UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Cvetana Krstev

Jedan prilaz informatičkom modeliranju teksta i algoritmi njegove transformacije

Doktorska disertacija

BEOGRAD
jun 1997.

Zahvalnost

Svoju najtopliju zahvalnost želim da izrazim svome mentoru, profesoru Gordani Pavlović-Lažetić čije dugogodišnje brižljivo praćenje mojih istraživanja, duboko razumevanje problema sa kojima sam se susretala i plodotvorni saveti u njihovom rešavanju su suštinski ugrađeni u sadržinu ovoga teksta.

Iskrenu zahvalnost dugujem i svom prvom učitelju programerske veštine, profesoru Nedeljku Parezanoviću, koji me je još ranih osamdesetih usmerio ka području obrade teksta, slutećи značaj koje će ovo područje dobiti toliko godina kasnije. Profesor Marica Prešić me je i često i mudro bodrila da istrajem u ovome poduhvatu: njena podrška mi je bila stalan i važan oslonac tokom rada. Profesor Živojin Stanojčić, koji se prihvatio zadatka da iz jedne informatičke teze raščita matematičke formalizacije morfoloških fenomena u srpskom jeziku, je svojim dragocenim primedbama doprineo da ovaj tekst bude precizniji i bogatiji: izražavam mu svoju najdublju zahvalnost.

Tokom istraživanja, brojne kolege iz zemlje i inostranstva predavale su mi deo svojih znanja od kojih su neka bitno uticala na formiranje mojih pogleda o informatičkim aspektima obrade prirodnih jezika. Posebnu zahvalnost za ovakve uticaje dugujem profesoru Morisu Grosu i profesoru Ljubomiru Popoviću. Urednik Nolit-a Slobodan Đorđević nije dozvolio da odustanem tokom višegodišnjeg zajedničkog rada na novom izdanju Vukovih narodnih poslovica, te mu pripada deo moje zahvalnosti. Moje mlađe kolege, Vlado Kešelj, Goran Nenadić, Nebojša Vasiljević i Goran Lazić, su doprinele da tokom rada na tezi uspešno razrešim brojne tehničke teškoće na koje nužno nailaze ovakvi poduhvati: neka njihov udeo u mome radu ne ostane nezabeležen.

Meda i Duško, svaki prema svojim mogućnostima, pomagali su mi da svoja istraživanja privedem kraju. Svojoj porodici izražavam osobitu zahvalnost za pažnju i razumevanje koje je ispoljavalja tokom mog dugogodišnjeg rada na disertaciji.

U Beogradu,
jun 1997.

Cvetana Krstev

Sadržaj

Glava 1

Tekst kao predmet informatičke obrade

Tekst zabeležen u digitalnom obliku je postao deo svakednevne komunikacije pisanom rečju. Prirodno je stoga da se programerska pažnja usmerava na metode formalizovanja i obrade teksta prepoznajući u njemu značajan objekt informatičke obrade. S druge strane, tekstovi u digitalnom obliku, ili elektronski tekstovi, su značajan izvor jezičke građe koja bi potencijalno mogla biti premet lingvističkog, filološkog ili leksikografskog istraživanja.

Jedan isti elektronski tekst može biti predmet različitih obrada, kao što su:

- uređivanje i slaganje radi štampe (elektronski grafičar);
- otkrivanje i korigovanje tipografskih ili gramatičkih „grešaka" (elektronski korektor i lektor);
- unošenje u automatizovani korpus nekog jezika za potrebe lingvističkih ili filoloških istraživanja;
- deponovanje u dokumentalistički arhiv (elektronski dokumentalista);
- automatsko prevođenje s jednog na drugi prirodni jezik (elektronski prevodilac);
- prenošenje kroz računarske mreže (elektronski poštar).

Rasprostranjenost ličnih računara i nagli rast računarskih mreža značajno je doprineo da se većina tekstova u nekom trenutku svog životnog veka nađe u digitalnom obliku. Procene iz 1993. godine su govorile da se preko 90% tekstova u visokorazvijenim sredinama proizvodi posredstvom računara. Pri tome, mnogi od njih se nikad ne otisnu na papiru (na primer, elektronska pošta, *pomoć* (engl. *help*) koja prati većinu programskih paketa, i slično). Ove činjenice zahtevaju razvoj posebnih informatičkih alata za razumevanje teksta.

Pre svega treba pomenuti programske jezike koji su namenjeni nenumeričkim obradama. Rodonačelnik ovakvih jezika je SNOBOL koji je nastao još ranih šezdesetih godina [29], [91] a za čijim su primerom razvijeni i drugi jezici kao što su Awk [8], Perl [149], Icon [43] i drugi. Njihovu zajedničku crtu predstavlja rad sa *niskom* (engl. *string*) kao primerenim objektom za raznovrsne obrade teksta. Za niske se definišu operatori, pre svega, operator *dopisivanja* (*konkatenacije*) i operacije, kakva je operacija *sravnjivanja niskovnih obrazaca* (engl. *string pattern matching*). Sravnjivanje niskovnih obrazaca ugrađeno u ove jezike je različite snage. Tako Awk sravnjuje obrasce zadate regularnim izrazima, dok SNOBOL4 sravnjuje i neke obrasce koji pripadaju klasi kontekstno zavisnih jezika [74]. Posebnu crtu ovih jezika čine

asocijativni nizovi kod kojih proizvoljna niska može biti i vrednost i supskript elementa niza. Pod uticajem potrebe za nenumeričkim obradama uključen je i u jezike opšte namene objekt *niska* kao tip podataka, kao i operatori i funkcije za rad sa njima. Ilustrativan je primer je programskog jezika Fortran koji je prvobitno bio namenjen numeričkim obradama ali u čije je kasnije verzije, Fortran77, a posebno Fortran90 [99], ne samo ugrađen niskovni tip podataka već i brojne odgovarajuće funkcije.

Postoje i drugi informatički alati koji su specijalizovani za potrebe obrade teksta. Na primer, *uređivači* (engl. *editor*), namenjeni pripremi tekstualnih datoteka, razlikuju se po repertoaru i realizaciji funkcija od uređivača opšte namene. Osnovne funkcije ovih uređivača su automatsko deljenje redova, poravnavanje margina i deljenje stranica. Iz iskustva uređivača tekstualnih datoteka razvijeni su *procesori reči* (engl. *word processor*), *formateri* (engl. *formatter*) i sistemi za *stono izdavaštvo* (engl. *desk-top publishing*) koji uključuju obilje olakšica u pripremi teksta za prikazivanje na klasičnom (papirnom) medijumu ili na ekranu računara. Posebnu klasu programskih paketa čine prelistači (engl. *browser*) i gledači (engl. *viewer*) koji se koriste na računarskim mrežama i koji dopuštaju vizuelizovanje i pregledanje tekstova pripremljenih u raznovrsnom hradverskom i softverskom okruženju i napisanih na raznim prirodnim jezicima.

Baze podataka su nastale iz potrebe za čuvanjem i rukovanjem velikom količinom, prvenstveno, numeričkih podataka. Razvijeni su različiti modeli baza podataka od kojih je relacioni model u najširoj upotrebi. Ovaj model se zasniva na konceptima koji nisu prilagođeni obradi teksta čija je osnovna crta da predstavlja potencijalno neograničenu sekvenciju karaktera. Koncept relacionog modela, kao i upitni jezik SQL, je na različite načine proširivan da bi obuhvatio tekstualni tip podatka [101]. Relacioni model podataka u sistemu ORACLE [98] proširen je novim tipovima podataka: polja mogu sadržati liste ključnih reči i tekst dužine do 64kB a standardni upitni jezik SQL je nadgrađen na takav način da obuhvati mogućnosti specijalizovanih sistema za pronalaženje informacija. Jedan drugi model tekstualne baze podataka, definisane gramatikom, prikazan je u glavi 3.

Sistemi za pretraživanje informacija su u tesnoj vezi sa bazama podataka jer omogućavaju efikasno pronalaženje u njima uskladištenih informacija. Posebnu klasu ovih sistema čine *sistemi za pretraživanje teksta* (engl. *text retrieval systems*) koje karakteriše pretraživanje punog (kontinuiranog, nesegmentiranog) teksta. Klasični sistemi za pretraživanje teksta (na primer, Mistral i Stairs), imaju najčešće statičku organizaciju u obliku invertovanih datoteka a njihov jezik za pretraživanje koristi ključne reči i Bulovske operatore. Dodatne olakšice u pronalaženju informacija pružaju sredstva za poboljšanje dokumentalističke pretrage, kao što su na primer tezaurusi koji među ključnim rečima definišu veze sinonimije, podređenosti, nadređenosti i asocijativnosti.

Posebnost teksta, kao objekta informatičke obrade ogleda se u činjenici da je on organizovan određenim prirodnim jezikom. Bilo bi otuda prirodno da informatički alati koji se koriste za njegovu obradu zavise od prirodnog jezika na kome je tekst napisan. Svi gore navedeni alati, razvijeni prevashodno za potrebe obrade teksta, premda mogu obaviti nesumnjivo korisne funkcije, veoma malo ili nikako ne zavise od prirodnog jezika na kome je tekst zapisan. Na primer, uređivači teksta i formateri se najčešće prilagođavaju prirodnom jeziku na kome je tekst napisan samo kroz izbor azbuke i odgovarajućih slovnih garnitura (fontova), zatim preko procedura za rastavljanje reči na kraju retka ili procedura za otkrivanje tipografskih grešaka u tekstu. Ove procedure zasnivaju se na obradi teksta do nivoa formalne reči. Procedure za rastavljanje reči na kraju retka se zasnivaju bilo na listama reči sa unapred obeleženim mestima na kojima se reč može rastaviti ili na listama obrazaca (koji

predstavljaju podniske „reči“) [87], [80]. Sličan je slučaj i sa procedurama za otkrivanje tipografskih grešaka koje se zasnivaju ili na listama trigrama bilo na listama „ispravnih“ reči, koje faktički predstavljaju uopštenje sa trigramske na n-gramsku nisku a stvaraju utisak o prisustvu rečnika i gramatičkih pravila [100]. Koliko malo znanja o jeziku je ugrađeno u uređivače i formatere teksta pokazuje primer funkcije promene malih u velika slova (i obrnuto). Ta funkcija u nekim uređivačima ne utiče na ona slova izabranog alfabeta koja se u kodnoj sekvenciji ne poklapaju sa pozicijama slova engleskog alfabeta. Sličan primer pružaju i pojedine prezentacije na Internetu gde su tekstovi napisani na srpskom jeziku svedeni na uopštenu latinicu, tako da se specifična latinična slova, na primer, *ć* i *ć* zamenjuju sa *c*.

Osim pomenutih sistema koji su našli široku komercijalnu primenu, razvijen je i izvestan broj programskih sistema koji su namenjeni prvenstveno potrebama istaživanja jezika. Pomenimo, prvo, programe za izradu *konkordancija* (AURORA [141], OCP [52], WordCruncher [148]) koji su našli široku primenu u lingvističkim, filološkim i leksikografskim istraživanjima. Konkordancije, koje se još nazivaju i *KWIC indeksima* (skraćenica od engleskog *Key Words In Context*), sastoje se od ispisivanja svakog pojavljivanja oblika formalne reči iz obrađenog teksta sa kontekstom, koji se obično izražava formalno brojem karaktera ili reči ispred i iza ključne reči. Ključne reči se izlistavaju u unapred zadatom redosledu (obično u alfabetskom poretku). Tako sastavljane konkordancije imaju onoliko redova koliko je obrađeni tekst imao reči. Sastavljanje programa za izradu konkordancija se zasniva na utvrđivanju precizne definicije formalne reči: skup karaktera se rastavlja na alfabetske i separatorske. Ponekad se iz izrade konkordancija isključuju neke reči, obično najfrekventniji sloj gramatičkih reči, njihovim navođenjem u takozvanoj *stop-listi*. Elementi flektivne paradigme mogu biti veoma udaljeni u alfabetski uređenim konkordancijama (na primer, *zao* i *gori*) a homomorfizmi su spojeni (na primer, *mila* od *mio* i *miti*). Premda se konkordancije mogu sastavljati u raznim varijantnim oblicima (na primer, paralelizovane konkordancije bilingvalnih korpusa [83]), njihova izrada se svodi na obradu teksta kao sekvencije karaktera.

Dalji primer pružaju programski paketi za analizu sadržaja (engl. *content analysis*) koje koriste istraživači iz oblasti sociologije i psihoanalize za analizu otvorenih upitnika, novinskih vesti kao i transkribovanog govora (na primer, psihoanalitičke seanse) [90]. Kod ovih programskih paketa je prvi korak sličan programima za izradu konkordancija: oni registruju pojavljivanje svih oblika reči u tekstu, bilo kao indeksa ili konkordancija. Za razliku od lingvističkog ili leksikografskog istraživanja koje podrazumeva svođenje oblika na leksičku reč analiza sadržaja podrazumeva svođenje oblika reči na *kategorije* pod čime se podrazumevaju grupisanje oblika reči po nekoj temi, ili tački gledišta. Na primer, jedan sistem kategorija za temu *anksioznost* može biti: stid, ranjavanje, krivica, odvajanje. Zajednička crta najvećeg broja sistema za analizu sadržaja je pridruživanje kategorija oblicima reči iz teksta i njegova dalja analiza prema pridruženim kategorijama pri čemu se u najvećoj meri zanemaruje jezički aspekti analiziranog teksta.

Na osnovu izloženog, možemo razlikovati dva nivoa programskih sistema za obardu teksta:

- nivo izvršavanja operacije nad ulaznom niskom kojom se ona transformiše u novu nisku bez uključivanja znanja o prirodnojezičkoj prirodi enkodiranoj ulaznom niskom, što u osnovi odgovara transformaciji *karakter* → *karakter*;
- prevođenje ulazne niske u jedinicu, snabdevenu informacijom o njenoj jezičkoj funkciji, čemu bi odgovarala transformacija *karakter* → *reč*.

U većinu navedenih programskih sistema za obradu teksta ugrađene su pretežno komponente prve vrste. Karakteriše ih da je rezultat njihovog rada namenjen ljudskoj interpretaciji.

Programski sistemi koji su namenjeni transformaciji prirodnojezičkog ulaza iz jednog u drugi oblik bez posredovanja čoveka moraju raspolagati komponentama koje vode računa o jezičkoj organizovanosti teksta. Ovakvu vrstu transformacije najbolje ilustruje sistema za potpuno automatsko prevođenje iz jednog jezika u drugi.

Za ovaj drugi vid transformacije neophodno je da se tekst unet u računar oslobodi od grafičkih i pravopisnih dvosmislica koje proističu iz samog načina zapisivanja teksta kako bi se kroz njegovu prethodnu redakciju tekst mogao sagledati kao linearna niska relevantnih jedinica.

Jedinice obrade u komponenatama druge vrste su lingvistički obejkti koji se nazivaju *ulazna rečenica* [144] a koji odgovaraju pojmu realizovane rečenice (engl.*utterence*) nad kojom je izvršena prethodna redakcija. Niske karaktera kojom su enkodirani lingvistički objekti (grafeme, reči, fraze i slično) su, po sebi, neadekvatna reprezentacija ulazne rečenice koja tada poseduje neformalizovano svojstvo „prirodnojezičke organizovanosti". Obrada prirodnojezičkog ulaza, unesenog posredstvom nekog uređaja za unos teksta, međutim, zavisi od mogućnosti da se zapis teksta predstavi kao sekvencija ulaznih rečenica. Procedure koje vrše konverziju zapisa teksta u sekvenciju ulaznih rečenica ne moraju biti eksplicitno navedene. Ova faza konverzije je poznata kao faza *prethodne redakcije*. Ako je određeni prirodni jezik predstavljen svojim korpusom, koriste se i različite strategije za aproksimativno ostvarenje ovakvog zadatka koje se tada nazivaju zajedničkim imenom *obeležavanje* (engl. *tagging*) [30].

Konverzija karakterske reprezentacije teksta u sekvenciju ulaznih rečenica podrazumeva sledeće korake:

1. konverzija karakterske u grafemsku reprezentaciju;
2. određivanje leksičkih jedinica;
3. snabdevanje ulazne rečenice gramatičkim informacijama.

Proces transformacije zapisa teksta u sekvenciju ulaznih rečenica se zasniva na znanju o jeziku. Stepen formalizovanosti znanja predstavlja neposredno meru mogućnosti automatizovanja procedure konverzije zapisa teksta u sekvenciju lingvistički relevantnih jedinica. U odnosu na ovaj kriterijum, mogu se razlikovati različiti stepeni neophodne ljudske intervencije. Krajnje slučajeve predstavlja potpuno ručna redakcija teksta, s jedne, i potpuno automatizovano prethodna redakcija, s druge strane. Između ove dve krajnje mogućnosti mogu se posmatrati različite kombinovane strategije u kojima se zadaci u čitanju dele između čitaoca i automatizovanih procedura. Prema tome gde je locirano znanje o jeziku može se odrediti pripadnost ovih strategija jednom ili drugom od krajnjih slučajeva.

Prvi korak, po obavljenoj prethodnoj redakciji, u postupku konverzije karakterske reprezentacije teksta u sekvenciju ulaznih rečenica predstavlja postupak segmentacije ulazne niske karaktera u nisku *tekstualnih reči* (preciznije govoreći, formalnih reči koje aproksimiraju tekstualne reči). Drugi korak u ovom postupku, poznat kao *morfološka analiza* podrazumeva postupak koji datoj tekstualnoj reči dodeljuje skup morfoloških informacija koje opisuju njeno pojavljivanje u zapisu teksta i pridružuje joj, eventualno, leksičku reč. Ovaj postupak je, po svojoj prirodi, nedeterministički jer za datu tekstualnu reč može postojati više leksičkih reči iz kojih ona, pod određenim uslovima, može biti izvedena. Strategije u realizaciji postupka morfološke analize predstavljaju određeni kompromis u raspodeli informacija između gramatike i rečnika. Jedan prilaz koji se temelji isključivo na primeni pravila i koje razvijen u okviru paradigme generativne fonologije, je dvorazinski modelu morfološke analize Kimma Koskenniemija u čijoj je osnovi uspostavljanje veze između leksičke i površinske forme posredstvom konačnih transduktora preko kojih se definiše sistem alternacija konkretnog prirodnog

jezika [144]. Drugi prilaz, u kome su sve informacije potrebne za morfološku analizu potisnute u rečnik, morfološku analizu zamenjuje konceptom *leksičkog prepoznavanja*. Ovaj postupak koji se zasniva se na modelu elektronskog rečnika, razvio je Max Silberztein [128]. Treći korak u postupku konverzije karakterske reprezentacije teksta u ulaznu rečenicu pretpostavlja postupak sintaksičke analize ulazne rečenice koji se zasniva na rezultatima morfološke analize. U glavi 5 i 6 biće predstavljene komponente za obradu teksta na srpskom jeziku koje obavljaju prvi i drugi korak iz opisanog postupka.

Dok se zapis teksta za potrebe lingvističke analize može, bez gubitka informacija posmatrati kao linearna niska, za potrebe vizuelne komunikacije se predstavlja u dvodimenzionalnom obliku uz pomoć sredstava koja ne pripadaju neposredno planu lingvističke analize. Ovakav dvodimenzionalan zapis teksta organizuje njegov jezički sadržaj kroz logičku i grafičku strukturu. *Logička struktura* zapisa teksta kao sredstvo za predstavljanje jedinica sadržaja teksta i *grafička struktura* zapisa teksta kao sredstvo za predstavljanje jedinica njegovog izgleda. Pod logičkom strukturom se podrazumeva rezultat postupnog rastavljanja sadržaja teksta u manje jedinice, *logičke objekte*, na osnovu njihovog značenja. Logičkih objekti su, na primer, glave, pasusi, liste, naslovi i slično. Pod *grafičkom strukturom* se podrazumeva rezultat rastavljanja sadržaja teksta na manje delove, *grafičke objekte*, na osnovu mogućnosti njihove vizuelne reprezentacije. Grafički objekti su, na primer, stranice, redovi, blokovi (koji sadrže slike), i slično.

Lep primer uzajamnih veza između logičke i grafičke strukture s jedne strane i prirodno-jezičke organizovanosti teksta s druge strane pružaju zakonsko-pravni tekstovi detaljno opisani u [142]. Tekstovi „izmena i dopuna" zakona se sastoje od tipičnih funkcija uređivača teksta — *izbaci*, *promeni*, *zameni* i *ubaci* — zadatih u obliku prirodno-jezičkih iskaza. Te instrukcije upućuju na osnovni tekst zakona preko njegove logičke i grafičke strukture kao i preko sadržaja organizovanog prirodnim jezikom. Primeri ovakvog referisanja su:

- referisanje logičke strukture:
  - U **članu 35.** ispred reči „uprava" briše se reč „opštu"...;
  - Ispod **naslova** ... dodati ...
- referisanje grafičke strukture:
  - U **sedmom redu odozdo** dodati ...
- referisanje jezički organizovanog sadržaja:
  - U tački člana ... **koja reguliše prodaju kafe** ...;
  - Reč ... zameniti rečju ... u **odgovarajućim padežima**;
  - Deo teksta od ... do **kraja rečenice** se briše.

Interna reprezentacija teksta, stoga, mora da obuhvati logičku i grafičku strukturu izraženu pomoću odgovarajućih kodova. Stoga, zapis teksta, kao informatički objekt, ne predstavlja tekst kao objekt organizovan prirodnim jezikom, već je njegova dvostruka aproksimacija: grafemski nivo teksta aproksimira se karakterskim sekvencijama dok se njegov jezički nivo aproksimira zadavanjem logičke i grafičke strukture i eksplicitnim obeležavanjem pojedinih jezičkih jedinica. Problem razumevanja teksta na osnovu zapisa teksta se, dakle, sastoji u rekonstrukciji objekata lingvističke analize na osnovu informacija o karakterskim sekvencijama i enkodiranoj logičkoj i grafičkoj strukturi.

Ako se izuzmu rani pokušaji reprezentovanja teksta u mašinski čitljivom obliku bez beleženja njegove logičke i grafičke strukture, u okviru većine sistema za obradu teksta, razvi-

jen je jezik za njegovo obeležavanje koji sa različitim stepenom formalizovanosti izražava gornji zahtev. Tako, na primer, svaki formater teksta koristi neku metodu obeležavanja logičke i (ili) grafičke strukture teksta. Ta metoda se najčešće svodi na jezik obeležavanja pri čemu su oznake koje se nedvosmisleno mogu razlikovati od samog teksta uklopljene u njega. Korisnik ponekad mora poznavati jezik obeležavanja i sam unositi oznake zajedno sa tekstom — kao što je slučaj sa TEX-om [77] — ili se, pak, one unose posredno, preko menija, kao u slučaju paketa *Ventura Publisher* [26] (tada korisnik ne mora poznavati jezik za označavanje)

Tekstovima koji su namenjeni lingvističkim i filološkim istraživanjima se za potrebe konkretnog istraživanja obeležavaju na odgovarajući način: svaki od programskih paketa za izradu konkordancija koristi svoj (ograničeni) jezik za obeležavanje [141], [52], [148]. I korpusi namenjeni ovakvim istraživanjima se obeležavaju u skladu sa izabranim ili specijalno razvijenim programskim alatima (primer Brown i LOB korpusa engleskog jezika [30]).

Upotreba specijalizovanih jezika za obeležavanje teksta značajno smanjuje upotrebnu vrednost teksta u mašinski čitljivom obliku i skraćuje njegov životni vek. Nije nepoznata nemogućnost obrade teksta pripremljenog uz pomoć jednog formatera nekim drugim formaterom. Šta više, reprezentacioni jezici mogu se razlikovati od verzije do verzije istog formatera. Problemi se najčešće javljaju na liniji *autor* → *izdavač* → *dalji korisnik*, pri čemu *dalji korisnik* može biti štamparija, tekstualna baza podataka, korpus, globalna mreža i tako dalje. Ovakvi konflikti se mogu razrešiti:

- ponovnim unošenjem teksta;
- primenom postupka „razobeležavanja" i ponovnog obeležavanja;
- izradom programa, ili primenom nekog postojećeg programa, za prevođenje jednog jezika obeležavanja u drugi.

Ovo usklađivanje jezika obeležavanja ne samo da oduzima vremene, već ono uvek vodi gubitku informacija i donosi grešake.

Jedno rešenje ovih problema predstavlja propisivanje standarda za jezike obeležavanja teksta. Polaznu osnovu za izradu standarda za obeležavanje teksta, predstavljaju s jedne strane postojeći jezici za obeležavanje, prvenstveno u smislu opisivanja svojstava teksta koja treba standardom obuhvatiti, a s druge strane već izrađeni standardi u oblasti kodiranja dokumenata u cilju njegove obrade i prenosa. U ovom domenu od značaja je standard ODA (*Office Document Architecture*) [58] koji je namenjen opisu arhitekture kancelarijskih dokumenata i formata za njihovu razmenu. Premda ovaj standard pruža mnoge mogućnosti za obeležavanje dokumenata, karakteriše ga ograničena funkcionalnost koja proističe iz činjenice da je semantika jezika za obeležavanje ugrađena u sam standard. Veće mogućnosti, posebno sa stanovišta lingvističkih, filoloških i leksikografskih istraživanja pruža standardni generalizovani jezik za obeležavanje teksta SGML [57] koji će biti detaljno predstavljen u glavi 2 ovog rada. U glavi 3 biće predstavljene mogućnosti njegove primene dok će u glavi 4 biti prikazan jedan programski sistem za analizu označenih SGML dokumenata.

Nivoi reprezentovanja teksta koji su naznačeni u ovoj glavi kao i model reprezentacije koji će biti predstavljen u narednim glavama biće ilustrovan na primeru jedne pesme Duška Radovića[1]. Kopija tri stranice na kojima je pesma zapisana data je na sledećoj strani. Jedan mogući zapis u elektronskom obliku je sledeći:

[1] Pesma „KAKO SE SMANJIVALA REALNA VREDNOST JEDNOG KAUBOJA" iz ciklusa „WESTERN" preuzeta je iz knjige „Utorak" poezije i proze za decu Duška Radovića koju su u četiri knjige izdali Beogradski izdavačko-grafički zavod i Narodna knjiga 1983. godine

КАКО СЕ СМАЊИВАЛА РЕАЛНА ВРЕДНОСТ ЈЕДНОГ КАУБОЈА

100%

У ИДЕЈИ — СЈАЈАН,
У ПРИНЦИПУ — ЛАФ,
У ПОБУДИ — БАЈАН,
У НАМЕРИ — ПАФ!
У ЦРТЕЖУ — БЉЕСАК,
У КОНЦЕПТУ — ФЛИТ,
У НАЧЕЛУ — ТРЕСАК,
КАО ЦАКА — ХИТ!
ТЕОРЕТСКИ — ЧУДО,
У ЖЕЉАМА — ГРОМ,
У ПРОЈЕКТУ — ЛУДО,
ЗА ПОЧЕТАК — ЛОМ!

80%

МЕЂУТИМ,
ЧИМ НЕГДЕ
ПРАСНЕ
— ОН
ЗА НИЈАНСУ
СПЛАСНЕ.
ЧИМ НЕГДЕ
НЕШТО
ЛУПИ
— ОН СЕ
ПРИМЕТНО
СКУПИ.

60%

БУДЕ ЛИ
НЕГДЕ
ТРЕСКА
— ОН СЕ
БУКВАЛНО
СПЉЕСКА.

АКО ГА
НЕШТО
ТАКНЕ
— ОН СЕ
ЗА ПРОЦЕНАТ
СМАКНЕ.

40%

ПОЧНЕ ЛИ
РАФАЛ
ДА КРЧКА
— ОН СЕ
ДИРЕКТНО
ЗБРЧКА.

АКО
И С БОКА
КОКА
— ЈОШ СЕ
ЗА РЕЦКУ
СКЉОКА.

ЈОШ АКО
И С ЛЕЂА
ГРУНЕ
— ОН
УПАДЉИВО
ТРУНЕ.

20%

ДВСИ СЕ,
СТРЕЛА
ВРИСНЕ
— ОН СЕ
ЗА ПЕДАЉ
СТИСНЕ.
ПОГЛЕДА:
СЕКИРА
ТРЧИ
— ЈОШ СЕ
ЗА ПЕДАЉ
ЗГРЧИ.
ПЕДАЉ
ПО ПЕДАЉ
СТРАЂИ
— И ГРОБ
ЗАСЛУЖИ
КРАЂИ

KAKO SE SMANJIVALA REALNA VREDNOST JEDNOG KAUBOJA Primer 1
100%
U IDEJI - SJAJAN,
U PRINCIPU - LAF,
U POBUDI - BAJAN,
U NAMERI - PAF!
U CRTEŽU - BLJESAK,
U KONCEPTU - FLIT,
U NAČELU - TRESAK,
KAO CAKA - HIT!
TEORETSKI - ČUDO,
U ŽELJAMA - GROM,
U PROJEKTU - LUDO,
ZA POČETAK - LOM!
80%
MEĐUTIM, ČIM NEGDE PRASNE
- ON ZA NIJANSU SPLASNE.
ČIM NEGDE NEŠTO LUPI
- ON SE PRIMETNO SKUPI.
60%
BUDE LI NEGDE TRESKA
- ON SE BUKVALNO SPLJESKA.
AKO GA NEŠTO TAKNE
- ON SE ZA PROCENAT SMAKNE.
40%
POČNE LI RAFAL DA KRČKA
- ON SE DIREKTNO ZBRČKA.
AKO I S BOKA KOKA
- JOŠ SE ZA RECKU SKLJOKA.
JOŠ AKO I S LEĐA GRUNE
- ON UPADLJIVO TRUNE.
20%
DESI SE, STRELA VRISNE
- ON SE ZA PEDALJ STISNE.
POGLEDA: SEKIRA TRČI
-JOŠ SE ZA PEDALJ ZGRČI.
PEDALJ PO PEDALJ STRAĆI
- I GROB ZASLUŽ I KRAĆI.

U ovom zapisu, tekst pesme je predstavljen isključivo kao sekvencija karaktera koji odgovaraju odgovarajućim grafemama i upotrebljenim interpunkcijskim znacima. Jedino obeležje koje nije iz ovog skupa a koje ovu sekvenciju karaktera povezuje sa njegovim uobičajenim dvodimenzionalnim zapisom je karakter (ili karakteri) koji označavaju kraj jednog i početak drugog reda. Programski paket koji ovu sekvenciju karaktera treba na neki način da obradi suočava se sa sledećim, i mnogim drugim, nedoumicama:

- o kakvoj vrsti teksta se radi;

- na kom jeziku je taj tekst napisan;
- kakva je veza karaktera i grafemskog sastava tog jezika;
- kakva je veza karaktera i skupa interpunkcijakih znakova;
- kojim pismom je tekst originalno vizuelizovan;
- koja slovna garnitura je korišćena za vizuelizaciju teksta;
- da li su karakteri koji označavaju kraj reda od značaja za vizuelizaciju teksta i njegovo razumevanje;
- da li se prvi red teksta razlikuje od ostalih.

Na ova, i slična pitanja, čitalac može lako da odgovori. Znanje na kojima se ti odgovori zasnivaju treba učiniti eksplicitno dostupnim programskim paketima koji će tekst obrađivati. Jedna mogućnost je da se tekst obeleži korišćenjem SGML jezika za obeležavanje pri čemu je semantička vrednost obeležja — *etiketa* u SGML terminologiji — definisana TEI preporukama za obeležavanje teksta (videti 3.3.1 i dodatak B). Početak izabrane pesme obeležene na ovaj način bio bi:

Primer 2

```
<TeiHeader>
<titleStmt>
   <title>Utorak</title><author>Duš ko Radović</author>
<language id=SCO wsd='JUS.B1' alpha='Cyrillic'>
.....................................
<div2 type='ciklus'>
   <head>Western</head>
<div3 type='pesma'>
   <head>KAKO SE SMANJIVALA REALNA &#RS;&#RE;VREDNOST JEDNOG KAUBOJA</head>
<head rend='bold 14pt'>100%</head>
<lg1 type='strofa' rend='roman 14pt'>
<l>U IDEJI &mdash; SJAJAN,</l>
<l>U PRINCIPU &mdash; LAF,</l>
<l>U POBUDI &mdash; BAJAN,</l>
<l>U NAMERI &mdash; PAF!</l>
<l>U CRTEŽU &mdash; BLJESAK,</l>
<l>U KONCEPTU &mdash; FLIT,</l>
<l>U NAČELU &mdash; TRESAK,</l>
<l>KAO CAKA &mdash; HIT!</l>
<l>TEORETSKI &mdash; ČUDO,</l>
<l>U ŽELJAMA &mdash; GROM,</l>
<l>U PROJEKTU &mdash; LUDO,</l>
<l>ZA POČETAK &mdash; LOM!</l>
</lg1>
<head='bold 12pt'>80%,</head>
<lg1 type='strofa' rend='roman 12pt'>
<lg2>
<l>MEĐ UTIM,<l>
<i rend='1ind'>ČIM NEGDE</l>
```

```
<l rend='2ind'>PRASNE</l></lg2>
<lg2>
<l>&mdash; ON</l>
<l rend='1ind'>ZA NIJANSU<l>
<l rend='2ind'>SPLASNE.</l></lg2>
<lg2>
<l>ČIM NEGDE</l>
<l rend='1ind'>NEŠTO </l>
<l rend='2ind'>LUPI</l></lg2>
<lg2>
<l>&mdash; ON SE</l>
<l rend='1ind'>PRIMETNO</l>
<l rend='2ind'>SKUPI.<l></lg2>
</lg1>
```

Odgovori na gornja pitanja eksplicirani su sledećim obeležjima i njihovim atributima:

- o kome delu i autoru se radi dato je u okviru TEI zaglavlja (`TeiHeader`) dok je vrsta teksta, `pesma`, vrednost atributa `type` odgovarajućeg odeljka;
- na kom jeziku je tekst napisan, koje pismo se koristi za njegovo zapisivanje kao i veza karakterskog skupa i grafemskog sastava su sastavni deo etikete `language` koja je deo TEI zaglavlja;
- interpunkcijski znaci kojima nije pridružen karakter u karakterskom skupu predstavljeni su referencom entiteta (`&mdash;`);
- svaka promena slovne garniture naznačena je kao vrednost atributa `rend` logičkog elementa u okviru koga se ta slovna garnitura koristi. Prisutne su i druge naznake izgleda vizuelizovanog teksta, koje se sve u ovom slučaju odražavaju na njegovu poruku;
- svaki red teksta predstavlja zasebnu logičku celinu, sadržaj elementa `l` koji je od značaja za razumevanje i predstavljanje pesničkog sadržaja;
- prvi red teksta je naslov pesme te je stoga drukčije obeležen od ostalih redova — on je sadržaj elementa `head`.

Na ovaj način su istaknuti i učinjeni dostupnim programskim sistemima za obradu teksta elementi logičke i grafičke strukture izabranog teksta. Sadržaj elemenata logičke strukture koji se dalje ne raščlanjavaju je i ovde predstavljen kao karakterska niska kao i u primeru 1. Tako se sa stanovišta programskog sistema za obradu ovako obeleženog teksta niska „ČIM NEGDE“ i „PRIMETNO“ suštinski ne razlikuju jer obe predstavljaju samo niske karaktera koje se dalje ne raščlanjavaju. Činjenica da prva od njih sadrži dve tekstualne reći (odnosno njihove aproksimacije) a druga samo jednu ovim sistemima ne može biti poznata. Još manje oni mogu razlikovati niske „PRASNE“ i „MEĐUTIM,“ koje obe aproksimiraju po jednu tekstualnu reč: prva je oblik glagola a druga rečca. Da bi ovakvo karakterisanje niski bilo moguće potrebno je izvršiti morfološku analizu zapisa teksta. Rezultat analize kraja izabrane pesme koja se zasniva na korišćenju elektronskog rečnika srpskog jezika bio bi:

Primer 3

```
<head rend='bold 6pt'>20%,</head>
<lg1 type='strofa' rend='roman 6pt'>
```

```
<lg2>
<l><w a='desiti.V33.51.3:P3s'>DESI</w>
   <w a='se.Par'>SE,</w></l>
<l rend='1ind'><w a='strela.N70.01-E%#E4.03:fsn-'>STRELA</w></l>
<l rend='2ind'><w a='vrisnuti.V23.50.3:P3s'>VRISNE</w></l></lg2>
<lg2>
<l>&mdash; <w a='on.ProN05:msn*'ON</w>
          <w a='se.Par'>SE</w></l>
<l rend='1ind'><w a='za.Pre'>ZA</w>
              <w a='pedalj.N22.51-*:msa->PEDALJ</w></l>
<l rend='2ind'><w a='stisnuti.V23.50.3:P3s'>STISNE.</w></l></lg2>
<lg2>
<l><w a='pogledati.V01.50.2:P3s'>POGLEDA:</l>
<l rend='1ind'><w a='sekira.N70.01-E%#E2.12:fsn-'>SEKIRA</w></l>
<l rend='2ind'><w a='trčati.V31.00.3:P3s'>TRČI</w></l></lg2>
<lg2>
<l>&mdash;<w a='još.Adv'>JOŠ</w>
         <w a='se.Par'>SE</w></l>
<l rend='1ind'><w a='za.Pre'>ZA</w>
              <w a='pedalj.N22.51-*:msa->PEDALJ</w></l>
<l rend='2ind'><w a='zgrčiti.V33.51.3:P3s'>ZGRČI.</w></l></lg2>
<lg2>
<l><w a='pedalj.N22.51-*:msn-'>PEDALJ</w></l>
<l rend='1ind'><w a='po.Adv'>PO</w>
              <w a='pedalj.N22.51-*:msn-'>PEDALJ</w></l>
<l rend='2ind'><w a='stračiti.V33.51.3:P3s'>STRAĆI</w></l></lg2>
<lg2>
<l>&mdash; <w a='i.Con'>I</w>
          <w a='grob.N08.01:msa-'>GROB</w></l>
<l rend='1ind'><w a='zaslužiti.V33.01.3:P3s'>ZASLUŽI</w></l>
<l rend='2ind'><w a='kratak.A10.03*:k@msa-'>KRAĆI.</w></l></lg2>
</lg1><div2>
```

Tektaulne reči su ovom zapisu sadržaj elementa w čiji jedini atribut a sadrži sve informacije pridružene tekstualnoj reči u procesu morfološke analize, kao što su pridružena leksička reč sa vezom na varijantne oblike te oznake vrste reči, morfološke klase i realizovanih gramatičkih kategorija. Opis izrade ovakvog zpisa teksta kao i mogućnosti njegove primene sadržaj su ovog rada.

Glava 2

Uvod u SGML

Standardni generalizovani jezik za označavanje — SGML (skraćenica od engl. *Standard Generalized Markup Language*) [57] je međunarodni standard reprezentovanja teksta u mašinski čitljivom obliku koje je nezavisno od računarskog uređaja i njegove programske opreme. SGML je ustvari *metajezik* za formalno opisivanje *jezika označavanja* pod čime se podrazumeva skup konvencija označavanja koje se koriste za enkodiranje teksta. Jedan jezik označavanja mora da specifikuje koje oznake su dozvoljene, koje su obavezne, kako se oznake razlikuju od teksta i šta te oznake znače. SGML je jezik koji omogućava specifikovanje jezika koji zadovoljavaju prva tri od navedenih uslova. Sam SGML je lišen semantike koju svaki korisnik treba da ugradi prema svojim potrebama (videti odeljak 3.3).

Kao najvažnija svojstva SGML-a mogu se istaći sledeća:

- *apstraktna sintaksa*, koja se sastoji od pravila za opisivanje strukture i atributa dokumenta. Ova pravila se primenjuju u svim implementacijama SGML-a. Apstraktna sintaksa je u standardu specifikovana formalnim sintaksnim pravilima izvođenja od kojih svako definiše jednu sintaksnu promenljivu. Za specifikovanje se koristi jedna od varijanti BNF-notacije kakve su uobičajene za kontekstno-slobodne jezike;
- *konkretna sintaksa*, koja povezuje pravila apstraktne sintakse sa konkretnim skupom karaktera i konkretnim veličinama (videti odeljak 2.8);
- *deklaracije obeležja*, koje omogućavaju definisanje korisničkog vokabulara generičkih obeležja i njihovih atributa u okviru *definicije tipa dokumenta*, skraćeno DTD (videti odeljke 2.1 i 2.2);
- *zamena teksta*, koja predstavlja tehniku za zamenu jedne niske karaktera u dokumentu drugom niskom koja fizički može biti smeštena unutar iste ili druge datoteke (videti odeljak 2.4). Ovim se obezbeđuje prenosivost podataka iz jednog hardverskog ili softverskog okruženja u drugo bez gubitka informacija;
- *proizvoljnost sadržaja podataka*, koja omogućuje uključivanje u dokument sadržaja koji nisu tekstualne prirode, kao, na primer, slika, grafikona, matematičkih formula i slično (videti odeljak 2.5);
- *definisanje jedinstvenih identifikatora* koji omogućavaju referisanje u tekstu imenovanih elemenata (videti odeljak 2.3);
- *uključivanje i isključivanje delova teksta* prema potrebi pojedinačnih obrada (videti odeljak 2.6);

- *opciona svojstva jezika* koja omogućavaju, između ostalog, minimizaciju obeležja (videti pododeljak 2.2.3), definisanje sponskih procesa za preslikavanje dokumenta iz jednog u drugi tip (videti odeljak 2.7) i istovremeno korišćenje više skupova obeležja, odnosno DTD-a (videti odeljak 2.7.3).

2.1 Struktura SGML dokumenta

Svaki SGML dokument se sastoji od *SGML prologa* i *instancije dokumenta*. Prolog se sastoji od *SGML deklaracije* i *definicije tipa dokumenta*.

U SGML deklaraciji preciziraju se svi detalji SGML dijalekta koji se u dokumentu koristi, a to su, redom:

- *karakterski skup dokumenta*, u kome se navodi koji se karakteri u dokumentu mogu koristiti i koje su njihove uloge.
- *skup memorijskih kapaciteta* koji su potrebni za obradu dokumenta. Ovaj skup podataka određuje memorijski prostor koji SGML raščlanjivač rezerviše za imena entiteta, imena elemenata, tekst zamene svih entiteta, tekst vrednosti svih atributa i slično.
- *konkretna sintaksa* koja se koristi unutar dokumenta. SGML deklaracija se uvek zapisuje korišćenjem *referentne konkretne sintakse* (videti odeljak 2.8);
- *domen konkretne sintakse* koji specifikuje da li se konkretna sintaksa primenjuje na celi dokument ili samo na tekst dokumenta a u tom slučaju se ceo prolog, a ne samo SGML deklaracija, zapisuje korišćenjem referentne konkretne sintakse.
- *SGML svojstva* koja se koriste unutar dokumenta, kakva su minimizacija obeležja, konkurentne strukture i slično.
- *posebne informacije za aplikaciju* koje su potrebne da bi se dokument obradio.

Primer SGML deklaracije dat je u dodatku A.

Validnost instancije dokumenta utvrđuje se u odnosu na definiciju tipa dokumenta. U najjednostavnijem slučaju definicija tipa dokumenta se sastoji samo od *osnovne definicije tipa dokumenta* koja prethodi instanciji dokumenta. Na primer,

```
<!DOCTYPE MOJ.DTD [
      <!-- sve deklaracije iz MOJ.DTD dolaze ovde -->
      ...
]>
<MOJ.DTD>
      Ovo je jedna instancija dokumenta za MOJ.DTD tip dokumenta.
</MOJ.DTD>
```

S obzirom da definicija tipa dokumenta može biti veoma složena te da ona obično opisuje jednu klasu dokumenata, kao što su pisma, radovi, izveštaji i slično, i da se može primeniti na mnoge instancije dokumenata, često se DTD smešta u zasebnu datoteku iz koje se na odgovarajući način poziva:

```
<!DOCTYPE rad.u.zborniku system "rad-u-zb.dtd" [
]>
<rad.u.zborniku>
      Ovo je jedna instancija nemodifikovanog tipa dokumenta
      'rad u zborniku'.
</rad.u.zborniku>
```

U ovom primeru definicija tipa dokumenta `rad.u.zborniku` nije eksplicitno data ali je naznačeno da se ona može pročitati iz datoteke čiji je sistemski identifikator naveden između navodnika. Uglaste zagrade se u ovom slučaju mogu, ali ne moraju, navesti.

Unutar uglastih zagrada navodi se *podskup definicije tipa dokumenta* čija je namena da modifikuje DTD koji se poziva iz zasebne datoteke:

```
<!DOCTYPE rad.u.zborniku system "rad-u-zb.dtd" [
      <!ENTITY dtd "definicija tipa dokumenta">
      <!ELEMENT moja.etiketa - - (#PCDATA)>
      <!-- sve ostale deklaracije i ponovljene definicije
           specifične za ovu instanciju dokumenta dolaze ovde.-->
]>


<rad.u.zborniku>
      Ovo je jedna instancija modifikovanog tipa dokumenta
      'rad u zborniku' koja koristi <moja.etiketa> moju
      specijalnu etiketu </moja.etiketa> i u kome se termin
      koji se često koristii uključuje referisanjem
      opšteg entiteta &dtd;.
</rad.u.zborniku>
```

Na snazi je u ovom slučaju definicija tipa dokumenta koja sadrži, pre svega, DTD podskup naveden između uglastih zagrada a zatim i sadržaj datoteke navedene iza ključne reči `system`. Redosled je veoma bitan jer je za SGML od značaja samo prva deklaracija entiteta na koju naiđe. Ako bi, dakle, i datoteka "rad-u-zb.dtd" sadržala deklaraciju entiteta *dtd*, ta deklaracija bila bi ignorisana. Višestruko deklarisanje entiteta je potpuno legalno i obično se koristi za modifikovanje postojećih DTD skupova. S druge strane, elementi se smeju samo jednom deklarisati što znači da ukoliko bi i datoteka "rad-u-zb.dtd" sadržala deklaraciju elementa `moja.etiketa`, SGML raščlanjivač bi signalizirao grešku.

