "candidates") argument arises from the critical pairs $\langle \mu' a.c \parallel \tilde{\mu} x.d \rangle$. Since neither of the expressions here can be identified as the preferred redex one cannot define candidates by induction on the structure of types.

The "symmetric candidates" technique in [6, 125] uses a fixed-point technique to define the candidates and suffices to prove strong normalisation for simply-typed $\overline{\lambda}\mu\tilde{\mu}$, but the interaction between intersection types and symmetric candidates is technically problematic.

In order to prove that typeable expressions are SN we first construct *pairs* (R, E) given by two non-empty sets $T \subseteq \Lambda_r$ and $C \subseteq \Lambda_e$. The pair (R, E) is *stable* if for every $r \in R$ and every $e \in E$, the command $\langle r \parallel e \rangle$ is SN. A pair (R, E) is *saturated* if

- whenever $\mu' a.c$ satisfies $\forall e \in E, c[\alpha := e]$ is SN then $\mu' a.c \in R$, and
- whenever $\tilde{\mu} x.c$ satisfies $\forall r \in R, c[x := r]$ is SN then $\tilde{\mu} x.c \in E$.

A pair (R, E) is *simple* if no term in R is of the form $\mu' a.c$ and no coterm in E is of the form $\tilde{\mu} x.c$.

We show that if the original pair is stable and simple, then we may always construct the saturated, stable extension. To achieve this we define the maps: $\Phi_r : 2^{\Lambda_e} \to 2^{\Lambda_r}$ and $\Phi_e : 2^{\Lambda_r} \to 2^{\Lambda_e}$ by

$$\begin{aligned} \Phi_r(Y) = \{ & r \mid r \text{ is of the form } \mu\alpha.c \text{ and } \forall e \in Y, c[\alpha := e] \text{ is SN} \} \\ & \cup \{ r \mid r \text{ is simple and } \forall e \in Y, \langle r \parallel e \rangle \text{ is SN} \} \end{aligned}$$

$$\begin{aligned} \Phi_e(X) = \{ & e \mid e \text{ is of the form } \tilde{\mu} x.c \text{ and } \forall r \in X, c[x := r] \text{ is SN} \} \\ & \cup \{ e \mid e \text{ is simple and } \forall r \in X, \langle r \parallel e \rangle \text{ is SN.} \} \end{aligned}$$

Since each of Φ_e and Φ_r is antimonotone, the maps $(\Phi_r \circ \Phi_e) : \Lambda_r \to \Lambda_r$ and $(\Phi_e \circ \Phi_r) : \Lambda_e \to \Lambda_e$ are monotone, so each of these maps has a complete lattice of fixed points, ordered by set inclusion.

We define different saturated pairs to interpret types depending on whether the type to be interpreted is (i) an arrow-type or its dual or (ii) an intersection or its dual.

If R is a simple set of SN terms let $R^{\uparrow}$ be the least fixed point of $(\Phi_r \circ \Phi_e)$ with the property that $R \subseteq R^{\uparrow}$. Analogously, $E^{\uparrow}$ is the least fixed point of $(\Phi_e \circ \Phi_r)$ such that $E \subseteq E^{\uparrow}$.

For interpreting the types that are intersections or their duals, we use the fact that the collection of fixed points of $(\Phi_r \circ \Phi_e)$ (and that of $(\Phi_e \circ \Phi_r)$) carries its own lattice structure under inclusion. We need the following definitions.

- Let $\mathsf{Fix}_{(\Phi_r \circ \Phi_e)}$ be the set of fixed points of the operator $(\Phi_r \circ \Phi_e)$. If $R_1, \ldots, R_k$ are fixed points of $(\Phi_r \circ \Phi_e)$, let $(R_1 \curlywedge \ldots \curlywedge R_k)$ denote the meet of these elements in the lattice $\mathsf{Fix}_{(\Phi_r \circ \Phi_e)}$.
- Let $\mathsf{Fix}_{(\Phi_e \circ \Phi_r)}$ be the set of fixed points of the operator $(\Phi_e \circ \Phi_r)$. Let $(E_1 \curlywedge \ldots \curlywedge E_k)$ denote the meet of fixed points of $(\Phi_e \circ \Phi_r)$.

Interpretation of types For each type T we define the set $[\![T]\!]$, maintaining the invariant that when T is a term-type then $[\![T]\!]$ is a fixed point of $(\Phi_r \circ \Phi_e)$ (set of terms) and when T is a coterm-type then $[\![T]\!]$ is a fixed point of $(\Phi_e \circ \Phi_r)$ (set of coterms).

- When T is $\bot$ then $[\![T]\!]$ is the set of SN commands.
- When T is a type variable we set R to be the set of term variables, then construct the pair $(R^\uparrow, \Phi_e(R^\uparrow))$. We then take $[\![T]\!]$ to be $R^\uparrow$ and $[\![T^\circ]\!]$ to be $\Phi_e(R^\uparrow)$.
- Suppose T is $(\bigcap A_i \to B)$. Set E to be $\{r \bullet e \mid \forall i, r \in [\![A_i]\!] \text{ and } e \in [\![B^\circ]\!]\}$ then construct the pair $(\Phi_r(E^\uparrow), E^\uparrow)$. We then take $[\![T]\!]$ to be $\Phi_r(E^\uparrow)$ and $[\![T^\circ]\!]$ to be $(E^\uparrow)$.
- When T is $(A_1 \cap A_2 \cdots \cap A_n)$, $n \geqslant 2$, we take $[\![T]\!]$ to be $([\![A_1]\!] \curlywedge \ldots \curlywedge [\![A_n]\!])$ and then take $[\![T^\circ]\!]$ to be $\Phi_e([\![T]\!])$.
- When T is $(A_1^\circ \cap A_2^\circ \cdots \cap A_n^\circ)^\circ$, $n \geqslant 2$, we take $[\![T^\circ]\!]$ to be $([\![A_1^\circ]\!] \curlywedge \ldots \curlywedge [\![A_n^\circ]\!])$ and then take $[\![T]\!]$ to be $\Phi_r([\![T^\circ]\!])$.

The following collects the information we need to prove Type soundness.

(1)
(2) For each type T, $[\![T]\!]$ is a set of SN (co)terms. $[\![(\bigcap A_i \to B)^\circ]\!] \supseteq \{r \bullet e \mid \forall i, r \in [\![A_i]\!] \text{ and } e \in [\![B^\circ]\!]\}$.
(3) $(\lambda x.b) \in [\![(\bigcap A_i \to B)]\!]$ if for all r such that $\forall i, r \in [\![A_i]\!]$ we have $b[x := r] \in [\![B]\!]$.
(4) $(\mu' a.c) \in [\![A]\!]$ if for all $e \in [\![A^\circ]\!]$ we have $c['a := e]$ SN. Similarly, $(\tilde{\mu} x.c) \in [\![A^\circ]\!]$ if for all $r \in [\![A]\!]$ we have $c[x := r]$ SN.
(5) $[\![(T_1 \cap \cdots \cap T_k)]\!] \subseteq ([\![T_i]\!] \cap \cdots \cap [\![T_k]\!])$.

Theorem (Type soundness). *If expression t is typable with type T then t is in $[\![T]\!]$.*

Since each $[\![T]\!]$ consists of SN expressions Type soundness implies that all typable expressions are SN.

General consideration of symmetry led us in [48, 49] to consider intersection *and union* types in symmetric λ-calculi. These papers characterised strong normalisation for call-by-name and call-by-value restrictions of the $\overline{\lambda}\mu\tilde{\mu}$-calculus, whereas the results in this work apply to unrestricted reduction. We might argue that if a term has type $A \cap B$, meaning that it denotes values which inhabit both A and B, then it can interact with any continuation that can receive an A-value *or* a B-value: such a continuation will naturally be expected to have the type $A \cup B$. But any type that can be the type of a variable can be the type of a coterm (via

the $\widetilde{\mu}$-construction) and any type that can be the type of a covariable can be the type of a term (via the μ-construction). This would suggest having intersections *and* unions for terms and continuations. It is well-known [123, 7] that the presence of union types causes difficulties for the Subject reduction property; unfortunately our attempt to recover Subject reduction in [48] was in error, as was pointed out to us by Hugo Herbelin [95]. Hence, this work only takes into account intersection types. The use of an explicit involution operator allows us to record the relationship between an intersection $(A \cap B)$ and its dual type $(A \cap B)^\circ$. The "classical" nature of the underlying logic is reflected in the "double-negation".

10.3. Dual calculus. Wadler's Dual calculus was introduced in [151, 152] as a term calculus which corresponds to classical sequent logic. In Dougherty et al. [52], we investigate some syntactic properties of Wadler's Dual calculus and establish some of the key properties of the underlying reduction.

We give now the syntax and reduction rules of Wadler's Dual calculus (although in our slightly altered notation). We distinguish three syntactic categories: *terms*, *coterms*, and *statements*. Terms yield values, while coterms consume values. A statement is a cut of a term against a coterm.

If r, q range over the set Λ_r of terms, e, f range over the set Λ_e of coterms, and c ranges over statements, then the syntax of the Dual calculus is given by the following:

$$\begin{array}{lrcl} \text{Term:} & r, q & ::= & x \mid \langle r,\, q\rangle \mid \langle r\rangle\text{inl} \mid \langle r\rangle\text{inr} \mid [e]\text{not} \mid \mu\alpha\,.\,c \\ \text{Coterm:} & e, f & ::= & \alpha \mid [e,\, f] \mid \text{fst}[e] \mid \text{snd}[e] \mid \text{not}\langle r\rangle \mid \widetilde{\mu}x\,.\,c \\ \text{Command:} & c & ::= & (\!|\, r \bullet e \,|\!) \end{array}$$

where x ranges over a set of term variables Var_{R}, $\langle r,\, q\rangle$ is a pair, $\langle r\rangle$inl ($\langle r\rangle$inr) is an injection on the left (right) of the sum, $[e]$not is a complement of a coterm, and $\mu\alpha\,.\,c$ is a covariable abstraction. Next, α ranges over a set of covariables Var_{L}, $[e,\, f]$ is a case, fst$[e]$ (snd$[e]$) is a projection from the left (right) of a product, not$\langle r\rangle$ is a complement of a term, and $\widetilde{\mu}x\,.\,c$ is a variable abstraction. Finally $(\!|\, r \bullet e \,|\!)$ is a cut. The term variables can be bound by μ-abstraction, whereas the coterm variables can be bound by $\widetilde{\mu}$-abstraction. The sets of free term and coterm variables, Fv_{R} and Fv_{L}, are defined as usual, respecting Barendregt's convention [10] that no variable can be both, bound and free, in the expression.

The reduction rules for an unrestricted calculus are given in Figure 18.

$$\begin{array}{lrcl} (\beta\widetilde{\mu}) & (\!|\, r \bullet \widetilde{\mu}x\,.\,c \,|\!) & \to & c[x := r] \\ (\beta\mu) & (\!|\, \mu\alpha\,.\,c \bullet e \,|\!) & \to & c[\alpha := e] \\ (\beta\wedge) & (\!|\, \langle r,\, q\rangle \bullet \text{fst}[e] \,|\!) & \to & (\!|\, r \bullet e \,|\!) \\ (\beta\wedge) & (\!|\, \langle r,\, q\rangle \bullet \text{snd}[e] \,|\!) & \to & (\!|\, q \bullet e \,|\!) \\ (\beta\vee) & (\!|\, \langle r\rangle\text{inl} \bullet [e,\, f] \,|\!) & \to & (\!|\, r \bullet e \,|\!) \\ (\beta\vee) & (\!|\, \langle r\rangle\text{inr} \bullet [e,\, f] \,|\!) & \to & (\!|\, r \bullet f \,|\!) \\ (\beta\neg) & (\!|\, [e]\text{not} \bullet \text{not}\langle r\rangle \,|\!) & \to & (\!|\, r \bullet e \,|\!) \end{array}$$

FIGURE 18. Reduction rules for the Dual calculus

The basic system is not confluent, inheriting the well-known anomaly of classical cut-elimination. Wadler recovers confluence by restricting to reduction strategies corresponding to (either of) the call-by-value or call-by-name disciplines.

The two subcalculi Dual_R and Dual_L are obtained by giving the priority to $(\widetilde{\mu})$ redexes or to (μ) redexes, respectively. Dual_R is defined by refining the reduction rule $(\beta\mu)$ as follows

$$\langle\!| \mu\alpha.c \bullet \underline{e} |\!\rangle \to c[\alpha := \underline{e}] \qquad \text{provided } \underline{e} \text{ is a coterm not of the form } \widetilde{\mu}x.c'$$

and Dual_L is defined similarly by refining the reduction rule $(\beta\widetilde{\mu})$ as follows

$$\langle\!| \underline{r} \bullet \widetilde{\mu}x.c |\!\rangle \to c[x := \underline{r}] \qquad \text{provided } \underline{r} \text{ is a term not of the form } \mu'a.c'.$$

We show that once the "critical pair" in the reduction system is removed by giving priority to either the "left" or to the "right" reductions, confluence holds in both the typed and untyped versions of the term calculus. Although the critical pair can be disambiguated in two ways, the proof we give dualises to yield confluence results for each system. The proof is an application of Takahashi's parallel reductions technique [146], analogous to the one used in [111] and with details of the proves given in [110].

A complementary perspective to that of considering the Dual calculus as term-assignment to logic proofs is that of viewing sequent proofs as typing derivations for raw expressions. The set Type of types corresponds to the logical connectives; for the Dual calculus the set of types is given by closing a set of *base types* X under conjunction, disjunction, and negation

$$A, B \ ::= \ X \mid A \wedge B \mid A \vee B \mid \neg A.$$

Type bases have two components, the *antecedent* a set of bindings of the form $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, and the *succedent* of the form $\Delta = \alpha_1 : B_1, \ldots, \alpha_k : B_k$, where x_i, α_j are distinct for all $i = 1, \ldots, n$ and $j = 1, \ldots, k$. The judgements of the type system are given by the following:

$$\Gamma \vdash \Delta, \boxed{r : A} \qquad \boxed{e : A}, \Gamma \vdash \Delta \qquad c : (\Gamma \vdash \Delta)$$

where Γ is the antecedent and Δ is the succedent. The first judgement is the typing for a term, the second is the typing for a coterm and the third one is the typing for a statement. The box denotes a distinguished output or input, i.e., a place where the computation will continue or where it happened before. The type assignment system for the Dual calculus, introduced by Wadler [151, 152], is given in Figure 10.3.

We prove strong normalisation (SN) for unrestricted reduction of typed terms, including expansion rules capturing extensionality. The proof is a variation on the "semantical" method of reducibility, where types are interpreted as *pairs* of sets of terms. Our proof technique uses a fixed-point construction similar to that in [6] but the technique is considerably simplified.

The approach is similar to the one given for [51] so we just present the details that differ. The pairs are defined analogously, as well as the notion of stable, saturated, and simple pairs.

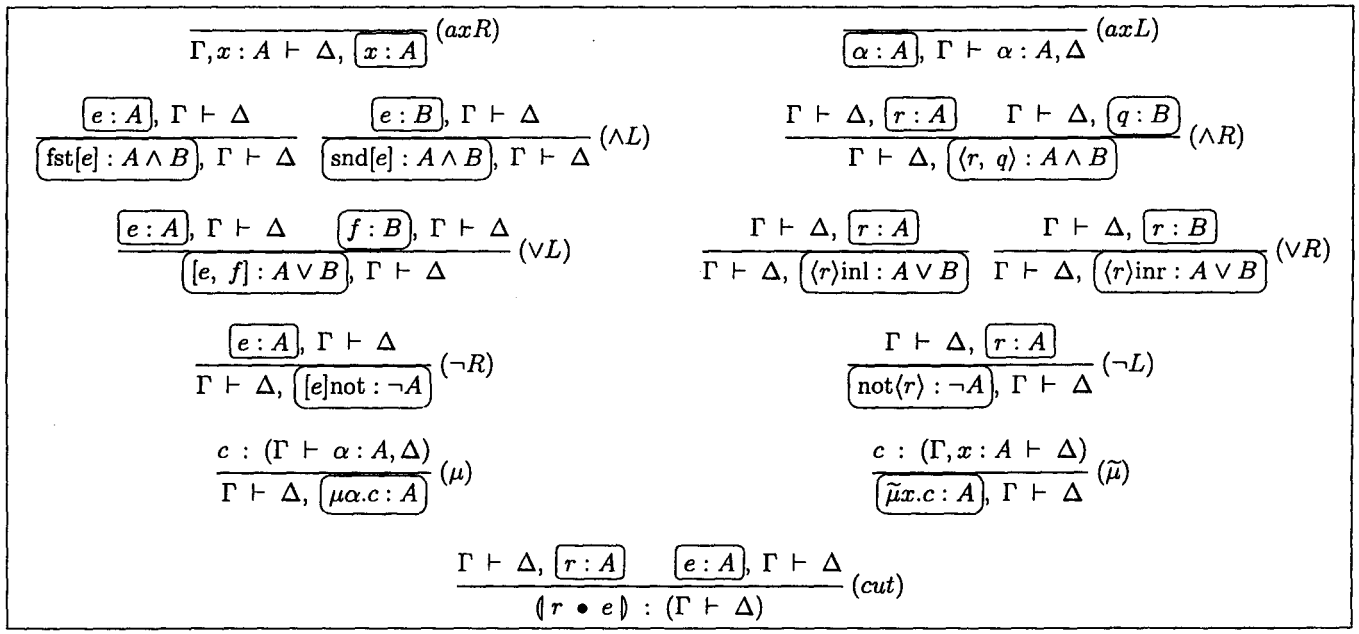


FIGURE 19. Type system for the Dual calculus

We can always expand a pair to be saturated. Also if the original pair is stable and simple, then we may always construct the saturated, stable extension.

We define the following constructions on pairs, where script letters denote pairs, and if $\mathcal{P}$ is a pair, $\mathcal{P}_{\mathrm{R}}$ and $\mathcal{P}_{\mathrm{L}}$ denote its component sets of terms and coterms.

Let $\mathcal{P}$ and $\mathcal{Q}$ be pairs.

- The pair $(\mathcal{P} \curlywedge \mathcal{Q})$ is given by:
  - $(\mathcal{P} \curlywedge \mathcal{Q})_{\mathrm{R}} = \{\langle r_1, r_2\rangle \mid r_1 \in \mathcal{P}_{\mathrm{R}},\ r_2 \in \mathcal{Q}_{\mathrm{R}}\}$
  - $(\mathcal{P} \curlywedge \mathcal{Q})_{\mathrm{L}} = \{\mathrm{fst}[e] \mid e \in \mathcal{P}_{\mathrm{L}}\} \cup \{\mathrm{snd}[e] \mid e \in \mathcal{Q}_{\mathrm{L}}\}$.
- The pair $(\mathcal{P} \curlyvee \mathcal{Q})$ is given by:
  - $(\mathcal{P} \curlyvee \mathcal{Q})_{\mathrm{R}} = \{\langle r\rangle\mathrm{inl} \mid r \in \mathcal{P}_{\mathrm{R}}\} \cup \{\langle r\rangle\mathrm{inr} \mid r \in \mathcal{Q}_{\mathrm{R}}\}$
  - $(\mathcal{P} \curlyvee \mathcal{Q})_{\mathrm{L}} = \{[e_1, e_2] \mid e_1 \in \mathcal{P}_{\mathrm{L}}.\ e_2 \in \mathcal{Q}_{\mathrm{L}}\}$.
- The pair $\mathcal{P}^\circ$ is given by:
  - $(\mathcal{P}^\circ)_{\mathrm{R}} = \{[e]\mathrm{not} \mid e \in \mathcal{P}_{\mathrm{L}}\}$
  - $(\mathcal{P}^\circ)_{\mathrm{L}} = \{\mathrm{not}\langle r\rangle \mid r \in \mathcal{P}_{\mathrm{R}}\}$.