Instancija dokumenta predstavlja sam sadržaj dokumenta. Ona sadrži samo tekst, oznake i reference opštih entiteta. Nove deklaracije unutar instancije dokumenta nisu dozvoljene. Velika dokumenta se obično grade na modularan način tako što se unutar DTD podskupa deklarišu opšti entiteti za svaku zasebnu celinu, odnosno modul:

```
<!DOCTYPE knjiga system "knjiga.dtd" [
      <!ENTITY glava1 system "glava1.txt">
      <!ENTITY glava2 system "glava2.txt">
      <!ENTITY glava3 "tek treba napisati!">
]>
<knjiga>
      <prednji.deo> Ovde dolazi uvodni deo.  </prednji.deo>
      <telo.knjige>
          &glava1;
          &glava2;
          &glava3;
          ...
      </telo.knjige>
</knjiga>
```

U ovom primeru je DTD koji je sadržan u datoteci "knjiga.dtd" proširen deklaracijama opštih entiteta za svaku glavu knjige. Prva dva entiteta referišu datoteke od kojih svaka sadrži tekst odgovarajuće glave knjige. Deklaracija trećeg entiteta predstavlja jednostavnu niskovnu supstituciju, koja u ovom slučaju treba da označi da odgovarajući tekst još ne postoji. Da bi se dobio željeni sadržaj, reference entiteta tipa `&glava1;` unutar instancije dokumenta razrešava raščlanjivač (videti odeljak 4.1). Datoteke "glava1.txt", "glava2.txt", itd. mogu da sadrže samo označeni tekst i, eventualno, nove reference entiteta. Referisanje entiteta unutar entiteta može ići do dubine koja je propisana vrednošću parametra `ENTLVL` u konkretnoj sintaksi koja se koristi (videti odeljak 2.8).

2.2 Elementi

Element je termin koji se u SGML-u koristi za označavanje tekstualnih jedinica, odnosno strukturnih komponenti teksta. Svakom elementu se pridružuje različito ime koje se u standardu naziva *generički identifikator*. SGML propisuje načine za ekspliciranje veza između elemenata različitog tipa kroz definisanje njihovog *sadržaja*.

Pridruživanje značenja elementu određenog tipa nije predmet standarda i SGML jezik ga ne podržava. Standard, međutim, obezbeđuje izbor određene vrste sadržaja što omogućava predstavljanje tekstualnih jedinica različite vrste. Tako je moguće:

- hijerarhijsko strukturiranje dokumenta, kakvo se sreće u dokumentima svih vrsta. Na primer, uobičajeno je da se struktura knjige opisuje drvetom: knjiga se sastoji od prednjeg dela, tela knjige i zadnjeg dela. Telo knjige se, pak, sastoji od delova, delovi od glava, glave od odeljaka i tako dalje.
- izdvajanje strukturnih delova dokumenta, koji se ne uklapaju u njegovu hijerarhijsku strukturu. Takve su, na primer, liste koje mogu biti prisutne u gotovo svim strukturnim delovima dokumenta, do nivoa pasusa. Sama lista se obično sastoji od članova liste koji se pak mogu sastojati od više pasusa i novih lista, ali ne i od glava, odeljaka i pododeljaka. Moguće je, dakle, uspostaviti složeniju strukturu SGML dokumenta od one koju predstavlja struktura drveta.
- isticanje delova teksta, koje može biti podstaknuto različitim motivima. Tako se u tekstu mogu, prema potrebi, istaći delovi koji predstavljaju termine, strane reči, lična

imena, datume i slično. Ovakvo isticanje je, u principu, moguće na svim hijerarhijskim nivoima dokumenta.

- obeležavanje delova dokumenta koji ne sadrži tekst u užem smislu, pod čime se obično podrazumeva neki prirodno-jezički zapis. Ovakvi delovi dokumenta mogu imati različiti sadržaj: to mogu biti matematičke ili hemijske formule, dijagrami, slike i slično. Oni su obično zapisani korišćenjem drugačijeg karakterskog sastava i drugačijih konvencija od onih koji se koriste za tekst u užem smislu što je opisano u pridruženoj notaciji (videti odeljak 2.5). U fazi obrade dokumenta oni obično zahtevaju drugačiji tretman.
- obeležavanje referentnih tačaka u dokumentu na koje je moguće pozivanje iz istog ili iz drugih dokumenata.

Inventar elemenata nekog tipa dokumenta uspostavlja se u prologu dokumenta i to u delu koji sadrži definiciju tipa dokumenta. Svaki element se u DTD mora eksplicitno deklarisati na sledeći način:

Mdo "ELEMENT" *tip_elementa minimizacija deklaracija_sadržaja* **Mdc**

gde su terminali, odnosno tokeni, graničnici **Mdo** i **Mdc** i ključna reč "ELEMENT" dok su *tip_elementa*, *minimizacija* i *deklaracija_sadržaja* sintaksičke promenljive. *tip_elementa* uvodi jedan ili više generičkih identifikatora za elemente na koje se deklaracija odnosi. Od ovih generičkih identifikatora formiraju se početna i krajnja etiketa koje obeležavaju početak odnosno kraj pojave elementa u instanciji dokumenta. Na primer, pojavljivanje elementa čije je generičko ime `glava` u instanciji dokumenta se obeležava etiketama `<glava>` i `</glava>`. *minimizacija* određuje da li je izostavljanje početne, odnosno krajnje etikete deklarisanih elemenata u instanciji dokumenta dozvoljeno ili ne, dok *deklaracija_sadržaja* propisuje vrstu sadržaja deklarisanih elemenata. Potpuna sintaksička definicija deklaracije elemenata kao i apstraktna sintaksa ostalih komponenata SGML-a data je u dodatku C.

2.2.1 Deklaracija sadržaja

Deklaracija sadržaja elementa specifikuje koji je dozvoljeni sadržaj pojavljivanja tog elementa. Sadržaj se može specifikovati bilo preko drugih elemenata bilo korišćenjem sledećih specijalnih rezervisanih reči:

- EMPTY znači da je sadržaj elementa prazan. Ovi elementi se obično koriste za postavljanje referentne tačke u dokumentu. Izostavljanje krajnje etikete ja za elemente sa praznim sadržajem obavezno.
- CDATA znači da su *karakterski podaci* sadržaj elementa. Nikakve oznake se unutar elementa sa ovakvim sadržajem ne prepoznaju, osim njegove krajnje etikete.
- RCDATA znači da su *zamenljivi karakterski podaci* sadržaj elementa. Unutar elementa sa ovakvim sadržajem, osim njegove krajnje etikete prepoznaju se i zamenjuju reference karakterskih entiteta i parametarskih entiteta.
- #PCDATA znači da su *raščlanjivi karakterski podaci* sadržaj elementa. Sadržaj ovako deklarisanog elementa se raščlanjuje i sve oznake se prepoznaju. U hijerarhijskoj strukturi dokumenta, većina elemenata pridružena listovima drveta imaće #PCDATA sadržaj. Takvi su, na primer, elementi `istaknuto` i `termin` iz primera datog u dodatku A.

Prve tri od navedenih rezervisanih reči predstavljaju takozvani *deklarisani sadržaj* koji u potpunosti opisuje sadržaj elementa kome je pridružen. Rezervisana reč #PCDATA, premda se često koristi samostalno kao u primeru elemenata `istaknuto` i `termin`, može se kombinovati sa drugim elementima u okviru *modela sadržaja*[1] koji opisuje kakve veze postoje između različitih elemenata. S obzirom da se rezervisana reč PCDATA može kombinovati sa drugim elementima u okviru modela sadržaja, graničnik # (**rci**) za indikaciju rezervisane reči koristi se za razlikovanje raščlanjivih karakterskih podataka od elementa čiji je generički identifikator PCDATA.

Model sadržaja se obavezno zapisuje unutar zagrada, odnosno, graničnika **gpo** i **gpc**, između kojih se navodi kako se kombinuju raščlanjivi karakterski podaci, elementi i drugi modeli sadržaja koji se u standardu nazivaju jednim imenom *tokeni sadržaja*. Tokeni sadržaja se povezuju unutar modela sadržaja korišćenjem sledećih *grupnih konektora* koji određuju redosled i selekciju tokena sadržaja unutar modela sadržaja:

seq	,	Svi se moraju pojaviti u datom redosledu.
and	&	Svi se moraju pojaviti u proizvoljnom redosledu.
or	|	Mora se pojaviti jedan i samo jedan.

seq, **and** i **or** su imena grupnih konektora u apstraktnoj sintaksi a u referentnoj konkretnoj sintaksi oni se redom realizuju kao karakteri ",", "&", odnosno "|". Formalno, jedna **and** grupa se svodi na jednu **or** grupu **seq** grupa permutacija. Na primer, model `( a & b )` je ekvivalentan grupi modela `(( a, b) | ( b, a ))` (videti 4.4).

Među grupnim konektorima nije propisana relacija prioriteta jer se unutar grupe (tj. između zagrada) može koristiti samo jedan grupni konektor. Međutim, model sadržaja ugnježden unutar modela sadržaja može koristiti neki drugi grupni konektor. Ugnježdenost modela sadržaja može ići do dubine koja je određena parametrom `GRPLVL` konkretne sintakse koja se koristi (videti odeljak 2.8).

Na svaki token sadržaja naveden unutar modela sadržaja može se primeniti *indikator pojavljivanja* koji ukazuje koliko se puta on može pojaviti. Token sadržaja koji nije modifikovan indikatorom pojavljivanja mora se pojaviti tačno jedanput. Indikatori pojavljivanja modifikuju pojavljivanje tokena sadržaja na sledeći način:

opt	?	Opcioni element (nula ili jedno pojavljivanje)
plus	+	Obavezni i ponovljiv element (jedno ili više pojavljivanja)
rep	*	Opcioni i ponovljiv element (nula ili više pojavljivanja)

opt, **plus** i **rep** su imena indikatora pojavljivanja u apstraktnoj sintaksi a u referentnoj konkretnoj sintaksi oni se redom realizuju kao karakteri "?", "+", odnosno, "*". Za token sadržaja #PCDATA se implicitno pretpostavlja da ima indikator pojavljivanja **rep**.

Indikatori pojavljivanja su višeg prioriteta od grupnih konektora. Stoga se model sadržaja `(a | b)+` razlikuje od modela sadržaja `(a+ | b+)`. Prvi dozvoljava proizvoljnu sekvenciju elemenata `a` i `b` dok drugi dozvoljava sekvenciju elemenata `a` ili sekvenciju elemenata `b`.

Deklaracija modela sadržaja elementa može biti rekurzivna. Na primer, elementi `a` i `b` su rekurzivno deklarisani. Ipak, deklaracija elementa `b` nije dozvoljena jer ne vodi ni do jedne korektne instancije dokumenta.

[1] U standardu se on naziva *grupa modela* dok je termin *model sadržaja* širi i obuhvata *grupu modela* i *izuzetke*. Terminologija je u ovom radu izmenjena u odnosu na standard u onim slučajevima kada više odgovara sintaksnom opisu primenjenom u izradi raščlanjivača.

```
<!ELEMENT a          - -  ( #PCDATA | a )*    >
<!ELEMENT b          - -  ( b )*              >
```

Ovakva rekurzivna deklaracija može se naći, na primer, u HTML DTD-u (videti 3.3.2) gde je element DIV koji se koristi uopšteno za razne vrste delova dokumenta deklarisan na sledeći način:

```
<!ELEMENT DIV - - (DIV|%heading|%block|HR|ADDRESS)*     >
```

U definicijama tipa dokumenta, često se nailazi i na indirektne rekurzivne deklaracije:

```
<!ELEMENT a          - -  ( b | #PCDATA )*    >
<!ELEMENT b          - -  ( a | #PCDATA )*    >
```

U ovom slučaju element a se može interpretirati kao *pasus* a element b kao *dugački citat.* Tada se gornje deklaracije mogu protumačiti na sledeći način: Pasus se sastoji od sekvencije karakterskih podataka i dugačkih citata dok se dugački citat sastoji od sekvencije karakterskih podataka i pasusa.

Ako se token sadržaja #PCDATA pojavljuje unutar modela sadržaja nekog elementa kaže se da element ima *mešoviti sadržaj*; u suprotnom se njegov sadržaj *sastoji od elemenata.* U drugom slučaju se sadržaj elementa definiše samo u odnosu na druge elemente i on ne sme sadržati karakterske podatke, osim praznih karaktera koji se ignorišu. U prvom slučaju prazni karakteri su dozvoljeni i oni se tretiraju kao deo raščlanjivih karakterskih podataka. Sledeći primer ilustruje razliku između ove dve vrste sadržaja. Neka se u definiciji tipa dokumenta nalaze sledeće deklaracije:

```
<!ELEMENT doc        - -  ( a & ( b | c ) )   >
<!ELEMENT (a, b, c)  - -  (#PCDATA)           >
```

Sadržaj elementa doc se sastoji od drugih elemenata. Njegov model sadržaja govori da se on sastoji od sekvencije dva tokena sadržaja u proizvoljnom redosledu: elementa a i modela (b | c) koji, pak, govori da će se realizovati bilo element b bilo element c. Stoga bi dve legalne realizacije elementa doc u instanciji dokumenta bile:

```
<doc><a>element a.</a>
     <b>element b.</b>
</doc>
<doc>
     <c>element c.</c><a>element a.</a></doc>
```

Blanko i tabulator karakteri koji se pojavljuju između krajnje etikete elementa a i početne etikete elementa b u prvom slučaju i početne etikete elementa doc i početne etikete elementa c biće, s obzirom na vrstu sadržaja elementa doc, ignorisani. Ako bi se pak u definiciji tipa dokumenta nalazile sledeće deklaracije:

```
<!ELEMENT doc        - -  ( a & ( #PCDATA | c ) )   >
<!ELEMENT ( a, c )   - -  (#PCDATA)                  >
```

onda bi element doc imao mešovit sadržaj s obzirom na prisustvo tokena #PCDATA u njegovom modelu. On se sastoji od sekvencije dva tokena sadržaja u proizvoljnom redosledu: elementa a i modela (#PCDATA | c), koji se realizuje bilo kao element c bilo kao raščlanjivi karakterski podaci. Sledeća realizacija elementa doc u instanciji dokumenta ne bi bila legalna, jer bi blanko i tabulator karakteri koji se pojavljuju između krajnje etikete elementa a i početne etikete elementa b zadovoljili token #PCDATA u modelu (#PCDATA | c), pa bi pojavljivanje i elementa c bilo nedozvoljeno:

```
<doc><a>element a.</a>
     <c>element b.</c>
</doc>
```

Iz sličnog razloga bila bi nedozvoljena i sledeća realizacija elementa doc:

```
<doc>Neki tekst
     <a>element b.</a>     </doc>
```

Niska karaktera 'Neki tekst' zadovoljava token #PCDATA u modelu (#PCDATA | c), pa je ponovno pojavljivanje karakterskih podataka iza krajnje etikete , kao niske blanko i tabulator karaktera, nedozvoljeno.

2.2.2 Izuzeci

Model sadržaja elementa omogućava opis takve strukture dokumenta u kojoj se neposredni konstituenti svakog elementa mogu identifikovati, tj. da se elementi nekog tipa dokumenta mogu predstaviti hijerarhijskom strukturom u čijem je vrhu jedan predak (deklarisan u DOCTYPE deklaraciji) a na listovima je mnogo dece (koja uglavnom imaju samo #PCDATA sadržaj). Takva je, na primer, pojednostavljena struktura rada u zborniku koja je data u primeru iz dodatka A: rad u zborniku se sastoji iz beleške o autoru i teksta rada u proizvoljnom redosledu, rad se dalje satoji iz imena autora, naziva rada, apstrakta, i tako dalje. Ovako jednostavna struktura se uspešno može primeniti na veliki broj dokumenata. Ona, međutim ne pokriva slučaj manje više slobodno plutajućih elemenata koji se mogu pojaviti na skoro svakom hijerarhijskom nivou u strukturi. Primeri tavih elemenata su fusnote i anotacije.

Jedno rešenje za ovakve elemente je njihovo uključivanje u modele sadržaja na svim hijerarhijskim nivoima na kojima se mogu primeniti. U slučaju veoma kompleksnih tipova dokumenta ovakvo rešenje može biti neadekvatno. U SGML je stoga predviđeno da se model sadržaja može modifikovati pomoću *liste izuzetaka*. Postoje dve vrste izuzetaka: lista *uključivanja* koju čine elementi koji se mogu uključiti na bilo kome mestu u modelu sadržaja ili u bilo kom od njenih konstitutivnih elemenata i lista *isključivanja* koju čine elementi koji se isključuju iz modela.

Izuzeci se mogu navoditi samo za one elemente čiji je sadržaj opisan modelom. Ako postoje, navode se na kraju modela sadržaja u obliku dve opcione liste generičkih identifikatora elemenata koji modifikuju sadržaj elementa kao isključivanje, odnosno uključivanje. Ako se u bilo kojoj tački instancije dokumenta neki element može primeniti i kao uključivanje i kao isključivanje, on se tretira kao isključivanje.

Efekat uključivanja na model sadržaja može se opisati na sledeći način: Ako je Q neki generički identifikator ili grupa u modelu sadržaja, x njen indikator pojavljivanja a R i S su generički identifikatori elemenata koji se mogu primeniti kao uključivanja, onda se token sadržaja Qx interpretira kao:

```
(R | S)*, (Q, (R | S)*)x
```

Efekat isključivanja svodi se na onemogućavanje opcija koje bi inače bile dozvoljene. Isključivanje ne sme modifikovati sadržaj ni na koji drugi način te ga nije dozvoljeno primeniti na druge tokene osim na one koji su primenljivo uključivanje, imaju indikator sadržaja koji govori da su opcioni element (**opt** ili **rep**) ili su unutar grupe povezani grupnim konektorom **or**. Takode nije dozvoljeno menjati status obaveznosti tokena. Tako nije dozvoljeno isključiti sve članove nekog obaveznog modela kao što pokazuje sledeći primer:

```
<!ELEMENT doc    - -  ( #PCDATA & txt )      -(b, c, d) >
<!ELEMENT txt    - -  ( a & ( b | c | d ) )             >
```

Ove dve deklaracije govore da se u instanciji dokumenta u okviru elementa doc ne smeju pojaviti elementi b, c i d. To znači da se oni ne smeju pojaviti ni u okviru elementa txt u okviru koga je, međutim, obavezno pojavljivanje tokena sadržaja (b | c | d).

Elementi i karakterski podaci sadržaja moraju biti u saglasnosti sa tokenima grupe modela i izuzecima u sledećem redosledu prioriteta:

1. ponavljanje poslednjeg zadovoljenog tokena, ako on ima indikator ponavljanja **rep** ili **plus**;
2. neki drugi token iz grupe modela, eventualno modifikovan izuzecima isključenja;
3. token koji je primenljivo uključenje;

Na primer, u nekoj instanciji sledećeg elementa:

```
<!ELEMENT doc    - -  ( a+ | b )+                       >
```

sukcesivni elementi a će zadovoljiti ponavljanje tog tokena a ne ponavljanje grupe modela. Za saglasnost sadržaja sa tokenima grupe modela ilustrativan je i sledeći primer. Model sadržaja elementa doc

```
<!ELEMENT doc    - -  ( a+ & b )+                       >
```

formalno je ekvivalentan sa sledećim modelom

```
<!ELEMENT doc    - -  (( a+ , b ) | ( b, a+ ))+         >
```

Realizacija modela sadržaja (a+ , b) neposredno posle realizacije modela sadržaja (b, a+) je samo parazitska mogućnost zbog navedenog redosleda sravnjivanja tokena: posle pojavljivanja elementa b, sva sukcesivna pojavljivanja elementa a će zadovoljiti model sadržaja (b, a+). Stoga je ova deklaracija ekvivalentna sa sledećom deklaracijom:

```
<!ELEMENT doc    - -  ((( a+ , b )+ , ( b, a+ )*) |
                       ( b, a+)+ )                     >
```

2.2.3 Minimizacija etiketa

SGML predviđa četiri tehnike za smanjivanje broja i dužine etiketa obeležavanja u instanciji dokumenta. To su:

1. skraćivanje etiketa (SHORTTAG);
2. grupisanje etiketa (RANK);
3. automatsko prepoznavanje etiketa (DATATAG);
4. izostavljanje etiketa (OMITTAG).

Tehnike minimizacije su opciona svojstva SGML-a tako da je korišćenje svake pojedinačne tehnike neophodno eksplicitno omogućiti u okviru FEATURE klauzule u SGML deklaraciji (videti odeljak 2.8).

Skraćivanje etiketa

Skraćivanje etiketa se može obaviti izostavljanjem generičkog identifikatora iz početne etikete (<>) ili krajnje etikete elementa (</>), to jest korišćenjem *praznih etiketa*. Ova tehnika

se može primeniti samo na elemente iz baznog dokumenta. Za praznu početnu etiketu se, ukoliko je omogućeno i izostavljanje etiketa, pretpostavlja generički identifikator elementa koji je poslednji otvoren a ukoliko izostavljanje etiketa nije omogućeno, prazna početna etiketa se interpretira kao generički identifikator elementa koji je poslednji zatvoren. Za praznu početnu etiketu se pretpostavlja da ima praznu listu atributa (videti odeljak 2.3). Prazna krajnja etiketa se interpretira kao krajnja etiketa poslednjeg otvorenog elementa iz baznog dokumenta. Na primer, ako bi elementi doc i a bili deklarisani na sledeći način:

```
<!ELEMENT doc    - -  ( a+ )        >
<!ELEMENT a      - -  ( #PCDATA )   >
```

jedna instancija dokumenta koja ne koristi ni jednu tehniku za minimizaciju etiketa bi mogla biti enkordirana na sledeći način:

```
<doc><a>1. pojavljivanje elementa a.</a>
     <a>2. pojavljivanje elementa a.</a>
     <a>3. pojavljivanje elementa a.</a>
</doc>
```

Ako se primeni tehnika praznih etiketa ova ista instancija dokumenta bi mogla biti enkodirana na sledeći način:

```
<doc><a>1. pojavljivanje elementa a.</>
     <>2. pojavljivanje elementa a.</>
     <>3. pojavljivanje elementa a.</>
</doc>
```

Druga mogućnost za skraćivanje etiketa je izostavljanje graničnika za zatvaranje etikete (**tagc**, a to je > u referentnoj konkretnoj sintaksi), tj. korišćenje *nezatvorenih etiketa*. Nezatvorena početna etiketa može se primeniti samo ako iza nje sledi graničnik **stago** ili **etago** (tj. < odnosno </ u referentnoj konkretnoj sintaksi). Ova tehnika se primenjuje u slučajevima kada se u dokumentu uzastopno javlja više etiketa. Sve, osim poslednje, tada mogu biti nezatvorene. Ova tehnika ne nameće nikakva ograničenja u korišćenju atributa. Gornji primer bi korišćenjem samo ove tehnike minimizacije izgledao:

```
<doc<a>1. pojavljivanje elementa a.</a>
     <a>2. pojavljivanje elementa a.</a>
     <a>3. pojavljivanje elementa a.</a</doc>
```

Treću tehniku za skraćivanje etiketa predstavlja korišćenje *nultih krajnjih etiketa* pomoću kojih se kraj nekog elementa može specifikovati pomoću samo jednog karaktera. Taj karakter, koji se naziva **NET** (skraćenica od **null end-tag**), je u referentnoj konkretnoj sintaksi kosa crta /. Da bi se ova mogućnost koristila potrebno je prethodno kreirati početnu etiketu koja dozvoljava NET tako što se u njoj karakter za zatvaranje etikete **tagc** zameni NET karakterom. Korišćenjem ove tehnike gornji primer bi se mogao enkodirati i na sledeći način:

```
<doc><a/1. pojavljivanje elementa a./
     <a/2. pojavljivanje elementa a./
     <a/3. pojavljivanje elementa a./
</doc>
```

Ova tehnika je posebno korisna za elemente čiji je sadržaj deklarisan rezervisanim rečima RCDATA ili CDATA jer je za takve elemente prisustvo krajnje etikete obavezno. Kod njenog korišćenja treba biti oprezan, jer se NET karakter ne sme pojaviti unutar sadržaja elementa niti unutar nekog u njemu umetnutog elementa.

Nulte krajnje etikete mogu biti umetnute do istog nivoa do koga je dozvoljeno umetanje modela sadržaja elementa. Nulta krajnja etiketa uvek zatvara poslednji element koji je otvoren početnom etiketom koja dozvoljava NET.

Grupisanje etiketa — rang

Kada deklaracija elementa sadrži *rangnu osnovu* i *rangni sufiks* tada početna etiketa elementa u instanciji dokumenta može sadržati samo rangnu osnovu pod uslovom da je u nekoj prethodnoj tački u dokumentu početna etiketa elementa navedana kompletna: rangna osnova i rangni sufiks. Na primer, deklaracijama:

```
<!ELEMENT doc 0  - -  ( #PCDATA | a | b )*        >
<!ELEMENT doc 1  - -  ( #PCDATA | a | b )*        >
<!ELEMENT doc 2  - -  ( #PCDATA | a | b )*        >
```

deklarišu se elementi čija su generička imena redom doc0, doc1 i doc2. Kako su u pitanju rangirani elementi za koje je u deklaraciji odvojeno navedena rangna osnova doc a odvojeno rangni sufiks 0, 1 i 2 u instanciji dokumenta se početna etiketa može sastojati samo od rangne osnove. Da bi se ova mogućnost primenila mora biti utvrđen *tekući rang* kao vrednost rangnog sufiksa u poslednjoj početnoj etiketi elementa u kojoj je naveden potpun generički identifikator: rangna osnova doc i rangni sufiks. U tom slučaju, ako početna etiketa sadrži samo rangnu osnovu, puno generičko ime elementa dobija se kao konkatenacija rangne osnove i tekućeg ranga, kao što pokazuje primer sledeće instancije dokumenta koja koristi gornje deklaracije:

```
<doc1>Element doc, nivo 1
      <a>U njega umetnut element a.</a></doc1>
<doc>Element doc, isti nivo 1
     <b>U njega umetnut element b.</b></doc>
```

U ovom slučaju početna etiketa <doc> otvara element sa generičkim imenom doc1 koje se izvodi konkatencijom rangne osnove doc i tekućeg ranga 1 koji je utvrđen korišćenjem etikete <doc1> u kojoj je naveden puni generički identifikator.

Rangirani elementi se mogu grupisati u *rangnu grupu* ako dele isti model sadržaja. Rangni sufiks se tada primenjuje na sva imena u rangnoj grupi, kao što pokazuje sledeći primer:

```
<!ELEMENT (a | b) 1  - -  ( #PCDATA | a1 | a2 | b1 | b2 | c)* >
<!ELEMENT (a | b) 2  - -  ( #PCDATA | a1 | a2 | b1 | b2 | c)* >
```

Nivoi elemenata sa rangnom osnovom a i b su međusobno povezani. U sledećoj instanciji dokumenta:

```
<a1>Element a, nivo 1
      <c>Element c.</c>
      <b>U njega umetnut element b, istog nivoa 1.</b></a1>
<a2>Element a, nivo 2
     <b>U njega umetnut element b, istog nivoa 2.</b></a2>
<a>Element a, istog nivoa 2 .....
```

Unutar rangne grupe, koristi se isti tekući rang za sve elemente iz grupe za koje u početnoj etiketi nije naveden rangni sufiks. Tako se prvo pojavljivanje etikete <b> tumači kao otvaranje elementa b1 jer je u tom trenutku tekući rang 1 utvrđen etiketom <a1> elementa a1 koji pripada istoj rangnoj grupi kao i b1.

Automatsko prepoznavanje etiketa

Etikete se mogu automatski prepoznati kada se određeni karakteri sa nekom pravilnošću pojavljuju u elementu. U tom slučaju ti karakteri predstavljaju i deo podataka i krajnju etiketu elementa. Da bi se to postiglo model sadržaja elementa treba deklarisati na sledeći način:

```
<!ELEMENT doc   - -  ( [ a, ", ", " "], b ) >
<!ELEMENT a     O O  ( #PCDATA )            >
<!ELEMENT b     O O  ( #PCDATA )            >
```

Model sadržaja elementa doc je seq grupa koja govori da se on sastoji od elementa a iza koga sledi element b. Međutim, element a je u modelu sadržaja naveden u okviru *grupe podatkovne etikete* koja je navedena između graničnika **dtgo** za otvaranje grupe podatkovne etikete i graničnika **dtgc** za zatvaranje grupe podatkovne etikete, a to su [i] u referentnoj konkretnoj sintaksi. Ova grupa se sastoji od tri člana:

- generički identifikator elementa čija se krajnja etiketa automatski prepoznaje. U gornjem primeru to je a.
- jedan ili, eventualno, grupa više *šablona podatkovne etikete* koji predstavlja nisku karaktera čije pojavljivanje označava kraj pripadajućeg elementa. Šablon nije deo sadržaja ni elementa a ni elementa b već je deo raščlanjivih karakterskih podataka koji se implicitno pretpostavljaju. U gornjem primeru naveden je samo jedan šablon a to je niska ", " što znači da zarez iza koga sledi blanko označava kraj podataka elementa a;
- *šablon dopunjavanja podatkovne etikete* koji predstavlja nisku karaktera koja eventualno sledi iza šablona podatkovne etikete i zajedno sa njim formira podatkovnu etiketu. U gornjem primeru to je niska " " što znači da više blankova može da sledi iza obaveznog zareza i blanka i da oni zajedno čine podatkovnu etiketu.

Neka pojavljivanja elementa doc u instanciji dokumenta mogla bi biti:

```
<doc>Element a, Element b</doc>
<doc>Element a,   Element b</doc>
```

U prvom primeru podatkovna etiketa je šablon ", " dok je u drugom primeru podatkovna etiketa šablon dopunjen šablonom dopunjavanja " ".

Treba napomenuti da su u ovom primeru elementi a i b deklarisani tako da omogućavaju izostavljanje i početne i krajnje etikete o čemu će biti govora u sledećoj tački. Ako se ne bi primenila ni jedna od tehnika minimizacije, deklaracije elemenata doc, a i b bile bi:

```
<!ELEMENT doc   - -  ( a, #PCDATA, b )      >
<!ELEMENT a     - -  ( #PCDATA )            >
<!ELEMENT b     - -  ( #PCDATA )            >
```

a primeri pojavljivanja elementa doc u instanciji dokumenta bili bi kodirani na sledeći način:

```
<doc><a>Element a</a>, <b>Element b </b></doc>
<doc><a>Opet a</a>,   <b>Opet b</b></doc>
```

Izostavljanje etiketa

Izostavljanje etiketa je tehnika minimizacije koja dozvoljava da se početna ili krajnja etiketa potpuno izostave prilikom kodiranja dokumenta. Ona se može primeniti samo ukoliko pro-

gram koji kodirani tekst obrađuje može na osnovu definicije tipa dokumenta nedvosmisleno da zaključi gde se koja izostavljena etiketa nalazi.

Ova tehnika, kao i ostale tehnike minimizacije, mora biti eksplicitno omogućena u okviru SGML deklaracije. U tom slučaju deklaracija svakog elementa u okviru DTD-a mora sadržati dva *parametra minimizacije izostavljanjem etiketa* koja se u deklaraciji navode iza generičkog identifikatora elementa (ili grupe generičkih identifikatora). Prvi parametar se odnosi na izostavljanje početne etikete a drugi na izostavljanje krajnje etikete. Ovaj parametar može imati dve karakterske vrednosti. Vrednost - (crtica) znači da izostavljanje etikete nije dozvoljeno dok vrednost O znači da je izostavljanje etikete dozvoljeno.

Početna etiketa se može izostaviti iz dokumenta ako je u pitanju element čije je pojavljivanje obavezno u odgovarajućem kontekstu a pojavljivanje bilo kog drugog elementa je u tom kontekstu opciono. Izuzetak su elementi koji imaju prazan sadržaj (deklarisani kao `EMPTY`) ili deklarisani sadržaj (deklarisan kao `RCDATA` ili `CDATA`) i elementi koji imaju obavezan atribut (videti odeljak 2.3). Ovo, praktično, znači da se ne mogu izostavljati etikete elemenata čije je pojavljivanje u modelu sadržaja kvalifikovano indikatorom pojavljivanja jer on onemogućava da se odredi koji je to element koji se sledeći mora pojaviti.

Pravila koja regulišu izostavljanje krajnjih etiketa su manje restriktivna. Krajnja etiketa se za neki element može izostaviti ako iza nje sledi krajnja etiketa nekog drugog u tom trenutku otvorenog elementa ili ako iza nje sledi element ili SGML karakter čije pojavljivanje nije dozvoljeno u njegovom sadržaju.

Kako se omogućava izostavljanje etiketa i kako se i kada ono može primeniti ilustrovaćemo na primeru sledećih deklaracija.

```
<!ELEMENT doc          - -  ( ( a & b ), c, d? )   >
<!ELEMENT (a,b,c,d)    O O  ( #PCDATA )            >
```

Etikete elementa `doc` se ne smeju izostavljati dok je izostavljanje i početne i krajnje etikete elemenata `a`, `b`, `c` i `d` dozvoljeno. To, međutim, ne znači da je njihovo izostavljanje zaista i moguće u svakom kontekstu. Na primer, u sadržaju elementa `doc` se početne etikete elemenata `a` i `b` ne mogu izostavljati jer model sadržaja tog elementa počinje **and** grupom koja govori da njegov sadržaj počinje elementima `a` i `b` koji se mogu pojaviti u proizvoljnom redosledu. Njihove krajnje etikete se, međutim, mogu izostaviti jer ako element `doc` počinje, na primer, elementom `a` iza njega obavezno sledi `b` a iza ovog `c`.

U okviru istog elementa početna etiketa elementa `c` može se izostaviti, jer se on obavezno pojavljuje iza elemenata `a` i `b`. Može se izostaviti i njegova krajnja etiketa jer se o njenom prisustvu može zaključiti bilo na osnovu početne etikete elementa `d`, koji nije dozvoljen sadržaj elementa `c`, bilo na osnovu krajnje etikete elementa `doc` koji je tekući otvoreni element.

```
<doc2><b>deo B</b>
      <a>deo A</a>
      deo C
      <d>deo D
</doc2>
```

Izostavljanje početne etiketa ne mora, međutim, biti dozvoljeno u svim pojavama elementa `doc`. Ako je izostavljena krajnja etiketa elementa kojim se završava sravnjivanje **and** grupe (`a` ili `b`), početna etiketa elementa `c` je obavezna.

```
<doc2><a>deo A
      <b>deo B
      <c>deo C
      <d>deo D
</doc2>
```

U kontekstu elementa doc, početna etiketa elementa d se ne sme izostaviti jer je njegovo pojavljivanje modifikovano indikatorom pojavljivanja.

Pojavljivanje elementa koje je nedozvoljeno u određenom kontekstu jer je element isključen ima isti efekat kao i pojavljivanje elementa za koji ne postoji odgovarajući token u grupi modela. To znači da se na osnovu pojavljivanja takvog elementa zaključuje o prisustvu na tom mestu krajnje etikete jednog ili više ugnježdenih elemenata iz čijeg je sadržaja element koji se otvara isključen. Na primer, sledeće deklaracije govore da je iz sadržaja elementa a isključen element c što praktično znači da je opciono pojavljivanje tog elementa u sadržaju elementa b onemogućeno, ali samo onda kada se element b pojavljuje u okviru elementa a.

```
<!ELEMENT doc          - -  ( a , b )          >
<!ELEMENT a            O O  ( b )*      -(c)   >
<!ELEMENT b            O O  ( c | d )          >
<!ELEMENT (c,d)        - O  ( #PCDATA )        >
```

Neka imamo sledeće pojavljivanje elementa doc u instanciji dokumenta:

```
<doc>
      <b><d>deo D, 1<b><d>deo B, 2
      <c>deo C
</doc>
```

Ukoliko bi sve početne i krajnje etikete bile prisutne, ista instancija dokumenta bila bi kodirana na sledeći način:

```
<doc>
   <a><b><d>deo D, 1</d></b>
      <b><d>deo B, 2</d></b></a>
   <b><c>deo C</c></b>
</doc>
```

Iz ovog primera se vidi da element c koji je isključen iz elementa a i svih u njega ugnježdenih elemenata zatvara redom elemente d, b i a. Unutar elementa doc iza elementa a obavezno sledi element b čiji je sadržaj, koji ovog puta nije isključen, element c.

2.2.4 Nedvosmislenost

Model sadržaja ne sme biti dvosmislen, što znači da jedan element ili niska karaktera u instanciji dokumenta mogu da zadovolje bez korišćenja preduvidnog simbola samo jedan token sadržaja. Dvosmislenost u modelu sadržaja može nastupiti, osim kao rezultat korišćenja konektora i indikatora pojavljivanja i njihovog kombinovanja, i u mešovitim modelima sadržaja. Primer dvosmislenosti mešovitog modela sadržaja je dat na kraju pododeljka 2.2.1 a ovde će biti dati primeri dvosmislenosti koji potiču od drugih uzroka. Dvosmislenost modela sadržaja elementa doc iz sledećeg primera potiče usled neodgovarajućeg korišćenja konektora **seq** i indikatora pojavljivanja **opt**.

```
<!ELEMENT doc - - ( ( a , c , b? ) , b )      >
```

U sledećoj instanciji dokumenta koja koristi element doc ne može se zaključiti da li element b zadovoljava token iz grupe (a , c , b?) ili, s obzirom na opciono pojavljivanje tokena b u ovoj grupi, token iz grupe ((...) , b).

```
<doc>
   <a>deo A</a>
   <c>deo C</c>
   <b>deo B</b>
</doc>
```

Kada bi se koristio preduvidni simbol, a to je u ovom slučaju krajnja etiketa </doc>, moglo bi se ispravno zaključiti da u ovom slučaju element b zadovoljava token iz grupe ((...) , b). Modeli sadržaja koji se oslanjaju na ovakvo korišćenje preduvidnih simbola u SGML jeziku nisu dozvoljeni.

Model sadržaja elementa doc iz sledećeg primera je takođe dvosmislen, usled neadekvatnog korišćenja konektora **or**.

```
<!ELEMENT doc - - ( ( a , c ) | a )          >
```

Na osnovu pojavljivanja početne etikete elementa a unutar elementa doc ne može se bez korišćenja preduvidnog simbola, etikete <c> ili </doc>, zaključiti da li element a zadovoljava token iz grupe (a , c) ili token iz grupe ((...) | a). Ovde treba primetiti da sledeći model sadržaja, premda ekvivalentan prethodnom jer se oba svode na isti regularni izraz, više nije dvosmislen:

```
<!ELEMENT doc - - ( a , c? )                 >
```

Dvosmislenost modela sadržaja koji koriste **and** konektor potiče iz istih razloga iz kojih su dvosmisleni modeli koji koriste **seq** i **or** konektor, kao što pokazuje primer sledećih dvosmislenih modela, koji su dobijeni zamenom u gornjim primerima **seq** odnosno **or** konektora **and** konektorom.

```
<!ELEMENT doc - - ( ( a , c , b? ) & b )     >
<!ELEMENT doc - - ( ( a , c ) & a )          >
```

O dvosmislenosti modela sadržaja biće više reči u pododeljku 4.4.3 sledeće glave kada se bude govorilo o konstrukciji SGML raščlanjivača.

2.3 Atributi

Atributi predstavljaju informaciju koja opisuje određeno pojavljivanje nekog elementa a koja sama nije deo sadržaja tog elementa. Za elemente koji imaju definisane atribute, vrednosti se pridružuju atributima u instanciji dokumenta u obliku para *atribut = vrednost* unutar početne etikete elementa. Oni se obično koriste kada je potrebno:

- identifikovati status elementa, na primer `<glava status = 'radna verzija'>`;
- odrediti način formatiranja elementa, na primer `<termin format = 'italik'>`;
- identifikovati tekst koji treba sistemski generisati, na primer `<lista numeracija = 'rimski broj'>`;
- definisati izvor ili veličinu eksterno smeštenih podataka, na primer `<slika x = '120mm' y = '90mm'>`;
- jednoznačno identifikovati određeno pojavljivanje elementa, na primer `<slika id = 'sl.23'>`;

- referisati određeno pojavljivanje elementa, na primer `<refslika idref = 'sl.23'>`;
- istaći neko svojstvo elementa koje može biti od značaja za kasniju obradu, na primer `<pasus jezik = 'engleski'>`.

Atributi se definišu u okviru definicije tipa dokumenta i njihova definicija je uvek vezana za određeni element ili notaciju (videti odeljak 2.5). Ona ima sledeći oblik:

Mdo "ATTLIST" *pridruženi_element_notacija*
ime_atributa deklarisana_vrednost pretpostavljena_vrednost
ime_atributa deklarisana_vrednost pretpostavljena_vrednost
..... Mdc

gde su terminali, odnosno tokeni, graničnici **Mdo** i **Mdc** i ključna reč "ATTLIST" dok su *pridruženi_element_notacija*, *ime_atributa*, *deklarisana_vrednost* i *pretpostavljena_vrednost* sintaksičke promenljive. *pridruženi_element_notacija* se realizuje kao jedno ili više generičkih imena elemenata ili notacija za koje se atributi definišu. Atributi se mogu pridružiti nekom elementu ili notaciji samo jednom u okviru jednog podskupa definicija tipa dokumenta.

ime_atributa je jedinstveno ime određenog atributa. Treba primetiti da je to ime jedinstveno samo u okviru jedne definicije dok atribut sa istim imenom može biti pridružen većem broju elemenata ili notacija i ti atributi mogu, šta više, imati različite deklarisane vrednosti. *deklarisana_vrednost* predstavlja skup dozvoljenih vrednosti atributa. Taj skup definiše koje se vrednosti mogu dodeliti određenom atributu u početnoj etiketi pridruženog elementa u instanciji dokumenta. On se može zadati bilo kao zagrađena lista dozvoljenih vrednosti atributa bilo kao ključna reč koja određuje koja vrsta podatka može biti vrednost atributa. Spisak ključnih reči i njihovo značenje je dat u sledećoj tabeli.

Ključna reč	Skup vrednosti
CDATA	Karakterski podaci koji uključuju sve dozvoljene SGML karaktere
ENTITY	Ime nekog deklarisanog poddokumenta ili opšteg entiteta
ID	Jedinstveni identifikator elementa
IDREF	Referenca jedinstvenog identifikatora elementa
IDREFS	Lista referenci jedinstvenih identifikatora
NAME	Važeće SGML ime
NAMES	Lista važećih SGML imena
NMTOKEN	Važeći imenski token
NMTOKENS	Lista važećih imenskih tokena
NOTATION	Jedno ili više imena notacija koja se zadaju kao zagrađena lista koja pobliže određuje ovu ključnu reč
NUMBER	Broj
NUMBERS	Lista brojeva
NUTOKEN	Brojčani token
NUTOKENS	Lista brojčanih tokena

Pretpostavljena_vrednost je vrednost koja se dodeljuje atributu ako u instanciji dokumenta nikakva vrednost tom atributu nije dodeljena u početnoj etiketi elementa, odnosno ako je ta etiketa izostavljena. Pretpostavljena vrednost se zadaje bilo kao određena vrednost atributa, koja u tom slučaju mora biti u saglasnosti sa deklarisanom vrednošću, bilo kao jedna

od ključnih reči. Ključnim rečima u ovom slučaju prethodi rni karakter (# u referentnoj konkretnoj sintaksi) da bi se izbegla eventualna nedoumica radi li se o ključnoj reči ili o vrednosti atributa. Spisak ključnih reči i njihovo značenje je dat u sledećoj tabeli.

Ključna reč	Objašnjenje
#REQUIRED	Atributu mora biti dodeljena vrednost u početnoj etiketi elementa
#CURRENT	Ako atributu nije dodeljena vrednost, pretpostavlja se poslednja vrednost dodeljena tom atributu
#IMPLIED	Ako atributu nije dodeljena vrednost, aplikacija može da je dodeli
#CONREF	Atribut se koristi za referisanje; vrednost može biti referenca u obliku teksta ili u obliku jedinstvenog identifikatora
#FIXED	Sledi zadatu pretpostavljenu vrednost i znači da je vrednost atributa nepromenljiva

Razgraničavanje između korišćenja elemenata i atributa za opis podataka nije uvek moguće jednoznačno obaviti. Na primer, možemo deklarisati element Ime i njegove atribute na sledeći način:

```
<!ELEMENT Ime          - -  ( #PCDATA )                          >
<!ATTLIST Ime
          Tip    ( Grad, Zemlja, Narod, Osoba )    #REQUIRED >
```

Element Ime ima obavezan atribut Tip koji govori o kakvoj se vrsti imena radi. Alternativno je moguće deklarisati četiri elementa na sledeći način:

```
<!ELEMENT Grad         - -  ( #PCDATA )                          >
<!ELEMENT Zemlja       - -  ( #PCDATA )                          >
<!ELEMENT Narod        - -  ( #PCDATA )                          >
<!ELEMENT Osoba        - -  ( #PCDATA )                          >
```

Druga mogućnost ima prednosti ukoliko svaki od ova četiri elementa ima svoj poseban skup atributa. Na primer, elementu Osoba bi se mogao pridružiti atribut Norm čija je vrednost puno ime osobe koje u dokumentu ne mora biti dato. Sa sledećom deklaracijom,

```
<!ELEMENT Grad         - -  ( #PCDATA )                          >
<!ELEMENT Osoba        - -  ( #PCDATA )                          >
<!ATTLIST Osoba
          Norm         CDATA                  #IMPLIED          >
```

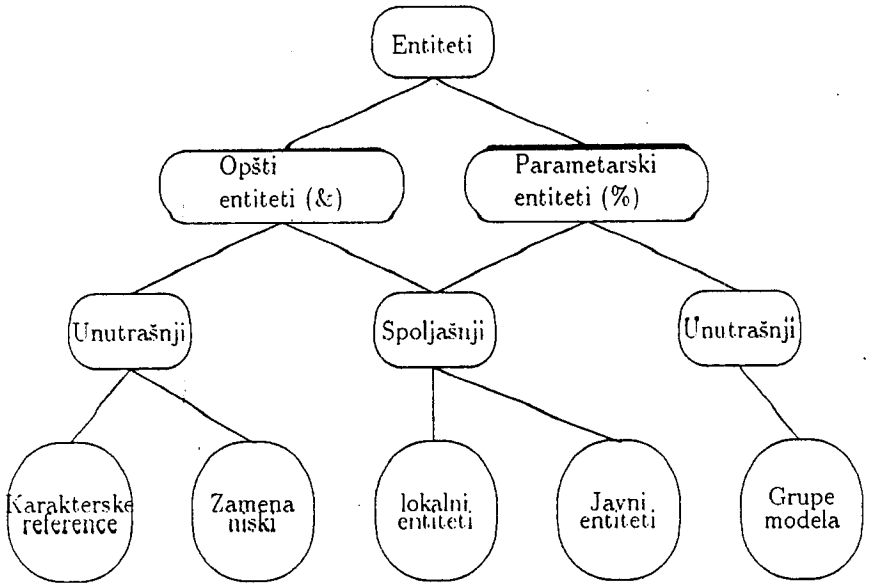
moguće je na sledeći način kodirati poslovicu [72]:

```
Inat je <Osoba Norm='Kraljević, Marko'>Marka</Osoba>
u <Grad>Stambol</Grad> zajmio.
```

Strukturiranost podataka takođe može uticati na izbor elemenata ili atributa za opis podataka. Neka je deo dokumenta koji se opisuje pismo — poslovno ili privatno. Ako se struktura privatnog pisma razlikuje od strukture poslovnog pisma, treba deklarisati različite elemente:

```
<!ELEMENT priv.Pismo  - -  ( Datum?, Kome?, Tekst )                  >
<!ELEMENT posl.Pismo  - -  ( Datum, od.Koga, Kome, o.Cemu, Tekst ) >
```

Ako pak pismo uvek ima istu strukturu, vrsta pisma se može odrediti na osnovu atributa Vrsta:



Slika 2.1: Različite vrste entiteta

```
<!ELEMENT Pismo  - -  ( Datum?, od.Koga?, Kome?, o.Cemu?, Tekst ) >
<!ATTLIST Pismo
          Vrsta       ( privatno, poslovno )      #REQUIRED       >
```

Specifična obrada dokumenta može na osnovu vrednosti atributa odrediti, na primer, različito formatiranje poslovnog odnosno privatnog pisma. Sledeći primer je, na primer, HTML DTD (videti 3.3.2) u kome je težnja da broj elemenata bude što manji ali da se omogući njihova kategorizacija pomoću atributa. Tako pasus može otpočeti etiketom `<P CLASS=STANZA.COUPLET>` ako je pitanju stih u strofi pesme. S druge strane TEI DTD (videti 3.3.1) definiše pet ugnježdenih nivoa grupa redova `lg1`, `lg2`, `lg3`, `lg4` i `lg5` za specifikovanje stihova, strofa i ostalih delova pesama.