Each of $(\mathcal{P} \curlywedge \mathcal{Q})$, $(\mathcal{P} \curlyvee \mathcal{Q})$, and $\mathcal{P}^\circ$ is simple and we show that if $\mathcal{P}$ and $\mathcal{Q}$ are stable pairs, then $(\mathcal{P} \curlywedge \mathcal{Q})$, $(\mathcal{P} \curlyvee \mathcal{Q})$, and $\mathcal{P}^\circ$ are each stable.

The type-indexed family of pairs $\mathcal{S} = \{\mathcal{S}^T \mid T \in \mathsf{Type}\}$ is defined as follows, which is our notion of reducibility candidates for the Dual calculus:

- When T is a base type, $\mathcal{S}^T$ is any stable saturated extension of $(Var_{\mathrm{R}}, Var_{\mathrm{L}})$.
- $\mathcal{S}^{A\wedge B}$ is any stable saturated extension of $(\mathcal{S}^A \curlywedge \mathcal{S}^B)$.
- $\mathcal{S}^{A\vee B}$ is any stable saturated extension of $(\mathcal{S}^A \curlyvee \mathcal{S}^B)$.
- $\mathcal{S}^{\neg A}$ is any stable saturated extension of $(\mathcal{S}^A)^\circ$.

Next we prove that typeable terms and coterms lie in the candidates $\mathcal{S}$, i.e., if term r is typeable with type A then r is in $\mathcal{S}^A_{\mathrm{R}}$ and if coterm e is typeable with type A then e is in $\mathcal{S}^A_{\mathrm{L}}$. Since $\mathcal{S}^A_{\mathrm{R}}$ and $\mathcal{S}^A_{\mathrm{L}}$ consist of SN expressions, it follows that typeable terms and coterms are SN. If $t = c$ is a typeable statement then it suffices to observe that, taking $`a$ to be any covariable not occurring in $c$, the term $\mu`a.c$ is typeable. This proves the strong normalisation of all typeable expressions of the calculus.

10.4. Symmetric calculus. Another interesting calculus expressing a computational interpretation of classical logic is the Symmetric Lambda Calculus of Barbanera and Berardi [6], which was originally used to extract the constructive content of classical proofs. In Dougherty et al. [50] we explore the use of intersection types for symmetric proof calculi. More specifically we characterise termination in the (propositional version of) the Symmetric Lambda Calculus of Barbanera and Berardi [6].

The syntax of λ^{sym} expressions is given by the following:

$$t := x \mid \langle t_1, t_2\rangle \mid \sigma_1(t), \mid \sigma_2(t), \mid \lambda x.c \mid (t_1 * t_2).$$

We depart from [6] in that we treat the operator $*$ as syntactically commutative.

The reduction rules of the calculus are

$$(\lambda x.b * a) \to b[x := a] \qquad (\langle t_1, t_2\rangle * \sigma_i(u)) \to \langle t_i, u\rangle$$
$$\lambda x.(b * x) \to b \text{ if } x \text{ not free in } b.$$

The set Type of *raw types* is generated from an infinite set $TVar$ of type-variables as follows

$$A, B ::= TVar \mid A \wedge B \mid A \vee B \mid A^{\perp} \mid A \cap B.$$

We consider raw types modulo the equations

$$A^{\perp\perp} = A \qquad (A \wedge B)^{\perp} = A^{\perp} \vee B^{\perp} \qquad (A \vee B)^{\perp} = A^{\perp} \wedge B^{\perp}.$$

A *type* is either an equivalence class modulo these equations or the special type $\perp$. Note that by orienting the equations above left-to-right each type has a normal form, in which the $(\cdot)^{\perp}$ operator is applied only to type variables or intersections. It is then easy to see that each type other than $\perp$ is uniquely of one of the following forms (where τ is a type variable):

$$\tau \quad \tau^{\perp} \quad (A_1 \wedge \cdots \wedge A_n) \quad (A_1 \vee \cdots \vee A_n) \quad (A_1 \cap \cdots \cap A_n) \quad (A_1 \cap \cdots \cap A_n)^{\perp}.$$

The type assignment system $\mathcal{B}$ is given by the typing rules in Figure 20.

$$\frac{}{\Sigma,\ x : (T_1 \cap \cdots \cap T_k) \vdash x : T_i}\ (\text{ax})$$

$$\frac{\Sigma \vdash t_1 : A_1 \qquad \Sigma \vdash t_2 : A_2}{\Sigma \vdash \langle t_1, t_2 \rangle : A_1 \wedge A_2}\ (\wedge) \qquad \frac{\Sigma \vdash t : A_i}{\Sigma \vdash \sigma_i(t) : A_1 \vee A_2}\ (\vee)$$

$$\frac{\Sigma,\ x : A \vdash c : \perp}{\Sigma \vdash \lambda x.c : A^{\perp}}\ (\perp) \qquad \frac{\Sigma \vdash p : A \qquad \Sigma \vdash q : A^{\perp}}{\Sigma \vdash (p * q) : \perp}\ (\text{cut})$$

FIGURE 20. Typing rules of the system $\mathcal{B}$

The symmetry in classical calculi blocks a straightforward adaptation of the traditional reducibility technique which uses the fact that function types are "higher" in a natural sense than argument types, permitting semantic definitions to proceed by induction on types. In this paper we adapt the symmetric candidates technique to the intersection-types setting. As we can see, this technique applies generally to all of the symmetric proof-calculi we have investigated, including the $\overline{\lambda}\mu\tilde{\mu}$-calculus of Curien and Herbelin [34, 51], and the Dual calculus of Wadler [151, 152].

The key to the symmetric candidates technique is to interpret types in certain families of *saturated* sets which are closed under inverse β-reduction. The problem in the intersection types setting arises since in standard semantics of intersection types, the interpretation of an intersection type $(A \cap B)$ is the intersection of the interpretations of A and B and in general *intersections of saturated sets are not saturated,*

A consequence of this fact is that the standard typing rule for intersection-introduction is not sound. So our type system has an intersection-elimination rule

only. This is not a problem since intersection-introduction is not needed for characterising termination. In the absence of intersection-introduction, terms receive a type which is an intersection by double-negation elimination.

Theorem (Main result). *A λ^{sym} term is terminating if and only if it is typable in $\mathcal{B}$.*

The direction "every terminating term is typable" follows the standard pattern from traditional λ-calculus where the standard intersection-introduction typing rule is not needed.

The proof that every typable term is terminating is analogous to the one given for [51] and [52]. We only briefly account for the differences.

We consider pairs $\{X_0, X_1\}$ which are *stable* if for every $r \in X_0$ and every $e \in X_1$, the command $(r * e)$ is terminating. They are *saturated* if for each i,

whenever $\lambda x.c$ satisfies: $\forall e \in X_i, c[x := e]$ is terminating, then $\lambda x.c \in X_{1-i}$.

An expression is *simple* if it is not a 'l-abstraction; a set X is simple if each term in X is simple.

We define the map $\Phi : 2^{\Lambda} \to 2^{\Lambda}$ by

$$\Phi(X) = \{e \mid e \text{ is of the form } \lambda x.c \text{ and } \forall r \in X, c[x := r] \text{ is terminating}\}$$
$$\cup \{e \mid e \text{ is simple and } \forall r \in X, (r * e) \text{ is terminating}\}.$$

If $X \neq \emptyset$ then $\Phi(X)$ is a set of terminating terms and if $X \subseteq SN$ then all variables are in $\Phi(X)$. Φ is antimonotone, hence $(\Phi \circ \Phi) = \Phi^2$ is monotone and has a complete lattice of fixed points, ordered by set inclusion.

If X is a simple set of terminating terms we denote by $X^{\uparrow}$ the least fixed point of Φ^2 with the property that $X \subseteq X^{\uparrow}$. Furthermore, let Fix_{Φ^2} be the set of fixed points of the operator Φ^2. If $R_1, \ldots, R_k$ are fixed points of Φ^2, let $(R_1 \curlywedge \ldots \curlywedge R_k)$ denote the meet of these elements in the lattice Fix_{Φ^2}.

Interpretation of types For each type T we define the set $[\![T]\!]$ as follows.

(1) When T is $\bot$ then $[\![T]\!]$ is the set of terminating terms.
(2) When T is a type variable we set R to be the set of term variables, then construct the pair $(R^{\uparrow}, \Phi(R^{\uparrow}))$. We then take $[\![T]\!]$ to be $R^{\uparrow}$ and $[\![T^{\bot}]\!]$ to be $\Phi(R^{\uparrow})$.
(3) Suppose T is $(A_1 \wedge A_2)$. Set R to be $\{\langle t_1, t_2\rangle \mid t_i \in [\![A_i]\!],\ i = 1, 2\}$. We then take $[\![T]\!]$ to be $(R^{\uparrow})$ and $[\![T^{\bot}]\!] = [\![A_1{}^{\bot} \vee A_2{}^{\bot}]\!]$ to be $\Phi(R^{\uparrow})$.
(4) When T is $(A_1 \cap A_2 \cdots \cap A_n)$, $n \geqslant 2$, we take $[\![T]\!]$ to be $([\![A_1]\!] \curlywedge \ldots \curlywedge [\![A_n]\!])$ and then take $[\![T^{\bot}]\!]$ to be $\Phi([\![T]\!])$.