U [134] je opisano kako se atributi mogu zameniti ugnježdenim ili obuhvatnim elementima. Druge primene atributa su ilustrovane u dodatku A.

2.4 Entiteti

U SGML-u postoji jednostavan i fleksibilan način za kodiranje i imenovanje proizvoljnih delova sadržaja dokumenta na prenosiv način. Imenovani delovi obeleženog dokumenta nazivaju se *entiteti*. Entiteti mogu biti niske karaktera, i u tom slučaju se nazivaju *unutrašnjim entitetima*, ili cele datoteke, kada se nazivaju *spoljašnjim entitetima*. Za njihovo uključivanje u dokument koriste se *reference entiteta*. Na slici 2.1 prikazane su različite vrste entiteta.

Entiteti se obično koriste za:

- uvođenje skraćenica za duge niske karaktera koje se u dokumentu često koriste, da bi se olakšalo kucanje;
- unošenje karaktera za koje ne postoji taster na tastaturi;
- unošenje u dokument delova koji su u spoljašnjim datotekama, što omogućava da se velika dokumenta i dokumenta sa raznovrsnim sadržajem razbiju u više manjih radi

lakšeg rukovanja;

- uključivanje u dokument dinamički izračunatih objekata, kakav je, na primer, sistemski datum;
- okupljanje i imenovanje listi atributa i grupa modela radi jednostavnijeg i preciznijeg opisa strukture dokumenta.

2.4.1 Karakterski entiteti

Karakterski entiteti su specijalna vrsta entiteta koja dozvoljava referisanje svih valjanih SGML karaktera koji se mogu pojaviti u dokumentu, bez obzira da li oni mogu biti uneti pritiskom na taster tastature ili ne. Njihova specifičnost u odnosu na druge vrste entiteta je da se oni ne deklarišu.

Za referisanje karakterskih entiteta koristi se graničnik otvaranja karakterske reference, **cro**, koji je u referentnoj konkretnoj sintaksi definisan kao **&#**. Iza graničnika može da sledi broj koji predstavlja kod ili poziciju karaktera u kolacionoj sekvenciji, ili ime funkcije određenog karaktera koje je tom karakteru pridruženo u konkretnoj sintaksi koju dokument koristi. Referentnom konkretnom sintaksom definisana su imena **RE**, **RS** i **SPACE** koja su pridružena redom karakterima čije su funkcije kraj sloga, početak sloga i razmak. Referenca karakterskog, i svih drugih, entiteta se završava graničnikom za zatvaranje reference entiteta, **refc**, koji je definisan kao ; u referentnoj konkretnoj sintaksi, kodom za kraj sloga ili bilo kojim drugim karakterom koji ne može biti deo imena entiteta.

Tipične karakterske reference imaju oblik:

```
&#13;       &#RE;  {ili, ekvivalentno}
&#13        &#RE
```

Ako se dokument obrađuje na računaru koji koristi ASCII kodnu sekvenciju, obe ove karakterske reference odnose se na karaker koji označava kraj sloga. Ako pak računar ne koristi tu kodnu sekvenciju samo druga referenca se odnosi na karaker koji označava kraj sloga.

To nije jedina razlika između korišćenja broja i imena u karakterskoj referenci: brojčane karakterske reference se uvek pamte u tom obliku, što znači da ih SGML-zasnovan program za obradu ne zamenjuje njihovim internim kodom, kao što je slučaj sa imenskim karakterskim referencama. Posledica toga je da brojčane karakterske reference ne mogu, na primer, biti prepoznate kao graničnici od strane SGML-zasnovanog programa.

2.4.2 Opšti entiteti

Opšti entiteti deklarišu se u okviru definicije tipa dokumenta na sledeći način:

Mdo "ENTITY" *ime_entiteta tekst_entiteta* **Mdc**

gde su terminali, odnosno tokeni, graničnici **Mdo** i **Mdc** i ključna reč "ENTITY" dok su *ime_entiteta* i *tekst_entiteta* sintaksičke promenljive. *ime_entiteta* je jednoznačno ime entiteta koje se koristi prilikom njegovog referisanja iza graničnika **ero** za otvaranje reference entiteta koji je u referentnoj konkretnoj sintaksi definisan kao **&**. Višestruke deklaracije istog entiteta su dozvoljene pri čemu se sve, osim prve, ignorišu.

tekst_entiteta se može realizovati na četiri različita načina:

- parametarski literal[2];

[2] Vrste literala su opisane u dodatku C. O njima će biti više reči u odeljku 4.1.

- podatkovni tekst;
- zagrađeni tekst;
- specifikacija spoljašnjeg entiteta;

Ako je parametarski literal tekst zamene entiteta, onda interpretirani parametarski literal zamenjuje referencu opšteg entiteta u dokumentu. Na primer, ako entitet Zamena deklarišemo na sledeći način:

```
<!ENTITY  Zamena  "Kraj reda &#RE;&#RS; Novi red"  >
```

njegova referenca u dokumentu:

```
... Ovako se zatvara jedan red &Zamena; i otvara novi ...
```

biće interpretirana kao:

```
... Ovako se zatvara jedan red Kraj reda
Novi red i otvara novi ...
```

Ovaj primer ilustruje da se literal koji predstavlja tekst zamene entiteta interpretira prilikom referisanja entiteta, što znači da se interpretiraju reference karakterskih i parametarskih entiteta koje su sadržane u tekstu zamene. Ovakvo ugnježđeno referisanje entiteta može ići do dubine koja je propisana odgovarajućom veličinom (ENTLVL) konkretne sintakse koja se koristi. Treba napomenuti da referisanje samog entiteta koji se deklariše unutar teksta zamene nije dozvoljeno.

Ako je parametarski literal u deklaraciji modifikovan ključnom reči CDATA, SDATA ili PI, onda to znači da se interpretirani parametarski literal tretira kao karakterski podaci, bez obzira na kontekst u kome se referenca pojavljuje. Na primer, deklaracija:

```
<!ENTITY  pasus  CDATA  "<pasus>"  >
```

znači da će referenca &pasus; uvek biti tretirana kao niska <pasus> a ne kao početna etiketa za element pasus. Ključna reč SDATA znači da parametarski literal sadrži sistemski zavisne podatke a ključna reč PI da sadrži instrukcije za obradu.

Ako je parametarski literal u deklaraciji modifikovan ključnom reči STARTTAG, ENDTAG, MS ili MD onda to znači da se interpretirani parametarski literal zagrađuje graničnicima početne etikete, krajnje etikete, označenog odeljka odnosno deklaracije oznaka. Na primer, deklaracija:

```
<!ENTITY  formula  STARTTAG  "form tip=TeX"  >
```

znači da će referenca &formula; biti zamenjena sa <form tip=TeX>. Alternativna bi bila sledeća deklaracija:

```
<!ENTITY  formula  "<form tip=TeX>"  >
```

Prednost prve deklaracije u odnosu na drugu je nezavisnost od karaktera, odnosno niski karaktera, koji se u konkretnoj sintaksi koriste za određene graničnike.

Tekst entiteta može biti specifikacija spoljašnjeg etniteta. Ona se sastoji od *spoljašnjeg identifikatora* koji ukazuje na tip entiteta koji se deklariše i, eventualno, na izvor podataka koje treba uključiti u dokument. Spoljašnji entitet može, pre svega, biti specifičan za određeni sistem, dakle lokalan, a može biti poznat na više sistema koji poznaju SGML, dakle javan. Za specifikaciju lokalnog entiteta koristi se u deklaraciji ključna reč SYSTEM. Na primer,

```
<!ENTITY  datoteka.txt SYSTEM >
```

datoteka.txt je ime spoljašnjeg entiteta koji je sistemski zavisan. U ovoj deklaraciji se pretpostavlja da sistem prepoznaje ime entiteta kao referencu datoteke koju treba uključiti u

dokument. Ako to nije slučaj, uz ključnu reč SYSTEM može se navesti i *sistemski identifikator* koji na jedinstven način identifikuje izvor entiteta.

```
<!ENTITY  vuk-a SYSTEM "c:\Nolit\Vuk\e-tekst\vuk-a.txt" >
```

Sistemski identifikator je literal koji može sadržati sve valjane SGML karaktere.

Javni spoljašnji entiteti sadrže deklaracije, tekst ili drugu vrstu podataka koji su nastali sa ciljem da ih koriste više SGML zasnovanih sistema. Za specifikaciju javnog entiteta koriste se ključna reč PUBLIC i *javni identifikator* koji se realizuje kao minimalni literal.

```
<!ENTITY  ime  PUBLIC "javni identifikator" >
```

Osim podele na lokalne i javne, spoljašnji entiteti se razlikuju i prema tome kakvu vrstu podataka sadrže. Tako postoje:

- Entiteti koji sadrže SGML tekst koji u deklaraciji ne traže bliže određenje;
- Entiteti koji sadrže SGML poddokumente (videti odeljak 2.7.2), koji se u deklaraciji pobliže određuju korišćenjem ključne reči SUBDOC:

  ```
  <!ENTITY  izvestaj2 SYSTEM "c:\CV\Izvestaji\izv2txt" SUBDOC >
  ```

- Entiteti koji ne sadrže tekst ili sadrže tekst koji nije SGML označen, u deklaraciji se pobliže određuju ključnim rečima CDATA, SDATA ili NDATA da bi se označili karakterski podaci, sistemski specifični podaci i podaci zapisani specijalnom notacijom koji ne sadrže valjane SGML karaktere. Iza ovih ključnih reči mora se navesti *ime notacije* koje identifikuje korišćeni tip kodiranja a iza njega, eventualno, i *specifikacija podatkovnih atributa* (videti odeljak 2.5). Na primer, u deklaraciji:

  ```
  <!NOTATION Tex-teka SYSTEM "emtex"                    >
  <!ENTITY   glava2   SYSTEM "c:\CV\Tekstovi\gl2.tex"
                      CDATA "Tex-teka"                  >
  ```

  je rečeno da entitet glava2 koji sadrži tekst zapisan u TEX notaciji sadrži valjane SGML karaktere ali nije SGML kodiran. Ključna reč CDATA određuje da su karakterski podaci sadržaj datoteke dok ime notacije koje sledi iza nje pobliže određuje korišćenu notaciju. Ime notacije mora biti deklarisano u istoj definiciji tipa dokumenta (videti odeljak 2.5).

2.4.3 Parametarski entiteti

Parametarski entiteti predstavljaju specijalnu vrstu entiteta koja se koristi samo u okviru deklaracija SGML oznaka. Oni se deklarišu na sledeći način:

Mdo "ENTITY" **pero** *ime_entiteta tekst_entiteta* **Mdc**

Jedina razlika u odnosu na deklaraciju opšteg entiteta je korišćenje graničnika **pero** ispred imena entiteta. Ovaj graničnik je u referentnoj konkretnoj sintaksi definisan kao %. Treba napomenuti da je razmak između ovog graničnika i imena entiteta obavezan.

Parametarski entiteti se obično koriste za imenovanje grupe modela koja se koristi u više deklaracija elemenata ili za imenovanje atributa koji su zajednički za više elemenata. Tako se, na primer, može deklarisati entitet fraza:

```
<!ENTITY  % fraza   "( #PCDATA ) | istaknuto | termin " >
```

čiji je tekst zamene grupa modela koja se može koristiti u deklarisanju velikog broja elemenata: `fusnota`, `paragraf`, `clan.liste` itd. što je ilustrovano u primeru iz dodatka A. Slično se može deklarisati i entitet `opsti.atributi`:

```
<!ENTITY  % opsti.atributi      "
          jezik                 CDATA          #IMPLIED
          izgled                CDATA          #IMPLIED
          identifikacija        ID             #IMPLIED   "   >
```

Ovaj entitet se može koristiti u definiciji atributa svih elemenata čime bi se omogućilo da opšti atributi `jezik`, `izgled` i `identifikacija` budu zajednički za sve elemente. Ta mogućnost je takođe ilustrovana u primeru iz dodatka A.

Kao i opšti entiteti, i parametarski entiteti mogu biti unutrašnji i spoljašnji a spoljašnji — lokalni i javni. Postoje ipak neka ograničenja. Ako je tekst entiteta podatkovni tekst, onda se ključne reči `CDATA` i `SDATA` za parametarske entitete ne smeju koristiti. Takodje, spoljašnji identifikator se za spoljašnje parametarske entitete ne određuje bliže tipom entiteta.

U upotrebi su često spoljašnji javni parametarski entiteti. Na primer,

```
<!ENTITY  % ISOcyr2 PUBLIC
    "ISO 8879-1986//ENTITIES Non-Russian Cyrillic//EN" >
```

`ISOcyr2` je ime javnog skupa entiteta koji je sastavila međunarodna organizacija za standardizaciju (ISO), kao podršku SGML standardu. Tekst entiteta je *formalni javni identifikator*[3] koji identifikuje kreatora skupa, opisuje sadržaj skupa i jezik na kome je napisan. U gornjem primeru kreator skupa je ISO, skup sadrži entitete koji opisuju ćirilična slova koja nisu u repertoaru ruske ćirilice a za opis se koristi engleski jezik. Da bi se entiteti iz ovog skupa mogli koristiti treba ih uključiti u definiciju tipa dokumenta referencom entiteta `%ISOcyr2;`. Na taj način postaje dostupan skup entiteta koji opisuju neruska ćirilična slova od kojih su ovde navedena samo neka:

```
<!ENTITY  djcy   SDATA "[djcy  ]" --=small dje, Serbian-->
<!ENTITY  DJcy   SDATA "[DJcy  ]" --=small DJE, Serbian-->
<!ENTITY  gjcy   SDATA "[gjcy  ]" --=small gje, Macedonian-->
```

Ovi entiteti su deklarisani kao opšti entiteti i kao takvi se mogu referisati u tekstu dokumenta gdegod se pojavi potreba za neruskom ćirilicom. Na primer,

```
... Singing the very famous song
    <phr lang=Serbian>&DJ;ur&dj;ev dan</phr>to the ...
```

Kao što se vidi, svi ovi entiteti su deklarisani kao sistemski zavisni što znači da će tekst zamene zavisiti od konkretnog računara, kodne šeme ili programa. Na primer, moguće zamene za entitet `djcy` su:

```
<!ENTITY  djcy   SDATA "&#128" --=small dje, Serbian-->
<!ENTITY  djcy   SDATA "\dj " --=small dje, Serbian-->
```

Prva deklaracija bi omogućila zamenu reference entiteta `&djcy;` kodom ćiriličnog slova *đ* u ISO 8859-5 kodnoj tablici dok bi druga omogućila zamenu odgovarajućom TEX komandom.

[3] Sintaksa i značenje formalnih javnih identifikatora je detaljno opisano u tekstu standarda [57].

2.5 Notacije

U dokument često želimo da uključimo podatke koje program za SGML obradu ne treba da raščlanjuje jer su oni zapisani korišćenjem neke drukčije notacije. Ti podaci mogu biti matematičke formule, grafikoni, formatiran tekst, digitalizovane slike, muzički ili video materijal. Ovakvi podaci se obično opisuju notacijama koje se u okviru definicije tipa dokumenta deklarišu na sledeći način:

Mdo "NOTATION" *ime_notacije identifikator_notacije* **Mdc**

gde su terminali, odnosno tokeni, graničnici **Mdo** i **Mdc** i ključna reč "NOTATION" dok su *ime_notacije* i *identifikator_notacije* sintaksičke promenljive. *ime_notacije* je jednoznačno ime notacije koje se koristi prilikom njegovog referisanja u, na primer, deklaraciji entiteta ili atributa. *identifikator_notacije* je spoljašnji identifikator koji mora sadržati dovoljno informacija da se u odgovarajućoj situaciji pozove potrebni interpretator notacije sa ispravnim parametrima.

Na primer, neka dokument koristi dve notacije za zapis formula: TEX i Unix notaciju Eqn. Tada bi se element `formula` mogao deklarisati na sledeći način:

```
<!NOTATION    Tex        SYSTEM     >
<!NOTATION    Eqn        SYSTEM     >
<!ELEMENT     Formula - - CDATA >
<!ATTLIST     Formula
              notacija NOTATION ( Tex | Eqn )  #CURRENT   >
```

Element ima atribut `notacija` čija vrednost može biti ime jedne od deklarisanih notacija za zapis formule. Pretpostavka je da na osnovu imena notacija `Tex` i `Eqn` sistem može nedvosmisleno da izabere odgovarajući programski paket koji će interpretirati sadržaj elementa `formula`. Na primer, u instanciji dokumenta:

```
<formula notacija=Eqn>
     < m sup 3 sub psi > over < m sup 2 sub < eta sub c > >
</formula>
```

sadržaj elementa `formula` biće interpretiran kao $m_{\psi}^3/m_{\eta_c}^2$.

Notacijama se, kao i elementima, mogu pridružiti atributi. Da bi se ime notacije razlikovalo od imena elementa, njemu u deklaraciji atributa prethodi ključna reč `#NOTATION`, što je ilustrovano u sledećim primerima.

```
<!NOTATION    PostScript PUBLIC "-//NOTATION Post Script//EN"     >
<!ELEMENT     slika  - - ( crtez, naziv? )                       >
<!ELEMENT     naziv  - - ( #PCDATA  )                            >
<!ELEMENT     crtez  - - CDATA                                   >
<!ATTLIST     crtez      okvir    ( nema | linije | kvadrat ) nema
                         notacija  NOTATION      #IMPLIED
                         sirina    CDATA         #IMPLIED
                         visina    CDATA         #IMPLIED       >
```

Neformalno govoreći, u ovom primeru je, pre svega, deklarisana notacija `PostScript` [1]. Zatim su deklarisani elementi `slika` koji se sastoji od crteža i opcionog naziva, `naziv` koji sadrži raščlanjive karakterske podatke i konačno `crtez` koji sadrži karakterske podatke. Elementu `crtez` su pridruženi atributi `okvir`, koji opisuje prezentaciju sadržaja entiteta u dokumentu, `notacija` koji specifikuje korišćenu notaciju i `sirina` i `visina` koji određuju dimenziju crteža

prilikom prezentacije. U sledećem primeru prikazan je deo instancije dokumenta koji koristi gornje deklaracije:

```
<slika>
   <crtez okvir=kvadrat notacija="PostScript" sirina="15cm" visina="10cm">
      %!
      gsave /d {rlineto} def /m {moveto} def/lw {setlinewidth} def
      /s {stroke} def ...
   <naslov>Primer jedne slike</naslov>
</slika>
```

Karakterski podaci zapisani u PostScript notaciji ne moraju biti deo osnovnog teksta dokumenta već se mogu uključiti iz spoljašnje datoteke referisanjem entiteta. Moguće su, dakle, i sledeće deklaracije.

```
<!NOTATION    PostScript PUBLIC "-//NOTATION Post Script//EN" >
<!ATTLIST     #NOTATION  PostScript
              sirina     CDATA          #IMPLIED
              visina     CDATA          #IMPLIED                    >
<!ENTITY      moja.slika SYSTEM "C:\mojdir\m_slika.eps"
                         CDATA PostScrpit [ sirina = "15cm"
                                            visina = "10cm" ]      >
<!ELEMENT     slika  - - ( crtez, naziv? )                          >
<!ELEMENT     naziv  - - ( #PCDATA  )                               >
<!ELEMENT     crtez  - - EMPTY                                      >
<!ATTLIST     crtez      teka   ENTITY       #IMPLIED
                         okvir  ( nema | linije | kvadrat ) nema >
```

Notaciji PostScript su sada pridruženi atributi sirina i visina čije vrednosti će prilikom referisanja notacije odrediti dimenzije na koje treba podesiti odgovarajući zapis. Zatim sledi deklaracija spoljašnjeg entiteta moja.slika koga identifikuje sistemski identifikator "C:\mojdir\m_slika.eps". Ovaj entitet sadrži karakterske podatke zapisane notacijom PostScript. Unutar graničnika dso i dsc koji su definisani kao [odnosno], u referentnoj konkretnoj sintaksi dodeljene su vrednosti 15cm i 10cm atributima sirina odnosno visina notacije PostScript. Te vrednosti govore da PostScript procesor treba na te dimenzije da podesi grafičke podatke iz datoteke "C:\mojdir\m_slika.eps". Element crtez sada ima prazan sadržaj ali mu je pridružen novi atribut teka koji je tipa ENTITY što znači da vrednost tog atributa u početnoj etiketi elementa crtez treba da bude ime nekog deklarisanog entiteta. Efekat navođenja imena entiteta za atribut tipa ENTITY je uključivanje entiteta, prema pravilima koja važe za referisanje entiteta, u dokument (videti odeljak 2.4). Kodiranje iste instancije dokumenta sada izgleda ovako:

```
<slika>
   <crtez teka="moja.slika" okvir=kvadrat>
   <naslov>Primer jedne slike</naslov>
</slika>
```

2.6 Označeni odeljci

Označeni odeljci se koriste za označavanje delova dokumenta koje ne treba uopšte obrađivati ili ih treba obrađivati pod određenim uslovima, odnosno na određeni način. Ovakvi delovi

dokumenta koji zahtevaju specifično rukovanje se označavaju na sledeći način:

Mdo Dso *statusne_ključne_reči* Dso ... označeni odeljak ... Msc Mdc

Mdo, Mdc, Dso i Msc su graničnici koji su u referentnoj konkretnoj sintaksi definisani kao <!, >, [odnosno]]. *statusne_ključne_reči* je sintaksička promenljiva koja se realizuje kao lista ključnih reči koje su razdvojene blanko karakterima. U ovom kontekstu mogu se realizovati sledeće ključne reči:

- CDATA koja ukazuje da označeni odeljak sadrži karakterske podatke i, prema tome, ne sadrži nikakve SGML oznake;
- RCDATA koja ukazuje da označeni odeljak sadrži zamenljive karakterske podatke što znači da će se prilikom obrade razrešiti sve karakterske reference i uključiti svi entiteti tipa CDATA i RCDATA;
- IGNORE koja ukazuje da će označeni odeljak prilikom obrade biti ignorisan;
- INCLUDE koja ukazuje da će označeni odeljak biti normalno obrađivan;
- TEMP koja označava da se radi o delu dokumenta koji je privremenog karaktera i koji će možda kasnije biti uklonjen. Prilikom obrade tretira se isto kao ključna reč IGNORE.

Ako je za neki odeljak navedeno više ključnih reči, na njih se primenjuje sledeći prioritet: IGNORE, CDATA, RCDATA i INCLUDE. Ako ni jedna nije navedena, pretpostavlja se status INCLUDE. Označeni odeljci mogu biti i ugnježdeni i to do nivoa koji je određen tekućom vrednošću TAGLVL parametra. Umetanje nije dozvoljeno u odeljke čiji je status CDATA i RCDATA iz očiglednih razloga: njihov sadržaj se ne raščlanjuje i nikakve oznake se ne prepoznaju. Statusi CDATA i RCDATA se koriste kada je potrebno privremeno sprečiti prepoznavanje SGML etiketa. Očigledan primer je SGML označen udžbenik o SGML-u. Da bi u tekst bio uključen primer kao što je `<crtez teka="moja.slika" okvir=kvadrat>` može se koristiti entitet tipa CDATA ili RCDATA ili, alternativno, označeni odeljak istog statusa:

```
<![ CDATA [
   <crtez teka="moja.slika" okvir=kvadrat>
]]>
```

Statusi INCLUDE i IGNORE se koriste kada svi delovi dokumenta ne ulaze u sve njegove verzije. Primer je paralelni višejezički dokument koji se uvek realizuje u jednom od mogućih jezika:

```
<!ENTITY % Srpski "INCLUDE" >
<!ENTITY % Engles "IGNORE"  >
.............................
<![ %Srpski; [
   Ovaj deo je napisan za potrebe...
]]>
<![ %Engles; [
   This part has been written to be used ...
]]>
```

Statusna ključna reč je u ovom slučaju tekst zamene odgovarajućeg parametarskog entiteta. Jednostavnom zamenom ključnih reči INCLUDE ili IGNORE realizuje se verzija dokumenta na jednom ili drugom jeziku. Primetimo da je u deklaraciji označenog odeljka dozvoljeno korišćenje referenci parametarskih entiteta jer je ona deo deklaracije oznaka, tj. sadržana je između graničnika Mdo i Mdc (videti pododeljak 4.1.1).

Statusi INCLUDE i IGNORE se koriste i za obradu velikih dokumenata koji se u fazi pripreme obrađuju deo po deo. Na primer,

```
<!ENTITY aslovo SYSTEM "e-vuk-a.txt" >
<!ENTITY bslovo SYSTEM "e-vuk-b.txt" >
<!--ENTITY % aodelj 'INCLUDE' -->
<!ENTITY % aodelj 'IGNORE'  >
<!ENTITY % bodelj 'INCLUDE' >
<!--ENTITY % bodelj 'IGNORE'  -->
.....
<!-- Ovde dolaze poslovice na slovo A                    -->
                <![ %aodelj; [ &aslovo; ]]>
<!-- Poslovice na slovo B                                -->
                <![ %bodelj; [ &bslovo; ]]>
```

Prvo su deklarisana dva spoljašnja opšta entiteta aslovo i bslovo za referisanje delova na spoljašnjem medijumu. Za svaki od tih delova deklarisan je po jedan parametarski entitet aodelj i bodelj čiji je tekst zamene INCLUDE ili IGNORE u zavisnosti od toga koji deo knjige se obrađuje. Označeni odeljci u samom dokumentu pri tome se ne moraju nikad menjati.

Označeni odeljak koji se koristi više puta u dokumentu može se smestiti u entitet, na sledeći način:

```
<!ENTITY u.prip MS "TEMP [ (ovaj deo je tek u pripremi)" >
```

Tekst zamene ovog entiteta je zagrađeni tekst, a graničnici koji se u ovom slučaju koriste određeni su ključnom reči MS. Graničnike za otvaranje i zatvaranje označenog odeljka, Mso i Msc u tekst zamene entiteta automatski dodaje program za obradu.

Treba napomenuti da u obrađenom dokumentu više nema nikakve informacije o tome da je neki njegov deo bio označeni odeljak. O tome će biti više reči u odeljku 4.1.

2.7 Dodatne mogućnosti

SGML standard predviđa niz dodatnih mogućnosti čije korišćenje nije predviđeno referentnom konkretnom sintaksom. Korišćenje svake od tih mogućnosti mora se zahtevati eksplicitno u SGML deklaraciji. Sa njihovim korišćenjem treba biti pažljiv, naročito prilikom razmene dokumenata, jer ih svaka SGML zasnovana aplikacija ili program ne mora podržavati. O mogućnosti minimizacije etiketa je već bilo reči u odeljku 2.2.3 a ovde će biti ukratko pomenute i ostale.

2.7.1 Kratke reference

Kratke reference su karakteri, ili niske karaktera, koje unutar teksta dokumenta omogućavaju, pod određenim uslovima, referisanje entiteta. Njihovo korišćenje omogućava da se, na primer, neoznačeni tekst interpretira kao da su oznake prisutne. One se obično koriste za prevođenje uobičajenih tipografskih konvencija u etikete označavanja.

Referentna konkretna sintaksa propisuje koji nealfanumerički karakteri koji se ne koriste za oznake mogu biti graničnici kratkih referenci. Taj skup je u celini naveden u tekstu standarda [57]. Korišćenje tog skupa graničnika u dokumentu obezbeđuje se navođenjem ključne reči SGMLREF u SHORTREF klauzi SGML deklaracije. Taj skup se može proširiti navođenjem drugih graničnika iza ključne reči SGMLREF.

Korišćenje svake kratke reference obezbeđuju tri deklaracije u definiciji tipa dokumenta. To su:

- deklaracija *preslikavanja kratke reference* koja definiše ime preslikavanja koje preslikava kratku referencu u referencu entiteta;
- deklaracija opšteg entiteta definiše nisku karaktera u koju se preslikava kratka referenca;
- deklaracija *korišćenja kratke reference* u kojoj se daju imena svih elemenata unutar kojih je određeno preslikavanje na snazi.

Deklaracija preslikavanja kratke reference je sledećeg oblika:

Mdo "SHORTREF" *ime_preslikavanja*
parametarski_literal ime
parametarski_literal ime
..... **Mdc**

gde su terminali, odnosno tokeni, graničnici **Mdo** i **Mdc** i ključna reč "SHORTREF" dok su *ime_preslikavanja*, *parametarski_literal* i *ime* sintaksičke promenljive. *ime_preslikavanja* je jednoznačno ime preslikavanja koje se koristi u deklaraciji korišćenja kratke reference. *parametarski_literal* je graničnik kratke reference koji se preslikava u *ime* opšteg entiteta deklarisanog u istoj definiciji tipa dokumenta.

Deklaracija korišćenja kratke reference je sledećeg oblika:

Mdo "USEMAP *ime_preslikavanja pridruženi_elementi* **Mdc**

gde su terminali, odnosno tokeni, graničnici **Mdo** i **Mdc** i ključna reč "USEMAP" dok su *ime_preslikavanja* i *pridruženi_elementi* sintaksičke promenljive. *ime_preslikavanja* je jednoznačno ime preslikavanja koje je prethodno deklarisano u deklaraciji preslikavanja kratke reference. To preslikavanje postaje aktivno preslikavanje unutar svih elemenata čija su imena realizacija sintaksičke promenljive *pridruženi_elementi* i unutar svih u njih ugnježdenih elemenata, osim onih kojima je takođe pridruženo neko preslikavanje.

Deklaracije iz sledećeg primera omogućavaju da SGML zasnovan program automatski generiše etikete za elemente pasus i navod:

```
<!ELEMENT  tekst - - ( pasus )+                          >
<!ELEMENT  pasus - - ( #PCDATA | navod )+                >
<!ELEMENT  navod - - ( #PCDATA )                         >
<!ENTITY   p.et      STARTTAG "pasus"                    >
<!ENTITY   p.et.kr   ENDTAG "pasus"                      >
<!ENTITY   n.et      STARTTAG "navod"                    >
<!ENTITY   n.et.kr   ENDTAG   "navod"                    >
<!SHORTREF u.pasus  "&#RS;"          p.et                >
<!SHORTREF u.pasusu "&#RS;B&#RE;"    p.et.kr
                    '"'              n.et                >
<!SHORTREF u.navodu '"'              n.et.kr             >
<!USEMAP   u.pasus  tekst                                >
<!USEMAP   u.pasusu pasus                                >
<!USEMAP   u.navodu navod                                >
```

Ove deklaracije uvode tri preslikavanja i preciziraju kada se ona koriste:

1. preslikavanje u.pasus je aktivno unutar elementa tekst i unutar u njega ugnježdenih elemenata kojima nije pridruženo neko drugo preslikavanje (ni jedan element, u ovom slučaju). Ono preslikava karakter za početak sloga u početnu etiketu elementa pasus;
2. preslikavanje u.pasusu je aktivno unutar elementa pasus i ono preslikava navodnik u početnu etiketu elementa navod a sekvenciju karaktera kraj sloga, blanko karakteri, početak sloga u krajnju etiketu elementa pasus;
3. preslikavanje u.navodu je aktivno unutar elementa navod i ono preslikava navodnik u krajnju etiketu elementa navod.

Korišćenjem ovako deklarisanih preslikavanja kratkih referenci bi sledeći, potpuno neoznačeni tekst (predgovor iz [109]):

```
....Samo na taj način može se stvoriti idealna država.

Ali šta je u stvari za Platona idealna država?

U načelu, on mudro kaže da je "najbolja i najsloženija ona država u
kojoj za vlašću najmanje teže oni koji su izabrani da vladaju, a ako
država ima drukčije vladaoce, onda je u njoj obrnuto" (520d), a kad
strukturu ....
```

bio preveden u označeni tekst na sledeći način:

```
....Samo na taj način može se stvoriti idealna država.</pasus>
<pasus>Ali šta je u stvari za Platona idealna država?</pasus>
<pasus>U načelu, on mudro kaže da je <navod>najbolja i
najsloženija ona država u kojoj za vlašću najmanje teže
oni koji su izabrani da vladaju, a ako država ima drukčije
vladaoce, onda je u njoj obrnuto</navod> (520d), a kad strukturu ....
```

Deklaracija korišćenja preslikavanja kratkih referenci se, za razliku od drugih deklaracija može naći i unutar označenog teksta. Tako deklaracija `<!USEMAP ime.pres>` u kojoj nije navedno ime ni jednog elementa znači da je preslikavanje `ime.pres` aktivno unutar tekućeg elementa i unutar u njemu ugnježdenih elemenata kojima nije pridruženo neko preslikavanje. Deklaracija `<!USEMAP #EMPTY>` privremeno onemogućava sva preslikavanja.

Primer korišćenja kratkih referenci može se naći u HTML definiciji tipa dokumenta koja je opisana u 3.3.2.

2.7.2 Poddokumenta

Ponekad je dokument tako složen da se ne može opisati jednom strukturom dokumenta. Ako se različite strukture koriste individualno, odnosno uvek je na snazi samo jedna od njih, govori se da se dokument sastoji od *poddokumenata*. Ako se delovima teksta mogu pridružiti različite uloge onda se višestruke strukture koriste paralelno pa se govori o konkurentnim strukturama dokumenta, o kojima će biti više govora u narednom pododeljku.

Ako se dokument sastoji od više odvojenih delova od kojih svaki ima zasebnu strukturu ti delovi se mogu tretirati kao poddokumenta smeštena izvan dokumenta. Svaki od njih se mora na početku glavnog dokumenta deklarisati kao spoljašnji opšti entitet pobliže određen ključnom reči `SUBDOC`. Na odgovarajućem mestu u glavnom dokumentu koristi se referenca entiteta ili se dodeljuje vrednost atributu tipa `ENTITY` da bi se poddokument uključio u

glavni dokument. Korišćenje poddokumenata se mora omogućiti u FEATURE klauzi SGML deklaracije, postavljanjem zastavice uz ključnu reč SUBDOC na vrednost YES.

Svaki poddokument je celina za sebe i sastoji se od definicije tipa dokumenta i teksta kodiranog korišćenjem entiteta, elemenata, atributa i notacija deklarisanih u tom lokalnom DTD-u. Sva poddokumenta moraju, međutim, da koriste istu SGML deklaraciju. Preciznije rečeno, poddokument ne sme da sadrži SGML deklaraciju već se na njega primenjuje SGML deklaracija iz glavnog dokumenta.

Poddokumenta se najčešće koriste kada se dokument sastoji od delova koji imaju sasvim različite strukture. Na primer, ako bi u udžbenik koji ima strukturu knjige trebalo uključiti terminološki rečnik, koji ima sasvim drukčiju strukturu, postupilo bi se na sledeći način:

```
<!DOCTYPE udzbenik SYSTEM "knjiga.dtd"  [
   <!ENTITY term.rec SYSTEM "c:\udzbenik\term_rec.txt" SUBDOC >
   <!ELEMENT opc.poglav   - - EMPTY                          >
   <!ATTLIST opc.poglav   teka   ENTITY   #REQUIRED          >
]>
<udzbenik>
.....
<!-- Ovde pocinje novo poglavlje sa svojom strukturom      -->
<opc.poglav teka=term.rec>
</udzbenik>
```

U ovom primeru udžbenik je opisan definicijom tipa dokumenta koja je pripremljena za strukturu knjige — `kniga.dtd`. Ova definicija je modifikovana uvođenjem poddokumenta koji sadrži terminološki rečnik. Da bi on bio uključen na odgovarajući način u tekst udžbenika, modifikovan je postojeći element `opc.poglav` iz DTD `knjiga.dtd` čiji je sadržaj sada prazan a ima jedan obavezni atribut tipa ENTITY preko koga se poddokument referiše.

Sadržaj poddokumenta iz datoteke "c:\udzbenik\term_rec.txt" bi sada mogao izgledati ovako:

```
<!DOCTYPE terminol SYSTEM "recnik.dtd" [
]>
<terminol>
<odrednica n=P1>apstraktna sintaksa<def>
.....
</terminol>
```

Struktura ovog poddokumenta je opisana unapred pripremljenom definicijom tipa dokumenta za rečnike — `recnik.dtd`.

Treba istaći da obrada jednog poddokumenta počinje sasvim iz početka. Iz glavnog dokumenta se ne prepoznaju entiteti, notacije niti jedinstveni identifikatori a, na primer, tekući rang računa se iz početka. S druge strane, poddokumenta mogu biti ugnježđena do nivoa koji je propisan vrednošću parametra ENTLVL u konkretnoj sintaksi koja se koristi (videti odeljak 2.8). Rezultat toga je da SGML raščlanjivač prilikom referisanja entiteta tipa SUBDOC treba da zapamti sve deklaracije i vrednosti svih parametara da bi ih mogao ponovo uspostaviti kada se obrada poddokumenta završi.

2.7.3 Konkurentne strukture

Kada istovremeno treba opisati više struktura jednog dokumenta onda na njegovom početku treba specifikovati više definicija tipa dokumenta. Da bi to bilo moguće, u FEATURE klauzi

SGML deklaracije, treba zastavicu uz ključnu reč CONCUR postaviti na YES i uz to specifikovati koliko najviše dodatnih definicija tipa dokumenta će se koristiti.

Svaka dozvoljena struktura dokumenta mora se deklarisati na njegovom početku, odmah iza SGML deklaracije. Prva definicija tipa dokumenta koja se navede je definicija *baznog dokumenta* — njegovo ime se koristi kao početna etiketa dokumenta. Ostale definicije mogu se navesti u proizvoljnom redosledu ali pre teksta dokumenta.

```
<!DOCTYPE platon SYSTEM "knjiga.dtd"                          >
<!DOCTYPE stranice [
   <!ELEMENT stranice - O   ( strana )+                       >
   <!ELEMENT strana   - -
                ( zaglavlje?, tekst, fusnote*, podnozje? ) >
   <!ATTLIST strana   r.broj  NUMBER      #IMPLIED            >
   ...............
]>
<!DOCTYPE original [
   <!ELEMENT original - O   ( org.str )+                      >
   <!ELEMENT org.str  - -   ( org.deo )+                      >
   <!ATTLIST org.str  r.broj  NUMBER      #IMPLIED            >
   <!ELEMENT org.deo  - -   ( #PCDATA )                       >
   <!ATTLIST org.deo  r.broj  NMTOKEN     #IMPLIED            >
]>
```

Ovaj primer ilustruje pojednostavljeni opis različitih struktura jednog konkretnog izdanja složenog teksta kakav je Platonova *Država* [109]. Za opis bazne strukture koristi se definicija tipa dokumenta pripremljena za knjige. Tip dokumenta **stranice** opisuje raspored teksta po stranicama konkretnog izdanja dok tip dokumenta **original** omogućava označavanje brojeva stranica i delova iz referentnog Stefanusovog izdanja. Korišćenjem ovih deklaracija kraj četrnaeste i početak petnaeste stranice navedenog izdanja ovog dela bio bi kodiran na sledeći način:

```
<platon>
...........
pustimo i najmanju stvar, a da je ne izvedemo na či-
</(stranice)strana>
<(stranice)strana r.broj=15>
stinu.  Ti je možda i poznaješ prijatelju, ali mi je, čini
mi se, ne možemo pronaći.  Vi, pametni, pre nego što se
<(original)org.str r.broj=337>
<(original)org.deo r.broj='a'>
na nas naljutite, morate imati sažaljenja prema nama.
<odeljak r.broj='XI.'>Tada se on podrugljivo nasmeja i reče:
<g govori='Telemah'>
Tako mi boga, ovo je ona uobičajena ironija<nap ref='N14'>
Sokratova.....
</platon>
```

Prilikom označavanja teksta treba za svaki element navesti kojoj strukturi pripada. To se radi tako što se ime odgovarajućeg tipa dokumenta unutar zagrada navodi ispred imena elementa u početnoj i krajnjoj etiketi elementa. Jedino za elemente koji pripadaju baznoj

strukturi ne treba navoditi ime tipa dokumenta. Slična pravila se koriste i za referisanje entiteta koji nisu deklarisani u baznoj definiciji tipa dokumenta.

2.7.4 Sponski procesi

Opciono svojstvo SGML-a je i korišćenje definicija sponskih procesa, koji specifikuju kako treba obraditi izvorni dokument da bi se proizveo rezultujući dokument drugog tipa, na primer, formatirani dokument. Sponski procesi se specifikuju u *deklaraciji tipa spone* koja je nezavisna od definicija tipa dokumenta i svih ostalih oznaka. Spone mogu biti:

- jednostavne, što znači da su kontrolisane atributima koji su pridruženi uz element baznog dokumenta. Da bi se one koristile, zastavica `SIMPLE` u `FEATURE` klauzi SGML deklaracije mora biti postavljena na `YES`.
- implicitne, što znači da njih određuje program za formatiranje. Da bi se one koristile zastavica uz `IMPLICIT` u `FEATURE` klauzi SGML deklaracije mora biti postavljena na `YES`.
- eksplicitne, što znači da njih eksplicitno definiše kreator dokumenta. Da bi se one koristile zastavica `EXPLICIT` u `FEATURE` klauzi SGML deklaracije mora biti postavljena na `YES`.

Definicije sponskih procesa specifikuju kako treba spojiti definicije tipa dokumenta da bi se dokument jednog, izvornog tipa transformisao u dokument drugog, rezultujućeg tipa. Na primer, kako se mogu generisati oznake koje dokument opisuju u kategorijama rezultujuće izlazne strukture (stranice, kolone, blokovi teksta, reci) na osnovu izvornih logičkih oznaka (glave, odeljci, pasusi).

Na primer, neka su u izvornom dokumentu deklarisane tri vrste istaknutih fraza `ist1`, `ist2` i `ist3` i neka je element `blok` deklarisan u rezultujućem dokumentu. Tada se može deklarisati sledeća spona:

```
<!LINK slaganje.dok ist1 blok [ fam="Times" lik="italic" pt=11pt ]
                    ist2 blok [ fam="Times" lik="bold" pt=11pt ]
                    ist3 blok [ fam="Times" lik="bold-it" pt=11pt ]
>
```

Sve tri vrste istaknutih fraza iz izvornog dokumenta biće transformisane u isti element `blok` kome će biti pridruženi odgovarajući atributi čije su vrednosti navedene u listi specifikacije atributa u deklaraciji spone `slaganje.dok`. Tako će sledeći izvorni tekst ([109], (334b)):

```
....jer on hvali Odisejvog deda po materinoj strani, Autolika, i
kaže da se <ist1>među ostalim ljudima ističe prevarom i
krivokletstvom</ist1>. I po tvome shvatanju ...
```

postati sledeći rezultujući tekst:

```
....jer on hvali Odisejevog deda po materinoj strani, Autolika,
i kaže da se <blok fam="Times" lik="italic" pt=11pt>među
ostalim ljudima ističe prevarom i krivokletstvom</blok>. I po
tvome shvatanju ...
```

Više o spajanju različitih struktura jednog dokumenta može se naći u [57] i [23].

2.8 Apstraktna prema konkretnoj sintaksi

SGML vrlo precizno određuje ulogu svakog karaktera da bi razlikovanje oznaka od podataka bilo moguće. Svaka oznaka je zasnovana na skupu uloga graničnika koje određuju koji delovi teksta će biti protumačeni kao oznake. Svaki od ovih graničnika se identifikuje imenom u *apstraktnoj sintaksi* kome se u konkretnoj implementaciji pridružuje niska karaktera. Na primer, u odeljku 2.2 je dat sledeći opis sintakse deklaracije elementa korišćenjem apstraktne sintakse:

Mdo "ELEMENT" *tip_elementa minimizacija deklaracija_sadržaja* Mdc

U SGML standardu je ponuđena jedna konkretna sintaksa koja se naziva *referentna konkretna sintaksa*. Kada se ona koristi, sintaksa deklaracije elementa izgledala ovako:

<!ELEMENT *tip_elementa minimizacija deklaracija_sadržaja* >

Referentna konkretna sintaksa se u SGML deklaraciji može promeniti čime se dobija *varijantna konkretna sintaksa*. Ukoliko bi se u njoj graničnik **Mdo** definisao kao `[?`, graničnik **Mdc** kao `]` a ključna reč **ELEMENT** kao `DEO`, sintaksa deklaracije elementa bi izgledala ovako:

`[?DEO` *tip_elementa minimizacija deklaracija_sadržaja* `]`

Treba napomenuti da je ovaj primer samo ilustracija i da ne znači da bi ovakva izmena referentne konkretne sintakse imala opravdanja.

U okviru SGML deklaracije definiše se varijanta SGML-a koja se koristi i to u okviru sledećih šest klauza:

- `CHARSET`, u okviru koje se detaljno opisuje jedan ili više karakterskih skupova koji se koriste u dokumentu. Jedan karakterski skup se identifikuje formalnim javnim identifikatorom kakav je, na primer, "ISO 646-1083//CHARSET..." u okviru podklauze `BASESET`. Za sve karaktere iz tog skupa se takođe propisuje u okviru podklauze `DESCSET` koja sledi da li im je dodeljeno značenje i ako jeste koje. U slučaju da karakteru nije dodeljeno značenje njegov kod je označen kao `UNUSED` što znači da se radi o karakterima koji nisu valjani SGML karakteri (ne-SGML karakteri);
- `CAPACITY`, koja definiše skup od 17 kapaciteta dokumenta kao niz vrednosti koje dokument ne bi smeo da prevaziđe. Na primer, kapacitet `ENTCHCAP` propisuje kolika je maksimalna ukupna dužina teksta svih entiteta merena brojem karaktera. Ključna reč `SGMLREF` znači da se koriste vrednosti iz skupa kapaciteta referentne konkretne sintakse, u kojoj je, na primer, `ENTCHCAP = 35000`. Promene se zadaju u obliku parova koji se sastoje od imena kapaciteta i pridružene vrednosti;
- `SCOPE`, koja određuje da li se konkretna sintaksa koja se definiše u sledećoj klauzi primenjuje i na definiciju tipa dokumenta ili samo na instanciju dokumenta;
- `SYNTAX`, koja opisuje konkretnu sintaksu koja se u dokumentu koristi;
- `FEATURES`, koja govori koja se opciona svojstva SGML-a u dokumentu koriste (videti, na primer, odeljak 2.7);
- `APPINFO`, u kojoj se prosleđuju informacije specifične za aplikaciju, ako postoje. Na primer, token `ArcBase` naveden u ovoj klauzi govori da je SGML dokument u saglasnosti sa DSSSL arhitekturom dokumenta (videti 3.1.1).

Najvažnija od ovih klauza je SINTAX jer ona propisuje koji se karakteri mogu koristiti za označavanje dokumenta. Ako se koristi referentna konkretna sintaksa, ona se u ovoj klauzi specifikuje formalnim javnim identifikatorom "ISO 8879-1986//SYNTAX Reference//EN". Varijantna konkretna sintaksa se zadaje u sledećih osam podklauza:

- SHUNCHAR, u kojoj se definiše koji su karakteri kontrolni karakteri ili neki ne-SGML karakteri, tj. koji karakteri nisu deo podataka;
- BASESET, u okviru koje se detaljno opisuje karakterski skup koji se koristi za konkretnu sintaksu. Karakterski skup se i ovde identifikuje formalnim javnim identifikatorom;
- DESCSET, u okviru koje se za sve karaktere iz specifikovanog skupa propisuje kako će biti korišćeni u konkretnoj sintaksi. Parovi podklauza BASESET i DESCSET mogu se više puta ponavljati za opis više karakterskih skupova;
- FUNCTION, u okviru koje se definiše koji se karakteri dodeljuju funkcijama koje sintaksa zahteva. Funkcijama RS, RE i SPACE se moraju obavezno pridružiti karakteri a mogu se uvesti i nove funkcije (npr. TAB). Za nove funkcije se mora navesti i kojoj klasi pripadaju. Na primer, moguća je klasa separatora, klasa karaktera koji privremeno obustavljaju prepoznavanje oznaka, i slično;
- NAMING, u okviru koje se specifikuju pravila koja se koriste za definisanje imena elemenata, entiteta, notacija itd. Tu se, pre svega, specifikuje od kojih karaktera se ime može sastaviti i kojim karakterima imena mogu početi, odvojeno za „mala slova" i „velika slova", a zatim se propisuje za koje SGML konstrukte postoji ekvivalentnost imena zapisanih „malim slovima" i „velikim slovima". U referentnoj konkretnoj sintaksi „mala" i „velika" slova su ekvivalentna, osim u slučaju imena entiteta;
- DELIM, koja govori šta su opšti a šta granični kratkih referenci u dokumentu. Ako se u podklauzama GENERAL i SHORTREF navede ključna reč SGMLREF, onda se koriste granični predviđeni referentnom konkretnom sintaksom. Međutim, iza ove ključne reči mogu slediti promene referentne konkretne sintakse u obliku parova koji se sastoje od imena graničnika, npr. Mdo i njemu pridružene karakterske niske, npr. "[?" za primer s početka ovog odeljka;
- NAMES, koja govori koja se konkretna imena koriste za rezervisane reči. Kao i u slučaju graničnika, ako se u ovoj klauzi navede ključna reč SGMLREF, koriste se imena predviđena referentnom konkretnom sintaksom. Promene referentne konkretne sintakse zadaju se u obliku parova koji se sastoje od imena rezervisane reči u referentnoj konkretnoj sintaksi i odgovarajućeg imena u varijantnoj konkretnoj sintaksi, na primer par ELEMENT DEO za primer s početka ovog odeljka;
- QUANTITY, koja definiše vrednosti za skup od 15 veličina dokumenta. Na primer, veličina NAMELEN propisuje maksimalnu dužinu imena, brojeva, tokena i slično. Ključna reč SGMLREF znači da se koriste vrednosti iz skupa veličina referentne konkretne sintakse, u kojoj je, na primer, NAMELEN = 8. Promene se zadaju u obliku parova koji se sastoje od imena veličine i pridružene vrednosti.

Važno je istaći da se sama SGML deklaracija zapisuje korišćenjem isključivo referentne konkretne sintakse, što je i logično jer tek SGML deklaracija može promeniti konkretnu sintaksu. Referentna konkretna sintaksa propisuje da se za deklaracije obeležavanja koristi karakterski skup ISO 646 [56] što znači da se cela SGML deklaracija mora zapisati korišćenjem

isključivo tog karakterskog skupa. Posledica toga je da ako se, na primer, definiše neka varijantna konkretna sintaksa koja za deklaracije obeležavanja koristi neki drugi karakterski skup i koja uz to uvodi nova imena za rezervisane reči, karakteri tih novih imena koji su izvan opsega uobičajenog ASCII skupa moraju biti uneti kao karakterske reference.

Takođe, rezervisane reči koje se pojavljuju u SGML deklaraciji ne mogu se menjati. Na primer, rezervisana reč ENTITY može u zavisnosti od konteksta imati tri značenja:

1. identifikator deklaracije entiteta;
2. ime opšteg entiteta, kao tip vrednosti atributa;
3. pravila za imenovanje entiteta (izmenljivost „malih“ i „velikih“ slova).

Rezervisana reč koja identifikuje prva dva značenja se može menjati dok se za identfikovanje trećeg značenja koje se pojavljuje samo u kontekstu SGML deklaracije mora koristiti ime ENTITY iz referentne konkretne sintakse.

Klauza CHARSET SGML deklaracije definiše karakterski skup dokumenta kao skup karaktera koji se koriste u dokumentu koji sledi. Kako dokument može da sadrži karaktere koji se ne koriste za obeležavanje dokumenta, karakterski skup dokumenta može da sadrži karaktere koji nisu u karakterskom skupu konkretne sintakse koji je definisan u podklauzama BASESET i DESCSET klauze SYNTAX. Ako je neki karakter iz karakterskog skupa dokumenta definisan kao UNUSED, onda se on u dokumentu uopšte ne koristi i tretira se kao ne-SGML karakter. Takav karakter može biti uključen u dokument samo preko spoljašnjeg entiteta čiji je sadržaj tipa NDATA. Treba stoga voditi računa da karakteri definisani kao UNUSED u konkretnoj sintaksi dokumenta budu isto tako definisani i u karakterskom skupu dokumenta.

Kada se koristi više baznih karakterskih skupova u karakterskom skupu dokumenta važno je voditi računa o tome da se svaki kod, u smislu bitovske kombinacije, navede samo jednom u onom delu koji opisuje karakterski skup (podklauza DESCSET klauze CHARSET). Sledeći primer ilustruje početak SGML deklaracije za dokument koji osim karakterskog skupa ISO 646 koristi i ISO 8859-5 [59] koji u gornjem delu kodne tablice sadrži kodove ćiriličnog alfabeta:

```
<!SGML "ISO 8879-1986"
  CHARSET BASESET "ISO 646-1983//CHARSET International
                   Reference Version (IRV)//ESC 2/5 4/0"
          DESCSET 0   9   UNUSED
                  9   2   9
                  11  2   UNUSED
                  13  1   13
                  14  18  UNUSED
                  35  95  32
                  127 1   UNUSED
          BASESET "ISO 8859-5//CHARSET 8-bit Single Byte Coded
                   Character set - Cyrillic Alphabet//ESC 2/5 4/0"
          DESCSET 128 128 128
          ..................
```

Puna SGML deklaracija data je u primeruu dodatku A.

Glava 3

SGML u primeni

SGML je nastao na osnovama IBM-ovog proizvoda GML (Generalized Markup Language) čiji je autor Charles Goldfarb [34]. Proteklih deset godina od objavljivanja SGML standarda su pokazale da je njegov značaj i uticaj na obradu dokumenta u najširem smislu mnogostruko prevazišao očekivanja kao što je i njegova primena obuhvatila daleko širu oblast ljudskog delovanja od tekstualnih i kancelarijskih sistema, kako stoji u kategorizaciji standarda. Već u trenutku njegovog objavljivanja potrebe za jednim takvim „sredstvom" bile su velike što pokazuje i činjenica da je SGML bio u upotrebi u Američkom ministarstvu odbrane i Kancelariji za zvanične publikacije Evropske zajednice i pre objavljivanja standarda. Samo dva meseca po objavljivanju, prodaja ovog standarda je premašila prodaju u tom trenutku 10 godina starog standarda programskog jezika Fortran, do tada najtraženijeg standarda iz oblasti obrade informacija.

Interesovanje za SGML-om nije jenjavalo. Polje njegove primene se širilo i stalno su pronalaženi novi načini njegove primene. Osim pomenutih institucija, i mnoge druge su pristupile njegovom korišćenju tako što su izradile definicije tipa dokumenta koje opisuju dokumenta iz oblasti njihovog rada. Pomenimo samo neke: Međunarodna organizacija za standardizaciju (ISO)[61], Državna štamparija Velike Britanije, Američko društvo izdavača [2], itd.

3.1 SGML alati

Premda SGML nije sistem za uređivanje i formatiranje teksta, nije sistem za tekstualne baze podataka, nije sistem za lingvističku obradu i nije sistem za prenos podataka, on je namenjen računarskoj obradi u pomenutim i mnogim drugim oblastima, ali se bez odgovarajuće programske podrške ne može koristiti. Među prvim programskim sistemima razvijeni su *SGML raščlanjivači* (engl. *parser*) koji paketno obrađuju dokument. U najjednostavnijem slučaju oni na ulazu dobijaju SGML dokument a na izlazu daju dva moguća odgovora: **da** ako je dokument ispravno sintaksički zapisan i **ne** ako dokument nije ispravno sintaksički zapisan. Među prve raščlanjivače spadaju **ASP-SGML** raščlanjivač, neformalno poznat kao Amsterdamski raščlanjivač koji su razvili Jos Warmeri i S. van Egmond [150] i **ARK-SGML** raščlanjivač koji je razvio Charles Goldfarb. Na osnovu njih nastalo je više drugih raščlanjivača, na primer **YASP** i **YAO**, a zatim i dva vrlo popularna raščlanjivača koje je razvio James Clark: **SGMLS** i njegov naslednik **SP** raščlanjivač koji podržava sva opciona svojstva SGML-a. Oba ova raščlanjivača na izlazu generišu **RAST** format koji je u skladu

sa ISO/IEC 13673:1994 standardom. Treba istaći da su svi pomenuti produkti javno dobro. Od ostalih raščlanjivača u značajnijoj upotrebi je MARK-IT raščlanjivač [125].

Korišćenje SGML raščlanjivač a ne oslobađa kreatore SGML dokumenata mukotrpnog posla oko označavanja dokumenta i pregledanja dokumenata pretrpanih etiketama označavanja. Razvijeni su stoga alati za *uređivanje i pregledanje SGML dokumenata* koji, između ostalih funkcija, omogućavaju „kompilaciju" SGML deklaracije i definicije tipa dokumenta i njihovu efikasnu upotrebu za sva dokumenta koja ih koriste. Osim toga, oni autoru nude, najčešće po principu menija, listu etiketa koje su primenljive u svakoj pojedinačnoj tački dokumenta. Od produkata koji su javno dobro treba pomenuti **PSGML** koji je razvio Lennart Staflin a koji predstavlja specijalan režim rada uređivača Emacs-a koji se korisiti za uređivanje SGML dokumenata. Od komercijalnih produkata treba pomenuti jedan od prvih, a to je WRITE-IT koji je proizvod Sega Group [126], i Author/Editor koji je proizvod SoftQuad.

I pored postojanja alata za uređivanje SGML dokumenata, mnogim autorima oni nisu dostupni ili oni, pak, ne žele da odustanu od korišćenja onih na koje su navikli. S ovim u vezi je i problem dokumenata koja su kreirana pre ere SGML-a. Za rukovanje i konverziju SGML dokumenata pogodni su jezici za obradu niski kakvi su, na primer, **Awk** [8], **Icon** [43] i **Perl** [149] za koji je razvijena i posebna biblioteka **PerlSGML** namenjena konverziji ovih dokumenata.

Razvijeni su i specifični alati za konverziju SGML podataka, od kojih će ovde biti pomenuti samo neki. **Rainbow** alati proizvode SGML dokument na osnovu formata određenog procesora reči s idejom da se iz njega ekstrahuje što je moguće više informacije o strukturi dokumenta. Ovakvi alati, koji se svi zasnivaju na istom SGML jeziku za razmenu, razvijeni su za više formata procesora reči: Interleaf, Ventura, i slično. **ICA** (Integrated Chameleon Architecture) je skup alata pomoću koga se generišu alati za konverziju SGML podataka, koji omogućavaju pretvaranje u, ili pretvaranje iz, ograničenog podskupa definicija tipa dokumenta. Tako je, na primer, razvijen alat za konverziju dokumenata opisanih definicijom tipa dokumenta 'knjiga' u odgovarajući LaTeX stil ili troff makro paket. **CoST** (Copenhagen SGML Tool) je takođe sistem za generisanje novih alata. Pomoću njega mogu se sastavljati specifikacije za konverziju koje se zasnivaju isključivo na strukturi instancije SGML dokumenta. Više o programskim alatima za rukovanje SGML dokumentima može se naći u [28] i na WWW stranicama [153].

3.2 SGML i tekstualne baze

Vrlo brzo se uvidelo da mogućnosti SGML-a daleko prevazilaze obradu kancelarijskih i drugih zvaničnih dokumenata. Takoreći od trenutka svog nastanka, SGML je našao svoje mesto u obradi tekstova u najširem smislu a posebno u najrazličitijim vidovima literarnih, lingvističkih i leksikografskih istraživanja [13]. Takav je, na primer, projekat Britanski nacionalni korpus (British National Corpus). Ipak, jedan od prvih i najinteresantnijih, koji je otvorio nove puteve u obradi teksta je projekat izrade drugog izdanja istorijskog rečnika engleskog jezika, *Oxford English Dictionary* (skraćeno OED) [15]. OED predstavlja kapitalno delo engleske leksikografije čije je prvo izdanje u 12 knjiga nastalo u periodu od 1884. do 1928. godine. Već 1933. godine publikovana je i jedna knjiga dodataka rečniku a u periodu od 1958. do 1986. godine nastale su još četiri knjige dodataka. Grandioznost poduhvata ilustruje podatak da OED i dodaci imaju zajedno 306.000 odrednica štampanih na ukupno 21.000 strana.

Oxford University Press je 1984. godine odlučio da pristupi izradi drugog izdanja *Oxford English Dictionary* (skraćeno OED2) sa ciljem da se spoje prvo izdanje i dodaci, unesu nove odrednice i revidiraju stare. Osim izrade novog izdanja u papirnom obliku, planirano je i stvaranje OED baze podataka koja bi omogućila leksikografima i drugim korisnicima efikasno pretraživanje ogromne količine podataka sadržanih u rečniku. Da bi ostvarivanje ovih ciljeva bilo moguće, pre svega je bilo neophodno pretvoriti papirnu verziju OED i dodataka u mašinski čitljiv oblik. Kompletni rečnici su ponovo prekucani pri čemu se vodilo računa da se iz papirne u mašinski čitljivu verziju prenese što je moguće više informacija. Tako je dobijena mašinski čitljiva verzija označena pretežno ne strukturnim već tipografskim oznakama koje su često ali ne uvek, i ne uvek dovoljno precizno, odražavale strukturu.

U trenutku početka rada na projektu OED2 ideja o kreiranju rečničke baze podataka paralelno sa izradom klasičnog, papirnog izdanja nije bila nova. Međutim, do tada izrađene rečničke baze podataka zasnivale su se na konceptu slogovno orijentisanih baza u kojima je tekst rečnika segmentiran na konvencionalne slogove i polja. Posebno prisustvo amorfnih komponenata u rečniku, kakve su ilustracije, tabele, unakrsne reference i slobodan tekst (diskurs) ne čini ovakav koncept sasvim pogodnim za velike tekstualne baze podataka. Pri koncipiranju izrade OED2 pristupilo se stoga drugim rešenjima. Pre svega, tekst rečnika nije segmentiran već je SGML označen [9] u skladu sa definicijom tipa dokumenta koja je dobijena analizom strukture odrednice u OED i novim potrebama. Dobijeni formalno precizni opis strukture odrednice omogućio je razvoj transduktora koji su tipografske oznake u mašinski čitljivoj verziji rečnika preveli u strukturne SGML oznake [73].

Pristupilo se zatim izradi modela baze podataka koji je prilagođen velikim, veoma strukturiranim tekstovima, razvoju struktura podataka koje efikasno podržavaju operacije nad modelom, konstrukciji upitnog modela koji bi zadovoljio raznovrsne zahteve za podacima, te razvoju jezika za izražavanje upita i algoritama za njihovo zadovoljavanje. Tako je nastao *model baze podataka definisan gramatikom* [37] i nad njim izgrađen jezik za manipulaciju podacima čije osnovne ideje će ovde biti predstavljene. Treba istaći da razvijeni model ne zavisi od konkretnog teksta i baze podataka, OED2, već se može primeniti i na druge strukturirane, referentne tekstove, kakve su enciklopedije, zakonska dokumenta, kolekcija novinskih isečaka, i slično. Šta više, ovaj model ne zavisi od konkretne reprezentacije tako da se može implementirati na različitom hardveru i za različite operativne sisteme.

Treba napomenuti da su u istom periodu razvijeni i drugi modeli tekstualnih baza podataka namenjenih strukturiranim tekstovima čija je struktura SGML reprezentovana a koji se zasnivaju na drugačijim principima, na primer, na objektno-orijentisanim tehnikama [154].

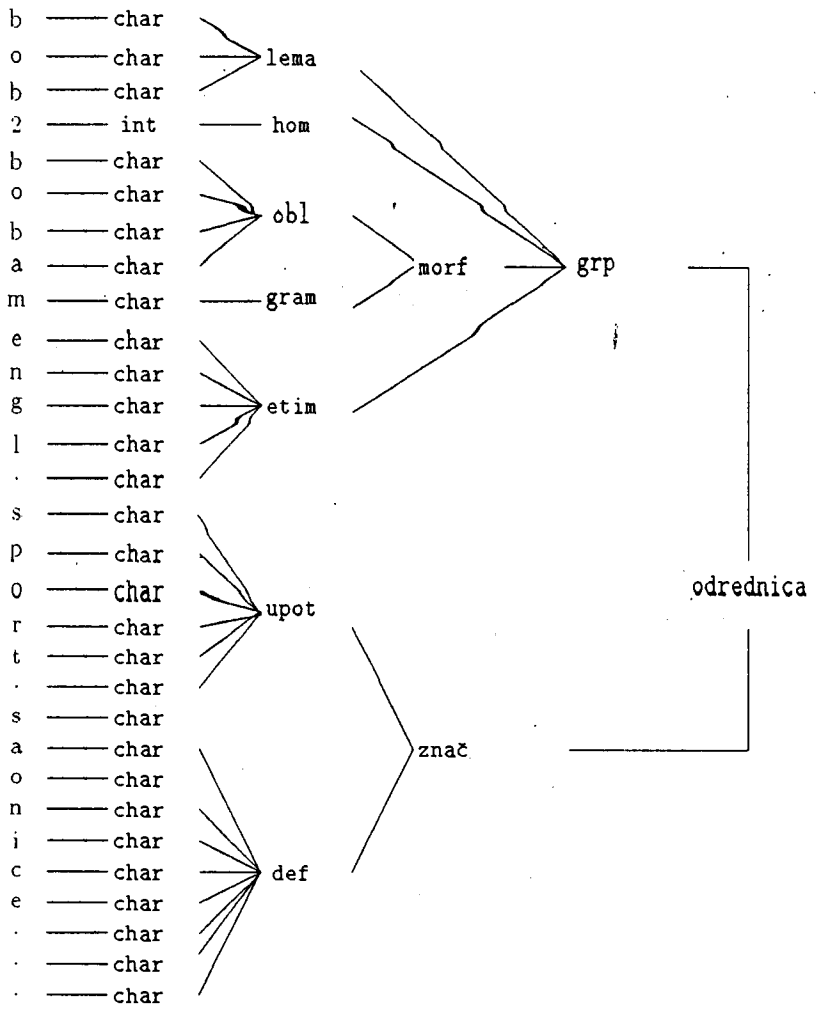
3.2.1 Model tekstualne baze podataka definisane gramatikom i jezik za manipulaciju podacima

```
rečnik:=     odrednica+;              lema:=  char+;
odrednica:=  grp znač;                gram:=  char+;
grp:=        lema hom? morf? etim?;   obl:=   char+;
morf:=       obl* gram;               etim:=  char+;
znač:=       upot? def;               upot:=  char+;
hom:=        dig+;                    def:=   char+;
```

Tabela 3.1: Gramatika za opis pojednostavljene strukture rečničke odrednice

Tekstualna baza se opisuje shemom koja se izražava preko gramatike. Primera radi, u tabeli 3.1 je data gramatika za opisivanje pojednostavljene strukture rečnika koja koristi uobičajenu BNF notaciju. Ona se može koristiti za raščlanjavanje i reprezentaciju kolekcije podataka kakav je rečnik. Na primer, sledeća rečnička odrednica iz [116] je saglasna sa sintaksom date gramatike:

bob^2, boba m engl. sport. *saonice sa više sedišta*

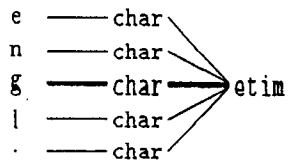


Slika 3.1: P-niska E za primer odrednice

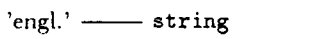
Valjana instancija baze podataka sadrži, dakle, podatke koji su u saglasnosti sa zadatom shemom. Podaci su u ovakvoj bazi smešteni u obliku raščlanjenih niski, koje se nazivaju *p-niskama*. Smešten u bazu korišćenjem gramatike iz tabele 3.1, gornji primer dobija oblik drveta raščlanjavanja sa slike 3.1, na kojoj su listovi drveta prikazani sa leve strane a koren drveta je na desnoj strani. Premda se predstavlja u obliku drveta raščlanjavanja, p-niska kao reprezentacija jedne instancije baze podataka, može se menjati putem operacija koje su ugrađene u jezik za manipulaciju podacima, te predstavlja, u suštini, apstraktni tip podataka.

Jezik za manipulaciju podacima sadrži operatore za konverziju podataka, za selekciju

pre svega, operatori za konverziju između tipova podataka niska i p-niska. Operator *string* ima jedan operand, p-nisku, a vraća karaktersku nisku koja je operandom reprezentovana. Operator *parsed by* ima dva operanda, nisku i neterminalni simbol, a vraća p-nisku koja je deo sheme opisane tim neterminalom. Tako, na primer, 'engl.' *parsed by* `etim` vraća sledeću p-nisku:

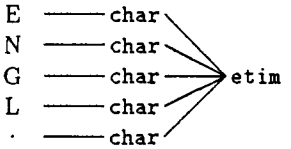


dok *string*('engl.' *parsed by* `etim`) vraća nisku 'engl.' Sama niska je specijalan slučaj p-niske čiji je koren obeležen sa `string` i čije jedino poddrvo je vrednost niske. Na primer,



Treći operator konverzije je *reparsed by* koji ima dva operanda, p-nisku i parcijalnu gramatiku, a vraća p-nisku koja je ponovo izračunata u skladu sa pravilima nove gramatike. Ovaj operator, ustvari, za svako pravilo nove gramatike `L` := $R_1 \ldots R_n$ zamenjuje svako pojavljivanje pod-p-niske `P` u čijem korenu je `L` p-niskom (*string* `P`) *parsed by* `L`.

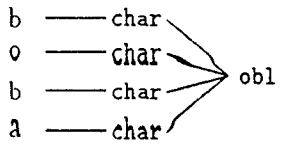
Mnogi od operatora koji su uključeni u jezik za manipulaciju podacima moraju da rade nad kolekcijama p-niski. Uključivanjem kolekcija u model baze zasnovan na p-niskama izbegava se uvođenje odvojenih operatora za rad sa kolekcijama. Tako se uvodi pojam *vektora* koji je i sam p-niska čiji je koren obeležen sa `vector`. *Skup* je p-niska koja nema istih poddrveta a njen je koren obeležen sa `set`. Za rad sa vektorima i skupovima uvode se mnoge funkcije koje se mogu proširiti i na proizvoljne p-niske. Tako funkcija *size* vraća broj poddrveta argumenta koji je p-niska. Na primer *size* ('engl.' *parsed by* `etim`) vraća vrednost 5. Operandi operatora spajanja (koji se označava zarezom) su vektori a rezultat je takođe vektor koji sadrži sva poddrveta vektora operanada. Operator *mapped onto* ima dva operanda: funkciju `f` sa jednim „slobodnim" argumentom i p-nisku `P`. Rezultat je p-niska u čijem je korenu takođe `P` a svako poddrvo je zamenjeno vrednošću funkcije `f` u kojoj je to poddrvo od `P` zamenilo „slobodni" argument funkcije. Tako. ako je `UVelika(x)` funkcija definisana nad p-niskama koja vraća kao vrednost p-nisku u kojoj su mala slova na listovima zamenjena odgovarajućim velikim slovom, onda izraz `UVelika()` *mapped onto* (`'engl.'` *parsed by* `etim`) vraća sledeću p-nisku:



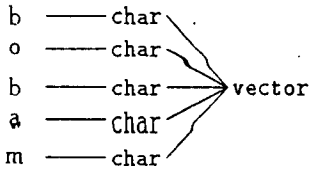
Operatori *root* i *subtrees* izdvajaju komponente operanda koji je p-niska. *root* `P` vraća neterminalni simbol koji je u korenu p-niske `P` a *subtrees* `P` vraća vektor p-niski koji se sastoji od svih poddrveta korena od `P`. Operator *with* ima dva operanda, neterminalni simbol i vektor p-niski a vraća p-nisku u čijem je korenu neterminalni simbol a poddrveta su mu ulazni vektor p-niski. Važi dakle:

$$root\ (\ n\ with\ L\ ) \equiv n$$
$$subtrees\ (\ n\ with\ L\ ) \equiv L$$

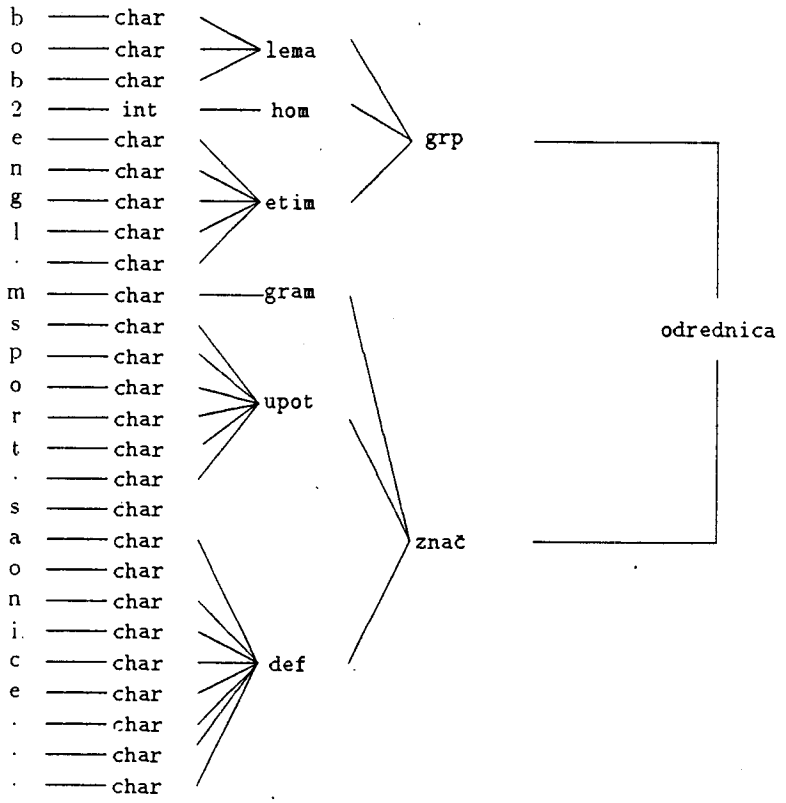
Operator selekcije je *in*. On ima dva operanda, neterminalni simbol i p-nisku a vraća p-nisku u čijem je korenu neterminalni simbol a do koje se prve dolazi u obilasku drva sa prvenstvom dubine (engl. *depth first search*). Na primer, `obl` *in* `E` ili, što je ekvivalentno, `obl` *in* `morf` *in* `E` vraća p-nisku:



Operator *every*...*in* za iste operande, neterminalni simbol i p-nisku, vraća vektor p-niski koje su sve podređene čvoru koji je obeležen ulaznim neterminalom, i to onim redom kako su posećene u obilasku sa prvenstvom dubine. Na primer, iskaz *every* `char` *in* `morf` vraća vektor:



Operatori transformacije transformišu strukturu p-niske. Prvi od njih je *transduced by* čiji su operandi p-niska `P` i gramatika `G` a izlaz je p-niska.



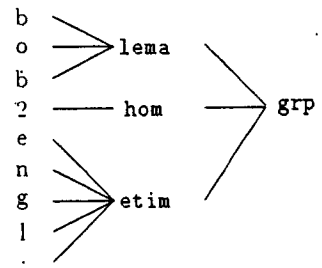
Slika 3.2: Transdukovana p-niska `E`

Rezultujuća p-niska dobija se na sledeći način: za svako pravilo L := $R_1 \ldots R_n$ gramatike G svako poddrvo od P u čijem je korenu L zamenjuje se sa L *with* $P_1 \ldots P_n$ gde su P_i:

$$P_i = \begin{cases} R_i \; in \; P, & \text{ako je } R_i \text{ neterminal} \\ R_i, & \text{inače} \end{cases}$$

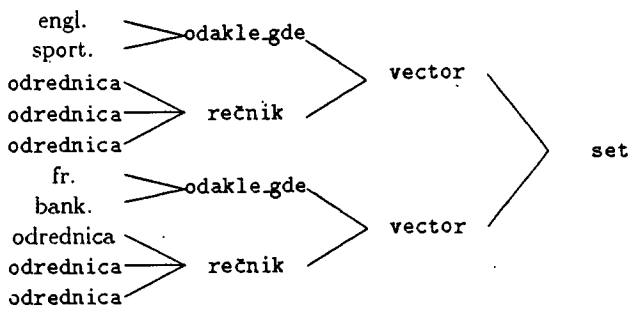
Na primer, ako je jedan operand gramatika G = { grp := lema hom etim; znač := gram upot def } a drugi p-niska E onda je vrednost iskaza E *transduced by* G p-niska E′ sa slike 3.2.

Operator transformacije *suppressing* uklanja nepotrebne detalje iz p-niske. On ima dva operanda, p-nisku i neterminal a vraća p-nisku iz koje je taj neterminal svugde uklonjen tako što su sva poddrveta čiji su koreni čvorovi obeleženi neterminalom direktno povezana sa roditeljem čvora. Tako izraz (grp *in* E′) *suppressing* char vraća p-nisku sa slike 3.3.



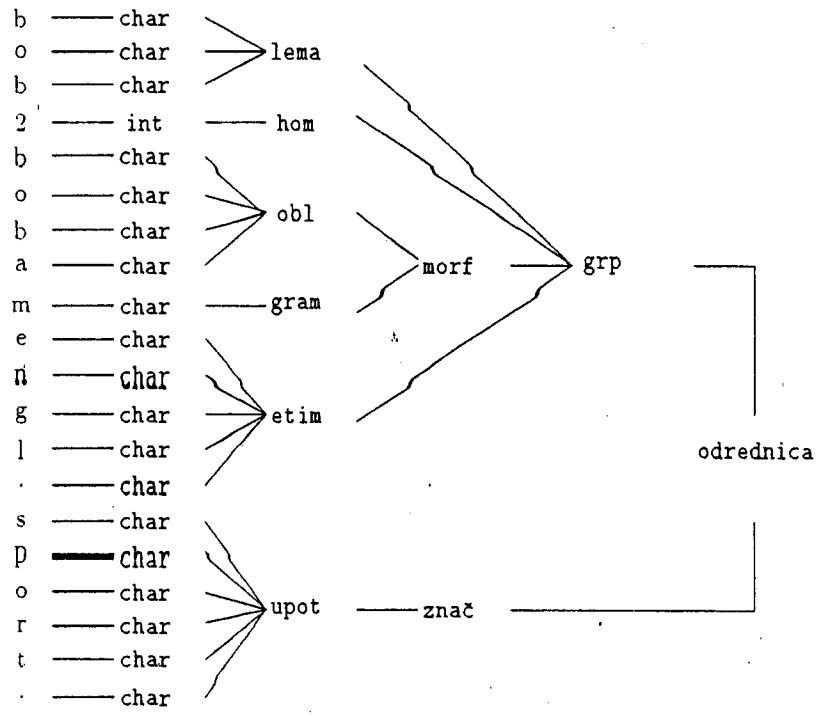
Slika 3.3: Vrednost izraza (grp *in* E′) *suppressing* char

Sledeći operator transformacije je *partitioned by* koji omogućava poređenja unutar p-niske a koji se može uporediti sa operatorom *group by* u SQL-u [103]. Za datu p-nisku P i funkciju F koja se može primeniti na svako poddrvo od P, operator *partitioned by* vraća p-nisku koja grupiše podrveta od P prema vrednosti funkcije F. Rezultat je p-niska koja predstavlja skup particija od kojih se svaka može predstaviti kao dvojka <F-vrednost, p-niska>, to jest kao vektor dimenzije dva, u kome druga komponenta ima koren obeležen sa P a za svako njeno poddrvo S je vrednost F(S) jednaka prvoj komponenti, F-vrednosti.



Slika 3.4: P-niska klasifikovanog rečnika

Na primer, ako je P p-niska koja predstavlja ceo rečnik i ako je OdakleGde sledeća funkcija:



Slika 3.5: Vrednost izraza E *where* BezDefinicije()

```
OdakleGde :=  proc(x)
              odakle_gde with (  string( etim in x ),
                                 string( upot in x ) )
              end;
```

onda izraz Rečnik *partitioned by* OdakleGde() proizvodi p-nisku sa slike 3.4. Izlazna p-niska je skup particija rečnika gde svaka particija sadrži one odrednice iz rečnika koje imaju poreklo u istom jeziku i koriste se u istom podjeziku.

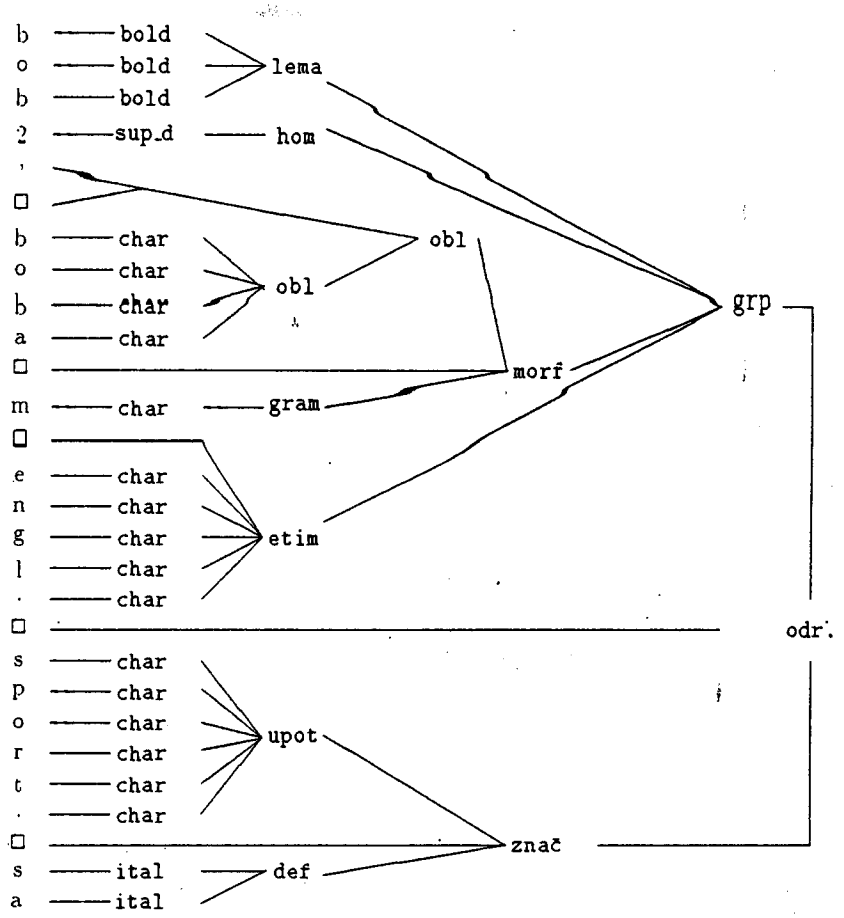
Poslednji operator transformacije je *where* koji ima dva operanda, p-nisku i bulovsku funkciju i vraća p-nisku iz koje su uklonjena sva poddrveta za koja je vrednost bulovske funkcije netačno. Operatori *where* i *partitioned by* su, očigledno povezani: P *where* F vraća p-nisku koja odgovara particiji skupa P *partitioned by* F koja je pridružena vrednosti tačno. Ovaj operator može se uporediti sa *where* klauzom SQL iskaza [103]. Na primer, za funkciju BezDefinicije:

```
BezDefinicije :=  proc(x)
                  root x <> def
                  end;
```

izraz E *where* BezDefinicije() vraća p-nisku sa slike 3.5.

Može se pokazati da jačina ovog jezika za manipulaciju podacima u strukturiranoj tekstualnoj bazi podataka ne zaostaje za jačinom upitnih jezika relacione baze podataka [37].

Na ovom mestu treba istaći da model strukturirane tekstualne baze podataka ne zavisi od upotrebe SGML-a. Šta više, neki elementi modela su koncipirani daleko pre nastanka SGML-a [36]. Oba koncepta su, međutim, nastala iz potrebe za isticanjem svojstva strukturiranosti teksta koji tekst suštinski razlikuje od niske karaktera. Nezavisnost modela od SGML-a može se ilustrovati na istom primeru pojednostavljenog rečnika, preformulisanjem gramatike iz 3.1 na takav način da se isti strukturni elementi rečnika izdvajaju na osnovu tipografskih i ortografskih konvencija papirne verzije rečnika registrovane u mašinski-čitljivom obliku. Pre-



Slika 3.6: P-niska za opis odrednice na osnovu njene tipografske i ortografske prezentacije

formulisana gramatika, konstruisana prema istom primeru iz rečnika [116], data je u tabeli 3.2 a p-niska iste odrednice na slici 3.6. Korišćenje ovakve gramatike za modeliranje strukturirane tekstualne baze pretpostavlja korišćenje vrlo preciznih i nedvosmislenih tipografskih i ortografskih konvencija što kod preuzimanja starih papirnih verzija teksta najčešće nije slučaj. Iz ovog primera se vidi da se tekst može modelirati i bez eksplicitnih SGML oznaka. Treba, međutim, naglasiti da i sam SGML, kroz korišćenje podatkovnih etiketa i kratkih referenci, omogućava implicitno obeležavanje teksta na osnovu sličnih konvencija.

3.2.2 GOEDEL — sistem za manipulaciju tekstualnih baza podataka

Model strukturirane tekstualne baze podataka, odnosno podatkovni tip p-niska, može se realizovati na više načina. Jedna mogućnost je realizacija drva raščlanjavanja kao rekurzivne strukture sa eksplicitno kodiranim pokazivačima poddrveta braće i roditelja. Druga mogućnost je implicitna reprezentacija drveta označavanjem teksta SGML etiketama. Treća mogućnost je izgradnja tabela preslikavanja neterminalnih simbola u slogove koji predstavljaju odgovarajuća poddrveta p-niske. Ovaj pristup je primenjen u interaktivnom jeziku GOEDEL[1][33] koji predstavlja parcijalnu reprezentaciju predstavljenog modela.

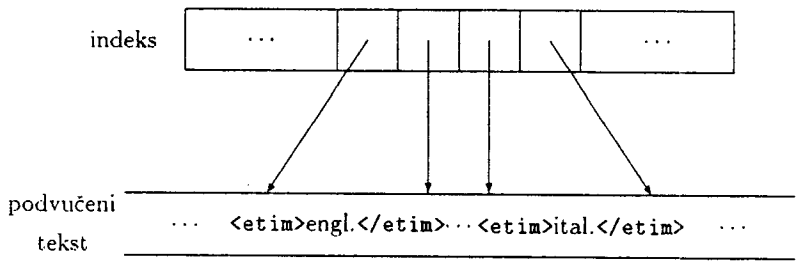
[1] Jezik je nazvan po čuvenom matematičaru K. Goedelu, izgleda prvenstveno zbog prisustva niske 'oed' u njegovom imenu.

rečnik:=	odrednica ('\n' odrednica)+	lema:=	bold+
odrednica:=	grp ' ' znač	gram:=	char+
grp:=	lema hom? morf? (' ' etim)?	obl:=	char+
morf:=	(', ' obl)* ' ' gram	etim:=	char+
znač:=	(upot ' ')? def	upot:=	char+
hom:=	super_dig+	def:=	ital+

Tabela 3.2: Gramatika za opis strukture rečničke odrednice na osnovu njene tipografske i ortografske prezentacije

Jezgro GOEDEL-a čini simbolički algebarski sistem Maple koji karakterišu dinamički tipovi podataka, efikasno upravljanje memorijom, snažna arhitektura procedura i efikasna implementacija skupova i listi. Njegova posebna karakteristika su asocijativni nizovi koji nalikuju SNOBOL tabelama i koji predstavljaju takvu strukturu u kojoj se vrednost indeksa pamti zajedno sa dodeljenom vrednošću [40], [41]. Nad ovo jezgro nadgrađene su osnovne funkcije za rad sa niskama, delimično su implementirane p-niske i podskup operatora za manipulaciju podacima.

Puna reprezentacija p-niske uključuje tip neterminala u korenu, pridruženu gramatiku i podvučeni tekst. Reprezentacija koja je ugrađena u jezik GOEDEL koristi samo neterminal u korenu i podvučeni tekst, što znači da se za kodiranje p-niske koristi samo identifikator neterminala i dva pokazivača teksta koja predstavljaju krajnje tačke podvučenog teksta. Ovakva implementacija koja se naziva *indeks p-niske* predstavljena je na slici 3.7. Ovakav indeks i sam predstavlja p-nisku koja je rezultat operatora *every* X *in* Y, gde je X indeksirani neterminal a Y ceo tekst, odnosno koren p-niske pod koju je podvučen ceo tekst. U indeksu p-niske svi pokazivači su registrovani u monotono rastućem redosledu.



Slika 3.7: Indeks p-niske za neterminal etim

Izgradnja ovakvih indeksa za svaki neterminal podržava efikasnu implementaciju operatora *in* i *every...in* a dodatak još nekih podatkovnih struktura omogućava raščlanjavanje p-niske prvog nivoa, to jest, pronalaženje broja i tipova njenih neposrednih naslednika.

Nad ovako reprezentovanom p-niskom realizovan je podskup operatora za manipulaciju podacima u koji nisu uključeni operatori *parsed by*, *reparsed by*, *transduced by* i *partitioned by*.

Premda koristi pojednostavljenu reprezentaciju p-niske i podskup operatora, GOEDEL se pokazao kao efikasno sredstvo za pronalaženje p-niski u bazi podataka i njihovu rekonstrukciju kao dinamičkih objekata. Iako se njegovo korišćenje može ograničiti na ekstrahovanje infor-

macija, i kao takvo sredstvo je intenzivno korišćen od strane leksikografa i lingvista s ciljem ekscerpcije najrazličitijih informacija iz OED2 baze podataka [88], njegove mogućnosti su daleko veće. On omogućava procesiranje baze koje uključuje odbacivanje nekih informacija, dodavanje novih i promenu postojećih.

Reprezentativna primena ovog sistema je izrada radne verzije trećeg izdanja *Shorter Oxford English Dictionary* (skraćeno SOED) na osnovu OED2 [19]. Prvo izdanje SOED rečnika iz 1933. godine izrađeno je na osnovu prvog izdanja OED rečnika kao njegova popularna verzija. No kroz više dopuna i drugo izdanje on se udaljio u velikoj meri od uzora, posebno u izboru primera i klasifikaciji značenja. Da bi treće izdanje SOED zadržalo svoj specifičan stil trebalo je izvršiti sledeću obradu OED2 baze:

- selekcija odrednica koja je obavljena na osnovu različitih informacija: datuma poslednjeg citata, autora citata, dela iz koga je citat isl.;
- izostavljanje delova članka uz odrednicu, kao što je, na primer, informacija o etimologiji, u zavisnosti od datuma, većine citata, isl;
- preuređivanje informacija, kao što je sortiranje značenja u redosledu datuma citata, izmeštanje informacija o frazama, kolokacijama i derivacijama iz pasusa pojedinačnog značenja na kraj članka odrednice isl;
- prevođenje informacija, kao što je prevođenje informacije o izgovoru iz jednog fonetskog alfabeta u drugi, prevođenje datuma citata iz jednog formata u drugi isl.

Imajući u vidu složenost postavljenog zadatka, te obim i kompleksnost OED2 baze, konačan uspeh projekta može se pripisati punoj obeleženosti teksta OED2 rečnika SGML etiketama koja je omogućila izradu indeksa p-niski i korišćenje jezika GOEDEL. Premda bi se izloženi zadatak mogao obaviti i korišćenjem nekog opšteg programskog jezika kakav je C ili nekog niskovno orijentisanog programskog jezika kakav je SNOBOL4 [32], takav pristup bi zahtevao nesrazmerno više programerskog rada koji bi rezultovao programom za jednokratnu upotrebu.

Uprkos više uspešnih primena, ovaj jezik nije ušao u širu primenu, pre svega zbog neefikasnosti svojstvene interpretatorima a zatim i zbog nepotpune implementacije izloženog modela. Stoga su 1991. godine gramatički definisani operatori ponovo dizajnirani i implementirani u obliku jezika za manipulaciju podacima koji se uključuje u C, i koji je ovog puta nazvan GDL [136].

3.2.3 PAT — sistem za pretraživanje tekstualnih baza podataka

U prethodnom odeljku predstavljen je programki jezik GOEDEL koji omogućava rukovanje bazom podataka definisanom gramatikom, realizovan nad pojednostavljenim konceptom p-niske i korišćenjem podskupa operatora za manipulisanje podacima. Na drugačijem konceptu teksta, a u okviru iste istraživačke laboratrije i istog OED2 projekta, nastao je programski sistem PAT [119] za pretraživanje teksta. Model teksta podvučen pod sistem PAT je takav da dopušta potpuno slobodno unošenje oznaka u tekst koje zadovoljava samo najopštije zahteve kakav je, na primer, pravilna ugnježdenost elemenata (zabranjeno je, dakle, označavanje `<a>...<b>...</a>...</b>`). Drugim rečima, oznake teksta ne moraju biti organizovane nikakvom gramatikom i sistem radi podjednako sa takvim tekstom kao i sa tekstom koji je organizovan gramatikom tipa SGML DTD.