Note that the interpretation $[\![A_1 \vee A_2]\!]$ of a disjunction-type is determined in part 3 above since any type $B_1 \vee B_2$ is the Dual of $B_1{}^{\bot} \wedge B_2{}^{\bot}$.

We prove the following, which is the key for proving the Type soundnes.

(1) $[\![T]\!]$ is a set of terminating terms.
(2) $[\![A_1 \wedge A_2]\!] \supseteq \{\langle t_1, t_2\rangle \mid t_i \in [\![A_i]\!],\ i = 1, 2\}$.
(3) $[\![A_1 \vee A_2]\!] \supseteq \{\sigma_1(p) \mid p \in [\![A_1]\!]\} \cup \{\sigma_2(p) \mid p \in [\![A_2]\!]\}$.
(4) $(\lambda x.c) \in [\![A]\!]$ if for all $e \in [\![A^{\bot}]\!]$ we have $c[x := e]$ terminates.

(5) $[\![(A_1 \cap \cdots \cap A_k)]\!] \subseteq ([\![A_1]\!] \cap \cdots \cap [\![A_k]\!])$.

Since each $[\![T]\!]$ consists of terminating expressions the following theorem implies that all typable expressions are terminating.

Theorem (Type soundness). *If expression t is typable with type T then $t \in [\![T]\!]$.*

11. Application in programming language theory

11.1. Functional languages. λ-calculus The basic concept of programming languages is the concept of a function, more precisely of intensional (or computational) function considered as a composition of computational steps, i.e., as algorithms (or methods). A universal model of computational functions is Church's λ-calculus [27]. λ-calculus as a simple language is very convenient to describe the semantics of programming languages (it is even used as a core for the languages Lisp, Algol, Scheme, ML, Haskell, etc).

The λ-calculus exists in basically two main flavours: call-by-name (of which Haskell implements the call-by-need variant) and call-by-value (as in Scheme, ML, C, Java, etc). Call-by-name has been extensively studied (see e.g., Barendregt [10], Krivine [105]) and call-by-value reasonably well too.

Classical λ-calculus. The $\lambda\mu$-calculus is an extension of λ-calculus with an operator similar to the `call-cc` operator that can be found in Scheme and ML. It also models weaker operators, such as `break` and `return` in C and Java.

The $\lambda\mu$-calculus is the prototypical formulation of a classical λ-calculus. As λ-calculus, $\lambda\mu$-calculus exists in call-by-name and call-by-value variants, the latter being a rather intricate structure to study [62, 133].

The $\overline{\lambda}\mu\tilde{\mu}$-calculus [34] is an improvement over $\lambda\mu$-calculus. It is an elegant calculus that exhibits different forms of symmetries. One of them is a symmetry between call-by-name and call-by-value which allows to significantly reduce the syntactic complexity of the call-by-value calculus compared to $\lambda\mu$-calculus.

Call-by-value and call-by-name delimited continuation. Historically, delimited control came with ad hoc operators for composing continuations: Felleisen [61] had a calculus that included a control operator *control* a delimiter *prompt* (denoted by $\mathcal{F}$ and #, respectively); Danvy and Filinski [36] had an operator `shift` to compose continuations and an operator `reset` to delimit them (these were also written $\mathcal{S}$ and $< \_ >$). Control operators are connected to classical logic, as first investigated by Griffin [94].

From [66], it is known that `shift` and `reset` are equivalent to the combination of Scheme's `call-cc`, Felleisen's *abort* and `reset`, and hence equivalent to $\mathcal{C}$ and *reset*. From [25], it is known that *control* and *prompt* are also equivalent to *shift* and *reset*, in spite that *control* is semantically more complex to study than $\mathcal{C}$ or `shift`. The simplicity of the semantics of `shift` together with its relevance for some programming applications contributed to set `shift` as a reference in delimited control. And this is so in spite (it seems that) it has never been studied until now as part of a dedicated λ-calculus of delimited control.

As shown by Ariola et al. [4], a fine-grained $\lambda\mu\widehat{\mathsf{tp}}$-calculus of delimited control of the strength of `shift` and `reset` is obtained if one starts from $\lambda\mu$-calculus and extends it first by a notation tp for the "toplevel" continuation, then by a toplevel delimiter. A possible interpretation for this toplevel delimiter is as a *dynamic* binder of tp, what justifies to interpret the resulting call-by-value calculus, called as an extension of call-by-value $\lambda\mu$-calculus with a single dynamically bound continuation variable $\widehat{\mathsf{tp}}$, where the hat on tp emphasises the dynamic treatment of the variable. A typical analogy for the dynamic continuation variable here is exception handling: each call to $\widehat{\mathsf{tp}}$ is dynamically bound to the closest surrounding $\widehat{\mathsf{tp}}$ binder, in exactly the same way as a raised exception is dynamically bound to the closest surrounding handler. The expressiveness of this calculus was shown by simulating the operational semantics of `shift` and `reset` and of most standard control operators, such as E and A (*abort*) of Felleisen's, `call/cc` (the implementation of `call-cc` in Scheme).

$$
\begin{array}{lcl}
\mathcal{S}\ M & \triangleq & \mu\alpha.[\widehat{\mathsf{tp}}](M\ \lambda x.\mu\widehat{\mathsf{tp}}.[\alpha]x) \\
< M > & \triangleq & \mu\widehat{\mathsf{tp}}.[\widehat{\mathsf{tp}}]M \\
\mathcal{A}\,M & \triangleq & \mu\_.[\widehat{\mathsf{tp}}]M \\
\mathcal{C}\ (\lambda k.M) & \triangleq & \mu\alpha_k.[\widehat{\mathsf{tp}}](M\ \lambda x.\mu\_.[\alpha_k]x) \\
\texttt{call/cc}\ (\lambda k.M) & \triangleq & \mu\alpha_k.[\alpha_k](M\ \lambda x.\mu\_.[\alpha_k]x)
\end{array}
$$

Herbelin and Ghilezan [98] proposed an approach to call-by-name delimited control. They devised a call-by-name variant of $\lambda\mu\widehat{\mathsf{tp}}$, for Ariola et al's call-by-value calculus of delimited control [4].

Continuation-passing-style semantics is given by a CPS transformation of the $\lambda\mu\widehat{\mathsf{tp}}$ into λ-calculus.

$$
\begin{array}{lcl}
x^* & \triangleq & x \\
(\lambda x.M)^* & \triangleq & \lambda(x,k).M^*\,k \\
(M\,N)^* & \triangleq & \lambda k.M^*\,(N^*,k) \\
(\mu\alpha.c)^* & \triangleq & \lambda k_\alpha.c^* \\
([\alpha]M)^* & \triangleq & M^*\,k_\alpha \\
(\mu\widehat{\mathsf{tp}}.c)^* & \triangleq & c^* \\
([\widehat{\mathsf{tp}}]M)^* & \triangleq & M^*
\end{array}
$$

FIGURE 21. Call-by-name CPS translation of $\lambda\mu\widehat{\mathsf{tp}}$

We present the behaviour of call-by-name $\lambda\mu\widehat{\mathsf{tp}}$ on standard examples that uses delimited control. We consider the example of list traversal that is used to emphasise the differences between Felleisen's operator $\mathcal{F}$ and *shift*. We extend $\lambda\mu\widehat{\mathsf{tp}}$ with a fixpoint operator, list constructors and a list destructor:

$$
\begin{array}{lcl}
M, N & ::= & \ldots \mid \nu_x.M \mid [] \mid M::N \\
& \mid & \texttt{if}\ M\ \texttt{is}\ x::y\ \texttt{then}\ M\ \texttt{else}\ M
\end{array}
$$

and we extend call-by-name reduction with the rules

$$\begin{array}{lcl}
\nu_x.M & \to & M[x := \nu_x.M] \\
\texttt{if } [] \texttt{ is } x::y \texttt{ then } M_2 \texttt{ else } M_1 & \to & M_1 \\
\texttt{if } M::N \texttt{ is } x::y \texttt{ then } M_2 \texttt{ else } M_1 & \to & M_2[x := M][y := N] \\
\texttt{if } \mu\alpha.c \texttt{ is } x::y \texttt{ then } M_2 \texttt{ else } M_1 & \to & \\
\multicolumn{3}{c}{\mu\alpha.c[\alpha := [\alpha]\,(\texttt{if } \Box \texttt{ is } x::y \texttt{ then } M_2 \texttt{ else } M_1)]}
\end{array}$$

In informal ML syntax, the example is the following

```
let traverse l = let rec visit l = match l with
| [] -> []
| a::l' -> visit (shift (fun k -> a :: k l'))
                        in reset (visit l) in traverse [1;2;3]
```

Translated into $\Lambda\mu$, it gives

$$v\ (n_1::n_2::n_3::[])$$

where v is $\nu_f.(\lambda l.\texttt{if } l \texttt{ is } a::l' \texttt{ then } f\ (\mu\alpha.a::[\alpha]l')\ \texttt{else } [])$. Translated into $\lambda\mu\widehat{\texttt{tp}}$, v is

$$\nu_f.(\lambda l.\texttt{if } l \texttt{ is } a::l' \texttt{ then } f\ (\mu\alpha.[\widehat{\texttt{tp}}]a::\mu\widehat{\texttt{tp}}.[\alpha]l')\ \texttt{else } []).$$