PAT se zasniva na konceptu indeksiranja teksta, pri čemu *indeksne elemente* korisnik sistema može sam da bira. To mogu biti pojedinačni karakteri, reči ili neki drugi segmenti teksta

```
fraza := indeksni_element [graničnik+] fraza;
indeksni_element := ograničeni_element | nametnuti_element;
ograničeni_element := elm_kar+;
nametnuti_element := samostalni_kar | signalni_kar elm_kar+;
elm_kar := a |... A |... d0 |... d9 |kosa_crta;
graničnik := blanko | tačka | zarez | veće | dvotačka |...
samostalni_kar := crtica;  signalni_kar := manje | amperzan;
   crtica := '-';
   manje := '<';      ...          amperzan := '&';
   a := 'a';          ...          z := 'z';
   A := 'A';          ...          Z := 'Z';
   d0 := '0';         ...          d9 := '9';
   kosa_crta := '/';
   blanko := ' ';     tačka := '.';     zarez := ',';
   veće := '>';       dvotačka := ':';  ...
```

Tabela 3.3: Gramatika za definisanje indeksnih elemenata u sistemu PAT

koje korisnik može sam definisati korišćenjem gramatike. U tabeli 3.3 data je jedna moguća gramatika za definisanje indeksnih elemenata. Indeksni elementi definisani ovom gramatikom su niske sačinjene od karaktera indeksnih elemenata razdvojene graničnicima, niske sačinjene od karaktera indeksnih elemenata koje počinju signalnim karakterom i samostalni karakteri. U poslednja dva slučaja indeksni elementi ne moraju biti razdvojeni graničnicima. Tako bi, na primer, niska 'baba-devojka' sadržala tri indeksna elementa: 'baba', '-' i 'devojka' dok bi niska 'i/ili' sadržala jedan indeksni element: 'i/ili'. Niske '<etim>' i '</etim>' obe sadrže po jedan indeksni element: '<etim', odnosno '</etim'. Skupovi graničnika, samostalnih karaktera, signalnih karaktera i karaktera indeksnih elemenata moraju biti disjunktni.

Svaki indeksni element je početak polubeskonačne niske, si-niske (skraćenica od *semi-infinite string*) koja se nastavlja sve do kraja teksta. Ukoliko se si-niska koristi i iza kraja teksta smatra se da je dopunjena specijalnim null-karakterima koji se razlikuju od svih mogućih karaktera u tekstu. Prilikom obrade upita, sistem pronalazi sve karaktere kojima počinju si-niske a koji se slažu sa obrascem zadatim u upitu. Svaka si-niska je jednoznačno određena pozicijom od koje počinje a za dati tekst ta pozicija se može jednostavno izraziti celim brojem. Najvažnija operacija na si-niskama je operacija leksikografskog poretka si-niski i ona se sastoji od poređenja njihovog sadržaja. Treba primetiti da osim ako se si-niska poredi sama sa sobom, dve si-niske nikad ne mogu biti jednake. Na primer, ako su indeksni elementi pojedinačni karakteri, onda su neke si-niske datog teksta:

```
Tekst        Ko s vragom tikve sadi, o glavu mu se razbijaju.

si-niska 1   Ko s vragom tikve sadi, o glavu mu se razbijaju.
si-niska 4   s vragom tikve sadi, o glavu mu se razbijaju.
si-niska 8   agom tikve sadi, o glavu mu se razbijaju.
si-niska 20  adi, o glavu mu se razbijaju.
si-niska 29  avu mu se razbijaju.
si-niska 45  aju.
```

Si-niske iz ovog primera porede se na sledeći način pod pretpostavkom da se koristi ASCII kodna sekvencija:

$$1 < 20 < 8 < 45 < 29 < 4$$

Na svaki upit PAT odgovara rezultujućim skupom koji može biti skup tačaka slaganja kojima otpočinju si-niske ili skupom regija, pri čemu je regija podniska teksta koja počinje i završava na određenim pozicijama. Regije se kreiraju pomoću definicije regije koja određuje uslove za određivanje prvog i poslednjeg karaktera regije. Definicija regije odgovara novoj gramatici koja ponovo raščlanjava tekst i omogućava da se u njemu identifikuju novi delovi. Regije se u rezultujućem skupu regija ne smeju preklapati. Rezultujući skupovi se mogu imenovati i tako imenovani koristiti u narednim upitima.

pravilo indeksiranja

```
fraza :=
  indeksni_element [graničnik+] fraza;
A := 'A';
...
Z := 'Z';
```

pravilo normalizacije

```
fraza :=
  indeksni_element ' ' fraza;
A := 'a';
...
Z := 'z';
```

Tabela 3.4: Pravila prevođenja koja normalizuju obrazac i tekst

Operaciju pretraživanja PAT obavlja tek pošto izvrši normalizaciju i fraze i obrasca. Normalizacija ugrađena u PAT preslikava sve graničnike u jedan blanko ali korisnik može izabrati svoju normalizaciju zadavanjem skupa pravila prevođenja koja redefinišu one delove gramatike indeksnih elemenata koji podležu normalizaciji. U tabeli 3.4 data su neka moguća pravila prevođenja za gramatiku indeksnih elemenata iz 3.3. Ova pravila prevođenja govore da se u procesu normalizacije sekvencija graničnika prevodi u jedan blanko karakter a sva velika slova zamenjuju se odgovarajućim malim slovima. Prilikom definisanja svojih pravila prevođenja korisnik mora da vodi računa da se zamena mora izabrati iz iste klase iz koje je i karakter koji se zamenjuje.

Sve PAT operacije mogu se klasifikovati u šest osnovnih tipova: leksičko pretraživanje, poziciono pretraživanje, frekvencijsko pretraživanje, definicija regija, restrikcija, augmentacija. Rezultujući skup leksičkog, pozicionog i frekvencijskog pretraživanja je obavezno skup tačaka slaganja a rezultujući skup definicije regija je skup regija. Rezultat restrikcije i augmentacije može biti i skup tačaka slaganja i skup regija što zavisi od operanada. Precizan opis sintakse PAT izraza i njihovo značenje dat je u [119], dok će ovde operacije biti predstavljene samo u osnovnim crtama.

U leksičkom pretraživanju obrazac se može zadati bilo kao obična niska s ili kao interval $s_1..s_2$, pri čemu su s_1 i s_2 niske. Ako je obrazac niska s onda su u rezultujućem skupu sve pozicije u tekstu na kojima otpočinje fraza koja se slaže sa s. Ako je obrazac $s_1..s_2$ onda su u rezultujućem skupu pozicije svih fraza koje počinju bilo niskom s_1, bilo niskom s_2 bilo nekom trećom niskom koja se nalazi između ove dve u leksikografskom poretku. Leksikografski poredak je ovde indukovan kolacionom sekvencijom, o čemu će biti reči kasnije. Na primer, ako je tekst:

```
<hi>Pretraživanje</hi> je traženje u kome se traži
```

onda obrazac `tra` vraća skup od dva elementa $\{27, 46\}$. To su pozicije u tekstu onih fraza koje počinju traženim obrascem. U ovom, i u narednim, primerima uvek se pretpostavlja

gramatika iz 3.3 za definisanje indeksnih elemenata i pravila normalizacije iz 3.4. Treba voditi računa da fraza uvek otpočinje indeksnim elementom, te stoga rezultujući skup ne sadrži poziciju 8. Ako bi obrazac bio `j..pu` rezultujući skup bi bio $\{5, 24, 38\}$.

Poziciono pretraživanje se koristi za pronalaženje karaktera koji je na apsolutno zadatoj poziciji u tekstu ili za pronalaženje karaktera koji se nalaze na datom rastojanju, ulevo ili udesno, od tačaka slaganja iz datog skupa. Na primer, izraz `shift.5 </hi` primenjen na gornji tekst vraća poziciju blanko karaktera iza etikete zatvaranja `</hi>`, odnosno vraća skup $\{23\}$.

Frekvencijsko pretraživanje omogućava pronalaženje najfrekventnijih podniski koje su celi indeksni elementi a kojima počinju fraze iz skupa tačaka slaganja koji je operand izraza. Na primer, izraz `signif 'nik'` vraća skup tačaka slaganja koje odgovaraju prvom karakteru pojavljivanja u tekstu najfrekventnije reči koja počinje sa 'nik'. Druga vrsta frekvencijskog pretraživanja pronalazi podniske koje se ponavljaju u tekstu tako što iz skupa tačaka slaganja izdvaja one koje imaju najduže (normalizovano) proširenje takvo da u skupu postoje bar dva takva proširenja. Tako izraz `lrep.5 'nik'` vraća skup tačaka slaganja kojima otpočinju fraze koje sve počinju istom (najdužom) niskom koja počinje sa 'nik' a ima bar 5 karaktera.

Za definisanje regija koristi se izraz `docs` $e_1..e_2$ gde su e_1 i e_2 skupovi tačaka slaganja. e_1 daje uslove za prvi karakter nove regije a e_2 uslove za poslednji karakter regije. Na primer, izraz `docs (shift.4 "<hi")..(shift.-1 "</hi")` vraća skup regija u kome je svaki element podniska koja počinje prvim karakterom iza etikete otvaranje elementa `hi` a završava poslednjim karakterom pre etikete zatvaranja. Pretpostavka je ovde da etikete otvaranja ne sadrže definicije vrednosti atributa. U rezultujućem skupu regija elementi se ne mogu preklapati, tako da, recimo iz teksta:

```
... <hi>Ovo je <hi>specijalno</hi> istaknuti tekst</hi>...
```

gornji izraz vraća skup regija koji sadrži samo `specijalno` ali ne i podnisku koja je okružuje.

op	*skup* e_1	*objašnjenje*
including	reg.	sadrži bar neki element iz e_2
^	tač./reg.	poklapa se sa nekim elementom iz e_2
-	tač./reg.	ne poklapa se ni sa jednim elementom iz e_2
fby.*n*	tač./reg.	prethodi neki element iz e_2 za najviše n karaktera
near.*n*	tač./reg.	odvojen od nekog elementa iz e_2 za najviše n karaktera
within	tač./reg.	sadržan u nekoj regiji iz e_2

Tabela 3.5: Operatori restrikcije ugrađeni u PAT

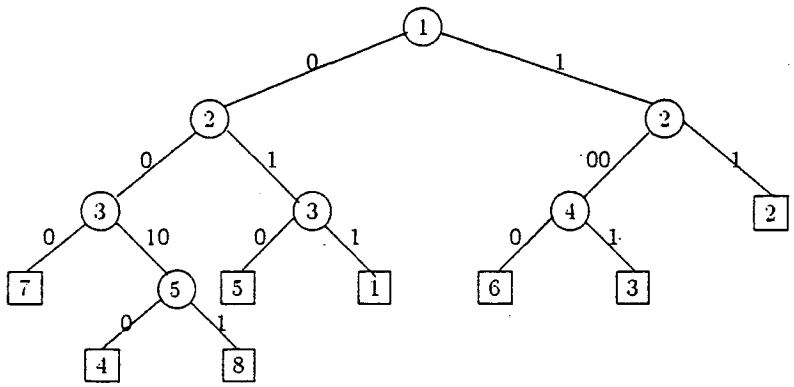
U PAT-u postoje binarni operatori koji vrše restrikciju datog skupa. Oni se koriste u obliku e_1 *op* e_2 i vraćaju skup koji je podskup skupa koji odgovara izrazu e_1 i sadrži one elemente koji zadovoljavaju uslov izražen kroz *op* e_2. Rezultujući skup može biti skup regija ili skup tačaka slaganja što zavisi od toga kakav je rezultujući skup izraza e_1. Dozvoljeni operatori restrikcije i njihove funkcije dati su u tabeli 3.5.

Operatori restrikcije '^' i '-' odgovaraju skupovnim operacijama preseka i razlike koje su prisutne u većini dokumentalističkih sistema za pretraživanje. Njih dopunjuje operator augmentacije '+' koji odgovara skupovnoj operaciji unije i koji kao rezultujući skup vraća uniju rezultujućih skupova operanada. Rezultujući skup može biti skup regija ili skup tačaka

slaganja što zavisi od toga kakvi su rezultujući skupovi operanada. Ukoliko je samo jedan operand skup regija ili ako se regije u rezultujućim skupovima preklapaju, ti skupovi regija se prevode pre izvršavanja operacije u skupove tačaka slaganja koji sadrže početne tačke regija.

Efikasnost sistema PAT kao i repertoar predviđenih operanada čvrsto su vezani za reprezentaciju teksta unutar sistema [38]. Reprezentacija teksta zasniva se na strukturi *Patrica* drveta koja predstavljaju varijantu binarnih digitalnih drveta [124]. Njih karakteriše postojanje dve vrste čvorova. Spoljašnji čvorovi sadrže ključne vrednosti a unutrašnji čvorovi sadrže redni broj bita koji se poredi (ili, alternativno, pomak brojača bita) i pokazivače dva podrveta, levog na koje se prelazi ako je tekući bit 0 i desnog na koje se prelazi ako je tekući bit 1. Beleženje brojača bita poređenja dozvoljava da se eliminišu unutrašnji čvorovi sa jednim naslednikom, to jest, u Patrica drvetima svaki unutrašnji čvor proizvodi „korisno" grananje. Koncepcija Patrica drveta pretpostavlja izjednačavanje ove dve vrste čvorova što nije od značaja za prikaz korišćenja ove strukture u sistemu PAT.

Karakteristika PAT drveta je da za tekst dužine n drvo ima tačno n spoljašnjih čvorova i tačno $n-1$ unutrašnjih čvorova, što znači da je veličina drveta izražena brojem čvorova $O(n)$. Kao i kod ostalih binarnih digitalnih drveta, struktura PAT drveta zavisi samo od bitovske strukture ključeva, a ne i od redosleda bitova. Za normalan tekst drvo bi trebalo da bude relativno dobro balansirano, što znači da bi broj bitova poređenja pri pretraživanju bio proporcionalan sa $\log n$. Na slici 3.8 prikazano je Patrica drvo za tekst koji je predstavljen sekvencijom bitova 01100100010111, pošto je u drvo umetnuto prvih 8 si-niski. (Jednostavnosti radi tekst je ovde predstavljen sekvencijom bitova; najmanji indeksni element teksta je obično karakter.) Pozicije bitova broje se od jedan nadalje, sleva udesno. Spoljašnji čvorovi u slučaju PAT-a sadrže si-niske, odnosno, pozicije njihovih početaka.

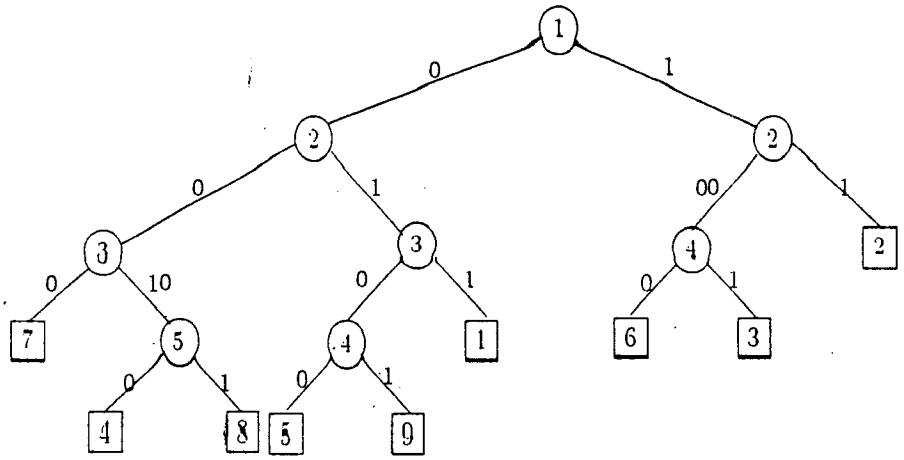


Slika 3.8: Patrica drvo za prvih 8 si-niski bitovske sekvencije

U PAT drvetu grane postoje samo za realizovane niske. Na primer, u datom primeru zasebne grane postoje za sekvencije '0' i '00' jer se u tekstu realizuju i sekvencije '00' i '01', odnosno '000' i '001'. Sekvencija '000' se završava spoljašnjim čvorom jer je jedina u zadatom tekstu. Sekvencijom '00' stiže se u najlevlji unutrašnji čvor drveta koji poredi treći bit. Desnom granom se iz tog čvora stiže u čvor koji poredi peti bit, što je rezultat činjenice da sekvencije '00100' i '00101' postoje u tekstu ali ne i sekvencije koje na četvrtoj poziciji imaju bit 1. U procesu pretraživanja, do spoljašnjih čvorova 4 i 8 stiže se bitovskim obrascima '00100', '00101', odnosno, '00110' i '00111'. Tek sravnjivanjem obrasca sa tekstom

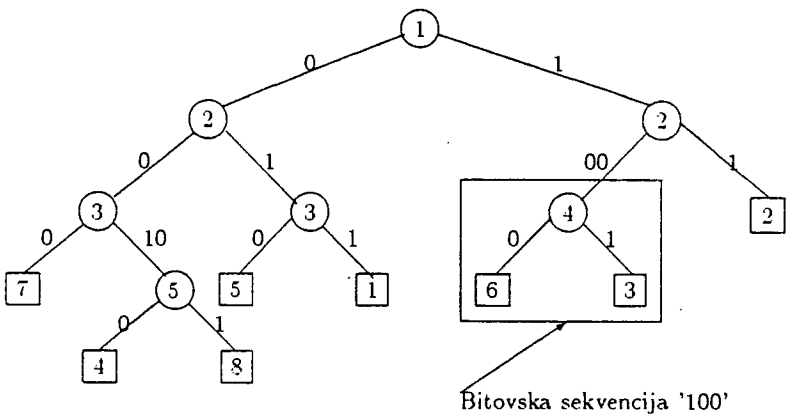
na pozicijama zapisanim u spoljašnjem čvoru može se utvrditi da li je obrazac prisutan na određenoj poziciji ili ne. Treba se podsetiti da bitovske sekvencije zapisane uz grane drveta na slikama nisu zaista i zabeležene u strukturi drveta.

Da bi se u PAT drvo uključila i si-niska na poziciji 9 spoljašnji čvor koji odgovara si-niski na poziciji 5 treba zameniti unutrašnjim čvorom koji poredi, u ovom slučaju, četvrti bit jer se na tom bitu razlikuju si-niske sa pozicije 5 i pozicije 9. Tako se dobija drvo sa slike 3.9.



Slika 3.9: Patrica drvo za prvih 9 si-niski bitovske sekvencije

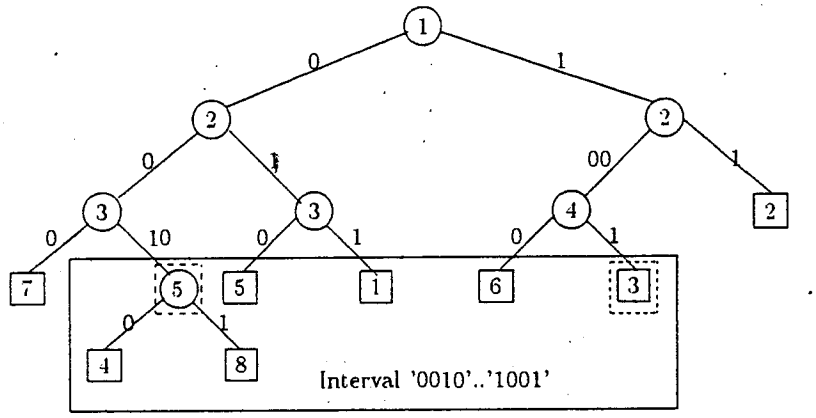
Korišćenjem ovakve strukture podataka, jednostavno i efikasno mogu se implementirati upiti ugrađeni u sistem PAT, što će i biti pokazano na primerima operacija leksičkog pretraživanja, frekvencijskog pretraživanja i restrikcije. Na osnovu konstrukcije, svako poddrvo PAT drveta sadrži sve si-niske sa datim prefiksom. Prema tome, traženje prefiksa u PAT drvetu sastoji se od poređenja sve dok se prefiks ne iscrpi ili dok se ne stigne do spoljašnjeg čvora. U oba slučaja treba još proveriti da li su određeni bitovi opravdano preskočeni a za to je potrebno proveriti **ma koju** si-nisku iz poddrveta. Ako se traži, recimo bitovska sekvencija '100', rezultujući skup odgovara svim spoljašnjim čvorovima koji se nalaze ispod unutrašnjeg čvora u koji se stiže tom sekvencijom, kao što je prikazano na slici 3.10.



Slika 3.10: Traženje prefiksa 100 u prvih 8 si-niski bitovske sekvencije

Traženje bitovske sekvencije '101' vodi u isti unutrašnji čvor ali sravnjivanje sa ma kojom od si-niski koju on sadrži pokazuje da treći bit nije opravdano preskočen te je rezultujući skup prazan. Dubina nasumičnih Patrica drveta je $O(\log n)$ [75], što znači da se prefiks u PAT drvetu može naći u vremenu $O(\log n)$ i to vreme uopšte ne zavisi od veličine odgovora. U praksi je dužina obrasca manja od $O(\log n)$ pa je vreme pretraživanja proporcionalno dužini obrasca.

Ako je obrazac leksičkog pretraživanja interval, u datom primeru recimo interval '0010' .. '1001', rezultujući skup obuhvata sve spoljašnje čvorove koji se nalaze ispod unutrašnjih koji odgovaraju granicama intervala, kao i sve spoljašnje čvorove koji se nalaze „između" njih. Drugim rečima, od čvora račvanja, uzimaju se sve desne grane čvorova koje su na višem nivou od leve granice intervala i sve leve grane čvorova koje su na višem nivou od desne granice (slika 3.11).

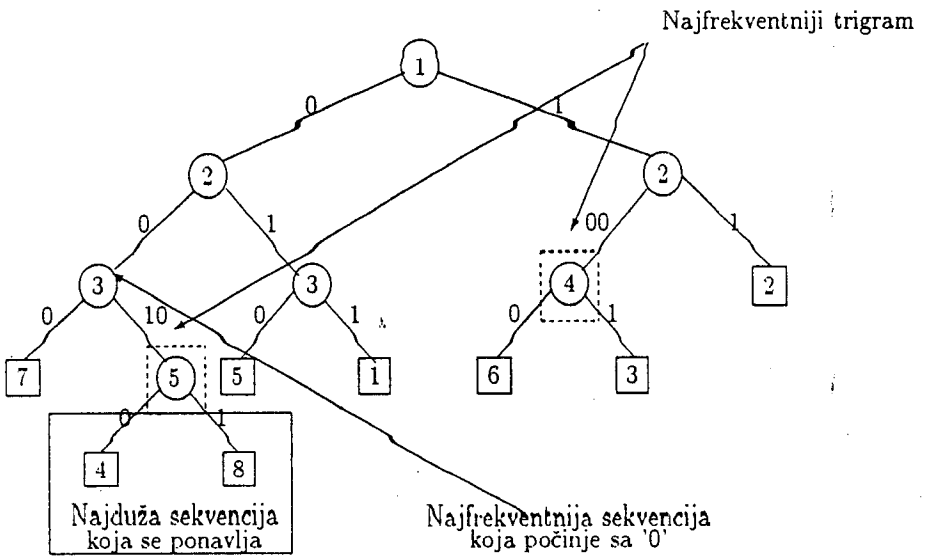


Slika 3.11: Traženje si-niski iz intervala '0010'..'1001' među prvih 8 si-niski bitovske sekvencije

Treba primetiti da odgovor na intervalni upit sadrži i u najgorem slučaju $O(dubina)$ poddrveta (kada ih ima tačno $2 * dubina - 1$) pa je potrebno vreme pretraživanja $O(\log n)$ i ono ne zavisi od veličine odgovora.

Frekvencijsko pretraživanje može podrazumevati pronalaženje najfrekventnije niske date dužine, na primer, pronalaženje najfrekventnijeg trigrama što se svodi na pronalaženje poddrveta sa najvećim brojem čvorova čiji koren je na dubini od 3 simbola od korena drveta. U datom primeru, najfrekventniji trigrami su bitovske sekvencije '001' i '100'. Frekvencijsko pretraživanje može, takođe, podrazumevati pronalaženje najfrekventnije niske proizvoljne dužine koja počinje datim prefiksom. U datom primeru, od bitovskih sekvencija koje počinju sekvencijom '0' najfrekventnija je '00' kojom otpočinju tri si-niske na pozicijama 4, 7 i 8. Pronalaženje najduže niske koja se ponavlja (bar dva puta) svodi se na pronalaženje najdubljeg unutrašnjeg čvora u smislu rednog broja bita koji se poredi. Tako je u datom primeru, najduža bitovska sekvencija koja se ponavlja '0010' (slika 3.12).

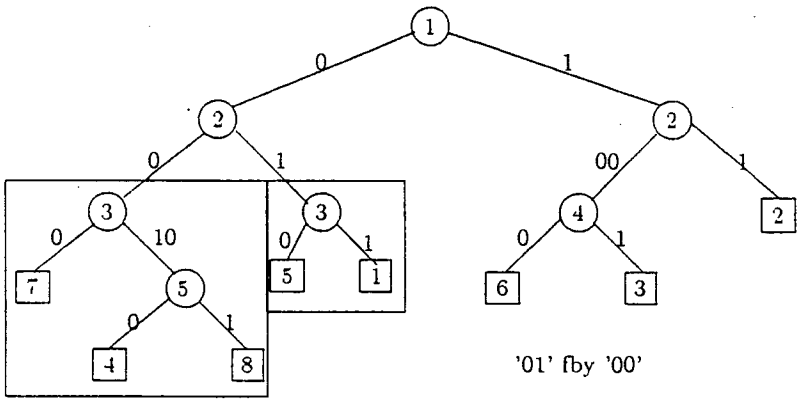
Najduže ponavljanje za dati tekst se može pronaći u trenutku konstruisanja PAT drveta. Pronalaženje (nekog) najdužeg ponavljanja među si-niskama koje dele zajednički prefiks se takođe može efiksano uraditi u vremenu $O(dubina)$ ukoliko se u svakom unutrašnjem čvoru čuva jedan bit informacije o tome koje je poddrvo dublje. Za pronalaženje svih najdužih ponavljanja potreban je još jedan bit informacije za slučaj kada su oba poddrveta iste dubine. Tada je za pretraživanje potrebno vreme logaritamski proporcionalno dubini drveta a linearno



Slika 3.12: Frekvencijska pretraživanja među prvih 8 si-niski bitovske sekvencije

proporcionalno broju odgovora.

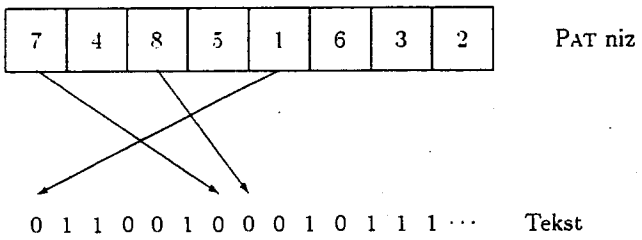
Restrikcija jednog skupa pod uslovom izraženim drugim skupom svodi se na pronalaženje spoljašnjih čvorova koji odgovaraju izrazima koji definišu te skupove. Ovde se primenjuje sledeći jednostavan algoritam: prvo se pronalaze oba skupa, zatim se manji od njih sortira a na kraju se prolazi kroz nesortirani skup i proverava svaka si-niska u njemu u odnosu na sve si-niske u sortiranom. Na sledećoj slici ilustrovan je operator restrikcije *fby* na primeru '01' fby '00': rezultujući skup $\{4,7,8\}$ obrasca '00' poredi se sa rezultujućim skupom $\{1,5\}$ obrasca '01'. Rezultat operacije je onaj podskup od $\{1,5\}$ u kome su samo one pozicije, koje se uvećane za dužinu obrasca '01' nalaze u skupu $\{4,7,8\}$, a to je skup $\{5\}$ (slika 3.13).



Slika 3.13: Restrikcija skupa si-niski drugim skupom si-niski

Ako su m_1 i $m_2, (m_1 < m_2)$ veličine rezultujućih skupova ovaj algoritam se završava u vremenu $(m_1 + m_2) \log m_1$ što može biti neadekvatno samo u slučaju kada su m_1 ili m_2 proporcionalni sa n.

Za vrlo dugačke tekstove memorijski zahtevi koje nameće struktura Patrica drveta su tako veliki da ih je teško zadovoljiti. Svaki unutrašnji čvor zahteva od 3 do 4 memorijske reči a svaki spoljašnji čvor jednu memorijsku reč a to znači da je potrebno ukupno $18n$ karaktera za indeksiranje n karaktera. Za reprezentaciju Patrica drveta je moguće primeniti različite tehnike za povećanje efikasnosti obilaženja drveta i za uštedu memorijskog prostora, kakvo je, na primer, smeštanje spoljašnjih čvorova u vedrice. U okviru sistema PAT Patricia drveta su realizovana u obliku Pat nizova koji omogućavaju skladištenje istih informacija sa istom efikasnošću u daleko manje memorijskog prostora. Ideja se sastoji u tome da se spoljašnji čvorovi smeste u niz u istom relativnom odnosu u kome se nalaze u drvetu, kao što pokazuje slika 3.14.



Slika 3.14: Realizacija Pat nizova

Sravnjivanje niske se onda može realizovati pomoću indirektnog binarnog pretraživanja preko PAT niza. Lako se može videti da ovakav niz sadrži većinu informacija sadržanih u drvetu a vreme obilaska drveta se povećava za faktor $O(\log n)$, ali samo za neke operacije. Sve u svemu, može se reći da za tekst dužine n pretraživanje PAT indeksa zahteva $O(\log n)$ pristupa disku, što za tekst kakav je OED2, čini oko 50–60 pristupa disku. Koncipirano je i više nad-indeksa koji dalje redukuju broj pristupa disku i čine sistem pogodnim za korišćenje na sporijim medijumima, kakav je CD-ROM.

Najnovije verzije PAT-a uključuju još dve zanimljive mogućnosti. To je pre svega pretraživanje zasnovano na regularnim izrazima i realizovano konceptom Mealy-jevih mašina [136]. Druga mogućnost je aproksimativno sravnjivanje niske koje dozvoljava da se pronalaze sve niske koje su na udaljenosti ne većoj od k od date niske, a gde neslaganja mogu proisticati od umetanja, izbacivanja ili zamene karaktera. Ova mogućnost se zasniva na primeni efikasnih algoritama za aproksimativno sravnjivanje niski [11].

Efikasnost sa kojom sistem PAT pretražuje velike kolekcije tekstova prevazilazi efikasnost većine sistema slične namene. Njegova fleksibilnost ostvarena kroz mogućnost korisnika da sam bira indeksne elemente te da pretraživanje podjednako zasniva kako na sadržaju teksta tako i na njegovoj strukturi, izraženoj kroz lako prepoznatljive etikete, povećavaju njegovu upotrebljivu vrednost. Treba međutim istaći da su ove prednosti ponekad ostvarene nauštrb opštosti i upotrebne vrednosti dobijenih rezultata. Neki od tih nedostataka će ovde biti navedeni.

Pre svega treba ukazati na ograničenja koja potiču od jednostavnosti gramatike kojom se opisuju indeksni elementi, odnosno, kojom se definišu reči. Zahtev da skupovi graničnika, signalnih karaktera, samostalnih karaktera i karaktera indeksnih elemenata budu disjunktni u velikoj meri degradira pojam reči u odnosu na njegovo značenje u prirodnom jeziku. Na

primer, ne postoji način da se uloga nekog karaktera definiše u zavisnosti od konteksta. Za gramatiku iz tabele 3.3, nisku "1.5" čine dva indeksna elementa, "1" i "5" premda bi u slučaju da niska "1.5" označava realan broj identifikacija jednog indeksnog elementa bila bolje rešenje. Takođe dva potpuno različita korišćenja crtice u, recimo, "baba-devojka" i "1991-1995" korišćenjem ovako ograničene gramatike ne mogu se razdvojiti. Na ova dva primera se, međutim, ne iscrpljuje različitost grafemskog od grafičkog svojstva teksta koja je detaljno opisana u [144].

Premda se korišćenje sistema PAT gotovo uvek ilustruje na tekstovima označenim SGML oznakama ili oznakama koje na njih nalikuju, korišćenje punih svojstava SGML-a je u velikoj meri ograničeno. Činjenica da je za PAT tekst samo neprekinuta sekvencija karaktera u potpunosti onemogućava korišćenje, recimo, entiteta. Da bi se olakšalo prepoznavanje strukturnih elemenata preko odgovarajućih početnih i krajnjih etiketa, isključuje se korišćenje minimizacije etiketa (videti 2.2.3) pa čak i atributa elemenata [134]. Prepoznavanje graničnih tačaka elementa u slučaju korišćenja svojstva minimizacije etiketa zahteva od programa koji tekst obrađuje njegovo raščlanjavanje a takav postupak u slučaju PAT-a, imajući u vidu njegovu koncepciju, nije moguć. Korišćenje atributa otežava prepoznavanje početnih etiketa jer njihova dužina u tom slučaju više ne može biti unapred poznata. Izraz `docs (shift.4 "<hi")..(shift.-1 "</hi")` više neće vratiti sadržaj elementa `hi` jer definicije atributa iz teksta nisu isključene. Prepoznavanje početne etikete u slučaju korišćenja atributa u svakom slučaju zahteva predefinisanje uloge karaktera ">" za zatvaranje etikete u samostalni karakter što može imati dodatne negativne implikacije.

Samo prepoznavanje etiketa je dosta pojednostavljeno. Upit kojim bi se tražile pozicije svih početnih etiketa u tekstu vratio bi, na primer, i nisku "<123" premda 123 nije dopušteno ime elementa u SGML-u. Prepoznavanje početne etikete u slučaju konkurentne strukture, na primer, "<(alt)deo>" bilo bi, takođe veoma otežano.

Ograničenja koja nastaju usled redukovanja SGML svojstava samo na najosnovnija mogu se, delimično, prevazići korišćenjem nezavisnog raščlanjivača koji će, pre svega, proveriti sinaksičku ispravnost teksta a zatim ga prevesti u „normalizovani" oblik u kome su zamenjeni entiteti, unete izostavljene etikete, atributi zamenjeni ugnježdenim ili obuhvatnim elementima i slično. Svaka i najmanja izmena izvornog teksta u tom slučaju zahteva i njegovo ponovno raščlanjavanje i prevođenje što može biti veoma zahtevan posao u slučaju velikih tekstova kojima je PAT namenjen.

Na kraju treba istaći probleme koji nastaju usled veze PAT-a i korišćene šeme kodiranja koja je u sistemu inherentno prisutna usled korišćenja strukture kakva je Patrica. Takva direktna veza sistema za pretraživanje teksta i kodne šeme može biti uzrok degradacije samog sistema, a posebno kod njegove primene izvan anglosaksonskog govornog područja. U slučaju sistema PAT, leksičko intervalno pretraživanje gubi svaku upotrebnu vrednost kada se primeni na bilo koju kodnu šemu našeg ćiriličnog ili latiničnog alfabeta jer ni u jednoj od njih alfabetski poredak nije zadržan [18], [122].

3.3 SGML aplikacije

Najgrublje govoreći, *SGML aplikacija* je jedna definicija tipa dokumenta (DTD) pripremljena za neku konkretnu primenu. U tom smislu, odnos SGML aplikacije prema SGML-u bi se mogao uporediti sa odnosom programa napisanog na nekom programskom jeziku prema tom programskom jeziku. Tako bi se DTD iz dodatka A mogao smatrati jednom SGML

aplikacijom koja je namenjena pripremi radova za Zbornike. Ovim nazivom se, međutim, obično ne nazivaju definicije tipa dokumenta proizvedene za neku konkretnu primenu, obradu, odnosno korisnika. SGML aplikacija je namenjena širokom krugu korisnika, pokriva široko polje primene i može se koristiti u dugačkom vremenskom intervalu. Ovde će biti predstavljene dve takve aplikacije dok će u narednom odeljku biti govora o još nekima koje imaju status međunarodnog standarda.

3.3.1 TEI - preporuke za kodiranje teksta

Text Encoding Initiative (skr. TEI) je ustanovljen kao kooperativni međunarodni istraživački projekat sa ciljem da se razvije skup opštih i fleksibilnih preporuka za pripremu i razmenu elektronskih tekstova [54], [55], [130]. Rezultat projekta su konvencije kodiranja koje pokrivaju širok opseg tekstova, od pisanih do govornih, na različitim jezicima, iz različitih epoha i različitog žanra. One se mogu koristiti u različitim aplikacijama, koje uključuju prirodno-jezičke obrade, pronalaženje informacija, hipertekst, elektronsko izdavaštvo, literarne i lingvističke analize, računarsku leksikografiju i sl. i mogu zadovoljiti potrebe različitih korisnika od istraživača u oblasti računarske lingvistike i raznim domenima društvenih nauka, do izdavača i bibliotekara i svih onih koji se brinu o skladištenju i pronalaženju informacija.

S obzirom da TEI konvencija kodiranja predstavlja jednu SGML aplikaciju ona je, ustvari, jedna definicija tipa dokumenta. Da bi definicija tipa dokumenta mogla da obuhvati širok opseg tekstova i njihovih svojstava, TEI DTD je modularno koncipiran tako što je razbijen na više skupova etiketa. Osnovnu grupu čine skupovi etiketa koji su namenjeni različitim tipovima tekstova. To su sledeći *bazni skupovi etiketa* koji definišu osnovne blokove određenog tipa teksta:

- *TEI.prose* za prozne tekstove;
- *TEI.verse* za poeziju;
- *TEI.drama* za dramske tekstove;
- *TEI.spoken* za transkribovane govorne tekstove;
- *TEI.dictionaries* za rečnike;
- *TEI.terminology* za terminološke baze podataka.

Osim ovih, postoje još dva bazna skupa etiketa koja omogućavaju mešanje u jednom TEI dokumentu više tipova tekstova. To su:

- *TEI.general* koji dozvoljava da različiti odeljci TEI dokumenta budu različiti tipovi teksta i koriste odgovarajući bazni skup;
- *TEI.mixed* koji dozvoljava da se unutar TEI dokumenta slobodno mešaju elementi višeg nivoa iz različitih baznih skupova;

Drugu grupu skupova etiketa čine oni delovi TEI DTD koje koriste sva, ili gotovo sva, TEI dokumenta. To su:

- *TEI.core* je jezgro TEI DTD-a koje sadrži definicije elemenata koji se mogu koristiti u svim tipovima tekstova i ono se automatski uključuje sa svakim baznim skupom etiketa;
- *TEI.header* sadrži elemente za opis zaglavlja TEI dokumenta koje predstavlja neku vrstu naslovne stranice elektronskog teksta i sadrži takve podatke kao što su izvor dokumenta, sistem kodiranja, istoriju svih promena, itd. Korišćenje ovog skupa etiketa,

- *TEI.structure* sadrži elemente za opis strukture teksta koja je zajednička većini tipova teksta;
- *TEI.front* sadrži elemente za opis strukture prednjeg dela koja je zajednička većini tipova teksta;
- *TEI.back* sadrži elemente za opis strukture zadnjeg dela koja je takođe zajednička većini tipova teksta;

Treću grupu čine dodatni skupovi etiketa koji sadrže opcione etikete za kojima se može javiti potreba, u principu, u svim tipovima tekstova. Neki od dodatnih skupova etiketa su:

- *TEI.linking* sadrži etikete za povezivanje, segmentaciju i poravnavanje;
- *TEI.transcr* sadrži etikete za transkripciju osnovnog izvora;
- *TEI.textcrit* sadrži etikete za kritička izdanja;
- *TEI.names.dates* sadrži etikete za imena i datume;
- *TEI.corpus* sadrži etikete za jezičke korpuse.

Osim ovih, korisnik može da sačini i svoj skup etiketa koji će pridodati uz jedan bazni i, eventualno, više dodatnih skupova etiketa.

Povezivanje ovih skupova etiketa u jedan DTD, uključivanje nekih a isključivanje ostalih postiže se intenzivnim korišćenjem parametarskih entiteta i označenih odeljaka. Svakom skupu etiketa je pridruženo ime parametarskog entiteta, koje je navedeno uz opis svakog od skupova čiji je tekst zamene ključna reč `INCLUDE` ili `IGNORE` koja rukovodi uključivanjem ili ignorisanjem označenih odeljaka, odnosno, skupova etiketa. Sve ovo se ostvaruje u okviru jedne datoteke; na operativnom sistemu MS-DOS to je datoteka *tei2.dtd*, koju svaki TEI dokument mora obavezno uključiti u deklaraciji tipa dokumenta `DOCTYPE`. Na primer, korisnik bi mogao da konfiguriše svoj DTD za, recimo, prozni tekst transkribovan iz rukopisa na sledeći način:

```
<!DOCTYPE TEI.2  system 'tei2.dtd' [
<!-- TEI bazni skup za prozni tekst                    -->
    <!ENTITY % TEI.prose 'INCLUDE' >
<!-- TEI dodatni skup za transkripcije                 -->
    <!ENTITY % TEI.transcr 'INCLUDE' >
]>
```

Struktura datoteke *tei2.dtd* je sledeća:

```
<!-- Korisničke modifikacije entiteta                  -->
<!ENTITY % TEI.extensions.ent ''                        >
%TEI.extensions.ent;
<!-- Entiteti za TEI generičke identifikatore          -->
<!ENTITY % TEI.elementNames system 'teigis2.ent'        >
%TEI.elementNames;
<!-- Entiteti za TEI ključne reči                      -->
<!ENTITY % TEI.keywords.ent system 'teikey2.ent'        >
%TEI.keywords.ent;
<!-- Klase elemenata za modele sadržaja, zajednički    -->
```

```
<!-- atributi za klase elemenata i globalni atributi.                    -->
<!ENTITY % TEI.elementClasses system 'teiclas2.ent'                       >
%TEI.elementClasses;
<!-- TEI elementi najvišeg nivoa: jedan za pojedinačne                   -->
<!-- tekstove a drugi za složene sa zajedničkim zaglavljem.              -->
<!ENTITY % TEI.2 'INCLUDE' >
<![ %TEI.2; [
<!ELEMENT %n.TEI.2;      - O  (%n.teiHeader;, %n.text;)                   >
<!ATTLIST %n.TEI.2;           %a.global; ,
          TEIform             CDATA                 'TEI.2'              >
]]>
<!-- TEI corpus je serija TEI.2 dokumenata ispred kojih je               -->
<!-- zajedničko TEI zaglavlje                                            -->
<!ENTITY % teiCorpus.2 'INCLUDE' >
<![ %teiCorpus.2; [
<!ELEMENT %n.teiCorpus.2;
                         - O  (%n.teiHeader;, (%n.TEI.2)+)                >
<!ATTLIST %n.teiCorpus.2;     %a.global;
          TEIform             CDATA                 'teiCorpus.2'        >
]]>
<!-- Skupovi etiketa. Prvo se uključuju korisnički                       -->
<!ENTITY % TEI.extensions.dtd ''                                          >
%TEI.extensions.dtd;
<!-- Etikete jezgra i zaglavlja se bezuslovno uključuju                  -->
<!ENTITY % TEI.header.dtd system 'teihdr2.dtd'                            >
%TEI.header.dtd;
<!ENTITY % TEI.core.dtd system 'teicore2.dtd'                             >
%TEI.core.dtd;
<!-- Uslovno uključivanje baznih skupova etiketa                         -->
<![ %TEI.prose [
<!ENTITY % TEI.prose.dtd system 'teipros2.dtd'                            >
%TEI.prose.dtd;
]]>
<!-- Slično se uključuju i ostali bazni skupovi.                         -->
.....................................................................
<!-- Uslovno uključivanje dodatnih skupova                               -->
<![ %TEI.linking [
<!ENTITY % TEI.linking.dtd system 'teilink2.dtd'                          >
%TEI.linking.dtd;
]]>
<!-- Slično se uključuju i ostali dodatni skupovi.                       -->
.....................................................................
]]>
```

Osim što rukovodi uključivanjem obaveznih skupova etiketa, odabranih baznih i dodatnih skupova etiketa, ova datoteka omogućava fleksibilni odabir generičkih identifikatora, TEI ključnih reči i definisanje klasa elemenata. Imenima generičkih identifikatora rukovodi sistemski parametarski entitet TEI.elementNames čija je predefinisana vrednost datoteka

teigis2.ent. Za svaki predviđeni element iz bilo kog od skupova etiketa u ovu datoteku je uključena jedna deklaracija parametarskog entiteta kao što su:

```
<!ENTITY % n.TEI.2 "TEI.2" >
<!ENTITY % n.teiCorpus.2 "teiCorpus.2" >
<!ENTITY % n.name "name" >
```

Korisnik može prilagoditi imena generičkih identifikatora svojim potrebama promenom teksta zamene. Kao što se vidi iz sadržaja datoteke *tei2.dtd*, deklarisanje određenog elementa vrši se indirektno pozivanjem odgovarajućeg parametarskog entiteta, u ovom slučaju `n.TEI.2` i `n.teiCorpus.2`.

Sistemski parametarski entitet `TEI.keywords.ent` dodeljuje „početne vrednosti" parametarskim entitetima koji upravljaju uključivanjem skupova entiteta. Ako korisnik ne kaže drukčije, svi oni dobijaju vrednost `IGNORE`. Deo sadržaja datoteke *teikey2.ent* koja je predefinisana vrednost entiteta `TEI.keywords.ent` je:

```
<!ENTITY % TEI.prose 'IGNORE'                    >
<!ENTITY % TEI.verse 'IGNORE'                    >
<!ENTITY % TEI.drama 'IGNORE'                    >
```

TEI DTD sadrži preko 400 različitih elemenata. Radi lakšeg razumevanja i eventualnog menjanja većina ovih elemenata je formalno klasifikovana uvođenjem dve vrsta klasa. Prvu, koja se naziva *a-klasa*, čine elementi koji dele neki skup SGML atributa a drugu, koja se naziva *m-klasa* čine elementi koji se mogu pojaviti na istom mestu u modelu sadržaja drugih TEI elemenata. Klase mogu imati podklase i nadklase i karakteristike nadklasa nasleđuju svi članovi podklase.

Oba tipa klasa su u TEI DTD-u predstavljena parametarskim entitetima. Za definisanje klase elemenata koji dele zajedničke SGML atribute koriste se takozvani *a-tačka entiteti* kojih u TEI DTD-u ima 14. Jedna od njih je klasa *names* koju čine elementi za označavanje vlastitih imena a to su: `name`. `placeName` i `persName`. Svi oni dele zajedničke atribute, `key` i `reg` za beleženje informacija o alternativnom identifikatoru i o normalizovanom obliku vlastitog imena. Ovakvo definisanje klase obezbeđuje da svaku eventualnu promenu u definiciji atributa automatski naslede svi članovi klase. Definicija klase *names* je sadržana u datoteci *teiclas2.ent* koja je predefinisana vrednost entiteta *TEI.elementClasses*:

```
<!ENTITY % a.names '
          key          CDATA              #IMPLIED
          reg          CDATA              #IMPLIED '     >
```

M-klasa se implementira definisanjem parametarskog entiteta, *m-tačka entiteta*, koji se koristi u deklaraciji modela sadržaja elementa. Tekst zamene entiteta je lista članova klase razdvojenih simbolom | (SGML **or** konektor). Za svaku klasu se uz m-tačka entitet deklariše i pomoćni *x-tačka entitet* čija je predefinisana vrednost prazna niska. Referenca tog entiteta se uvek uključuje u tekst zamene odgovarajućeg m-tačka entiteta, što korisniku omogućava dodavanje novih elemenata u klasu definisanjem nove vrednosti x-tačka entiteta. Na primer, klasa *bibl* koja ima tri člana: `bibl`, `bibl.full` i `bibl.struct` definiše se na sledeći način:

```
<!ENTITY % x.bibl '' >
<!ENTITY % m.bibl ' %x.bibl bibl | bibl.full | bibl.struct ' >
```

Korisnik koji želi da doda klasi novi član `moja.bib` treba da promeni tekst zamene x-tačka entiteta:

```
<!ENTITY % x.bibl 'moja.bib | ' >
```

Sistem m-klasa je strukturiran oko sledećih grupa elemenata:

- *komadi* su takvi elementi koji se mogu direktno pojaviti u tekstu ili njegovim poddelovima (elementima tipa div) ali se ne mogu pojaviti unutar drugih komada. Tipičan primer je pasus.
- *elementi na nivou fraze* mogu se pojaviti samo unutar komada ali ne i između njih (ne mogu biti dakle direktno sadržani u div). Tipični elementi su istaknute fraze, naslovi i slično.
- *međunivojski elementi* se mogu pojaviti između komada, kao direktna deca od div, ili unutar njih. Tipični primeri su liste, citati i slično.

Sve ove klase su definisane unutar datoteke *teiclas2.ent* koja je predefinisana vrednost entiteta *TEI.elementClasses*. Sadržaj te datoteke je:

```
<!-- M-klase: Klase niskog nivoa
<!-- Nivo fraze                                                    -->
<!ENTITY % x.hqphrase ''                                           >
<!ENTITY % m.hqphrase '%x.hqphrase distinct | emph | foreign |
          gloss | hi | mentioned | soCalled | term | title'       >
.....................................................................
<!-- Među nivo                                                     -->
<!ENTITY % x.bibl ''                                               >
<!ENTITY % m.bibl '%x.bibl bibl | biblFull | biblStruct'           >
.....................................................................
<!-- Razno: koristi se samo za neke skupove etiketa               -->
<!ENTITY % x.addrPart ''                                           >
<!ENTITY % m.addrPart '%x.addrPart postBox | postCode | street'    >
<!-- Klase višeg nivoa                                            -->
<!ENTITY % x.phrase ''                                             >
<!ENTITY % m.phrase '%x.phrase %m.data; | %m.edit; |
          %m.formPointers; | %m.hqphrase; | %m.loc; |
          %m.phrase.verse; | %m.seg; | %m.sgmlKeywords; |
          formula | handShift'                                     >
<!ENTITY % x.inter ''                                              >
<!ENTITY % m.inter '%x.inter %m.bibl; | %m.hqinter; | %m.lists;
          | %m.notes; | %m.stageDirection; | castList | figure
          | stage | table | text'                                  >
<!ENTITY % x.chunk ''                                              >
<!ENTITY % m.chunk '%x.chunk eTree | graph | l | lg | p | sp |
          tree'                                                    >
.....................................................................
<!-- Definicije entiteta specifične za pojedine skupove etiketa-->
<![ %TEI.verse [
<!ENTITY % TEI.verse.ent system 'teivers2.ent'                     >
%TEI.verse.ent;
]]>
<!-- Slično je i za ostale skupove etiketa                        -->
```

```
.....................................................................
<!-- Definicija standardnih modela sadržaja. To su: phrase,     -->
<!-- phrase.seq, component, component.seq, paraContent          -->
<!-- i specialPara,                                             -->
<!ENTITY % phrase '(#PCDATA | %m.phrase)'                        >
<!ENTITY % phrase.seq '(%phrase;)*'                              >
<!-- Ostale komponente zavise od baznog skupa.                  -->
<!-- Ako se bazni skupovi kombinuju, specijalan postupak        -->
<![ %TEI.mixed [
.....................................................................
]]>
<![ %TEI.general [
.....................................................................
]]>
<![ %TEI.prose [
<!ENTITY % component '(%m.common)'                               >
<!ENTITY % TEI.singleBase 'INCLUDE'                              >
]]>
<![ %TEI.verse [
<!ENTITY % component '(%m.common | %m.comp.verse)'               >
<!ENTITY % TEI.singleBase 'INCLUDE'                              >
]]>
<!-- Slično je i za ostale skupove etiketa                       -->
.....................................................................
<!ENTITY % component.seq '(%component;)*'                        >
<!ENTITY % paraContent '(#PCDATA | %m.phrase | %m.inter)*'        >
<!ENTITY % specialPara '(((%m.chunk), (%component.seq)) |
                        (%paraContent))'                         >
                                                                 -->
<!-- A-klase
<!ENTITY % a.names '
          key                CDATA              #IMPLIED
          reg                CDATA              #IMPLIED'       >
.....................................................................
<!-- Globalni atributi                                           -->
<!ENTITY % a.global '        %a.fs;
                             %a.analysis;
                             %a.linking;
                             %a.terminology;
          id                 ID                 #IMPLIED
          n                  CDATA              #IMPLIED
          lang               IDREF              %INHERITED
          rend               CDATA              #IMPLIED'       >
```

Za razumevanje načina konfigurisanja konkretnog DTD-a nekog TEI dokumenta potrebno je razumeti mehanizme funkcionisanja parametarskih entiteta i označenih odeljaka kao i značaj redosleda uvođenja deklaracija entiteta i elemenata. Premda se ova SGML aplikacija sastoji od niza običnih tekstualnih datoteka, od kojih su neke ovde detaljnije opisane, koje

korisnik može lako da menja, konstruisanje TEI aplikacije ne podrazumeva menjanje ni jedne od ovih osnovnih datoteka. Sve izmene moraju biti lokalizovane u osnovi na dve datoteke od kojih jedna sadrži korisnikove entitete a druga korisnikove etikete, odnosno, deklaracije elemenata. U dodatku B dat je sadržaj ove dve datoteke kao i definicija tipa dokumenta za elektronsko izdanje Vukovih narodnih poslovica [72] koje predstavlja jednu TEI aplikaciju.

Premda je TEI jedna SGML aplikacija, upotreba nekih svojstava SGML-a je ograničena, i to:

- kratke reference nisu dozvoljene;
- RANK svojstvo nije dozvoljeno;
- izostavljanje generičkih identifikatora u etiketama nije dozvoljeno;
- označeni odeljci koriste samo ključne reči `INCLUDE`, `IGNORE` i `CDATA`;
- poddokumenta se mogu uključiti samo zadavanjem reference odgovarajućeg entiteta kao vrednosti atributa.

Osim što ilustruje primenu koncepta modularnosti na koncipiranje jedne SGML aplikacije, TEI aplikacija ilustruje mogućnost korišćenja parametarskih entiteta kao programskih promenljivih i označenih odeljaka kao `if...then` programskih konstrukata. Uvodi se osim toga i nov koncept konkurentnog sadržaja preko baznih skupova za mešovite sadržaje koji se u znatnoj meri razlikuje od konkurentnih struktura koje predviđa SGML. Definisane su i tri nove vrste vrednosti atributa:

- `INHERITED` koji ukazuje da se vrednost atributa nasleđuje od obuhvatnog elementa;
- `ISO-date` koji ukazuje da je atribut valjani ISO datum;
- `extrptr` koji ukazuje da je atribut valjani izraz u TEI pokazivačkoj notaciji;

3.3.2 HTML — jezik za sporazumevanje globalnih računarskih mreža

Jezik za označavanje hiperteksta — HTML (skraćenica od *Hypertext Markup language*) [112] je jezik koji se koristi za kreiranje hipertekstualnih dokumenata koji su prenosivi sa jedne na drugu računarsku platformu. To je jezik koji razumeju gledači (engl. *viewer*) i prelistači (engl. *browser*) koji se koriste na mreži Internet koja pokriva ceo svet, WWW (skraćenica od engl. *World Wide Web*).

HTML predstavlja jednu SGML aplikaciju, odnosno SGML DTD, kome je pridružena generička semantika pogodna za predstavljanje informacija u najrazličitijim primenama, kao što su hipertekstualne vesti, pošta, dokumentacija, hipermedijalna dokumenta, meniji opcija, rezultati upita baza podataka i mnoge druge.

Globalna računarska mreža Internet koristi ovaj jezik počev od 1990. godine. O razvoju i standardizaciji jezika brine se organizacija Internet Engineering Task Force (IETF). Trenutno je u upotrebi verzija 3.2 ovog jezika. Verzije jezika se razvijaju tako da svaka naredna podržava sve prethodne, iako se sve mogućnosti prethodnih verzija ne moraju preporučivati. Sam jezik je organizovan u četiri nivoa: Nivo 0 podržava samo osnovnu HTML strukturu, nivo 1 omogućava najjednostavniju podršku slikama, nivo 2 omogućava korišćenje formulara dok nivo 3 obezbeđuje korišćenje tabela, matematičkih formula u notaciji koja nalikuje LaTeX-ovoj i pruža dodatnu podršku multimedijalnim dokumentima. Verzija 1 HTML podržava nivoe 0 i 1, verzija 2 nivoe 0, 1 i 2 a verzija 3 sve do sada predviđene nivoe [16].

HTML je koncipiran u skladu sa sledećim smernicama:

- HTML treba da bude jezik za sporazumevanje globalne svetske mreže s ciljem da bude zajedničko sredstvo za povezivanje informacija iz potpuno različitih izvora.
- Jezik treba da bude podjednako jednostavan kako za autore dokumenata tako i za konstruktore prelistača i gledača. Ostvarenost ovog zahteva u HTML-u smatra se jednim od najznačajnijih faktora u neverovatnom brzom širenju WWW.
- Mogućnost podešavanja treba da bude prisutna. U HTML-u se ono ostvaruje, pre svega kroz subkategorizaciju elemenata. Tako, na primer, globalni element P kojim se opisuju pasusi može se kategorisati korišćenjem atributa CLASS kao apstrakt, stih u strofi i slično. Podešavanje se postiže i pomoću URI (skraćenica od engl. *Uniform Resource Idenitifier*) zasnovanih spona za uključivanje informacija u drugim formatima.
- Jezik treba da bude potpuno nezavisan od platforme. HTML omogućava predstavlajnje dokumenata na širokom spektru uređaja od klasičnih alfanumeričkih terminala preko DOS, Windows i Machintosh radnih stanica pa do nepisanih madijuma koji koriste, na primer, govor ili Brajevo pismo. Prezentacije se podešavaju prema sredstvima prisutnim na klijentovoj mašini i prema njegovim željama.
- Jezik treba da opisuje sadržaj dokumenta a ne njegov izgled. Kreator dokumenta često ipak želi da u izvesnoj meri zadrži kontrolu nad konačnim izgledom dokumenta. Zadovoljenje ovog zahteva, koji je donekle suprotstavljen prethodnom, je u HTML-u postignuto vezivanjem uz svaki element informacija o stilu koje opisuju njegove prezentacione karakteristike. Stilski list (engl. *style sheet*) se može izraziti na način nezavisan od platforme ili se može koristiti za detaljniju kontrolu za određenu klasu klijenata ili izlaznih uređaja.
- Potrebno je obezbediti podršku kaskadnim stilskim listovima što je od posebnog značaja za rad na mreži gde autor, izdavač, klijent i krajnji korisnik mogu imati različite želje u vezi sa izgledom dokumenta. HTML podržava stilske listove preko elementa LINK pomoću koga se stilski list referiše preko URI-ja. Autor može da prevaziđe određeni stil navođenjem elementa STYLE u zaglavlju dokumenta.
- Nevizualni medijumi treba da budu podržani tako da sam autor dokumenta ne treba da piše posebne specifikacije za različite grupe klijenata.
- Jezik treba da podržava različite načine za kreiranje dokumenata. HTML dopušta i „ručno" kucanje teksta i oznaka s obzirom da je HTML dokument obična ASCII datoteka. Moguće je osim toga i korišćenje specijalnih WYSIWYG HTML uređivača kao i izvoznih filtera iz uobičajenih formata procesora reči.

Korišćeni HTML se može parametrizovati na više načina. Jedna mogućnost je navođenje parametra karakterskog skupa da bi se specifikovalo kodiranje HTML dokumenta kao sekvencije bajtova. Ukoliko ovaj parametar nije naveden pretpostavlja se korišćenje US-ASCII koda a ako se u dokumentu pojavi nešto izvan 7-bitnog koda, onda se pretpostavlja korišćenje ISO 8859-1 koda.

Osim toga, mogu se definisati dve formalne varijante jezika: restriktivna, koja dozvoljava korišćenje samo novih i nezastarelih mogućnosti jezika i opuštena, koja obezbeđuje kompatibilnost sa prethodnim verzijama jezika. Korišćenje ovih varijanti jezika obezbeđuje se postavljanjem vrednosti dve zastavice HTML.Recommended i HTML.Depricated. Ove zastavice se deklarišu kao entiteti: zastavicu HTML.Recommended treba postaviti na vrednost INCLUDE da bi se omogućilo restriktivno korišćenje jezika dok zastavicu HTML.Depricated

treba postaviti na vrednost INCLUDE ako se dozvoljava korišćenje mogućnosti iz ranijih verzija koje se više ne preporučuju. Početak dokumenta koji koristi restriktivnu varijantu jezika izgledao bi recimo ovako:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN"
  [ <!ENTITY % HTML.Recommended "INCLUDE" >
]>
```

Uključivanje opuštene varijante jezika automatski postavlja zastavicu HTML.Depricated na vrednost INCLUDE. To se postiže sledećim deklaracijama u okviru HTML DTD-a:

```
<![%HTML.Recommended [
   <!ENTITY % HTML.Deprecated "IGNORE" >
]]>
<!ENTITY % HTML.Deprecated "INCLUDE" >
```

Opšta je preporuka da se prilikom kreiranja dokumenta koristi restriktivna varijanta jezika dok dokument treba primati korišćenjem opuštene varijante jezika.

Varijantna konkretna sintaksa koju HTML koristi precizirana je u odgovarajućoj SGML deklaraciji. U njoj je specifikovano da se dokumenti kodiraju korišćenjem dva skupa: ISO 646 (neformalno, US–ASCII) i gornje stranice ISO 8859-1 (poznate kao Latin 1 skup). Za kodiranje oznaka koristi se samo skup ISO 646 kako je predviđeno i referentnom konkretnom sintaksom. U odnosu na referentnu konkretnu sintaksu povećani su neki kapaciteti (TOTALCAP i GRPCAP) i neke veličine (ATTSPLEN, LITLEN, NAMELEN, PILEN, TAGLEN, GRPGTCNT i GRPCNT) i uvedena je jedna nova funkcija karaktera TAB. Od opcionih svojstava jezika dozvoljena je samo minimizacija etiketa, i to izostavljanje i skraćivanje etiketa. Sva HTML dokumenta moraju da koriste ovu varijantnu konkretnu sntaksu.

U okviru definicije tipa dokumenta HTML-a deklarisano je više entiteta koji se koriste za definisanje različitih simbola, zajedničkih atributa i tokena sadržaja. Simboli se grupišu u tri grupe entiteta:

- Entiteti za simbole iz gornje stranice ISO 8859-1. Ova grupa se deklariše pomoću entiteta HTMLlat1 čiji je tekst zamene formalni javni identifikator "-//IETF//ENTITIES Added Latin 1 for HTML//EN". Ovu grupu čine svi entiteti iz grupe "ISO 8879:1986// ENTITIES Added Latin 1//EN" i neki entiteti iz grupe "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN". Primeri entiteta iz ove grupe su:

ü	ü	&#252	malo u, dijereza ili umlaut
ç	ç	&#231	malo c, cedilja
<	<	&#60	manje od

- Entiteti za matematičke simbole i slova grčkog alfabeta. Ova grupa se deklariše pomoću entiteta HTMLmath čiji je tekst zamene formalni javni identifikator "-//IETF//ENTITIES Math and Greek for HTML//EN". Imena entiteta iz ove grupe su, gde god je to moguće, usklađena sa imenima odgovarajućih komandi TEX-a [77]. Primeri entiteta iz ove grupe su:

&cdots;	\cdots	$\cdots$	tri tačke na istom nivou kao znak minus
	\;	$\;$	debeli razmak
ϵ	\varepsilon	ε	epsilon

- Entiteti za standardne ISO/WWW ikone. Oni se mogu koristiti umesto pretpostavljenih simbola za označavanje članova listi (preko atributa DINGBAT) ili kao deo hipertekstualnih spona. Njihovim korišćenjem štedi se vreme potrebno za spuštanje slika.

Prelistači ih mogu koristiti preko biblioteke slika ili preko URL/URN. Ova grupa se deklariše pomoću entiteta HTMLicons čiji je tekst zamene formalni javni identifikator "-//IETF//ENTITIES Icons for HTML//EN". Primeri deklaracija entiteta iz ove grupe su:

```
<!ENTITY ftp SDATA "ftp" -- ftp server -- >
<!ENTITY fixed.dsk SDATA "fixed.disk" -- pogon fiksnog medijuma -- >
<!ENTITY mail SDATA "mail" -- poruke e-pošte -- >
```

U HTML definiciji tipa dokumenta entiteti se koriste za definisanje atributa i to na dva načina: za deklarisanje mogućih vrednosti atributa i za deklarisanje liste atributa. Primeri prvih deklaracija su:

```
<!ENTITY % HTTP-Method "GET | POST" -- HTTP specifikacija -->
<!ENTITY % URI "CDATA"
   -- CDATA atribut čija je vrednost identifikator uniformnog
      resursa URI. Ova vrednost je ograničena veličinom
      LITLEN = 1024 --
>
```

Prvi primer specifikuje moguće varijacije HTTP protokola (skraćenica od engl. *Hypertext Transfer Protocol*). To su moguće vrednosti atributa METHOD elementa FORM koji se koristi za ulazne formulare. Vrednost tog atributa određuje koji metod server podržava. Drugi primer specifikuje vrednosti koje mogu imati sponski atributi elemenata. Koriste ga, recimo, atributi SRC elementa IMG za opis slika, elementa BANNER za opis transparenata i mnogi drugi koji pomoću URI-ja identifikuju odgovarajuće objekte.

U definiciji tipa dokumenta HTML-a se pomoću entiteta deklarišu i cele liste atributa ali pre svega u cilju skraćivanja zapisa a ne radi klasifikovanja elemenata u a-klase, kako je to učinjeno u TEI DTD-u. Deklarisano je nekoliko listi atributa od kojih je najčešće korišćena lista deklarisana entitetom attrs:

```
<!ENTITY % attrs  -- zajednički atributi elemenata --
   'id       ID       #IMPLIED -- kao cilj krajeva spona (href atr.) --
    lang     CDATA    "en.us"  -- ISO jezik, zemlja kod --
    class    NAMES    #IMPLIED -- za subkategorisanje elemenata.
                                  klase se interpretiraju hijerarhijski,
                                  s desna na levo, i razdvajaju tačkom --
>
```

Sa stanovišta HTML-a najzanimljiviji su atributi za opis spona. Tako se, na primer, atribut MD (skraćenica od engl. *message digest*) može koristiti uz sve one elemente uz koje se koristi atribut čija je vrednost URL (skraćenica od engl. *uniform resource locator*) zadata najčešće kao URI. Vrednost ovog atributa je sažeta poruka ili kriptografski proverni zbir koja se koristi da bi se proverilo da li je povezani objekat onaj koji se želi, odnosno da on nije promenjen ni na koji način. Može se koristiti, recimo, uz atribut SRC elementa IMG za opis slika, ili uz isti atribut elementa BANNER za opis transparenata kao i uz mnoge druge.

Za opis spona može se koristiti i grupa atributa opisana entitetom linkExtraAttributes. Oni omogućavaju da se imenuje vrsta spone koju objekat ostvaruje sa povezanim objektom (najčešće preko URI-ja) kao i da se imenuje vrsta obrnute spone (od povezanog objekta ka sidru, odnosno izvorištu spone). Koriste ih element LINK, koji ukazuje na vezu celog dokumenta sa drugim objektima, i element A koji se koristi za opisivanje hipertekstualnih

veza. Entiteti `url.link` i `linkExtraAttributes` deklarišu se na sledeći način:

```
<!ENTITY % url.link -- Atributi pridruženi URL sponama --
   "md CDATA #IMPLIED  -- sažeta poruka za sponski objekat --">
<!ENTITY % linkExtraAttributes
      "rel %linkType #IMPLIED -- tip unaprednog odnosa  --
       rev %linkType #IMPLIED -- tip obrnutog odnosa --
       title   CDATA #IMPLIED -- naslov sponskog objekta --
       methods NAMES #IMPLIED -- podržani javni metodi objekta:
                                 TEXTSEARCH, GET, HEAD, ... --
       ">
```

U HTML DTD-u entiteti se konačno koriste i za deklarisanje tokena sadržaja. Kao što će biti pokazano, ove deklaracije su znatno jednostavnije od onih iz TEI DTD-a pre svega zato što bilo kakva promena HTML DTD-a od strane korisnika, za razliku od TEI DTD-a, nije uopšte predviđena. Drugi razlog potiče od potpune definisanosti uloge HTML-a, a to je kreiranje hipertekstualnih dokumenata i njihovo predstavljanje na mreži, što je dovelo do pomeranja težišta u opisivanju dokumenta sa elemenata na njihove atribute.

Sadržaj najvećeg broja nepraznih elemenata opisuje se jednim od sledeća tri entiteta, ili nekom njihovom kombinacijom:

1. Oznake na nivou teksta deklarišu se entitetom `text`. Karakteristika ovih elemenata je da su za njih obavezne obe etikete. Osim elemenata sadržanih u deklaraciji ovog entiteta, ovim sadržajem se opisuju i element `P` za opis pasusa, elementi `H1`–`H6` za opis naslova različitog nivoa, `LI` za opis člana liste i mnogi drugi.

```
<!-- Elementi za opis stila fonta  -->
<!ENTITY % font " U | S | TT | I | BIG | SMALL">
<!-- Elementi za opis fraza -->
<!ENTITY % phrase "EM | STRONG | CODE | SAMP | KBD | VAR | CITE">
<!-- Elementi koji nose razne informacije -->
<!ENTITY % misc "Q | LANG | AU | DFN | PERSON | ACRONYM |
                 ABBREV | INS | DEL">
<!-- Specijalni elementi: tabele, mat. formule, spone, slike,
     tvrdi kraj reda -->
<!ENTITY % special "TAB | MATH | A | IMG | BR">
<!ENTITY % notmath "%font | %phrase | %special | %misc">
<!-- Konačno TEXT je sve prethodno navedeno, plus supskrpti,
     superskripti, polumasna slova i raščlanjivi karakterski
     podaci -->
<!ENTITY % text "#PCDATA | SUB | SUP | B | %notmath">
<!-- Rekurzivna definicija svih elemenata za opis stila, fraza i
     informacija. Na osnovu toga sledi da svi ovi elementi mogu
     biti ugnježdeni -->
<!ELEMENT (%font|B|%phrase|%misc) - - (%text)+>
```

2. Oznake plutajućih elemenata opisuje entitet `block`. To su blokovi teksta koji mogu biti direktno sadržani u strukturnim celinama dokumenta koje opisuje element `DIV`. Ovim sadržajem se direktno opisuju element `DD` za opis termina i element `LI` za opis člana liste a indirektno, preko tokena sadržaja `body.content` i mnogi drugi.

```
<!-- Neuredjene i uredjene liste -->
<!ENTITY % list "UL | OL">
<!-- Blokovski citati (za razliku od CITE) -->
<!ENTITY % blockquote "BQ">
<!-- Unapred formatiran tekst -->
<!ENTITY % preformatted "PRE">
<!-- Osim toga, blok sadrži pasuse, liste definicija, forme,
     fusnote, tabele, slike, napomene, indekse -->
<!ENTITY % block
     "P | %list | DL | %preformatted | %blockquote
        | FORM | ISINDEX | FN | TABLE | FIG | NOTE">
<!-- Plutajući elementi su sekvencija blokova -->
<!ENTITY % flow "(%block)*">
```

3. Oznake tela dokumenta opisuje entitet `body.content`. Njime se opisuje strukturiranost tela dokumenta u odeljke, liste, pasuse i slično. Ovim sadržajem se opisuju osim elementa `BODY` koji predstavlja telo dokumenta, i elementi `DIV` za opis odeljaka, `TH` i `TD` za opis ćelija tabele, `NOTE` za opis napomena i mnogi drugi.

```
<!-- Šest nivoa naslova. Oni se sastoje od tekstualnih
     elemenata -->
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!-- Elemente tela dokumenta č ine odeljci, naslovi,
     blokovski elementi, horizontalne linije i adrese -->
<!ENTITY % body.content "(DIV|%heading|%block|HR|ADDRESS)*"
```

Strukturiranost HTML dokumenata je veoma jednostavna. Svaki HTML dokument mora imati sledeću strukturu:

```
<HTML>
<HEAD> -- elementi glave --
<BODY> -- elementi tela --
</HTML>
```

Kao što je već rečeno, telo ima sadržaj opisan entitetom `body.content`. U njemu je predviđen samo jedan element za opis strukturnog dela dokumenta `DIV`. Njegov atribut `CLASS` opisuje o kakvom se strukturnom delu radi: deo, glava, odeljak, apstrakt i slično. U sprezi s njim koriste se i elementi `H1`–`H6` koji određuju nivoe naslova. Prezentacija naslova i njihovo numerisanje određeni su stilskim listom koji je na snazi.

Glava HTML dokumenta opisuje osobine dokumenta kao što su njegov naziv, linija alatki (engl. *toolbar*), stilski list i slično. Deklariše se na sledeći način:

```
<!ELEMENT HEAD O O  "TITLE & ISINDEX? & BASE? & STYLE?
                    & META* & LINK* & RANGE*">
```

Većina elemenata koji su u sastavu glave ima prazan sadržaj osim elemenata `TITLE` i `STYLE` čiji sadržaj su raščlanjivi karakterski podaci. Uloga ovih elemenata je ukratko sledeća:

- `TITLE` daje naziv dokumenta koji se ne prikazuje u tekstu dokumenta već se koristi za obeležavanje prozora u kome se dokument nalazi.

- BASE daje URL samog dokumenta u okviru HREF atributa, što može biti od značaja ako se dokument koristi izvan konteksta. Osim toga svi URL u dokumentu mogu se parcijalno zadavati, relativno u odnosu na bazni URL.

- ISINDEX informiše korisnika da se dokument može pretraživati po ključnim rečima.

- Pomoću elementa STYLE uključuju se informacije o prezentaciji i služi za prevazilaženje klijentovog pretpostavljenog ili povezanog stilskog lista. Atribut NOTATION specifikuje koja se od mogućih notacija koristi za zadavanje stilskog lista u okviru ovog elementa. Jedna od mogućih vrednosti ovog atributa je `dsssl-lite` koja specifikuje korišćenje pojednostavljenog podskupa DSSSL-a [67] (videti i 3.4.1).

- Element LINK ukazuje na vezu između HTML dokumenta i nekog drugog objekta koji se korišćenjem URI notacije specifikuje kao vrednost HREF atributa. Atributi REL i REV određuju odnos između dokumenta i objekta. Na primer, ako se definišu linija alatki ili navigaciona dugmeta menija, vrednosti atributa REL mogu biti `home`, `toc`, `index`, `glossary`, `copyright`, `up`, `previous`, `next`, `help`, `bookmark`, sa očiglednim značenjem.

- Elementom RANGE označava se opseg u dokumentu, recimo radi osvetljavanja delova koji zadovoljavaju neki kriterijum pretraživanja. Za definisanje opsega koriste se atributi FROM i UNTIL čije vrednosti su tipa ID.

- Element META se koristi za uključivanje meta informacija o dokumentu koje se ne mogu definisati pomoću drugih HTML elemenata.

Predstavljanje matematičkih formula uvedeno je sa nivoom 3 HTML-a kroz element MATH i još dvadeset elemenata koji se koriste u njegovom sadržaju. Način kodiranja se u velikoj meri oslanja na kodiranje u TEX-u. Osim za imena simbola i grčkih slova, o čemu je već bilo reči, koriste se ista imena i za druge komande i funkcije iz TEX-ovog matematičkog režima. Kako su sintakse TEX-a i SGML-a potpuno različite a da bi se skratio zapis matematičkog izraza i približio njegovoj dužini u TEX-u, koriste se za označavanje blokova i supskripta i superskripta kratke reference. Na primer:

x^2^	preslikava se u	`x<sup>2</sup>`
y_z_	preslikava se u	`y<sub>z</sub>`
{a+b}	preslikava se u	`<box>a + b</box>`

Poređenja radi, matematička formula:

$$\int_a^b \frac{f(x)}{1+x} dx$$

se u TEX-u kodira na sledeći način:

```
\int_a^b {f(x)\over{1+x}} dx
```

Ista formula kodirana u HTML-u izgleda ovako:

```
<math>&int;_a_^b^{f(x)<over>1+x}dx</math>
```

Kratke reference se u HTML DTD-u definišu na sledeći način:

```
<!ENTITY REF1   STARTTAG   "SUP">
<!ENTITY REF2   ENDTAG     "SUP">
<!ENTITY REF3   STARTTAG   "SUB">
<!ENTITY REF4   ENDTAG     "SUB">
<!ENTITY REF5   STARTTAG   "BOX">
<!ENTITY REF6   ENDTAG     "BOX">

<!USEMAP MAP1   MATH>
<!USEMAP MAP2   SUP>
<!USEMAP MAP3   SUB>
<!USEMAP MAP4   BOX>

<!SHORTREF MAP1 "^" REF1   "_" REF3   "{" REF5 >
<!SHORTREF MAP2 "^" REF2   "_" REF3   "{" REF5 >
<!SHORTREF MAP3 "_" REF4   "^" REF1   "{" REF5 >
<!SHORTREF MAP4 "}" REF6   "^" REF1   "_" REF3   "{" REF5 >
```

Ulazne forme su uvedene sa nivoom 2 HTML-a i proširene novim mogućnostima u nivou 3. One se mogu koristiti u različite svrhe, za upitnike, hotelske rezervacije, narudžbenice i slično. Forme se opisuju elementom FORM koji se u HTML DTD-u deklariše na sledeći način:

```
<!ELEMENT FORM - - %body.content -(FORM) +(INPUT|SELECT|TEXTAREA)>
<!ATTLIST FORM
   action %URI #REQUIRED -- Lokacija na kojoj se obrađuje forma --
   method (%HTTP-Method) GET -- HTTP protokol koji se koristi --
   enctype %Content-Type; "application/x-www-form-urlencoded"
                          -- Tip enkodiranja sadržaja forme --
   script %URI #IMPLIED -- veza sa klijentovim skriptom --
        >
```

Iz ove deklaracije sledi da forme ne mogu biti ugnježdene. Za opis sadržaja forme koristi se mali broj elemenata, osim onih koji su i inače dozvoljeni u okviru HTML dokumenta, i oni se zadaju u obliku uključenja. Za realizaciju forme od najvećeg značaju su atributi kojih, recimo, samo element INPUT ima sedamnaest. Pomenimo samo atribute TYPE kojim se određuje vrsta ulaza (tekst, lozinka, datoteke, i slično), NAME kojim se imenuje ulazno polje i VALUE kojim se precizira koje vrednosti ulazno polje može da sadrži.

HTML ne pruža direktnu podršku ograničavanju vrednosti koja se unose u tekstualna polja ili, za izvedena polja, vrednosti koje se izračunavaju na osnovu vrednosti drugih polja. Umesto da se proširi skup oznaka, forma može da se poveže sa skriptom putem URI-ja. Skript rukovodi izdavanjem poruka za različita ulazna polja i za formu u celini.

Treba napomenuti da HTMP nije jedini jezik namenjen razmenjivanju dokumenata na mreži. Jezik XML (skraćenica od *Extensible Markup Language* i se takođe zasniva na SGML-u sagiasan je sa vezijom 3.2 HTML-a. Osnovne razlike u odnosu na HTML su u mogućnosti korisnika da definiše svoje elemente i njihove atribute. Takođe, strukture dokumenta mogu biti ugnježdene do prizvoljnog nivoa.

3.4 Standardi zasnovani na SGML-u

U periodu od nastanka SGML-a u međunarodnoj organizaciji za standardizaciju ISO razvijen je veći broj standarda koji se u manjoj ili većoj meri na njemu zasnivaju. Tako neki od njih koriste prvenstveno njegov formalni alat — takav je, na primer, standardni jezik za opisivanje muzike SMDL [65] — dok drugi standardizuju alate koji podržavaju njegovo korišćenje, na primer sintaksno vođene uređivače [60]. Ovde će biti reči samo o nekim od ovih standarda koji slede jednu od osnovnih ideja koje su podstakle izradu samog SGML standarda, a to je proizvodnja dokumenata raznovrsnog sadržaja na papirnom ili elektronskom medijumu.

3.4.1 DSSSL — Jezik za semantiku stila dokumenta i specifikaciju

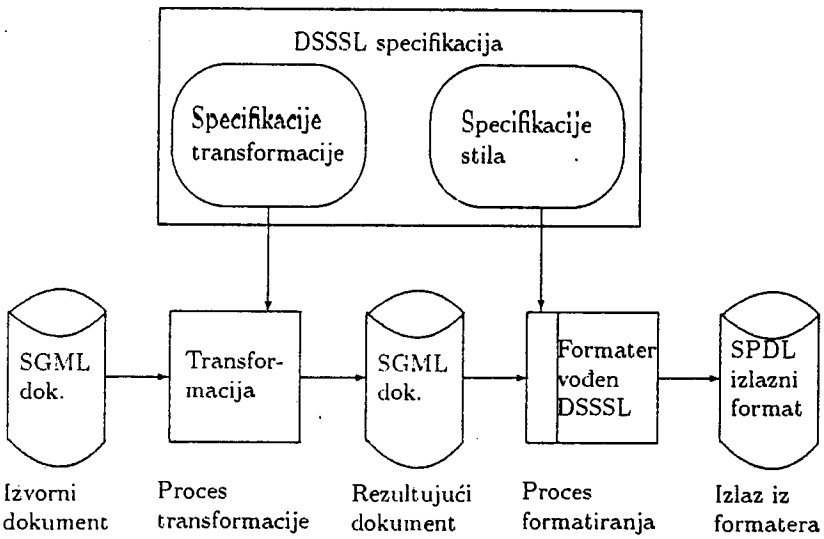
DSSSL standard (skraćenica od engl. *Document Style Semantics and Specification Language*) [67] fokusira se na dve osnovne namene, a to su formatiranje za papirni ili elektronski medijum i transformacija SGML dokumenata označenih prema proizvoljnom DTD-u. On to postiže definisanjem jezika pomoću kojih se na formalan i precizan način izražavaju formatiranje i druge specifikacije obrade dokumenta tako da ih formateri i drugi procesori mogu koristiti bilo direktno bilo posredstvom odgovarajućeg prevodioca. Ovi jezici su po prirodi deklarativni i, premda nisu potpuni programski jezici, sadrže mnoge konstrukte poznate iz postojećih programskih jezika. To čini da se DSSSL specifikacije mogu nedvosmisleno raščlanjavati i interpretirati pomoću heterogenih sistema za obradu.

Preciznije, DSSSL definiše semantiku, sintaksu i model obrade dva nezavisna jezika za specifikaciju obrade dokumenta. To su:

- *transformacioni jezik* kojim se transformišu SGML dokumenta označena u skladu sa jednom ili više definicija tipa dokumenta u SGML dokumenta označena u skladu sa nekim drugim definicijama tipa dokumenta. Transformisanje se zadaje u okviru specifikacije transformacije koja korisniku omogućava da kreira nove strukture, da postojeće strukture prepisuje, preuređuje i grupiše na nov način.
- *stilski jezik* pomoću koga se skup karakteristika formatiranja pridružuje delovima podataka onoliko precizno koliko to aplikacija zahteva, ostavljajući neke odluke, kao što su podela redaka ili kolona završnom procesu slaganja i preloma. Pod formatiranjem se ovde podrazumevaju sledeći postupci:
  - proces koji primenjuje prezentacioni stil na sadržaj izvornog dokumenta i određuje njegov položaj na prezentacionom medijumu;
  - selekcija i preuređivanje sadržaja rezultujućeg dokumenta u odnosu na njegovu poziciju u ulaznom dokumentu;
  - uključivanje materijala koji nije eksplicitno prisutan u izvornom dokumentu, kao što je generisanje novog materijala;
  - isključivanje materijala iz ulaznog dokumenta u rezultujućem dokumentu.

  DSSSL definiše vizuelni izgled formatiranog dokumenta na osnovu karakteristika koje se pridružuju jednom među drvetu koje se naziva *drvo plutajućih objekata*, pri čemu plutajući objekat predstavlja specifikaciju posla koju formater treba da obavi. Ovo drvo se gradi na osnovu raščlanjavanja SGML dokumenta i primene pravila formatiranja zadatih u specifikaciji stila. Izlaz procesa formatiranja može biti SPDL dokument (videti 3.4.2) ili dokument u nekom drugom formatu.

Ova dva jezika podržavaju model DSSSL obrade prikazan na slici 3.15 koji se sastoji od dva različita procesa — procesa transformacije i procesa formatiranja. Ta dva procesa mogu se koristiti zajedno ali i nezavisno jedan od drugoga.



Slika 3.15: Koncepcijski model DSSSL-a

Podršku ovim dvama jezicima pružaju još dva jezika koja se takođe standardizuju DSSSL standardom. To su:

- *upitni jezik* SDQL (skraćenica od engl. *Standard Document Query Language* koristi se za identifikovanje delova SGML dokumenta. Koristi ga i transformacioni i stilski jezik. On omogućava navigaciju kroz hijerarhijsku strukturu SGML dokumenta u toku koje se identifikuju relevantni delovi SGML oznaka i sadržaja nad kojima odgovarajuću obradu treba izvršiti. Jezik obuhvata mehanizme za obradu niski koji omogućavaju obradu neoznačenih podataka, koji se mogu identifikovati na osnovu sadržaja. Kao opciono svojstvo se za identifikovanje informacija može koristiti i HyTime modul za adresiranje lokacija (videti 3.4.3).
- *jezik izraza* služi za kreiranje i manipulaciju objektima. Koriste ga kako upitni jezik, tako i transformacioni i stilski jezik. Zasnovan je na shematskom programskom jeziku R^4RS onako kako je on definisan u IEEE standardu. Kao Lispolik jezik koristi latentne tipove podataka i prefiksnu notaciju i zagrade za izraze i podatke. Procedure su objekti jezika kao i svi drugi u čemu ovaj jezik nalikuje Common Lisp-u. Jezik ima blokovsku strukturu sa statičkim domenom objekata u čemu je takođe nalik Common Lisp dijalektu Lisp-a.

Sama DSSSL specifikacija je jedan SGML dokument koji je saglasan sa arhitekturom DSSSL dokumenta. DSSSL specifikacija, prema deklaraciji sadržaja odgovarajućeg elementa, `dsssl-specification`, sastoji se od ponavljanja u proizvoljnom redosledu specifikacija stila, specifikacija transformacija i spoljašnjih specifikacija. Sadržaj elementa koji odgovara specifikacijama stila, `style-specification-body`, kao i elementa koji odgovara specifikacijama transformacija, `transformation-specification-body`, označen je kao `CDATA` jer ove speci-

fikacije ne koriste SGML jezik već odgovarajući jezik definisan DSSSL standardom, a to znači stilski, odnosno, transformacioni jezik.

Svakoj od ovih specifikacija, kao i svima njima zajedno, mogu prethoditi deklaracije u kojima se navode, na primer, opciona svojstva jezika. Deklaracijama se takođe definiše jedinstveni repertoar karaktera koji uključuje sve karaktere koji se u DSSSL dokumentu koriste. Pomoću njih se, takođe, precizira kako se vrši normalizacija ulaznih karaktera na osnovu njihovog značenja i nezavisno od toga kako su u ulaznom dokumentu uneti (na primer, kao dva koda ili referenca parametarskog entiteta).

Za opis stila HTML dokumenta mogu se koristiti DSSSL jezici. Ovde će biti predstavljeni samo neki ilustrativni primeri. U sledećem primeru opisuje se formatiranje elementa `DIV` — odeljak — u kome se poravnavanje desne margine određuje na osnovu vrednosti atributa `ALIGN` ovog elementa.

```
(element DIV
 (let ((align (attribute-string "align")))
  (make display-group
        quadding: (case align
        (("LEFT") 'start)
        (("CENTER") 'center)
        (("RIGHT") 'end)
        (else 'justify))
        (process-children-trim))))
```

U narednom primeru se formatiranje elementa `SUP` — superskript — određuje kroz promenu veličine fonta i pozicioniranje, relativno u odnosu na tekući redak.