Let ϵ be an arbitrary continuation distinct from $\widehat{\texttt{tp}}$. We write l_i for $n_i::\ldots::n_3::[]$. We list the steps of the reduction of $[\epsilon](v\ l_1)$:

$$\begin{array}{l}
[\epsilon]v\ l_1 \\
\to\ [\epsilon](\lambda l.\texttt{if } l \texttt{ is } a::l' \texttt{ then } v\ (\mu\alpha.a::[\alpha]l')\ \texttt{else } [])\ l_1 \\
\to\ [\epsilon]\texttt{if } l_1 \texttt{ is } a::l' \texttt{ then } v\ (\mu\alpha.a::[\alpha]l')\ \texttt{else } [] \\
\to\ [\epsilon]v\ (\mu\alpha.n_1::[\alpha]l_2) \\
\to\!\!\to\ \texttt{if } (\mu\alpha.n_1::[\alpha]l_2) \texttt{ is } a::l' \texttt{ then } v\ (\mu\alpha.a::[\alpha]l')\ \texttt{else } [] \\
\to\ [\epsilon]\mu\alpha.n_1::[\alpha](\texttt{if } l_2 \texttt{ is } a::l' \texttt{ then } v\ (\mu\alpha.a::[\alpha]l')\ \texttt{else } []) \\
\to\ n_1::[\epsilon](\texttt{if } l_2 \texttt{ is } a::l' \texttt{ then } v\ (\mu\alpha.a::[\alpha]l')\ \texttt{else } []) \\
\to\ n_1::[\epsilon](v\ (\mu\alpha.n_2::[\alpha]l_3)) \\
\to\!\!\to\ n_1::[\epsilon](\mu\alpha.n_2::[\alpha](v\ l_3)) \\
\to\!\!\to\ n_1::n_2::[\epsilon](v\ l_3) \\
\to\!\!\to\ n_1::n_2::[\epsilon](\mu\alpha.n_3::[\alpha](v\ [])) \\
\to\!\!\to\ n_1::n_2::n_3::[\epsilon](v\ []) \\
\to\!\!\to\ n_1::n_2::n_3::[\epsilon][]
\end{array}$$

Otherwise said, the list traversal program copies its argument and shifts its continuation to the tail of the list.

11.2. Object-oriented languages. The aim of the following works was to give the basis for designing a calculus that combines class-based features with object-based ones. We propose two extensions of the "Core Calculus of Classes and Mixins" of [22], one with higher-order, composable mixins, the second one with incomplete objects.

Mixins [24] are subclass definitions parameterised over a superclass and were introduced as an alternative to some forms of multiple inheritance. A mixin can

be seen as a function that, given one class as an argument, produces a subclass, by adding and/or overriding certain sets of methods.

The calculus proposed in Bettini et al. [13, 14] extends the core calculus of classes and mixins of [22] with *higher-order* mixins. In this extension a mixin can: (*i*) be applied to a class to create a fully-fledged subclass; (*ii*) be *composed* with another mixin to obtain yet another mixin with more functionalities. In what we believe is quite a general framework, we give directions for designing a programming language equipped with higher-order mixins, although our study is not based on any actual object-oriented language.

In the calculus proposed in Bettini et al. [18, 17, 16, 15] we extend the core calculus of classes and mixins of [22] with *incomplete objects*. In addition to standard class instantiation, it is also possible to *instantiate mixins* thus obtaining *incomplete objects*.

Incomplete objects can be completed in two ways: (*i*) via *method addition*, (*ii*) via *object composition*, that composes an incomplete object with a complete one that contains all the required methods. When a method is added, it becomes an effective component of the host object, meaning that the methods of the host object may invoke it, but also the new added method can use any of its sibling methods. The type system ensures that all method additions and object compositions are type safe and that only "complete" methods are invoked on objects. This way the type information at the mixin level is fully exploited, obtaining a "tamed" and safe object-based calculus.

The metatheory of both extensions is studied in Likavec [110]. In particular, the soundness property is proved, to guarantee the absence of run-time "message-not-understood" errors.

In addition, in [18] the calculus is endowed with width subtyping on complete objects, which provides enhanced flexibility while avoiding possible conflicts between method names.

Part 3 – Related work

Related work on computational interpretation of logic. The $\lambda\mu$-calculus of Parigot [121] embodies a Curry–Howard correspondence for classical natural deduction. It was introduced in call-by-name style, followed by a call-by-value variant, proposed by Ong and Stewart [120].

Herbelin [96] proposed the first "sequent" λ-calculus, named $\bar{\lambda}$, for which bijective correspondence between normal simply typed terms and cut-free proofs of the appropriate restriction of the Gentzen's LJ was obtained. He considered a λ-calculus with an explicit operator of substitution and substitution propagation rules. Each cut-elimination step corresponds to β-reduction, a substitution propagation or concatenation. However, this bijection failed to extend to sequent calculus with cuts.

After that, intuicionistic sequent λ-calculi were proposed by several authors, Barendregt and Ghilezan [12], Dyckhoff and Pinto [55], Espirito Santo and Pinto [60], among others.

The $\overline{\lambda}\mu\widetilde{\mu}$-calculus of Curien and Herbelin [34] provides a symmetric computational interpretation of classical sequent style logic. Expressions in $\overline{\lambda}\mu\widetilde{\mu}$ represent derivations in the sequent calculus proof system and reduction reflects the process of cut-elimination. This calculus provides an environment for a more fine-grained analysis of calculations within languages with control operators. Since its introduction, Curien and Herbelin's calculus has had a strong influence on the further understanding between calculi with control operators and classical logic (see for example [3, 2, 151, 152]).

In the calculus of Urban and Bierman [147, 148] derivations correspond exactly to cut elimination.

This calculus inspired Lengrand's $\lambda\xi$ in [107] and further led to the development of $\mathcal{X}$ calculus of van Bakel et al. [5], and van Bakel and Lescanne [150]. In this work, a calculus which interprets directly the implicational sequent logic is proposed as a language in which many kinds of other calculi can be implemented, from λ-calculus to $\overline{\lambda}\mu\widetilde{\mu}$ through a calculus of explicit substitution and $\lambda\mu$.

Wadler's dual calculus [151, 152] corresponds to Gentzen's classical sequent calculus. Conjunction, disjunction, and negation are primitive, whereas implication is defined in terms of the other connectives.

One of the most recently proposed systems is λ^{Gtz}-calculus, developed by Espirito Santo [56], whose simply typed version corresponds to the sequent calculus for intuicionistic implicational logic.

Prior to Curien and Herbelin's $\overline{\lambda}\mu\widetilde{\mu}$ [34] several term-assignment systems for sequent calculus were proposed as a tool for studying the process of cut-elimination [126, 12, 147]. In these systems—with the exception of the one in [147]—expressions do not unambiguously encode sequent derivations.

The Symmetric Lambda Calculus of Barbanera and Berardi [6], although not based on sequent calculus, belongs in the tradition of exploiting the symmetries found in classical logic, in their case with the goal of extracting constructive content from classical proofs.

Related work on strong normalisation. Barbanera and Berardi [6] proved SN for their calculus using a "symmetric candidates" technique; Urban and Bierman [147] adapted their technique to prove SN for their sequent-based system. Lengrand [107] shows how simply-typed $\overline{\lambda}\mu\widetilde{\mu}$ and the calculus of Urban and Bierman [147] are mutually interpretable, so that the strong normalisation proof of the latter calculus yields another proof of strong normalisation for simply-typed $\overline{\lambda}\mu\widetilde{\mu}$. Polonovski [125] presents a proof of SN for $\overline{\lambda}\mu\widetilde{\mu}$ with explicit substitutions using the symmetric candidates idea of Barbanera and Berardi [6]. Pym and Ritter [129] identify two forms of disjunction for Parigot's $\lambda\mu$-calculus [121]; they prove strong normalisation for $\lambda\mu\nu$-calculus ($\lambda\mu$-calculus extended with such disjunction). David and Nour [37] give an arithmetical proof of strong normalisation for a symmetric $\lambda\mu$-calculus.

The larger context of related research includes a wealth of work in logic and programming languages. In the 1980's and early 1990's Reynolds explored the role that intersection types can play in a practical programming language (see for example the report [131] on the language Forsythe).

Related work on continuation semantics. Continuation-passing-style (cps) translations were introduced by Fischer and Reynolds in [67] and [130] for the call-by-value λ-calculus, whereas a call-by-name variant was introduced by Plotkin in [124]. Moggi gave a semantic version of a call-by-value cps translation in his study of notions of computation in [117]. Lafont [106] introduced a cps translation of the call-by-name λC-calculus [63, 64] to a fragment of λ-calculus that corresponds to the $\neg, \wedge$-fragment of the intuitionistic logic. Hence, continuation semantics can be seen as a generalization of the double negation rule from logic, in a sense that cps translation is a transformation on terms which, when observed on types, corresponds to a double negation translation.

As early as 1989 Filinsky [65] explored the notion that the reduction strategies call-by-value and call-by-name could be dual to each other in the presence of continuations. Filinski defined a symmetric λ-calculus in which values and continuations comprised distinct syntactic sorts and whose denotational semantics expressed the call-by-name vs call-by-value duality in a precise categorical sense.

Categorical semantics for both, call-by-name and call-by-value versions of Parigot's $\lambda\mu$-calculus [121] with disjunction types was given by Selinger in [137]. In this work the notion of *control category* is formally introduced and formalised as an extension of cartesian closed category with premonoidal structure. It is showed that the call-by-name $\lambda\mu$-calculus forms an internal language for control categories, whereas the call-by-value $\lambda\mu$-calculus forms an internal language for co-control categories. The opposite of the call-by-name model is shown to be equivalent to the call-by-value model in the presence of product and disjunction types. Hofmann and Streicher presented categorical continuation models for the call-by-name $\lambda\mu$-calculus in [100] and showed the completeness.

Lengrand gave categorical semantics of the *typed* $\overline{\lambda}\mu\tilde{\mu}$-calculus and the $\lambda\xi$-calculus (implicational fragment of the classical sequent calculus LK) in [107].

Ong [119] defined a class of categorical models for the call-by-name $\lambda\mu$-calculus based on fibrations. This model was later extended for two forms of disjunction by Pym and Ritter in [129].

References

[1] R. M. Amadio and P.-L. Curien, *Domains and lambda-calculi*, Cambridge University Press, Cambridge, 1998.