```
(define *ss-size-factor* 0.6)
(define *ss-shift-factor* 0.4)
(element SUP
  (make sequence
        font-size: (* (inherited-font-size)
                      *ss-size-factor*)
        position-point-shift: (+ (* (inherited-font-size)
                                    *ss-shift-factor*))
        (process-children-trim)))
```

U prethodnim primerima treba uočiti da, recimo, `define` i `let` izrazi pripadaju jeziku izraza, `element` pripada upitnom jeziku a `make` pripada stilskom jeziku.

3.4.2 SPDL — Standardni jezik za opis stranice

Ovim standardom definiše se jezik za specifikaciju elektronskih dokumenata koja se sastoje od teksta, slika ili grafike u crnom i belom tonu, u sivim tonovima ili u boji u obliku koji je pogodan za prezentaciju, to jest za štampanje ili prikazivanje na nekom drugom pogodnom medijumu.

Dokumenta koja su u saglasnosti sa ovim standardom nazivaju se SPDL dokumenta (skraćenica od engl. *Standard Page Description Language*). SPDL dokumenta mogu se razmenjivati, skladištiti za kasniju obradu i obrađivati na lokalnim uređajima za prezentaciju ili na uređajima povezanim putem OSI ili nekih drugih mreža.

SPDL dokument je dokument u završnom obliku, to jest u obliku u kome se dokument

više ne može menjati, i on nastaje kao rezultat procesa formatiranja i slaganja. U okviru tog procesa vrši se selekcija glifova za prezentaciju teksta, pozicionira se svaki glif, određuju se krajevi redaka i slično. U njemu logička struktura dokumenta više nije reprezentovana niti su prisutne specifikacije formatiranja.

Prvenstveni izvor SPDL dokumenta je aplikacija koja primenjuje proces formatiranja, slaganja i preloma na dokument koji je u izmenjivom obliku i u okviru koga mogu biti sadržane specifikacije formatiranja ili one, pak, mogu biti zadate spolja, na primer, u vidu DSSSL specifikacije stila. Osim toga, SPDL dokument može nastati i prevođenjem dokumenta formatiranog u nekom drugom formatu ili primenom procesa formatiranja i slaganja na dokument čiji su neki delovi u izmenljivom obliku a neki delovi u formatiranom obliku. Bitno je istaći da „ručna" izrada SPDL dokumenta nije predviđena.

SPDL dokument predstavlja idealnu sliku izrađenu od strane procesa formatiranja i slaganja. Proces prezentacije SPDL dokumenta je zadužen da na konkretnom uređaju izradi što bolju aproksimaciju te idealne slike. U toku procesa prezentacije se stoga podešavaju neki parametri koji su specifični za svaku konkretnu prezentaciju, odnosno uređaj. Ti parametri se specifikuju u okviru instrukcija za produkciju dokumenta koje mogu biti sastavni deo SPDL dokumenta a mogu se zadavati i nezavisno.

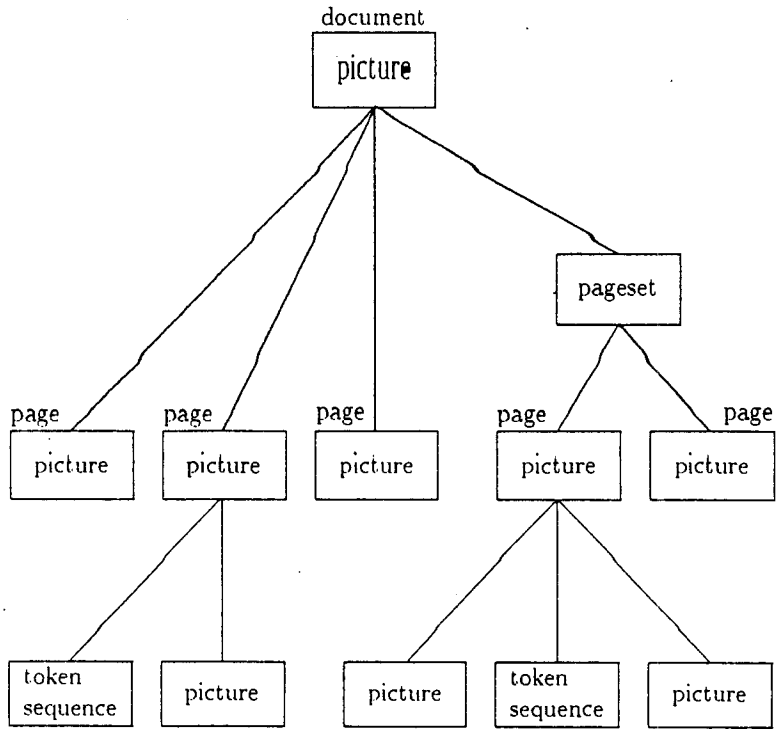
SPDL dokument ima strukturu i sadržaj. Struktura organizuje sadržaj u zasebne delove i identifikuje izvor za svakog od njih. Struktura je celina za sebe i može se obrađivati nezavisno od sadržaja. S druge strane, obrada sadržaja zavisi od obrade strukture koja uspostavlja kontekst u kome se obrada sadržaja odvija.

Struktura dokumenta nastaje kao rezultat podele dokumenta u sve sitnije delove. Primer strukture SPDL dokumenta prikazan je na slici 3.16. Struktura SPDL dokumenta je hijerarhijska i u njenom korenu je element `DOCUMENT` ispod koga mogu biti `PAGESET` i `PAGE` elementi. Deca `PAGESET` elementa su takođe `PAGESET` i `PAGE` elementi. Element `PAGE` sadrži opis onog dela dokumenta koji će biti prezentiran na jednoj instanciji medijuma za prezentaciju — stranici ili ekranu. Njegova deca mogu biti elementi `PICTURE` i `TOKENSEQUENCE`, od kojih samo ovaj drugi sadrži tekst dokumenta. Treba uočiti da se u ovoj strukturi `DOCUMENT` i `PAGE` razlikuju semantički, ali ne i sintaksički. `DOCUMENT` je samo slika koja je na najvišem nivou u hijerarhijskoj strukturi.

SPDL struktura se može lako raščlanjavati na čemu se mogu zasnivati mnoge obrade, uključujući i proces prezentacije. Da bi se podržalo raščlanjivanje SGML dokumenta u različitim okruženjima, standardom su specifikovana dva formata za reprezentaciju i razmenu koja se koriste za kodiranje strukture SPDL dokumenta. To su *binarni format za reprezentaciju i razmenu strukture* koji se zasniva na apstraktnoj sintaksnoj notaciji 1 (ASN.1) definisanoj u [62] i *format čistog teksta za reprezentaciju i razmenu strukture* koji se zasniva na SGML-u. Ova dva formata su potpuno ekvivalentna u smislu funkcionalnosti koje se njima mogu izraziti.

Standard takođe specifikuje i dva formata za reprezentaciju i razmenu koja se koriste za kodiranje sekvencije tokena, osnosno sadržaja. To su *binarni format za reprezentaciju i razmenu sadržaja* ili kompaktno binarno kodiranje koji se koristi kada se koristi binarno kodiranje strukture i *format čistog teksta za reprezentaciju i razmenu sadržaja* ili čitljiv čisti tekst koji se koristi kada se koristi kodiranje čistim tekstom strukture dokumenta.

Ilustracije radi, u donjem primeru je dat izvod iz SPDL definicije tipa dokumenta koji opisuje hijerarhijsku strukturu SPDL dokumenta:



Slika 3.16: Primer strukture SPDL dokumenta

```
<!ENTITY % pgstbdy "(pageset|picture|strctid)" --telo skupa stranica -->
<!ENTITY % body    "(picture|tknseqn|strctid)" --telo slike          -->
<!ENTITY % prlgref "(prologue|strctid)"        --prolog ili reference-->
<!ELEMENT pageset - - ((%prlgref;)?,%pgstbdy;) >
<!ELEMENT picture - - (%body; | nonSPDL | strctid ) >
<!ELEMENT pictbdy - - ((%prlgref;)?,%body;)    -- telo slike         -->
<!ELEMENT nonSPDL - - CDATA                    -- telo ne-SPDL slike -->
```

Deo ASN modula koji sadrži iste strukturne definicije dat je u narednom primeru:

```
Pageset                 ::= [APPLICATION 5] IMPLICIT SEQUENCE {
   comment                    Comment OPTIONAL,
   prologue-or-ref            [0] CHOICE {
      prologue                        Prologue
      reference                       External-Reference } OPTIONAL,
   body                       [1] IMPLICIT SEQUENCE OF REFERENCE {
      pageset                         Pageset,
      picture                         Picture,
      reference                       External-Reference} }
```

3.4.3 HyTime — standard za strukturirana hipermedijalna dokumenta

Osim pojma hiperteksta koji označava grupu tekstova koji se mogu čitati i pretraživati na nelinearan način, danas se sve više sreću pojmovi i drugih vrsta dokumenata, kao što su ([96]):

- multimedijalni dokument kao paket informacija koje su namenjene ljudskoj percepciji a koji koristi osim pisanog teksta i grafike i jedan, ili više, drugih medija. Prezentacija (ili „izvođenje") ovih dodatnih medija može da zauzme vreme i/ili prostor;
- hipermedijalni dokumenti su vrsta multimedijalnih dokumenta u koje su ugrađene hipertekstualne veze;
- dokumenti zasnovani na vremenu koji specifikuju jedan ili više vremenskih rasporeda koga se prezentacija dokumenta treba pridržavati. Primeri su muzičke partiture, animacija, simulacija realnih događaja itd.;
- dokumenti zasnovani na prostoru specifikuju dvodimenzionalni ili trodimenzionalni konačni koordinatni prostor i relativne pozicije koje u tom prostoru objekti sadržani u dokumentu treba da zauzmu prilikom prezentacije. Primer su prozori na ekranu računara koji sadrže objekte različite vrste: tekst, grafiku, ikone, itd.

HyTime predstavlja standardni jezik za reprezentovanje strukture ovih novih vrsta dokumenata (vidi [63], [35]). On omogućava da se na standardan način uspostavi veza sa bilo kakvim objektom koji se nalazi bilo gde i to u bilo kom trenutku, ne standardizujući pri tom same multimedijalne objekte, njihove notacije, semantiku tipova veza i ne zahtevajući nikakvu doradu postojećih dokumenata da bi njihov sadržaj bio povezan sa HyTime dokumentom.

HyTime predstavlja jednu aplikaciju SGML standarda. Pri tome, HyTime ne specifikuje jedan HyTime DTD. Stoga on nije, kao ni SGML, standard koji nešto propisuje već standard koji nešto omogućava. Formalno, on predstavlja skup pravila, koja se nazivaju arhitektonske forme, koja dizajner aplikacije može da primeni u svom DTD.

Arhitektonske forme se po pravilu specifikuju kroz definiciju atributa, od kojih je jedan HyTime atribut koji identifikuje formu. Postoje dve vrste arhitektonskih formi:

- forma tipa elementa koju definiše deklaracija tipa elementa u sprezi sa odgovarajućom listom definicija atributa;
- forma liste atributa koju definiše samo lista atributa.

Jedan tip elementa koji je u saglasnosti sa HyTime formom tipa elementa naziva se HyTime tip elementa, iako sam tip elementa nije definisan u HyTime standardu, već ga definiše aplikacija. Aplikacioni DTD može da sadrži i HyTime tipove elementa i one koji to nisu. Arhitektonska forma tipa elementa strogo specifikuje gde i kako se ove dve vrste elemenata mogu koristiti.

HyTime je konstruisan tako da se može koristiti modularno, što znači da samo oni delovi jezika koji su od značaja za određenu vrstu dokumenata treba da budu podržani. Primarni moduli su:

- bazni modul koji obezbeđuje osnovne funkcije i obavezan je;
- modul za adresiranje lokacija pomoću koga se ukazuje na određeni segment informacije, ma gde da se ona nalazi;
- modul hiperveza koji se koristi za uspostavljanje hipermedijalnih veza, za definisanje sidara (engl. *anchor*) i tkanja (engl. *weave*);
- modul konačnog koordinatnog prostora pomoću koga se specifikuju prostorne i vremenske relacije među objektima.

Uz ove, mogu biti prisutni i sekundarni moduli pomoću kojih se specifikuje prezentacija HyTime dokumenta.

HyTime podržava tri vrste adresiranja. To su:

- adresiranje preko imena koje omogućava da se adresiraju imenovani SGML entiteti i SGML elementi koji se mogu jedinstveno identifikovati u spoljašnjem SGML dokumentu. Ova vrsta adresiranja je najotpornija na promene u objektu koji se adresira.
- adresiranje preko pozicije koje dozvoljava adresiranje objekata u nekom prostoru koji koristi proizvoljan merni sistem. HyTime prostori su ograničena područja koja se mere korišćenjem prebrojivih atoma koji se nazivaju kvantumi. Dimenzije i pozicija ukazane lokacije specifikuju se odbrojavanjem od početka ili kraja koordinatne ose, referisanjem neke lokacije koja je već specifikovana na osi ili relativno u odnosu na neku postojeću specifikaciju dimenzije. Može se koristiti za neimenovane objekte ili proizvoljne delove objekata.
- adresiranje preko semantičkih konstrukata omogućava identifikovanje podskupa podataka sadržanih u nekom širem dokumentu a koje zahteva korišćenje sistema za interpretaciju tih podataka. Sami podaci uopšte ne moraju biti zapisani korišćenjem SGML jezika.

Otvorena je mogućnost da aplikacije kreiraju više forme adresiranja, kao što su, na primer, regularni izrazi, koje se razrešavaju u adresiranje preko imena ili preko pozicije.

U HyTime-u je definisano pet arhitektonskih formi za uspostavljanje hiperveza. To su:

- Nezavisne veze koje predstavljaju najopštiji oblik hiperveze. Ove veze mogu imati proizvoljan broj krajeva. Pri tome kraj veze može direktno da pokazuje sidro, putem SGML mehanizma #ID-#IDREF ili može da ga pokazuje indirektno, korišćenjem istog mehanizma za pokazivanje elementa koji sadrži razvijeniji opis adrese. HyTime takođe obezbeđuje sredstva za specifikovanje sa kog kraja veze putovanje može početi a sa kog je moguć povratak;
- veze osobina uvek imaju samo dva kraja i obično pridružuju osobinu, ime atributa i vrednost, nekom elementu. Ove veze se najčešće koriste za postavljanje tkanja nad dokumentom koji se može samo čitati.
- kontekstualne veze takođe imaju samo dva kraja od kojih je jedan lokacija same veze u dokumentu. Koriste se, na primer, u tekstu za referisanje fusnota.
- veze agregiranih lokacija povezuju više lokacija zajedno tako da se one mogu posmatrati kao jedna agregirana lokacija.
- veze prostiranja dozvoljavaju da se raščlanjena informacija sadržana u susednim SGML elementima posmatra kao da je neizdeljena umetnutim SGML oznakama. Ove veze se, na primer, koriste kada dokument koji se može samo čitati želimo da snabdemo oznakama saglasnim sa DTD koji nije onaj isti pomoću koga je dokument originalno zapisan.

Ovaj modul je zadužen za raspoređivanje objekata. Objekat može u HyTime-u da sadrži ma koju vrstu informacije: digitalizovanu video i/ili audio informaciju, grafičke objekte, programe, karakterski tekst, SGML raščlanjiv ili ne. Objekti se u konačnom koordinatnom prostoru pojavljuju kao sadržaj događaja koga opisuje element **event**. Događaj je kutija u koju objekat treba da se smesti. Svaki događaj ima skup specifikacija dimenzija (nalik onom

koji se koristi za adresiranje lokacije preko pozicije) koje određuju njegovu poziciju i „prostor" na koordinatnim osama konačnog koordinatnog prostora u kome se raspored događaja, tj. element evsched, koji sadrži taj događaj pojavljuje.

Jedan konačan koordinatni prostor može imati proizvoljan broj rasporeda događaja a svaki raspored događaja može da sadrži proizvoljan broj događaja. Dizajner aplikacije određuje kako će se događaji organizovati u rasporede događaja i kako će se rasporedi događaja organizovati među sobom.

Svaki konačni koordinatni prostor određuje svoj domen merenja i referentnu jedinicu za svaku osu. Za sve specifikacije dimenzija koriste se kvantumi kao delovi ili umnošci odgovarajuće jedinice. HyTime standard sadrži deklaracije mnogih kvantuma koje su zasnovane na SI metru i SI sekundi. Dizajner aplikacije može da uvede i druge jedinice prema svojim potrebama.

HyTime jezik je zamišljen tako da se za njega mogu realizovati programi, koji se nazivaju „HyTime mašine" koji obavljaju razrešavanje adresa, vrše povezivanje, podešavaju objekte u prostoru i obavljaju njihovu sinhronizaciju u vremenu, čime se olakšava razvoj aplikacije. Standard ne propisuje nikakvu posebnu arhitekturu za implementaciju ovakve mašine. Šta više, moguće je povezivanje HyTime obrade i aplikacionih programa. HyTime arhitektura je modularna te je potrebno da budu implementirane samo one mogućnosti koje su stvarno potrebne.

Očekuje se da ovaj standard omogući izdavačkoj industriji, pod čime se podrazumeva sveobuhvatna proizvodnja informacija u digitalnom obliku, izradu kvalitetnijih dokumenata, pogodnih za široku upotrebu i sa dugim vekom trajanja.

Glava 4

SGML raščlanjivač

U ovoj glavi biće opisana konstrukcija SGML raščlanjivača kome je na ulazu SGML dokument a čiji je izlaz „kompletan" dokument i odgovarajući indeksi ili poruka o grešci. Uloga SGML raščlanjivača je trostruka. Raščlanjivač prvo treba da učita i obradi SGML deklaraciju i time podesi sopstvene parametre a zatim mora da proveri da li je DTD dokumenta korektan, odnosno, da li poštuje sva pravila i ograničenja predviđena standardom. Na kraju, raščlanjivač mora da proveri da li je dokument saglasan sa DTD-em, da unese sve etikete koje su u SGML dokumentu izostavljene i da proizvede željene indekse.

SGML raščlanjivač koji će ovde biti opisan (u daljem tekstu MSGML) obavlja sva tri navedena zadatka. Moguć je i drugačiji pristup. Tako je *Amsterdamski SGML raščlanjivač* [150] podeljen u dva dela: prvi deo, *dtd raščlanjivač* proverava ispravnost DTD-a i generiše raščlanjivač, odnosno njegov drugi deo, *raščlanjivač dokumenta*, koji proverava saglasnost dokumenta sa DTD-em. Prvi pristup ima prednosti u slučajevima kada se koriste raznovrsni DTD-i ili je DTD dokumenta podložan promenama, na primer u vreme njegove izgradnje. Drugi pristup ima prednosti u slučajevima kada se stabilni DTD koristi za veliki broj dokumenata, jer obrada DTD-a oduzima dosta vremena. Osim toga, standardni programski alati FLEX i YACC mogu se tada koristiti i za sintaksičku analizu DTD-a i za sintaksičku analizu modela sadržaja.

Funkcionalnost SGML-a, kao i varijantna konkretna sintaksa koju SGML dokument koristi, definiše se u SGML deklaraciji. MSGML čita i obrađuje SGML deklaraciju SGML dokumenta ali ne podržava sva dozvoljena svojstva niti sve promene konkretne sintakse. Od opcionih svojstava, MSGML podržava samo izostavljanje etiketa koje je u SGML deklaraciji specifikovano opcijom `OMITTAG`. Amsterdamski SGML raščlanjivač uz izostavljanje etiketa podržava i kratke reference. Kao što će u daljem tekstu biti izloženo, koncepcija MSGML-a je otvorena za ugradnju i ostalih svojstava, osim sponskih procesa.

Amsterdamski SGML raščlanjivač podržava referentnu konretnu sintaksu i referentne skupove kapaciteta i veličina. MSGML takođe ima fiksiran najveći broj elemenata konkretne sintakse i najveći broj veličina i kapaciteta. Ipak, dozvoljene su neke promene kao, na primer, promena ključnih reči kao i promena veličine `LITLEN` koja specifikuje maksimalnu dužinu literala. U ostalim elementima, MSGML podržava referentnu sintaksu i referentne skupove sa izuzetkom izmena koje predviđa TEI (videti 3.3 i dodatak A). Koncepcija MSGML je takva da se mogućnost menjanja SGML deklaracijom svih elemenata referentne sintakse i referentnih skupova u njega ne može jednostavno ugraditi, kao što je recimo slučaj sa promenom skupa graničnika ili dužine imena, što će biti u narednim odeljcima objašnjeno.

SGML standard specifikuje u svom završnom odeljku koje opcione validacione usluge

SGML raščlanjivač može da obezbedi. Ovde će biti navedena lista tih validacionih usluga sa kratkim obrazloženjem i naznakom da li ih i kako MSGML obezbeđuje.

- GENERAL: ova usluga znači da se sve sintaksičke greške u DTD-u i dokumentu otkrivaju i prijavljuju. MSGML u potpunosti podržava ovu uslugu, ali većinu grešaka tretira kao fatalne te posle prijavljivanja greške završava sa radom.
- MODEL: ova usluga znači da se proverava dvosmislenost modela sadržaja svakog elementa (videti 2.2.4 i 4.4.3). MSGML u potpunosti podržava ovu uslugu a pojavu dvosmislenog modela sadržaja tretira kao fatalnu grešku posle koje završava sa radom.
- EXCLUDE: ova usluga znači da se prijavljuju greške do kojih dovode isključenja koja menjaju obavezni ili opcioni status grupe u modelu sadržaja. MSGML podržava ovu uslugu a pojavu takvog isključenja tretira kao fatalnu grešku posle koje završava sa radom.
- CAPACITY: ova usluga znači da se svako prekoračenje kapaciteta prijavljuje. MSGML delimično podržava ovu uslugu, odnosno prijavljuju se samo prekoračenja onih kapaciteta koji su u realizaciji vezani za dimenzije pridruženih veličina.
- SGML: ova usluga znači da se greške u SGML deklaraciji otkrivaju i prijavljuju. MSGML sintaksički proverava ispravnost kompletne SGML deklaracije iako podržava promenu samo manjeg broja parametara.
- NONSGML: ova usluga znači da se pojava nekih, ne obavezno svih, ne-SGML karaktera prijavljuje. MSGML podržava ovu uslugu.
- FORMAL: ova usluga znači da se greške u formalnim javnim identifikatorima otkrivaju. MSGML ovu uslugu ne podržava.

SGML raščlanjivač može se uporediti sa raščlanjivačem nekog višeg programskog jezika i sastoji se od istih blokova: leksičkog analizatora, sintaksičkog analizatora i semantičkog analizatora. Funkcije njegovog leksičkog i sintaksičkog analizatora su uobičajene pa se za njihovu realizaciju mogu koristiti neki od postojećih programskih alata. One će biti opisane u odeljcima 4.1, 4.2 i 4.3. Semantička analiza SGML raščlanjivača se znatno razlikuje od iste faze u raščlanjivaču višeg programskog jezika. Grubo govoreći ona se sastoji iz dve faze: u prvoj se model sadržaja prevodi u konačni automat koji prepoznaje odgovarajući regularni izraz a u drugoj se proverava da li proizvedeni konačni automat prihvata SGML dokument. O tome će biti reči u odeljcima 4.4 i 4.5.

4.1 Leksička analiza

Uloga leksičkog analizatora je pretvaranje struje ulaznih karaktera u struju tokena koji se prosleđuju na dalju obradu sintaksičkom analizatoru.[1] Osim ove, leksički analizator SGML raščlanjivača mora da obavi i sledeće funkcije, čija će realizacija u MSGML raščlanjivaču biti prikazana u ovom odeljku:

- prepoznavanje i odbacivanje komentara;
- prepoznavanje i odbacivanje blanko sekvencija;

[1] Treba obratiti pažnju da tokeni kao izlaz leksičkog analizatora nemaju nikakve veze sa tokenima sadržaja iz 2.2.1. Takođe, imenski i brojčani tokeni iz 2.3 su samo jedna vrsta tokena leksičkog analizatora.

- prepoznavanje i obrada referenci entiteta;
- prepoznavanje i obrada označenih odeljaka;

Leksički analizator MSGML-a je realizovan korišćenjem generatora leksičkih analizatora FLEX, koji je javni pandan paketa LEX [86].

4.1.1 Tokeni

Ime	Niska	Režim	Kontekst	Uloga
and	&	grp		and konektor
com	--	cxt md		početak ili kraj komentara
cro	&#	con lit	cref	otvaranje karakterske reference
dsc	]	ds md	ent	zatvaranje podskupa deklaracija
dso	[	cxt md		otvaranje podskupa deklaracija
ero	&	con lit	nms	otvaranje reference entiteta
etago	</	con (tag)*	gi	otvaranje krajnje etikete
grpc	)	grp		zatvaranje grupe
grpo	(	cxt grp md		otvaranje grupe
lit	"	grp lit md tag		početak ili kraj literala
lita	'	grp lit md tag		alternativni početak ili kraj literala
mdc	>	cxt md		zatvaranje deklaracije oznaka
mdo	<!	con dsm	dcl	otvaranje deklaracije oznaka
minus	-	md	ex	minus; isključenje
msc	]]	con dsm	mse	zatvaranje označenog odeljka
opt	?	grp		indikator opcionog pojavljivanja
or	—	grp		or konektor
pero*	%	dsm grp lit md	nms	otvaranje reference parametarskog entiteta
pic	>	pi		zatvaranje instrukcije obrade
pio	<?	con dsm		otvaranje instrukcije obrade
plus*	+	grp md	(ex)*	obavezan i ponovljiv; uključenje
refc	;	ref		zatvaranje reference
rep	*	grp		opcion i ponovljiv
rni	#	grp md		indikator rezervisanog imena
seq	,	grp		seq konektor
stago	<	con (tag)*	gi	otvaranje početne etikete
tagc	>	(cxt)* tag		zatvaranje etikete
vi	=	tag		indikator vrednosti

Tabela 4.1: Referentni skup graničnika

Leksički analizator MSGML-a prepoznaje pet vrsta tokena. To su:

- ključne reči i ostale vrste imena;
- brojevi;
- brojčani tokeni;

- graničnici;
- literali.

Kao što je u glavi 2 rečeno, SGML koristi veliki broj ključnih reči. Njihov kompletan spisak je dat u [23]. Pretraživanjem ovog spiska leksički analizator za svako ime može da utvrdi da li predstavlja ključnu reč SGML-a ili ne. U slučaju ključne reči, leksički analizator vraća celobrojni kod ključne reči dok se u protivnom vraća i sama leksema. Ukoliko SGML deklaracija promeni ključne reči u podklauzi **NAMES** klauze **SYNTAX**, potrebno je samo promeniti odgovarajuću globalnu tabelu koju zajednički koriste leksički i sintaksički analizator (videti 4.3).

MSGML prepoznaje 29 graničnika. Prema standardu, jedna niska karaktera prepoznaje se kao graničnik samo u određenim režimima prepoznavanja, a u nekim slučajevima i ako je uz to zadovoljeno i neko kontekstualno ograničenje. Uloge graničnika, režimi u kojima se prepoznaju, eventualna kontekstualna ograničenja i niske koje su im dodeljene u referentnoj konkretnoj sintaksi predstavljeni su u tabeli 4.1. Zvezdicom su u tabeli označena mesta koja su pri realizaciji MSGML raščlanjivača zahtevala promene u odnosu na standard. Režimi rada su opisani u tabeli 4.2 a kontekstualna ograničenja u tabeli 4.3. Oni su u leksičkom analizatoru MSGML-a realizovani korišćenjem operatora početnog uslova i pratećeg konteksta generatora leksičkog analizatora FLEX, o čemu će biti reči u sledećem pododeljku.

Prema standardu, granične niske se prepoznaju u redosledu pojavljivanja, bez preklapanja. Na primer, ako su 'abc' i 'bcd' granične niske, a dokument sadrži nisku 'abcd', leksički analizator uvek treba da prepozna graničnik 'abc' i da raščlanjavanje nastavi od 'd' čak i ako je prepoznati graničnik semantički neispravan. Osim toga, ako više graničnih niski počinje istim karakterom prepoznaje se samo najduži od njih bez obzira što on može biti semantički neispravan. U referentnom skupu graničnika, imajući u vidu režime i ograničenja, do takvih situacija ne može ni doći.

Režim	*Značenje*
con	*Sadržaj* (sve između početne i krajnje etikete u instanciji dokumenta) i *označeni odeljak* deklaracije označenog odeljka koja se pojavljuje u sadržaju
cxt	Kontekstualna sekvencija režima **con** i **cxt**
ds	*Podskup deklaracija* (pojavljuju se u definiciji tipa dokumenta, tipa spone i deklaraciji označenog odeljka)
dsm	*Podskup deklaracija* i *označeni odeljak* deklaracije označenog odeljka koja se pojavljuje u poskupu deklaracija
grp	*grupa* (sve između balansiranih **grpo** i **grpc** graničnika)
lit	*literal*
md	*deklaracija oznaka*
pi	*instrukcija obrade*
ref	*referenca opšteg entiteta*, *referenca parametarskog entiteta* i *karakterka referenca*
tag	*početna etiketa* i *krajnja etiketa*

Tabela 4.2: Režimi prepoznavanja

Definisanje alternativnog skupa graničnika u varijantnoj konkretnoj sintaksi u MSGML-u nije moguće zbog načina definisanja obrazaca u FLEX-u. Ugradnja ove mogućnosti zahtevala

Kontekst	*Značenje*
cref	Početni karakter imena ili cifra
dcl	Početni karakter imena, com, dso ili mdc
gi	Početni karakter imena, ili tagc ako je dozvoljeno skraćivanje etiketa, ili grpo ako su dozvoljene konkurentne strukture
mse	mdc
nms	Početni karakter imena, ili grpo ako su dozvoljene konkurentne strukture ili sponski procesi
ex	grpo
ent	Prepoznaje se samo u istom entitetu kome je pripadao i odgovarajući dso

Tabela 4.3: Kontekstualna ograničenja

bi reorganizaciju ovog modula. Iz istog razloga nije moguće menjati ni pravila imenovanja, odnosno nije moguće menjati skup karaktera imena i skup početnih karaktera imena.

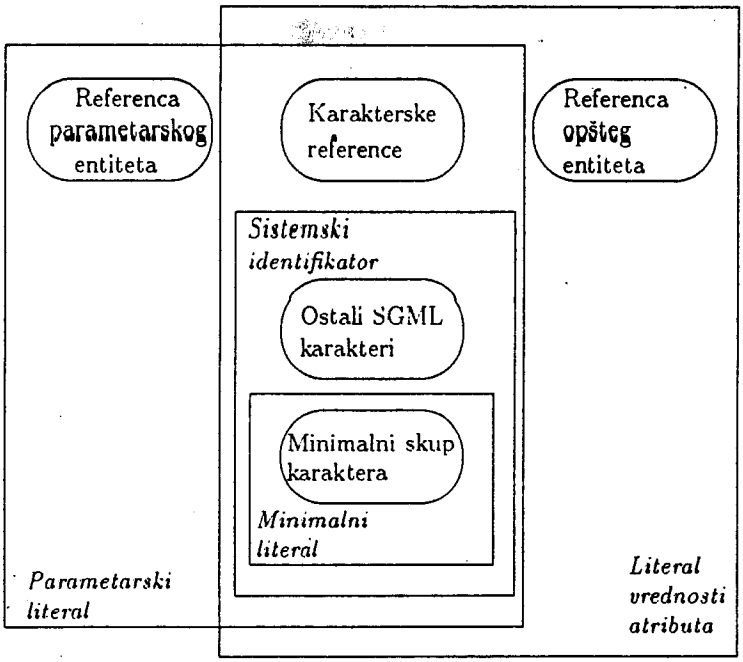
U Standardu su definisane četiri vrste literala, odnosno niski karaktera okruženih sa dva lit, odnosno, sa dva lita graničnika. To su:

- *minimalni literali*, koji se sastoje od minimalnog skupa karaktera u kome je osim malih i velikih slova (engleske abecede) i cifara još i ograničen skup specijalnih karaktera: space, rs, re, '()+,-./:=?. Oni se koriste, recimo, za specifikovanje javnih identifikatora.
- *sistemski identifikatori*, koji se sastoje od potpunog skupa SGML karaktera, što imajući u vidu referentnu konkretnu sintaksu znači da se taj skup u sintaksi FLEX obrazaca može predstaviti kao `[\t\r\n\x20-\x7e]`. (Dodat je samo funkcionalni karakter \t predviđen u TEI).
- *parametarski literal*, koji osim SGML karaktera može da sadrži karakterske reference i reference parametarskih entiteta. Oni se koriste, recimo, za definisanje teksta zamene entiteta.
- *literal vrednosti atributa*, koji osim SGML karaktera može da sadrži karakterske reference i reference opštih entiteta.

Odnos između ovih literala je šematski prikazan na narednoj slici. Leksički analizator MSGML-a prepoznaje pet tokena koji odgovaraju različitim literalima. To su niske koje sadrže samo karaktere iz minimalnog skupa, niske koje sadrže valjane SGML karaktere, niske koje uz valjane SGML karaktere sadrže i karakterske reference, niske koje uz valjane SGML karaktere i karakterske reference sadrže i reference parametarskih entiteta i niske koje uz valjane SGML karaktere i karakterske reference sadrže i reference opštih entiteta. Sadržaj koji karakteriše i razlikuje ove niske predstavljen je na slici ovalima. Sintaksički analizator od njih gradi standardom predviđene literale okupljanjem tokena pomoću odgovarajućih sintaksičkih pravila. Na primer, sintaksičko pravilo za `sistemski_identifikator` je:

```
sistemski_identifikator : minimalni_literal
| SGML_literal
```

Tim sintaksičkim pravilima odgovaraju kvadrati sa slike.



4.1.2 Stanja leksičkog skanera

U prethodnom odeljku bilo je reči o režimima prepoznavanja graničnika. Koncept režima prepoznavanja je dobro poznat iz programskih jezika. Obično su uvek prisutni režim komentara u kome se prepoznaje kraj komentara i režim niske u kome se prepoznaje kraj niske. U SGML je ovaj koncept ekspliciran i daleko složeniji nego u programskim jezicima. Ne samo što je prisutno mnogo više režima od ova dva uobičajena u programskim jezicima, već ti režimi u SGML-u mogu biti i ugnježdeni. Na primer, kad počne deklaracija oznaka, režim prepoznavanja postaje **md** (videti tabelu 4.2). Pojavljivanje **grpo** u tom režimu označava prelazak u režim **grp** u kome **mdc** neće biti prepoznat kao kraj deklaracije oznaka. Pojava **grpc** označava izlazak iz režima **grp** i povratak u režim **md**.

Graničnici nisu do kraja precizno opisani u standardu. Tako standard poznaje graničnik **plus** koji se koristi i za označavanje obaveznih ponovljivih tokena u modelu sadržaja i za uključenja elemenata u model sadržaja. To, međutim dovodi do problema u sintaksičkoj analizi (videti 4.2). S druge strane, za graničnik **pero** navodi se samo njegova uloga u referencama parametarskih entiteta i stoga se prepoznaje samo pod kontekstualnim ograničenjem **nms**. On se koristi i kod deklarisanja parametarskih entiteta a tada se prepoznaje pod drugim kontekstualnim ograničenjem — iza njega je obavezna blanko sekvencija.

Standard nije do kraja precizan i potpun ni u opisivanju režima prepoznavanja. Tako ne postoji poseban režim za komentar što znači da bi, potencijalno, unutar komentara mogli biti prepoznati i drugi graničnici. U standardu se takva situacija sprečava specijalnom naznakom „da se unutar komentara drugi graničnici osim **com** koji završava komentar ne prepoznaju“ koja u suštini implicitno uvodi režim komentara. U standardu se takođe navodi da se u režimu **tag** prepoznaju graničnici **stago** i **etago** premda to nema nikakvog opravdanja jer etikete ne mogu biti ugnježdene. Prema standardu, graničnik **msc** prepoznaje se u režimu **con** koji označava „Sadržaj i označeni odeljak deklaracije označenog odeljka koja se pojavljuje u sadržaju“ premda se on kao graničnik može prepoznati samo u deklaraciji označenog odeljka

stanje	mdo/ dcl	mdo/ dso	pio	tago/ gi	tagc/ gi	mdo/ com	dsc	com	dso
init	dozn		inob	etik					
dsk	dozn	ozod	inob				*pov*		
ozdsk	dozn	ozod	inob						
dozn								kom	dsk
etik									
grup									
inob									
kom								*pov*	
navn									
karref									
ozod									ozdsk ozsadmix ozsadelm ozodig *brn* + +
ozodig									
sadred					etik	dozn			
sadcd					etik	dozn			
sadmix		ozod	inob	etik	etik	dozn			
ozsadmix		ozod	inob	etik	etik	dozn			
sadelm		ozod	inob	etik	etik	dozn			
ozsadelm		ozod	inob	etik	etik	dozn			
sademp					etik				

Tabela 4.4: Promena stanja leksičkog analizatora

a ne i u sadržaju.

Poseban problem predstavlja prepoznavanje oznaka u sadržaju SGML dokumenta. Uz režim **con** je u standardu navedena naznaka „da većina graničnika neće biti prepoznata u sadržaju ako je on deklarisan kao `CDATA` ili `RCDATA`". Ovim su opet implicitno uvedeni novi režimi prepoznavanja. Osim toga, kao što je rečeno u 2.2.1, blanko sekvencije se drugačije tretiraju u elementu sa mešovitim sadržajem u odnosu na element koji se sastoji samo od drugih elemenata. Jasno je, stoga, da se leksička analiza sadržaja SGML dokumenta zasniva na deklaraciji elemenata te se leksički analizator mora oslanjati na rezultate rada ostalih modula raščlanjivača o čemu će biti reči u 4.3.

Iz svih ovih razloga, leksički analizator MSGML-a ima 19 stanja u odnosu na 9 režima prepoznavanja prisutnih u standardu a ugradnja svih opcionih svojstava jezika bi možda taj broj i povećala. Zanimljivo je da leksički analizator Amsterdamskog SGML raščlanjivača takođe ima 19 režima prepoznavanja, ali u [150] nije objavljeno koji su pa se ova dva pristupa ne mogu uporediti. U MSGML-u su to sledeća stanja:

- **init**, početno stanje;
- **dsk**, podskup deklaracija;
- **ozdsk**, podskup deklaracija unutar deklaracije označenog odeljka;
- **dozn**, deklaracija oznaka;
- **etik**, početna ili krajnja etiketa;
- **grup**, grupa;
- **inob**, instrukcija obrade;

- **kom**, komentar;
- **navn**, niska između navodnika;
- **karref**, karakterska referenca;
- **ozod**, deklaracija označenog odeljka;
- **ozodig**, označeni odeljak koji se ignoriše;
- **sadrcd**, sadržaj RCDATA;
- **sadcd**, sadržaj CDATA;
- **sadmix**, mešoviti sadržaj;
- **ozsadmix**, mešoviti sadržaj u deklaraciji označenog odeljka;
- **sadelm**, sadržaj se sastoji od elemenata;
- **ozsadelm**, sadržaj koji se sastoji od elemenata u deklaraciji označenog odeljka;
- **sademp**, prazan sadržaj.

stanje	lit ili lita	grpo	mdc	pic	grpc	tagc	cro	refc ili \n	msc/ mdc
init									
dsk									
ozdsk									$\underset{pov}{brn}$ – – i
dozn	navn	grup	*pov*						
etik	navn					sadrcd sadrc sadmix sadelm sademp *pov*			
grup	*brn* + +			$\underset{pov}{brn}$ – – i					
inob				*pov*					
kom									
navn	*pov*						karref		
karref								*pov*	
ozod			*pov*						*pov*
ozodig									$\underset{pov}{brn}$ – – i
sadrcd							karref		
sadcd									
sadmix							karref		
ozsadmix							karref		$\underset{pov}{brn}$ – – i
sadelm							karref		
ozsadelm							karref		$\underset{pov}{brn}$ – – i
sademp									

Tabela 4.4: Promena stanja leksičkog analizatora — nastavak

U tabeli 4.4 opisan je prelazak iz jednog stanja leksičkog analizatora u drugo. U poljima u kojima je navedeno više stanja to znači da je za prelazak u neko od navedenih stanja potrebno da bude ispunjen neki uslov. Na primer, graničnik **tagc** iz stanja **etik** vodi u stanje **sadrcd**, **sadcd**, **sadmix**, **sadelm**, odnosno, **sademp**, u zavisnosti od toga kako je deklarisan sadržaj elementa čija je etiketa upravo učitana. Moguć je i povratak u prethodno stanje ukoliko je učitana krajnja etiketa. Slična je situacija i u stanju **ozod** odakle graničnik **dso** vodi

u stanje **ozsadmix**, **ozsadelm** ili **ozdsk** u zavisnosti od stanja koje je prethodilo stanju **ozod**: **ozsadmix**, **ozsadelm**, odnosno, **dsk**. Moguć je i prelazak u stanje **ozodig** ukoliko se označeni odeljak ignoriše ali mu se mora pronaći odgovarajući **mdc**.

Neki graničnici ne dovode do promene stanja, već iziskuju neku drugu akciju koja će uticati na promenu stanja u nekom kasnijem koraku. Tako **grpo** u stanjima **dozn** i **grup** uvećava brojač nivoa ugnježdenosti zagrada, dok **grpc** u stanju **grup** umanjuje taj isti brojač a povratak na prethodno stanje vrši se samo u slučaju kada je taj brojač nula. Slična je situacija i sa graničnicima **mdo/dso** i **msc/mdc** osim u stanju **ozod** u kome se ugnježdenost ovih graničnika ne proverava već prvi **msc/mdc** zatvara označeni odeljak. (U ovom stanju se ostaje ako je označeni odeljak deklarisan kao `RCDATA` ili `CDATA`.)

Kao što je već rečeno, leksički analizator obrađuje označene odeljke i reference entiteta. Najveći deo obrade označenih odeljaka sastoji se u korektnom prepoznavanju graničnika i statusa odeljka. Za uključene odeljke (status `INCLUDE`) dalja obrada teče na uobičajen način za odgovarajuću vrstu sadržaja odeljka.

Da bi se korektno obradile reference entiteta, osim karakterskih, potrebno je znati tekst zamene entiteta i njegovu vrstu a to se može saznati konsultovanjem odgovarajuće tabele simbola koju popunjava sintaksički analizator (videti 4.3). Jednom kada su tekst zamene i vrsta entiteta iščitani, leksički analizator treba da preusmeri ulaznu struju. Ovakav koncept je opet poznat iz programskih jezika u kojima često postoji makro ili direktiva tipa `include` koja u tekst programa uključuje sadržaj neke druge datoteke. Ovaj koncept je u SGML-u složeniji jer se u SGML dokument može uključiti kako sadržaj neke datoteke tako i niska pridružena entitetu kao njegov tekst zamene.

Preusmeravanje se u FLEX-u može realizovati korišćenjem više ulaznih bafera, čemu podršku pruža više procedura i makroa. Da bi se realizovale reference entiteta bilo je, međutim, potrebno promeniti FLEX-ov makro YY_INPUT tako da se omogući punjenje tekućeg ulaznog bafera bilo iz neke datoteke bilo iz niske — teksta zamene entiteta. Kako reference entiteta mogu biti ugnježdene, otvoreni ulazni baferi čuvaju se na stogu čija je dubina određena veličinom `ENTLVL`. Za manipulaciju stogom je u MSGML-u korišćen skup makroa opisan u [53] koji koristi statički stog te zbog toga promena ove veličine u SGML deklaraciji za sada nije moguća.

Treba još istaći da se svaka referenca entiteta ne razrešava. Kako je rečeno, entiteti mogu biti ugnježdeni što znači da tekst zamene nekog entiteta može sadržati reference drugih entiteta. Prilikom obrade deklaracije entiteta u tabeli simbola treba zapamtiti njenu nisku zamene bez razrešavanja referenci entiteta koji u njemu mogu eventualno biti sadržani. Razrešavanje entiteta u ovoj fazi moglo bi da dovede do registrovanja neprihvatljivo dugačkih niski zamene. Kako se u ovoj tački pojavljuje konflikt između onoga što treba da uradi sintaksički analizator a to je prepoznavanje deklaracije entiteta i onoga što treba da uradi leksički analizator a to je razrešavanje entiteta, on je u MSGML-u razrešen uvođenjem specijalnog slučaja: reference entiteta se ne razrešavaju unutar literala koji je unutar deklaracije oznaka, tj. ne razrešavaju se u stanju **navn** ako je prethodno stanje **dozn**.

4.2 Sintaksička analiza

Sintaksički analizator MSGML-a realizovan je pomoću paketa BYACC koji je javni pandan YACC-a [69]. On prepoznaje LALR(1) gramatike i ima ugrađena pravila za razrešavanje dvosmislenosti. Gramatička pravila iz ulazne YACC-ove datoteke data su u dodatku C.

Treba podsetiti da celokupan SGML nije ovim pravilima obuhvaćen već samo onaj njegov deo koji ulazi u bazični SGML. Gramatika ima 338 pravila, 84 neterminala i 223 terminala a prebaci/svedi raščlanjivač ima 536 stanja.

Pri prevođenju pravila izvođenja koja su data u standardu u gramatička pravila YACC-a trebalo je rešiti određen broj problema koji su uglavnom nastali usled nepreciznosti samog standarda. Prvi od problema nastupa u prepoznavanju graničnika **plus** koji u deklaraciji oznaka može imati dvostruku ulogu: on može označavati da je model sadržaja obavezan i ponovljiv a takođe označava i prisustvo izuzetka uključenja. Prema standardu, on se prepoznaje u režimima **grp** i **md** pod ograničenjem **ex** koje u suštini govori da ograničenje postoji samo u režimu **md** a da ga u režimu **grp** nema. Međutim takva definicija graničnika **plus** dovodi do prebaci/svedi konflikta koji se ne može uspešno prevazići pravilima za razrešavanje dvosmislenosti.

Do konflikta dovode pravila gramatike označena brojevima 216, 218, 220, 237–241 u dodatku C koja opisuju sadržaj elementa u slučaju kad je on opisan modelom sadržaja. Ako u tim pravilima imena neterminala lišimo svake semantike a delove koji nisu od značaja za generisanje konflikta zamenimo terminalima, dobićemo sledeću gramatiku:

(1)	$A \rightarrow$	B C	(5)	$E \rightarrow$	ε
(2)	$C \rightarrow$	D E	(6)	$E \rightarrow$	+ lis
(3)	$D \rightarrow$	ε	(7)	$B \rightarrow$	grp
(4)	$D \rightarrow$	− lis	(8)	$B \rightarrow$	grp +

Kolekcija LR(1) skupova generisana iz ove gramatike je:

I_0:	$A' \rightarrow$	$\cdot A$, \$	$I_2 \xrightarrow{C} I_4$:	$A \rightarrow$	$BC\cdot$, \$
	$A \rightarrow$	$\cdot BC$, \$			
	$B \rightarrow$	$\cdot$**grp**, −/+/\$	$I_2 \xrightarrow{D} I_5$:	$C \rightarrow$	$D \cdot E$, \$
	$B \rightarrow$	$\cdot$**grp** +, −/+/\$		$E \rightarrow$	$\cdot$ + lis, \$
				$E \rightarrow$	$\cdot$, \$
$I_0 \xrightarrow{A} I_1$:	$A' \rightarrow$	$A\cdot$, \$			
			$I_2 \xrightarrow{-} I_6$:	$D \rightarrow$	$- \cdot$ lis, +/\$
$I_0 \xrightarrow{B} I_2$:	$A \rightarrow$	$B \cdot C$, \$			
	$C \rightarrow$	$\cdot DE$, \$	$I_3 \xrightarrow{+} I_7$:	$B \rightarrow$	**grp** + $\cdot$, −/+/\$
	$D \rightarrow$	$\cdot -$ lis, +/\$			
	$D \rightarrow$	$\cdot$, +/\$	$I_5 \xrightarrow{E} I_8$:	$C \rightarrow$	$DE\cdot$, \$
$I_0 \xrightarrow{\text{grp}} I_3$:	$B \rightarrow$	**grp**$\cdot$, −/+/\$	$I_5 \xrightarrow{+} I_9$:	$E \rightarrow$	+ $\cdot$ lis, \$
	$B \rightarrow$	**grp** $\cdot$ +, −/+/\$			
			$I_6 \xrightarrow{\text{lis}} I_{10}$:	$D \rightarrow$	− lis$\cdot$, +/\$
			$I_9 \xrightarrow{\text{lis}} I_{11}$:	$D \rightarrow$	+ lis$\cdot$, \$

Konflikt se generiše iz skupa I_3. Član $B \rightarrow$ **grp** $\cdot$ +, −/+/\$ generiše u tabeli simbola akciju „pomeri simbol '+' i pređi u stanje I_7". Drugi član ovog skupa, $B \rightarrow$ **grp**$\cdot$, −/+/\$ generiše akciju „svedi vrh stoga po gramatičkom pravilu (8)" za ulazne simbole −, + i \$. Na taj način

se u stanju 3 tabele simbola za simbol '+' u ulaznoj struji pojavljuje prebaci/svedi konflikt. Ovaj konflikt se može razrešiti uvođenjem dva različita simbola '+': jedan koji označava ponavljanje obavezne grupe u modelu sadržaja i drugi koja označava uključenje. To je u gramatici MSGML-a i učinjeno u pravilu 239. Leksički analizator pri tome nema nikakvih problema da razlikuje ova dva različita graničnika: ovaj drugi se prepoznaje u stanju **dozn** samo po uslovom da iza njega sledi **grpo**, dok se prvi prepoznaje u stanjima **dozn** i **grp** i ne zahteva prateći kontekst.

Drugu vrstu konflikta izazivaju pravila za specifikaciju vrednosti atributa (pravila 285–288). Pokazuje se da su ova pravila u standardu nepotpuna. Puna specifikacija vrednosti atributa bila bi:

```
spec_vrednosti_atributa : zamenljivi_literal
| Kar_podaci /* ako je dekl. CDATA */
| Ime        /* ako je ENTITY, ID, IDREF, NAME */
| ime_lista  /* ako je ENTITIES, IDREFS, NAMES */
| Broj       /* ako je NUMBER */
| broj_lista /* ako je NUMBERS */
| Brtoken    /* ako je NUTOKEN */
| brtoken_lista     /* ako je NUTOKENS */
| token_imena       /* ako je NMTOKEN */
| token_imena_lista /* ako je NMTOKENS */
ime_lista         : Ime
  | ime_lista Ime
broj_lista        : Broj
  | broj_lista Broj
brtoken_lista     : Brtoken
  | brtoken_lista Brtoken
token_imena_lista : token_imena
  | token_imena_lista token_imena
```

U primeni ovakvih gramatičkih pravila javlja se dosta problema. Pre svega, `Ime`, `Broj` i `Brtoken` su tokeni dok je `token_imena` neterminal definisan pravilima 270–272. Njegovo uključivanje u listu mogućnosti za specifikaciju vrednosti atributa dovodi do niza svedi/svedi konflikta koji se mogu razrešiti samo na semantičkom nivou. Za atribute čija je deklarisana vrednost `NMTOKEN`, raščlanjivač treba da proveri da li je specifikovana vrednost `Ime`, `Broj` ili `Brtoken`.

Veći problem prouzrokuju liste imena čije uključivanje među pravila za specifikaciju vrednosti atributa dovodi do prebaci/svedi konflikta. Neka je raščlanjivač u jednom od sledeća dva stanja:

```
 stanje a
spec_vrednosti_atributa : Ime .
ime_lista : Ime . Ime
specif_atributa : Ime . Vi spec_vrednosti_atributa

 stanje b
spec_vrednosti_atributa : lista_imena .
ime_lista : ime_lista . Ime
```

Kada se u ovim stanjima u ulaznoj struji pojavi token **Ime**, onda se taj simbol može prebaciti na stog prebaci/svedi raščlanjivača u nameri da se ime doda listi imena ili se, pak, vrh stoga može svesti pravilom za **spec_vrednosti_atributa** u uverenju da to ime označava ime sledećeg atributa u skupu specifikacija atributa (pravila 311–314).

Naznake za rešavanje ovog problema postoje u samom standardu i sadržane su u sledećim iskazima:

1. zamenljivi literal se interpretira kao specifikacija vrednosti atributa pošto se zamene reference svih entiteta unutar njega;
2. specifikacija vrednosti atributa može biti vrednost atributa koja nije zamenljivi literal ako sadrži samo karaktere imena i ako se pojavljuje unutar liste definicija atributa ili ako je dozvoljeno skraćivanje etiketa u SGML deklaraciji.

Drugi iskaz znatno ograničava specifikaciju vrednosti atributa. Dakle, jedine vrednosti atributa koje se mogu specifikovati izvan graničnika **lit** ili **lita** su **Ime**, **Broj** i **Brtoken** i to pod određenim uslovima. Prvi iskaz, međutim, govori da se zamenljivi literal mora interpretirati a tu interpretaciju izvršava sintaksički analizator koji koristiti „mali" leksički analizator koji će u zamenljivom literalu prepoznati imena, brojeve, brojčane tokene i njihove liste i rezultat analize uporediti sa deklarisanom vrednošću atributa.

Gramatička pravila za specifikaciju vrednosti atributa, kao i njihovo tumačenje, su u standardu nespretno izrečena i podložna su različitim tumačenjima. U MSGML raščlanjivaču je, stoga, usvojeno da se vrednost atributa može specifikovati samo kao ime, broj, brojčani token ili zamenljivi literal. Sve ostale mogućnosti se proveravaju prilikom interpretacije zamenljivog literala. Vrednost atributa koja nije specifikovana kao zamenljivi literal mora se dodatno proveriti: njeno pojavljivanje unutar početne etikete je dozvoljeno samo ako je specifikovano **SHORTTAG YES** u SGML deklaraciji.

Konačno, gramatička pravila koja opisuju instanciju dokumenta data u standardu su formalno dvosmislena u dva aspekta. Prvi se tiče praznih elemenata. Posle početne etikete, element sa praznim sadržajem je završen i ne sme imati krajnju etiketu. Pravilo za opisivanje elemenata u standardu je:

```
element : poc_etiketa? sadrzaj krj_etiketa?
```

koje je u sintaksičkom analizatoru MSGML-a zamenjeno pravilom 316 u kome su i početna i krajnja etiketa obavezne. Leksički analizator elementima čiji sadržaj je deklarisan kao **EMPTY** na početnu etiketu odmah dodaje i krajnju etiketu tako da sintaksički analizator sve elemente može da tretira na isti način. To se obavlja kroz proces saradnje leksičkog i sintaksičkog analizatora o čemu će biti govora u 4.3.

Druga dvosmislenost potiče od sadržaja elementa. On je u standardu definisan kao:

```
sadržaj : mešoviti sadržaj
        | sadržaj od elemenata
        | zamenljivi karakterski podaci
        | karakterski podaci
```

Međutim razlikovanje ovih sadržaja ne potiče od samog sadržaja već od njegove interpretacije koja pak zavisi od toga kako je on deklarisan. Ova dvosmislenost je, stoga, uklonjena drugačijim pravilima po kojima je sadržaj neuređena sekvencija karakterskih podataka, elemenata i dozvoljenih SGML deklaracija (321–324). Šta će zaista činiti sadržaj elementa

zavisi od interpretacije ulazne struje posle početne etikete a ona opet zavisi od deklaracije odgovarajućeg elementa. Leksički analizator zato i ima pet različitih stanja, **sadrcd**, **sadcd**, **sadmix**, **sadelm**, i **sademp** za razlikovanje pet različitih tipova sadržaja.

4.3 Saradnja leksičkog i sintaksičkog analizatora

Leksički analizator, učitavajući ulaznu datoteku, svaki izdvojeni token prosleđuje sintaksičkom analizatoru a uz token, prema potrebi, i odgovarajuću leksemu. Međutim, osim direktno, leksički i sintaksički analizator sarađuju i preko tabela simbola koje se grade u toku raščlanjavanja. Raščlanjivač MSGML gradi tabele simbola za osnovne konstrukte SGML-a a to su: entiteti, elementi, atributi i notacije.

Svim tabelama simbola se manipuliše preko odgovarajućih melanžnih tabela (engl. *hash*). Struktura ovih tabela, skup funkcija koje njima upravljaju kao i melanžna funkcija preuzeti su iz [53]. Melanžne tabele su ogranizovane po vedricama (engl. *bucket*) tako da svaka vedrica sadrži pokazivač stvarne tabele simbola, odnosno, pokazivač korisničkog prostora tabele simbola. Funkcije za upravljanje melanžnim tabelama pretpostavljaju da se na početku korisničkog prostora tabele simbola nalazi sam ključ tabele simbola, a ne pokazivač ključa. U tabelama simbola raščlanjivača MSGML ključevi su imena odgovarajućih konstrukata, entiteta, elemenata i slično, čija je dužina određena vrednošću veličine **NAMELEN**. Ovaj zahtev koji postavljaju funkcije za upravljanje melanžnim tabelama je razlog što se ova veličina u tekućoj implementaciji MSGML-a ne može menjati SGML deklaracijom.

Za saradnju između leksičkog i sintaksičkog analizatora od najvećeg su značaja tabele simbola entiteta i elemenata koje će biti predstavljene u ovom odeljku. Njihov sadržaj biće predstavljen definicijom odgovarajuće strukture u C jeziku.

U tabeli simbola entiteta se, osim imena entiteta i (pokazivača) teksta zamene, beleže podaci koji pobliže određuju vrstu entiteta: da li je parametarski ili opšti entitet, da li je tekst zamene unutrašnji, odnosno, literal ili spoljašnji. Dalje određenje je ponekad potrebno za unutrašnje parametarske entitete i za spoljašnje opšte entitete a za ove poslednje je u tom slučaju potrebno precizirati još i notaciju i vrednosti specifikovanih atributa (videti 2.4).