[2] Z. M. Ariola, H. Herbelin, and A. Sabry, *A type-theoretic foundation of continuations and prompts*; in: *Proc. 9th Internat. Conf. on Functional Programming ICFP '04*, pp 40–53, 2004.

[3] Z. M. Ariola and H. Herbelin, *Minimal classical logic and control operators*; in: *Proc. Annual Internat. Colloquium on Automata, Languages and Programming ICALP '03*, Lect. Notes Comput. Sci. 2719, Springer-Verlag, 2003, pp. 871–885.

[4] Z. M. Ariola, H. Herbelin, and A. Sabry, *A type-theoretic foundation of delimited continuations*, High.-Order Symb. Comput. 2007, to appear.

[5] S. v. Bakel, S. Lengrand, and P. Lescanne, *The language $\mathcal{X}$: circuits, computations and classical logic*; in: *Proc. 9th Italian Conf. on Theoretical Computer Science ICTCS '05*, Lect. Notes Comput. Sci. 3701, Springer-Verlag, 2005, pp. 81–96.

[6] F. Barbanera and S. Berardi, *A symmetric lambda calculus for classical program extraction*, Inf. Comput. 125(2):103–117, 1996.

[7] F. Barbanera, M. Dezani-Ciancaglini, and U. de' Liguoro, *Intersection and union types: syntax and semantics*, Inf. Comput. 119(2):202–230, 1995.

[8] F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro, *Intersection and union types: Syntax and semantics*, Inf. Comput. 119(2):202–230, 1995.

[9] H. P. Barendregt, *Lambda calculi with types*; in: S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, pp. 117–309 Oxford University Press, Oxford, 1992.

[10] H. P. Barendregt, *The Lambda Calculus: its Syntax and Semantics*, revised edition, North-Holland, Amsterdam, 1984.

[11] H. P. Barendregt, M. Coppo, and M. Dezani-Ciancaglini, *A filter lambda model and the completeness of type assignment*, J. Symb. Log. 48(4):931–940 (1984), 1983.

[12] H. P. Barendregt and S. Ghilezan, *Lambda terms for natural deduction, sequent calculus and cut-elimination*, J. Funct. Program. 10(1):121–134, 2000.

[13] L. Bettini, V. Bono, and S. Likavec, *A core calculus of higher-order mixins and classes*; in: *Proc. Workshop Types for Proofs and Programs TYPES '03 (Selected Papers)*, Lect. Notes Comput. Sci. 3085, pp. 83–98, Springer-Verlag, 2004.

[14] L. Bettini, V. Bono, and S. Likavec, *A Core Calculus of Higher-Order Mixins and Classes*; in: *Proc. 19th Annual ACM Symposium on Applied Computing, SAC '04*, pp. 1508–1509 ACM Press, 2004.

[15] L. Bettini, V. Bono, and S. Likavec, *A core calculus of mixin-based incomplete objects*; in: *Proc. 11th International Workshop on Foundations of Object-Oriented Languages, FOOL '04*, pp. 29–41, 2004.

[16] L. Bettini, V. Bono, and S. Likavec, *A core calculus of mixins and incomplete objects*; in: *Proc. Conf. on Object-Oriented Programming Systems, Languages, and Applications OOPSLA '04*, pp. 208–209, ACM Press, 2004.

[17] L. Bettini, V. Bono, and S. Likavec, *Safe and Flexible Objects*; in: *Proc. 20th Annual ACM Symposium on Applied Computing SAC '05*, pp. 1258–1263, ACM Press, 2005.

[18] L. Bettini, V. Bono, and S. Likavec, *Safe and Flexible Objects with Subtyping*, J. Object Technology 4(10), 2005, Special Issue on "The 20th ACM SAC - March 2005".

[19] G. M. Bierman, *A computational interpretation of the $\lambda\mu$-calculus*; in: *Proc. Symp. on Mathematical Foundations of Computer Science MFCS '98*, Lect. Notes Comput. Sci. 1450, pp. 336–345 Springer-Verlag, 1998.

[20] C. Böhm, *Alcune proprieta delle forme $\beta - \eta$-normali nel $\lambda - k$-calcolo*, Publ. Inst. Appl. Calc. 696:1–19, 1968.

[21] C. Böhm and M. Dezani-Ciancaglini, *λ-terms as total or partial functions on normal forms*, in: *λ-calculus and Computer Science Theory*, Lect. Notes Comput. Sci. 37, pp. 96–121, Springer-Verlag 1975.

[22] V. Bono, A. Patel, and V. Shmatikov, *A core calculus of classes and mixins*, in: *Proc Europ. Conf. on Object-Oriented Programming ECOOP '99*, Lect. Notes Comput. Sci. 1628, pp. 43–66. Springer-Verlag, 1999.

[23] V. Bono, B. Venneri, and L. Bettini, *A typed lambda calculus with intersection types*, Theor. Comput. Sci. 398(1-3):95–113, 2008.

[24] G. Bracha and W. Cook, *Mixin-based inheritance*, in: *Proc. Conf. on Object-Oriented Programming Systems, Languages, and Applications OOPSLA '90*, pp. 303–311, 1990.

[25] C.-c. Shan, *Shift to control*, in: *Proc. 5th Workshop on Scheme and Functional Programming*, pp. 99–107, 2004.

[26] S. Carlier and J. B. Wells, *Type inference with expansion variables and intersection types in system E and an exact correspondence with beta-reduction*, in: *Proc. 6th Conf. on Principles and Practice of Declarative Programming PPDP '04*, pp. 132–143, ACM, 2004.

[27] A. Church, *A set of postulates for the foundation of logic*, Ann. Math. II.33:346–366, 1932.

[28] A. Church, *A formulation of the simple theory of types*, J. Symb. Log. 5:56–68, 1940.

[29] M. Coppo and M. Dezani-Ciancaglini, *A new type-assignment for lambda terms*, Archiv Math. Logik 19:139–156, 1978.

[30] M. Coppo and M. Dezani-Ciancaglini, *An extension of the basic functionality theory for the λ-calculus*, Notre Dame J. Formal Logic 21(4):685–693, 1980.

[31] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri, *Principal type schemes and λ-calculus semantics*; in: J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 535–560, Academic Press, London, 1980.

[32] P.-L. Curien, *Abstract machines, control, and sequents*, in: *Applied Semantics, International Summer School, APPSEM 2000, Advanced Lectures*, Lect. Notes Comput. Sci. 2395, pp. 123–136, Springer-Verlag, 2002.

[33] P.-L. Curien, *Symmetry and interactivity in programming*, Bull. Symb. Log. 9(2):169–180, 2003.

[34] P.-L. Curien and H. Herbelin, *The duality of computation*; in: *Proc. 5th Internat. Conf. on Functional Programming, ICFP'00*, pp. 233–243, Montreal, Canada, 2000; ACM Press.

[35] H. B. Curry, J. R. Hindley, and J. P. Seldin, *Combinatory Logic*, volume II, North-Holland, Amsterdam, 1972.

[36] O. Danvy and A. Filinski, *A functional abstraction of typed contexts*, Technical Report 89/12, DIKU, University of Copenhagen, Copenhagen, Denmark, Aug. 1989.

[37] R. David and K. Nour, *Arithmetical proofs of strong normalization results for the symmetric $\lambda\mu$-calculus*; in: *Proc. Typed Lambda Calculus and Application, TLCA '05*, Lect. Notes Comput. Sci. 3461, pp. 162–178, Springer-Verlag, 2005.

[38] R. David and W. Py, *Lambda-mu-calculus and Böhm's theorem*, J. Symb. Log. 66(1):407–413, 2001.

[39] P. de Groote, *A CPS-translation of the $\lambda\mu$-calculus*, in: *Proc. Colloquium on Trees in Algebra and Programming, CAAP '94*, Lect. Notes Comput. Sci. 787, pp. 85–99, Springer-Verlag, 1994.

[40] P. de Groote, *On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control*; in: *Proc. Internat. Conf. on Logic Programming and Automated Reasoning, LPAR '94*, Lect. Notes Comput. Sci. 822, pp. 31–43, Springer-Verlag, 1994.

[41] M. Dezani-Ciancaglini and S. Ghilezan, *A lambda model characterizing computational behaviours of terms*, in: *Proc. Internat. Workshop on Rewriting in Proof and Computation RPC '01*, pp. 100–119, 2001.

[42] M. Dezani-Ciancaglini and S. Ghilezan, *A behavioural lambda model*, Schedae Informaticae Universitas Iagelonica, 12:35–47, 2003.

[43] M. Dezani-Ciancaglini and S. Ghilezan, *Two behavioural lambda models*; in: *Proc. Workshop Types for Proofs and Programs TYPES '02*, Lect. Notes Comput. Sci. 2646, pp. 127–147, Springer-Verlag, 2003.

[44] M. Dezani-Ciancaglini, S. Ghilezan, and S. Likavec, *Behavioural inverse limit models*, Theor. Comput. Sci. 316(1–3):49–74, 2004.

[45] M. Dezani-Ciancaglini, S. Ghilezan, and B. Venneri, *The "relevance" of intersection and union types*, Notre Dame J. Formal Logic, 38(2):246–269, 1997.

[46] M. Dezani-Ciancaglini and E. Giovannetti, *From Böhm teorem to observational equivalence: an informal account*; in: *Proc. Böhm theorem: applications to Computer Science Theory – BOTH '01*, Electr. Notes Theor. Comput. Sci. 50(2), 2001.

[47] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama, *Compositional characterization of λ-terms using intersection types*; in: *Proc. Mathematical Foundations of Computer Science MFCS '00*, Lect. Notes Comput. Sci. 1893, pp. 304–314, Springer-Verlag, 2000.

[48] D. Dougherty, S. Ghilezan, and P. Lescanne, *Characterizing strong normalization in a language with control operators*; in: *Proc. 6th Conf. on Principles and Practice of Declarative Programming PPDP '04*, pp. 155–166, ACM Press, 2004.

[49] D. Dougherty, S. Ghilezan, and P. Lescanne, *Intersection and union types in the $\bar{\lambda}\mu\tilde{\mu}$-calculus*; in: *Proc. Workshop on Intersection Types and Related Systems ITRS '04*, Electr. Notes Theor. Comput. Sci. 136:153–172, 2005.

[50] D. Dougherty, S. Ghilezan, and P. Lescanne, *A general technique for analyzing termination in symmetric proof calculi*; in: *Proc. 9th Internat. Workshop on Termination WST '07*, 2007.

[51] D. Dougherty, S. Ghilezan, and P. Lescanne, *Characterizing strong normalization in the Curien-Herbelin symmetric lambda calculus: extending the Coppo-Dezani heritage*; Theor. Comput. Sci. 398:114–128, 2008.

[52] D. Dougherty, S. Ghilezan, P. Lescanne, and S. Likavec, *Strong normalization of the dual classical sequent calculus*; in: *Proc. 12th Internat. Conf. on Logic for Programming, Artif. Intell., and Reasoning LPAR '05*, Lect. Notes Comput. Sci. 3835, pp. 169–183 Springer-Verlag, 2005.

[53] K. Došen and Z. Petrić, *The typed Böhm teorem*; in: *Proc. Böhm theorem: applications to Computer Science Theory - BOTH '01*, Electr. Notes Theor. Comput. Sci. 50(2), p.13, 2001.

[54] J. Dunfield and F. Pfenning, *Type assignment for intersections and unions in call-by-value languages*; in: *Proc. 6th Internat. Conf. on Foundations of Software Science and Computation Structures FOSSACS '03*, Lect. Notes Comput. Sci. 2620, pp. 250–266, Springer-Verlag, 2003.

[55] R. Dyckhoff and L. Pinto, *Cut-elimination and a permutation-free sequent calculus for intuitionistic logic*, Studia Logica 60(1):107–118, 1998.

[56] J. Espírito Santo, *Completing Herbelin's programme*; in: *Proc. Conf. on Typed Lambda Calculus and Applications TLCA '07*, Lect. Notes Comput. Sci. 4583, pp. 118–132, Springer-Verlag, 2007.

[57] J. Espírito Santo, S. Ghilezan, and J. Ivetić, *Characterising strongly normalising intuitionistic sequent terms*, in: *Proc. Workshop Types for Proofs and Programs TYPES '07 (Selected Papers)*, Lect. Notes Comput. Sci. 4941, pp. 85-99 Springer-Verlag, 2007.

[58] J. Espírito Santo, J. Ivetić, and S. Likavec, *Intersection type assignment systems for intuitionistic sequent calculus*, in: *Workshop on Intersection Types and Related Systems ITRS'08*, 2008.

[59] J. Espírito Santo, J. Ivetić, and S. Likavec, *Characterising strongly normalising intuitionistic terms*, submitted to Fundamenta Informaitcae.

[60] J. Espírito Santo and L. Pinto, *Permutative conversions in intuitionistic multiary sequent calculi with cuts*; in: *Proc. Conf. on Typed Lambda Calculus and Applications TLCA '03*, Lect. Notes Comput. Sci.2071, pp. 286–300, , Springer-Verlag, 2003.

[61] M. Felleisen, *The theory and practice of first-class prompts*, in: *Proc. 15th ACM Symp. on Principles of Programming Languages POPL '88*, pp. 180–190. ACM, 1988.

[62] M. Felleisen, D. P. Friedman, E. Kohlbecker, and B. F. Duba, *Reasoning with continuations*, in: *Proc. 1st Symposium on Logic in Computer Science LICS '86*, pp. 131–141, 1986.

[63] M. Felleisen, D. P. Friedman, E. Kohlbecker, and B. F. Duba, *A syntactic theory of sequential control*, Theor. Comput. Sci. 52(3):205–237, 1987.

[64] M. Felleisen and R. Hieb, *The revised report on the syntactic theories of sequential control and state*, Theor. Comput. Sci., 103(2):235–271, 1992,

[65] A. Filinski, *Declarative continuations and categorical duality*, Master's thesis, DIKU, Computer Science Department, University of Copenhagen, 1989, DIKU Rapport 89/11.

[66] A. Filinski, *Representing monads*, in: *Proc. 21st ACM Symp. on Principles of Programming Languages, POPL'94*, pp. 446–457 ACM, 1994.

[67] M. Fischer, *Lambda calculus schemata*, in: *Proc. ACM Conf. on Proving Assertions About Programs '72*, pp. 104–109 ACM Press, 1972.

[68] G. Frege, *Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought*, Halle, 1879; reprinted in Jan van Heijenoort, editor, *From Frege to Gödel, A Sourcebook in Mathematical Logic*, 1879–1931, Harvard University Press, 1967.

[69] J. H. Gallier, *Typing untyped λ-terms, or reducibility strikes again!*, Ann. Pure Appl. Logic 91:231–270, 1998.

[70] J. H. Gallier, *Constructive logics part i: A tutorial on proof systems and typed lambda-calculi*, Theor. Comput. Sci. 110(2):249–339, 1993.

[71] G. Gentzen, *Unterschungen über das logische Schliessen*, Math Z. 39 (1935), 176–210; in: M. Szabo, editor, *Collected papers of Gerhard Gentzen*, pp. 68–131, North-Holland, 1969.

[72] S. Ghilezan, *Inhabitation in intersection and union type assignment systems*, J. Log. Comput. 3(6):671–685, 1993.

[73] S. Ghilezan, *Application of typed lambda calculi in the untyped lambda calculus*, in: *Proc. Logical Foundations of Computer Science LFCS '04*, Lect. Notes Comput. Sci. 813, pp. 129–139, 1994.

[74] S. Ghilezan, *Generalized finitness of developments in typed lambda calculi*, J. Autom. Lang. Comb., 1(4):247–258, 1996.

[75] S. Ghilezan, *Strong normalization and typability with intersection types*, Notre Dame J. Formal Logic 37(1):44–52, 1996.

[76] S. Ghilezan, *Cut elimination in the simply typed lambda calculus*, in: *Proc. 1st Panhellenic Logic Symp. PLS '97*, pp. 21–24, 1997.

[77] S. Ghilezan, *Natural deduction and sequent typed lambda calculus*, Novi Sad J. Math. 29:209–220, 1999.

[78] S. Ghilezan, *Topologies in lambda calculus*, in: *Proc. 2nd Panhellenic Logic Symp. PLS '99*, pp. 102–106, 1999.

[79] S. Ghilezan, *Intersection types and topologies and lambda calculus*, in: *ICALP Satellite Workshops*, pp. 303–304, 2000.

[80] S. Ghilezan, *Full intersection types and topologies in lambda calculus*, J. Comput. Sys. Sci. 62(1):1–14, 2001.

[81] S. Ghilezan, *Types and confluence in lambda calculus*, in: *Proc. 3rd Panhellenic Logic Symp. PLS '01*, 2001.

[82] S. Ghilezan, *Terms for natural deduction, sequent calculus and cut elimination in classical logic*; in: *Reflections on Type Theory, Lambda Calculus, and the Mind – Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday*, 2007.

[83] S. Ghilezan and J. Ivetić, *Intersection types for λ^{gtz} calculus*, Publ. Inst. Math., Nouv. Sér. 82(96):85–91, 2007.

[84] S. Ghilezan and V. Kunčak, *Confluence of untyped lambda calculus via simple types*, in: *Proc. Italian Conf. on Theoretical Computer Science ICTCS '01*, Lect. Notes Comput. Sci. 2202, pp. 38–49, , Springer-Verlag, 2001.

[85] S. Ghilezan and V. Kunčak, *Reducibility method in simply typed lambda calculus*, Novi Sad J. Math. 31:27–32, 2001.

[86] S. Ghilezan, V. Kunčak, and S. Likavec, *Reducibility method for termination properties of typed lambda terms*, in: *Proc. 5th Internat. Workshop on Termination WST '01*, pp. 14–16, 2001.

[87] S. Ghilezan and P. Lescanne, *Classical proofs, typed processes and intersection types*, in: *Proc. Workshop Types for Proofs and Programs TYPES '03 (Selected Papers)*, Lect. Notes Comput. Sci. 3085, pp. 226–241, Springer-Verlag, 2004.

[88] S. Ghilezan and S. Likavec, *Reducibility: A Ubiquitous Method in Lambda Calculus with Intersection Types*, in: *Proc. Workshop on Intersection Types and Related Systems ITRS '02*, Electr. Notes Theor. Comput. Sci. 70, 2003.

[89] S. Ghilezan and S. Likavec, *Extensions of the reducibility method*, in: *Proc. 4th Panhellenic Logic Symp. PLS 04*, pp. 107–112, 2004.

[90] S. Ghilezan, J. Pantovic, and J. Žunic, *Separating Points by Parallel Hyperplanes – Characterization Problem*, *IEEE Transactions of Neural Networks* 18 (5), pp. 1356-1363, 2007.

[91] S. Ghilezan, J. Pantovic, and J. Žunic, *Partitioning Finite d-Dimensional Integer Grids with Application*, in T. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristic*, pp. 55-1–55-15, Taylor and Fransis Group, USA, 2005.

[92] J.-Y. Girard, *Une extension de l'interprétation de Gödel à l'analyse, et son application à l'elimination des coupures dans l'analyse et la théorie des types*, in: *Proc. 2nd Scandinav. Logic Symp.*, pp. 63–92. North-Holland, Amsterdam, 1971.

[93] J.-Y. Girard, *A new constrcutive logic: classical logic*, Math. Struct. Comput. Sci. 1(3):255–296, 1991.

[94] T. Griffin, *A formulae-as-types notion of control*, in: *Proc. 19th Annual ACM Symp. on Principles Of Programming Languages, POPL '90*, pp. 47–58, ACM Press, 1990.

[95] H. Herbelin, Private communication.

[96] H. Herbelin, *A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure*, in: *Proc. Conf. on Computer Science Logic, CSL '94*, Lect. Notes Comput. Sci. 933, pp. 61–75, Springer-Verlag, 1995.

[97] H. Herbelin, *Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*, Thèse, Université Paris 7 1995.

[98] H. Herbelin and S. Ghilezan, *An approach to call-by-name delimited continuations*, in: *Proc. 35th Annual ACM Symp. on Principles of Programming Languages POPL '08*, pp. 383–394, SIGPLAN Notices 43, ACM Press, 2008.

[99] J. R. Hindley, *Coppo–Dezani types do not correspond to propositional logic*, Theor. Comput. Sci. 28(1-2):235–236, 1984.

[100] M. Hofmann and T. Streicher, *Completeness of continuation models for λμ-calculus*, Inf. Comput. 179(2):332–355, 2002.

[101] W. A. Howard, *The formulas-as-types notion of construction*, in: J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 479–490, Academic Press, 1980.

[102] F. Joachimski and R. Matthes, *Standardization and confluence for ΛJ*, in: *Proc. Rewriting Techniques and Applications RTA '00*, Lect. Notes Comput. Sci. 1833, pp. 141–155, Springer-Verlag, 2000.

[103] A. J. Kfoury and J. B. Wells, *Principality and type inference for intersection types using expansion variables*, Theor. Comput. Sci. 311(1-3):1–70, 2004.

[104] G. Koletsos, *Church-Rosser theorem for typed functionals*, J. Symb. Log. 50:782–790, 1985.

[105] J.-L. Krivine, *Lambda-calcul types et modèles*, Masson, Paris, 1990.

[106] Y. Lafont, *Negation versus implication*, Draft, 1991.

[107] S. Lengrand, *Call-by-value, call-by-name, and strong normalization for the classical sequent calculus*, Electr. Notes Theor. Comput. Sci. 86, Elsevier, 2003.

[108] S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel, *Intersection types for explicit substitutions*, Inf. Comput. 189(1):17–42, 2004.

[109] S. Likavec, *Reducibility method for λ calculus with intersection types*, Master's thesis, University of Novi Sad, Serbia, 2005.

[110] S. Likavec, *Types for object oriented and functional programming languages*, PhD thesis, Università di Torino, Italy and ENS Lyon, France, 2005.

[111] S. Likavec and P. Lescanne, *On untyped Curien-Herbelin calculus*, in: *Proc. 1st Workshop on Classical Logic and Computation CLaC '06*, 2006.

[112] L. Liquori and S. Ronchi Della Rocca, *Intersection-types à la church*, Inf. Comput. 205(9):1371–1386, 2007.

[113] R. Matthes, *Characterizing strongly normalizing terms of a calculus with generalized applications via intersection types*, in: *ICALP Satellite Workshops*, pp. 339–354, 2000.

[114] G. Mints, *Normal forms for sequent derivations*, in: P. Odifreddi, editor, *Kreiseliana. About and Around Georg Kreisel*, pp. 469–492. A. K. Peters, Wellesley, 1996.

[115] J. C. Mitchell, *Type systems for programming languages*, in: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Volume B, pp. 415–431, Elsevier, Amsterdam, 1990.

[116] J. C. Mitchell, *Foundation for Programmimg Languages*, MIT Press, Boston, 1996.

[117] E. Moggi, *Notions of computations and monads*, Inf. Comput. 93(1), 1991.
[118] C. R. Murthy, *Classical proofs as programs: How, what, and why*, in: *Constructivity in Computer Science*, Lect. Notes Comput. Sci. 613, pp. 71–88, Springer-Verlag, 1991.
[119] C.-H. L. Ong, *A semantic view of classical proofs: type-theoretic, categorical, denotational characterizations*, in: *Proc. 11th IEEE Annual Symp. on Logic in Computer Science LICS '97*, pp. 230–241. IEEE Computer Society Press, 1997.
[120] C.-H. L. Ong and C. A. Stewart, *A Curry–Howard foundation for functional computation with control*, in: *Proc. 24th ACM Symp. on Principles of Programming Languages POPL '97*, pp. 215–227, 1997.
[121] M. Parigot, *An algorithmic interpretation of classical natural deduction*, in: *Proc. Internat. Conf. on Logic Programming and Automated Reasoning, LPAR '92*, Lect. Notes Comput. Sci. 624, pp. 190–201, Springer-Verlag, 1992.
[122] M. Parigot, *Proofs of strong normalisation for second order classical natural deduction*, J. Symb. Log. 62(4):1461–1479, 1997.
[123] B. C. Pierce, *Programming with intersection types, union types, and polymorphism*, Technical Report CMU-CS-91-106, Carnegie Mellon University, Feb, 1991.
[124] G. D. Plotkin, *Call-by-name, call-by-value and the λ-calculus*, Theor. Comput. Sci. 1:125–159, 1975.
[125] E. Polonovski, *Strong normalization of λμμ̃-calculus with explicit substitutions*, in: *Proc. 7th Internat. Conf. on Foundations of Software Science and Computation Structures, FOSSACS '04*, Lect. Notes Comput. Sci. 2987, pp. 423–437, Springer-Verlag, 2004.
[126] G. Pottinger, *Normalization as homomorphic image of cut-elimination*, Ann. Math. Log. 12:323–357, 1977.
[127] G. Pottinger, *A type assignment for the strongly normalizable λ-terms*, in: J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 561–577, Academic Press, London, 1980.
[128] D. Prawitz, *Natural Deduction*, Almqvist and Wiksell, 1965.
[129] D. Pym and E. Ritter, *On the semantics of classical disjunction*, J. Pure Appl. Algebra 159:315–338, 2001.
[130] J. C. Reynolds, *Definitional interpreters for higher-order programming languages*, in: *Proc. ACM Annual Conf.*, pp. 717–740, ACM Press, 1972.
[131] J. C. Reynolds, *Design of the programming language Forsythe*, Report CMU-CS-96-146, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.
[132] K. Rose, *Explicit substitutions: Tutorial and survey*, Technical Report LS-96-3, BRICS, 1996.
[133] A. Sabry and M. Felleisen, *Reasoning about programs in continuation-passing style*, Lisp and Symbolic Computation 6(3-4):289–360, 1993.
[134] P. Sallé, *Une extension de la théorie des types en lambda-calcul*, in: *Proc. 5th Internat. Conf. on Automata, Languages and Programming ICALP '79*, Lect. Notes Comput. Sci. 62, pp. 398–410, Springer-Verlag, 1978.
[135] A. Saurin, *Separation with streams in the λμ-calculus*, in: *Proc. 20th Annual IEEE Symp. on Logic in Computer Science LICS '05*, pp. 356–365 IEEE Computer Society Press, 2005.
[136] D. S. Scott, *Continuous lattices*, in: *Toposes, Algebraic Geometry and Logic*, Lect. Notes Math. 274, pp. 97–136, Springer-Verlag, 1972.
[137] P. Selinger, *Control categories and duality: On the categorical semantics of the lambda-mu calculus*, Math. Struct. Comput. Sci. 11(2):207–260, 2001.
[138] A. K. Simpson, *Categorical completeness results for simply typed lambda calculus*, in: *Proc. Conf. on Typed Lambda Calculus and Applications TLCA '95*, Lect. Notes Comput. Sci. 902, pp. 414–427, Springer-Verlag, 1995.
[139] M. H. Sørensen and P. Urzyczyn, *Lectures on the Curry–Howard isomorphism*, Studies in Logic and the Foundations of Mathematics, 149. Elsevier, 2006.
[140] R. Statman, *Completeness, invariance and λ-definability*, J. Symb. Log. 47(1):17–26, 1982.
[141] R. Statman, *Logical relations and the typed λ-calculus*, Inf. Control 65:85–97, 1985.

[142] T. Streicher and B. Reus, *Classical logic, continutation semantics and abstract machines*, J. Funct. Program. 8(6):543–572, 1998.

[143] G. J. Sussman and G. L. S. Jr, *Scheme: A interpreter for extended lambda calculus*, High.-Order Symb. Comput. 11(4):405–439, 1998.

[144] W. W. Tait, *Intensional interpretations of functionals of finite type I*, J. Symb. Log. 32:198–212, 1967.

[145] W. W. Tait, *A realizability interpretation of the theory of species*, in: *Logic Colloquium*, Lect. Notes Math. 453, pp. 240–251, Springer-Verlag, 1975.

[146] M. Takahashi, *Parallel reduction in λ-calculus*, Inf. Comput. 118:120–127, 1995.

[147] C. Urban and G. M. Bierman, *Strong normalisation of cut-elimination in classical logic*, in: *Proc. Conf. on Typed Lambda Calculus and Applications, TLCA '99*, Lect. Notes Comput. Sci. 1581, pp. 365–380 Springer-Verlag, 1999.

[148] C. Urban and G. M. Bierman, *Strong normalisation of cut-elimination in classical logic*, Fund. Inf. 45(1-2):123–155, 2001.

[149] S. van Bakel, *Intersection type assignment systems*, Theor. Comput. Sci. 38(2):246–269, 1997.

[150] S. van Bakel and P. Lescanne, *Computation with classical sequents*, Math. Struct. Comput. Sci. 18(3):555–609, 2008.

[151] P. Wadler, *Call-by-value is dual to call-by-name*, in: *Proc. 8th Internat. Conf. on Functional Programming ICFP '03*, pp. 189–201, 2003.

[152] P. Wadler, *Call-by-value is dual to call-by-name, reloaded*; in: *Proc. Conf. on Rewriting Technics and Applications RTA '05*, Lect. Notes Comput. Sci. 3467, pp. 185–203, 2005.