```
typedef struct ent_sim          /* tabela simbola entiteta */
{
   unsigned char ime[Namelen+1]; /* ime entiteta */
   unsigned int par_ops;    /* 0 ako parametarski 1 ako opsti        */
   unsigned int lit_spo;    /* 0 ako literal 1 ako spoljasnji        */
   unsigned int ent_tip;    /* ako lit:  ;            ako ops i spo: */
    /* 0 nije podat./zagradj.     0 nema tip  */
    /* 1 PI (podat.)              1 NDATA     */
    /* 2 CDATA (podat.-samo ops.) 2 CDATA     */
    /* 3 SDATA (podat.-samo ops.) 3 SDATA     */
    /* 4 STARTTAG (zagradj.)                  */
    /* 5 ENDTAG (zagradj.)                    */
    /* 6 MS (zagradj.)                        */
    /* 7 MD (zagradj.)                        */
    char *ent_txt;          /* pokazivac pridruzenog teksta          */

    struct notacije *notac; /* NULL ako nije opsti ent. ili nema tip */
```

```
    struct link_das *veza;  /* NULL ako nije opsti entitet ili       */
    /* nema tip - inace pokazivac prvog      */
    /* u listi specifikacija atributa        */
    struct link_das *eveza; /* NULL ili pokazivac poslednjeg  */
} ent_sim;
```

Premda se tabela simbola entiteta formira u toku sintaksičke analize, sam sintaksički analizator je takoreći ne koristi. Njeno jedino korišćenje u sintaksičkom analizatoru je provera jedinstvenosti imena entiteta. Glavno korišćenje ove tabele je od strane leksičkog analizatora. Kao što je već rečeno u 4.1, razrešavanje entiteta obavlja leksički analizator i to upravo korišćenjem ove tabele. Za referisani entitet se iz tabele simbola pronalazi tekst zamene a na osnovu vrste entiteta se on na sledeći način interpretira:

- ako je unutrašnji, ulazna struja se preusmerava na tekst zamene; ako je spoljašnji, unutrašnja struja se preusmerava na datoteku koju identifikuje tekst zamene;
- ako je parametarski, koristi se samo unutar deklaracije oznaka;
- ako je unutrašnji podatkovni entitet, podaci se tretiraju na način određen vrstom podataka. Na primer, ako je vrsta CDATA, tekst zamene se bez daljeg raščlanjavanja prepisuje u izlazni dokument; ako je unutrašnji zagrađeni entitet, onda se u ulaznu struju, odnosno, u ulazni bafer upisuje odgovarajuća oznaka koja dalje prolazi uobičajeno raščlanjavanje;
- ako je spoljašnji entitet specifičnog tipa, sadržaj datoteke identifikovane tekstom zamene biće interpretiran na način koji zavisi od pridružene notacije. Obrada ovakvih spoljašnjih entiteta u MSGML još nije ugrađena.

U tabeli simbola elemenata se osim imena elementa i njegovog jedinstvenog identifikacionog broja beleži još samo pokazivač zapisa koji sadrži sve ostale podatke koji opisuju sadržaj tog elementa. Ovakvo razdvajanje podataka vezanih za jedan element je učinjeno stoga što identičan sadržaj može biti pridružen većem broju elemenata navođenjem grupe imena u deklaraciji elemenata (videti 2.2). Jedinstveni identifikacioni broj elementa koriste nedeterministički konačni automati kojima se opisuje model sadržaja elemenata (videti 4.4).

```
typedef struct elm_sim                /* tabela simbola elemenata */
{
   unsigned char ime[Namelen+1]; /* ime elementa */
   unsigned int id_broj;        /* identifikacioni broj elementa */
   struct   element *sdr_elm;  /* pokazivac sadrzaja elementa -
* vise elemenata moze imati isti sadrzaj */
        /* kao  posledica koriscenja grupe imena */
   struct   elm_sim *sledeci;  /* sledeci element sa istim sadrzajem */
      /* kruzna lista */
} elm_sim;
```

Kao što je rečeno u 4.2, interpretacija sadržaja elementa u instanciji dokumenta zavisi od toga kako je sadržaj tog elementa deklarisan. Stoga, kada se u stanju **etik** leksičkog analizatora prepozna kraj početne etikete, leksički analizator mora utvrditi kakav je sadržaj elementa koji se otvara da bi znao kako da interpretira nisku karaktera do krajnje etikete istog elementa. Stoga se mora konsultovati vrednost polja `tip_sdr` strukture `element` tabele

simbola elemenata. U zavisnosti od vrste sadržaja, leksički analizator će preći u odgovarajuće stanje. O ostalim poljima zapisa **element** biće govora u narednim odeljcima.

```
typedef struct element          /* sadrzaj elementa iz tabele simbola
  elemenata */
{
   unsigned int min_p;     /* 1, izostavljanje pocetne etikete dozv. */
   unsigned int min_k;     /* 1, izostavljanje krajnje etikete dozv. */
   unsigned int tip_sdr;  /* tip sadrzaja: */
  /* 1 Sadrzaj elementa su CDATA podaci */
  /* 2 Sadrzaj elementa su RCDATA podaci */
  /* 3 Sadrzaj elementa je prazan */
  /* 4 Element moze imati bilo kakav sadrzaj */
  /* 5 Sadrzaj elementa je mesovit */
  /* 6 Sadrzaj elementa su drugi elementi */
   struct   nfa *poc_st;  /* pocetno stanje odgovarajuceg NKA */
  /* ili NULL za deklarisani sadrzaj  */
   struct   nfa *zav_st;  /* zavrsno stanje odgovarajuceg NKA */
  /* ili NULL za deklarisani sadrzaj  */
   SET      *obav_prol;   /* skup simbola bez kojih se ne moze    */
  /* proci od pocetnog do zavrsnog stanja */
   struct lista_elem *isklj;  /* lista elemenata koji se iskljucuju */
      /* iz modela sadrzaja */
   struct lista_elem *uklj;   /* lista elemenata koji se ukljucuju */
      /* u model sadrzaja */
   struct lista_atrb *atributi; /* lista definicja pridruzenih atributa */
} element;
```

4.4 Opis sadržaja elementa

Za element koji ima deklarisani sadržaj nije potreban nikakav dodatni opis tog sadržaja od onog koji daje polje **tip_sdr** strukture **element** tabele simbola elemenata koje obezbeđuje adekvatnu interpretaciju njegovog sadržaja u instanciji dokumenta. Za elemente čiji je sadržaj opisan modelom sadržaja mora se proveriti saglasnost sadržaja u instanciji dokumenta sa zadatim modelom.

Grupa modela koja opisuje sadržaj elemenata u velikoj meri podseća na regularne izraze iz teorije automata koja predstavlja teorijsku osnovu za neke aspekte pojma saglasnosti sa modelom sadržaja. U dodatku H iz [57] pojašnjeno je na koji način grupi modela odgovaraju regularni izrazi, a posebno u sledećim slučajevima:

1. **and** grupa je ekvivalentna sa **or** grupom **seq** grupa permutacija; na primer grupa modela (a & b) je ekvivalentna sa regularnim izrazom, odnosno grupom modela, ((a, b) | (b, a));
2. token sa indikatorom pojavljivanja **opt** se svodi na **or** grupu koja sadrži taj isti token i ništičav token. Na primer, (a?) je ekvivalentno regularnom izrazu (a |) koji sam nije valjana SGML grupa modela.

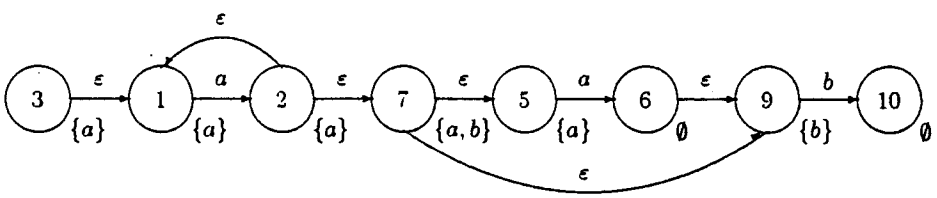
Postoje, međutim, odstupanja koncepta SGML grupe modela od pojma regularnih izraza. Pre svega, model sadržaja koji se svodi na ekvivalentan regularni izraz, ne mora mu biti ekvivalentan u druge svrhe. Tako, mogućnost izostavljanja početne etikete zavisi od samog oblika modela sadržaja. Na primer, modeli sadržaja (a?, b) i ((a, b) | b) svode se na iste regularne izraze ali u prvom slučaju se početna etiketa elementa b može izostaviti dok u drugom slučaju to ne bi bilo dozvoljeno.

4.4.1 Konstrukcija nedeterminističkih konačnih automata

Provera saglasnosti sa modelom sadržaja je u suštini ekvivalentna sa problemom prepoznavanja, odnosno prihvatanja ili odbacivanja, regularnih izraza. Kako su regularni izrazi ekvivalentni determinističkim konačnim automatima, skraćeno DKA, raščlanjivač bi mogao da obrađuje modele grupa svodeći ih na regularne izraze i konstruišući za njih DKA kod koga su putevi prelaska obeleženi tokenima iz grupe modela.

Kod primene ovog postupka javljaju se dva problema. Prvi potiče od svođenja **and** grupe na regularni izraz o čemu će biti reči u tački 4.4.2. Drugi problem proizlazi iz konstrukcije DKA. Najčešće korišćeni metod sastoji se iz konstrukcije nedeterminističkog konačnog automata, skraćeno NKA, direktno iz regularnog izraza a zatim transformacije tog NKA u ekvivalentan DKA što može zahtevati dosta vremena. Konstruisanje DKA za NKA koji odgovara modelu sadržaja nije, međutim, ni potrebno jer su modeli sadržaja koji su dvosmisleni ili zahtevaju preduvid, standardom zabranjeni (videti 2.2.4). Rezultat toga je, prema dodatku H iz [57] da se modeli sadržaja mogu svesti na regularne izraze čiji se NKA mogu deterministički obilaziti zato što za njih važi samo jedno od sledeća dva tvrđenja:

1. dati čvor NKA može napustiti samo jedna sekvencija lukova koja ne sadrži više od jednog obeleženog luka a taj obeleženi luk odgovara tekućem tokenu;
2. ako više takvih sekvencija lukova može napustiti čvor NKA, samo jedan od njih može ući u njega i tom luku pravila SGML daju prioritet.



Slika 4.1: NKA za model sadržaja (a+, a?, b)

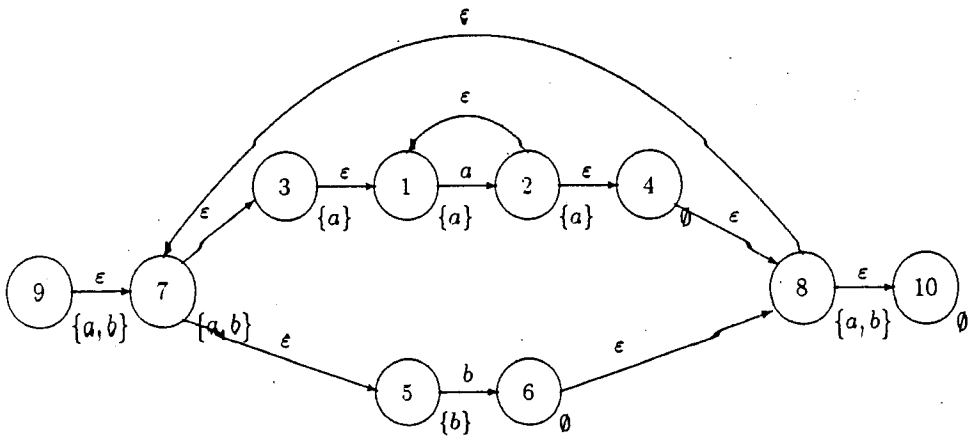
Prvo tvrđenje potiče od zahteva da modeli sadržaja ne smeju biti dvosmisleni dok druga dozvoljava takve modele u slučajevima koji se mogu razrešiti primenom prioriteta pri sravnjivanju modela. Ona se može ilustrovati na primeru modela sadržaja (a+, a?, b). Na slici 4.1 je prikazan NKA konstruisan za ovaj model sadržaja. Redosled čvorova potiče od Tompsonove konstrukcije NKA (videti 4.4.3). U donjem desnom uglu je naveden skup *PRVI* koji se pridružuje svakom čvoru konstruisanog automata a koji će biti definisan kasnije u ovoj istoj tački. Sa slike se vidi da iz čvora 2 vode tri sekvencije lukova koje sadrže samo jedan obeleženi luk: $\{2,1,2\}$, $\{2,7,5,6\}$ i $\{2,7,9,10\}$. U prve dve sekvencije obeleženi token odgovara tokenu a a samo prvi od njih ulazi u čvor 2 i njemu SGML pravila daju prioritet. Treba

uočiti da prema SGML pravilima ovaj model sadržaja nije dvosmislen ali token a? u ovom modelu sadržaja nikada neće biti zadovoljen.

Međutim, drugo tvrđenje u standardu nije precizno definisano i trebalo bi ge dopuniti trećim tvrđenjem:

3. ako više takvih sekvencija lukova može napustiti čvor NKA i sve i ulaze u njega, pravila SGML daju prioritet najkraćoj od njih.

Ovo se može ilustrovati na primeru modela sadržaja (a+ | b)+ iz 2.2.2. Na slici 4.2 je prikazan NKA koji odgovara ovom modelu sadržaja. Iz čvora 2 vode tri sekvencije lukova koje sadrže samo jedan obeleženi luk. To su sekvencije: $\{2,1,2\}$, $\{2,4,8,7,3,1,2\}$ i $\{2,4,8,7,5,6\}$. Prve dve od njih i ulaze u čvor 2 a obeleženi luk odgovara tokenu a. Pravila SGML-a daju piritet prvoj od njih.



Slika 4.2: NKA za model sadržaja (a+ | b)+

NKA se iz modela sadržaja konstruišu primenom poznatog Tompsonovog algoritma kojim se NKA konstruišu direktno iz regularnih izraza [7], [53]. Uobičajeni postupak se bez modifikacije može primeniti i na SGML modele sadržaja osim za **and** modele grupa. Opis konstrukcije NKA za taj slučaj biće dat u 4.4.2. Specifičnost automata konstruisanih Tompsonovim algoritmom je ta da se svaki čvor automata može predstaviti sledećom jednostavnom strukturom:

```
typedef struct nfa
{
   int luk;   /* Oznake za luk:                          */
      /*    identifikacioni broj elementa ili */
      /* -5 CDATA sadrzaj, obradjuje skaner   */
      /* -4 RCDATA sadrzaj, obradjuje skaner  */
      /* -3 EMPTY sadrzaj, obradjuje skaner   */
      /* -2 nema izlazni luk                  */
      /* -1 epsilon luk                       */
      /*  0 #PCDATA sadrzaj                   */
```

```
   struct nfa *sled;  /* Sledece stanje ili NULL ako ga nema)    */
   struct nfa *sled2; /* Drugo sledece stanje, samo ako je luk   */
      /* epsilon; NULL, ako se ne koristi        */
   SET    *s_prvi;    /* skup koji sadrzi id.br. elemenata koji  *
       * obezbedjuju dalji pomak kroz automat    */
   SET    *s_op;      /* skup koji sadrzi id.br.elemenata koji su*
       * obavezni za prolaz kroz opcioni put NKA */
   int    kakvo_st;   /* Specifikacija stanja:                   *
       * 1  pocetno stanje                       *
       * 2  zavrsno stanje                       *
       * 3  filter stanje                        *
       * -1 podautomat tokena and-sekvencije     *
       *    ima eps prolaz                       *
       * -2 podautomat tokena and-sekvencije     *
       *    nema eps prolaz                      */
} NFA;
```

Osim oznake luka i pokazivača najviše dva sledeća stanja, svaki čvor sadrži još tri podatka. Celobrojni podatak `kakvo_st` identifikuje specifična stanja kao što su početno i završno stanje automata i početno stanje podautomata za članove **and**-sekvencija. U ovom poslednjem slučaju beleži se takođe da li kroz podautomat postoji ε prolaz, to jest, da li se iz početnog u završno stanje podautomata člana **and**-sekvencije može stići korišćenjem ε lukova, odnosno, da li je član **and**-sekvencije opcioni (videti 4.4.2).

Skup *PRVI*, koji intuitivno odgovara skupu *FIRST* [7], definiše se na sledeći način: Neka je α sekvencija lukova koja izlazi iz čvora n nekog NKA, sadrži samo jedan luk koji nije obeležen sa ε i to je poslednji luk sekvencije. Neka je dalje A skup svih takvih sekvencija koje izlaze iz čvora n. Skup $Prvi(n)$ je onda skup svih ne-ε simbola koji se pojavljuju na lukovima skupa A. Sada se može definisati i skup *PRVI* koji pokazuje `s_prvi` iz strukture čvora NKA:

$$PRVI = \begin{cases} \emptyset, & \text{iz } n \text{ izlazi samo jedan luk i on je obeležen sa } \varepsilon \\ & \text{i } n \text{ je završno stanje nekog podautomata} \\ \subseteq Prvi(n), & \text{inače} \end{cases}$$

Skup $PRVI$ uz svaki čvor je samo podskup skupa *Prvi* kao posledica konstrukcije NKA i skupa $PRVI$ direktno iz modela sadržaja, bez naknadne obrade. Na primer, skup *Prvi* čvora 2 automata za model sadržaja `( a+, a?, b )` koji je predstavljen na slici 4.1 je $\{a, b\}$, dok je skup $PRVI = \{a\}$, kako je na slici i prikazano. U trenutku konstrukcije podautomata za token `b`, podautomat za model sadržaja `a+`, `a` je već konstruisan i naknadno se ne menja, izuzev, eventualno, početnog i završnog stanja. Kako se skup $PRVI$ uz stanje 2 koristi samo prilikom obilaska automata, nepotpunost skupa se prevazilazi algoritamski (videti 4.5.1). Sama konstrukcija NKA i formiranje ovog skupa biće detaljno opisani u 4.4.3.

Premda ovako nepotpuno definisan, skup *PRVI* ima višestruku ulogu:

- provera nedvosmislenosti modela sadržaja (4.4.3);
- deterministički obilazak NKA (4.5.1);
- implementacija podautomata za **and**-sekvencije (4.4.2).

Pokazivač s_op pokazuje skup koji sadrži one elemente čije je pojavljivanje obavezno i da bi se prolazeći automatom stiglo u njegovo završno stanje. Ovaj podatak se koristi radi implementacije izuzetka isključenja i koristi se u kombinaciji sa skupom Obav_prol iz strukture koja opisuje sadržaj elementa. Koristeći ove podatke utvrđuje se prilikom raščlanjivanja instancije dokumenta da li su primenljiva isključenja dozvoljena — isključenja koja menjaju status obaveznosti elementa, naime, nisu dozvoljena. Većini stanja automata se ovaj skup ne dodeljuje — on se pridružuje samo stanjima sa kojima otpočinju alternativi putevi kroz automat. O konstrukciji ovog skupa biće govora u 4.4.3.

Za predstavljanje automata svih elemenata koristi se jedan kontingentan memorijski prostor — veliki niz čiji je svaki elemenat struktura za opis stanja automata. Prema potrebi se stanju automata može pristupiti preko indeksa niza ili preko pokazivača. Imajući u vidu implementaciju MSGML-a pod operativnim sistemom MS-DOS, za opis jednog stanja automata potrebno je 20 bajtova. Kako pod MS-DOS-om kontingentan memorijski prostor ne može biti veći od jednog memorijskog segmenta od 64KB, ukupan broj stanja svih automata za jedan DTD ne sme biti veći od 3276. Upoređenja radi, NKA za DTD iz A ima 217 stanja.

Za predstavljanje skupa *PRVI*, kao i drugih skupova koje MSGML koristi, koristi se skup makroa i procedura za podršku radu sa skupovima iz [53]. Podatkovna struktura SET koja se koristi za reprezentaciju skupova pretpostavlja da su članovi skupa nenegativni celi brojevi te je stoga pogodna za predstavljanje skupa *PRVI* čiji članovi su elementi koji se mogu predstaviti svojim identifikacionim brojem. Ova struktura je dinamička — broj memorijskih reči koje se koriste za predstavljanje skupa zavisi od najvećeg broja u skupu. Kako se identifikacioni brojevi dodeljuju elementima redom, broj memorijskih reči koje se koriste za predstavljanje skupa *PRVI* je najviše $N \pmod{16}$ gde je N broj deklarisanih elemenata u DTD-u a 16 je broj članova skupa koji se mogu reprezentovati u jednoj reči. Kao što će biti prikazano u narednoj tački, za svaki čvor kome se pridružuje skup *PRVI* ne generiše se novi skup, već pokazivači uz više čvorova pokazuju isti skup.

4.4.2 Podautomat za and-sekvencije

Kao što je već rečeno, teorijski se **and**-sekvencije jednostavno svode na odgovarajući regularni izraz. Pri konstrukciji podautomata za **and**-sekvencije javlja se u praksi dosta problema. Pre svega svođenje **and**-sekvencije koja ima više od 2 člana na ekvivalentan regularni izraz proizvodi dvosmislenost. Na primer, grupa `( a & b & c )` se može svesti na sledeću ekvivalentnu grupu koja ne koristi **and** konektor:

```
(1)     ( ( a, b, c ) | ( a, c, b ) |
  ( b, a, c ) | ( b, c, a ) |
  ( c, a, b ) | ( c, b, a ) )
```

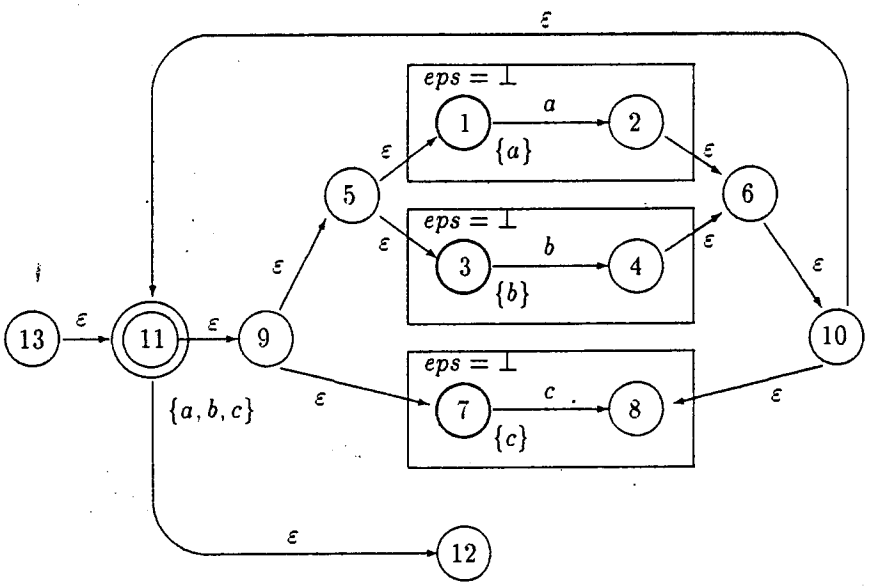
Ovaj model sadržaja je, međutim, dvosmislen jer se ne može utvrditi bez preduvida da li, na primer, token a zadovoljava grupu `( a, b, c )` ili grupu `( a, c, b )`. Dvosmislenost se daljim računom može izbeći na sledeći način:

```
(2)     ( ( a, ( ( b, c ) | ( c, b ) ) |
  ( b, ( ( a, c ) | ( c, a ) ) |
  ( c, ( ( a, b ) | ( b, a ) ) )
```

Ovakvo preformulisanje **and**-sekvencije ne samo što nije pogodno za konstruisanje automata u realnom vremenu kakvo obezbeđuje Tompsonov algoritam već ne eliminiše ni drugi problem

vezan za **and**-sekvencije a to je broj stanja automata. Naime, jedna **and**-sekvencija od n članova transformiše se u **or** grupu tipa (1) od $n!$ članova. Broj stanja NKA konstruisanog Tompsonovom metodom na osnovu grupe modela tipa (2) za **and**-sekvenciju od n članova može se predstaviti sledećom rekurentnom relacijom:

$$\begin{aligned} S_2 &= 8 \\ S_n &= 2 \cdot (n-1) + n \cdot (1 + S_{n-1}), \text{ za } n > 2 \end{aligned}$$



Slika 4.3: Podautomat sekvencije (a & b & c)

Vrednosti prvih članova ovog niza su $S_3 = 31, S_4 = 134, S_5 = 683, S_6 = 4144, S_7 = 29020$. Na primer, model sadržaja elementa <HEAD> iz HTML DTD-a (videti 3.3.2) definiše se **and**-sekvencijom od 7 članova. Ovako veliki broj stanja automata koji se može pojaviti u realnoj aplikaciji, uz činjenicu da sama konstrukcija automata zahteva dosta vremena, čini ovaj postupak svođenja grupe modela na ekvivalentan regularni izraz neadekvatnim.

U MSGML-u je stoga primenjen takav postupak koji **and**-sekvenciju prevodi u NKA čiji je broj stanja linearan u odnosu na broj članova **and**-sekvencije i koji se može konstruisati u realnom vremenu. Osim toga, konstruisanje i obilaženje podautomata ne zahteva dodatan memorijski prostor. Potrebno je samo dodatno obeležiti neka stanja podautomata za šta je dovoljan kratak ceo broj `kakvo_st`. Uz to se još koristi i skup *PRVI* koji je radi determinističkog obilaženja NKA već prisutan u strukturi koja opisuje stanje automata.

Podautomat za **and**-sekvenciju (a & b & c) je predstavljen na slici 4.3. On u velikoj meri podseća na podautomat za (a | b | c)+ osim što je pozicija završnog stanja promenjena. Sličnost je očigledna: tokeni a, b i c pojavljuju se u proizvoljnom redosledu. Razlika je u tome što se tokeni iz grupe (a & b & c) ponavljaju tačno tri puta a svaki od njih tačno jednom.

Konstruisani automat zadržava korisne osobine automata konstruisanih Tompsonovim algoritmom, kao što su:

- svi podautomati koji prepoznaju grupe tokena imaju jedno ulazno i jedno izlazno stanje;
- iz jednog stanja mogu izlaziti najviše dva luka;
- iz stanja mogu izlaziti samo sledeći lukovi: (a) jedan luk obeležen tokenom; (b) jedan luk obeležen ε; (c) dva luka, oba obeležena sa ε;
- automat konstruisan na osnovu jednog ili dva podautomata menja, eventualno, samo početno i završno stanje tih podautomata.

U konstrukciji i obilasku podautomata **and**-sekvencije prikazanog na slici 4.3 treba obezbediti sledeće:

- tokeni — članovi **and**-sekvencije ponavljaju se najviše onoliko puta koliki je broj članova **and**-sekvencije;
- svaki token iz sekvencije se pojavljuje najviše jednom, odnosno, tačno jednom ako ne postoji ε prolaz od početnog do završnog stanja podautomata tokena.

Oba uslova se obezbeđuju korišćenjem skupova *prvi* pridruženih uz svako stanje. Samo u slučaju specijalnog filter-stanja fs, koje je na slici 4.3 predstavljeno dvostrukim krugom, ovaj skup se u toku obilaska automata menja. Obilazak ovog automata odvija se na sledeći način:

1. Neka se automat nalazi u stanju fs, neka se u ulaznoj struji pojavljuje token x i neka je $x \in prvi(fs)$, tada automat iz stanja fs prelazi ε lukovima do početnog stanja ps podautomata člana **and**-sekvencije takvog da je $x \in prvi(ps)$. Stanja ps su na slici 4.3 predstavljena debljom linijom kruga. Tada postaje $prvi(fs) = prvi(fs) \backslash prvi(ps)$. Obilazak automata se nastavlja na uobičajen način.
2. Neka se automat nalazi u stanju fs, neka se u ulaznoj struji pojavljuje token x i neka $x \notin prvi(fs)$, tada automat iz stanja fs prelazi u završno stanje ako:
   (a) $prvi(fs) = \emptyset$, ili
   (b) za svako početno stanja ps podautomata člana **and**-sekvencije za koje važi da je $prvi(ps) \subseteq prvi(fs)$, postoji ε prolaz od početnog do završnog stanja podautomata.

   Po prelasku u završno stanje, skup $prvi(fs)$ dobija svoju početnu vrednost i to kao $\bigcup_{\forall ps} prvi(ps)$. Obilazak automata se nastavlja na uobičajeni način.

Kao što se vidi, obilazak podautomata **and**-sekvencije zahteva posebnu obradu koja se za svaki član te sekvencije sastoji od jedne operacije skupovnog oduzimanja, i eventualno, ako podautomat tog člana dozvoljava ε prolaz, jedne provere skupovne pripadnosti. Umesto $prvi(fs) = \bigcup_{\forall ps} prvi(ps)$, restauracija stare vrednosti skupa $prvi(fs)$ može se obaviti jednom operacijom skupovne dodele, o čemu će biti reči u 4.5.1.

Funkcionisanje ovog postupka obezbeđuje činjenica da za svaka dva početna stanja podautomata članova **and**-sekvencije, ps_1 i ps_2, važi da je $prvi(ps_1) \cap prvi(ps_2) = \emptyset$, a što potiče od nedvosmislenosti modela sadržaja.

4.4.3 Konstrukcija NKA iz modela sadržaja i provera nedvosmislenosti

Konstrukcija NKA prema opisanom modifikovanom Tompsonovom algoritmu postiže se definisanjem akcija uz gramatička pravila 216–232 iz dodatka C. Neterminalima iz ovih pravila, `model`, `grupa_tokena`, `sek_tokeni`, `ili_tokeni`, `i_tokeni` i `token_sadrzaja`, pridružuje se sledećih šest sintetizovanih atributa:

1. *poc* je početno stanje podautomata pridruženog neterminalu;
2. *krj* je završno stanje podautomata pridruženog neterminalu;
3. *prvi* je skup $PRVI(poc)$;
4. *eps* je logička vrednost koja dobija vrednost `TRUE` ako se iz *poc* u *krj* može stići ε-lukovima; u protivnom dobija vrednost `FALSE`;
5. *opc_kraj* je skup tokena x za koje važi sledeće: (a) iz čvora *poc* počinje sekvencija lukova koja vodi u stanje *krj* čiji je prvi obeleženi luk obeležen sa x, i (b) iz čvora *poc* se u *krj* može stići ε-lukovima;
6. *obav_prol* je skup tokena x koji se nalaze na svakom putu od *poc* do *krj*;

Osim što se računa atribut *prvi*, on se i dodeljuje određenim stanjima podautomata, i to: (a) početnom stanju, i (b) stanju koje obezbeđuje ponavljanje tokena. Ostalim stanjima se ne menja skup $PRVI$ pridružen prilikom konstrukcije podautomata, a završnom stanju se pridružuje prazan skup. Odatle sledi da u konstruisanom automatu, skup *Prvi* može biti nadskup njemu pridruženog skupa $PRVI$ kao što je već rečeno u pododeljku 4.4.1.

Slično važi i za atribut *obav_prol* koji se dodeljuje nekim stanjima podautomata. Kako se on koristi za proveru valjanosti isključenja (videti 4.5.2), ovaj skup se dodeljuje samo onim stanjima kojima otpočinje neki opcioni put kroz automat, bilo da se radi o **rep** ili **opt** konektoru ili o **or** sekvenciji, jer su samo tokeni iz opcionih puteva dozvoljeni za isključenje. Međutim, ako se jednom izabere taj opcioni put, tokeni obavezni za prolaz kroz njega se više ne smeju isključivati.

U ovoj tački će za svaki od neterminala biti prikazan konstruisani automat, dakle atributi *poc* i *krj*, kao i izračunavanje ostalih atributa i, na osnovu njih, provera nedvosmislenosti. U narednom prikazu koristiće se sledeće notacije:

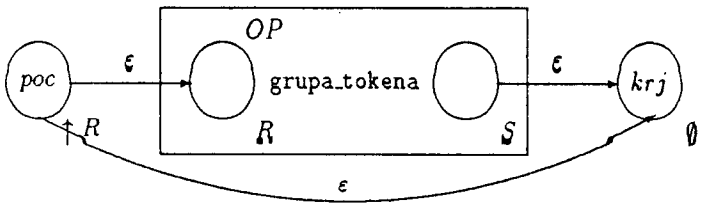
- $\mathcal{A}$(X) označava skup svih atributa pridruženih uz neterminal X. Prema tome, ako je X neterminal na levoj strani gramatičkog pravila, $\mathcal{A}$(X) = $\mathcal{A}$(Y) znači da su svi atributi neterminala X jednaki svim atributima neterminala Y;
- ako je X neterminal onda je X.*a* sintetizovani atribut *a* neterminala X;
- ako je X neterminal sa desne strane gramatičkog pravila, X.*n* neko stanje podautomata konstruisanog za taj neterminal, onda je X.*n.kakvo_st* specifikacija tog stanja.

Na slikama koje ilustruju konstruisanje automata koriste se sledeće konvencije:

- Dole desno uz čvor naznačen je skup $PRVI$ pridružen tom čvoru;
- Gore desno uz čvor naznačen je skup *obav_prol* tamo gde je taj skup čvoru pridružen;
- Ako je X skup pridružen čvoru, onda X znači da je taj skup formiran i pridružen čvoru; oznaka $\uparrow X$ znači da je čvoru pridružen pokazivač postojećeg skupa.

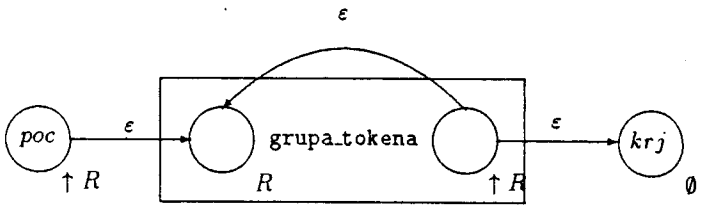
217 model : Grpo grupa_tokena Grpc Opt

$$\begin{aligned} \text{model}.eps &= \top \\ \text{model}.obav\_prol &= \emptyset \\ \text{model}.opc\_kraj &= \text{grupa\_tokena}.opc\_kraj \cup \\ &\quad \text{grupa\_tokena}.prvi \\ \text{model}.prvi &= \text{grupa\_tokena}.prvi \end{aligned}$$



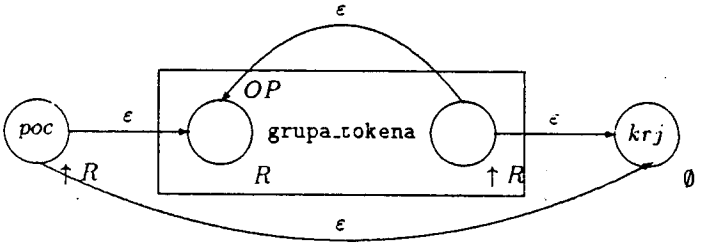
218 model : Grpo grupa_tokena Grpc Plus

$$\begin{aligned} \text{model}.eps &= \text{grupa\_tokena}.eps \\ \text{model}.obav\_prol &= \text{grupa\_tokena}.obav\_prol \\ \text{model}.opc\_kraj &= \text{grupa\_tokena}.opc\_kraj \\ \text{model}.prvi &= \text{grupa\_tokena}.prvi \end{aligned}$$



219 model : Grpo grupa_tokena Grpc Rep

$$\begin{aligned} \text{model}.eps &= \top \\ \text{model}.obav\_prol &= \emptyset \\ \text{model}.opc\_kraj &= \text{grupa\_tokena}.opc\_kraj \cup \\ &\quad \text{grupa\_tokena}.prvi \\ \text{model}.prvi &= \text{grupa\_tokena}.prvi \end{aligned}$$



220 model : Grpo grupa_tokena Grpc

$$\mathcal{A}(\texttt{model}) = \mathcal{A}(\texttt{grupa\_tokena})$$

221 grupa_tokena : sek_tokeni

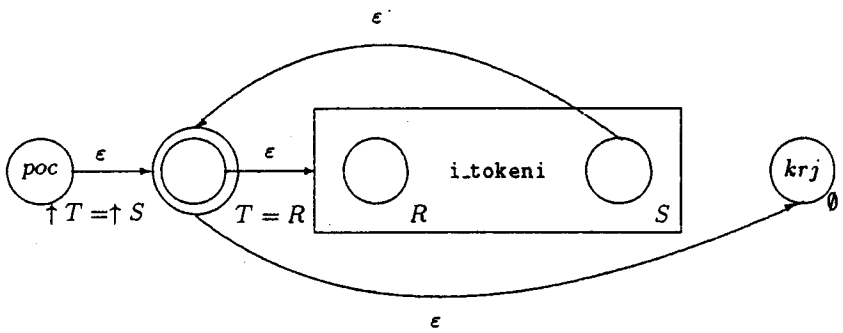
$$\mathcal{A}(\texttt{grupa\_tokena}) = \mathcal{A}(\texttt{sek\_tokeni})$$

222 grupa_tokena : ili_tokeni

$$\mathcal{A}(\texttt{grupa\_tokena}) = \mathcal{A}(\texttt{ili\_tokeni})$$

223 grupa_tokena : i_tokeni

$$\begin{aligned} \texttt{grupa\_tokena}.\mathit{eps} &= \texttt{i\_tokeni}.\mathit{eps} \\ \texttt{grupa\_tokena}.\mathit{obav\_prol} &= \texttt{i\_tokeni}\mathit{obav\_prol} \\ \texttt{grupa\_tokena}.\mathit{opc\_kraj} &= \texttt{i\_tokeni}.\mathit{opc\_kraj} \\ \texttt{grupa\_tokena}.\mathit{prvi} &= \texttt{i\_tokeni}.\mathit{prvi} \end{aligned}$$



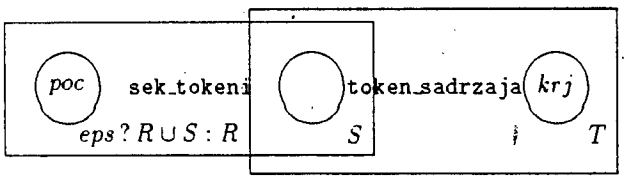
Kako se vidi na slici, skup *prvi* pridružen uz filter stanje je isti kao skup *prvi* pridružen uz početno stanje podautomata konstruisanog za neterminal i_token. Ipak, filter stanju se pridružuje skup a ne pokazivač odgovarajućeg skupa, kako se radi u sličnim slučajevima (videti pravila 217–219). Razlog je taj što se skup *prvi* uz filter stanje menja prilikom obilaska automata a ponovo se restaurira kad podautomat uđe u svoje završno stanje jednom operacijom skupovne dodele, umesto skupovne unije skupova *prvi* uz sva početna stanja podautomata članova and-sekvencije.

224 sek_tokeni : token_sadrzaja

$$\mathcal{A}(\texttt{sek\_tokeni}) = \mathcal{A}(\texttt{token\_sadrzaja})$$

225 sek_tokeni : sek_tokeni[1] Seq token_sadrzaja

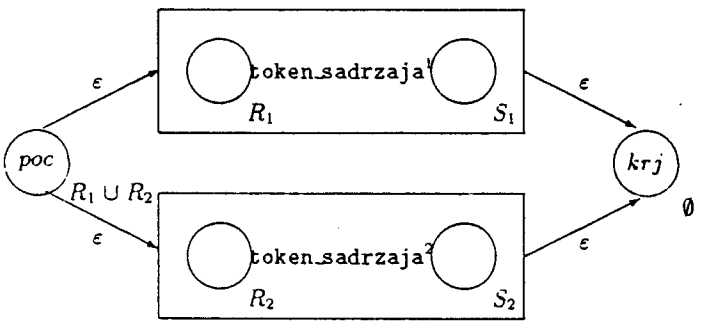
$$
\begin{aligned}
\texttt{sek\_tokeni}.eps &= \texttt{sek\_tokeni}^1.eps \wedge \texttt{token\_sadrzaja}.eps \\
\texttt{sek\_tokeni}.obav\_prol &= \texttt{sek\_tokeni}^1.obav\_prol \cup \\
&\quad \texttt{token\_sadrzaja}.obav\_prol \\
\texttt{sek\_tokeni}.opc\_kraj &= (\texttt{token\_sadrzaja}.eps\_prol) \\
&\quad ? \ \texttt{sek\_tokeni}^1.opc\_kraj \cup \\
&\quad \texttt{token\_sadrzaja}.opc\_kraj \\
&\quad : \ \texttt{token\_sadrzaja}.opc\_kraj \\
\texttt{sek\_tokeni}.prvi &= (\texttt{sek\_tokeni}^1.eps) \\
&\quad ? \ \texttt{sek\_tokeni}^1.prvi \cup \\
&\quad \texttt{token\_sadrzaja}.prvi \\
&\quad : \ \texttt{sek\_tokeni}^1.prvi
\end{aligned}
$$



$$\texttt{if}(\texttt{sek\_tokeni}^1.opc\_kraj \cap \texttt{token\_sadrzaja}.prvi \neq \emptyset)$$
"Model sadržaja je dvosmislen"

226 i_tokeni : token_sadrzaja[1] And token_sadrzaja[2]

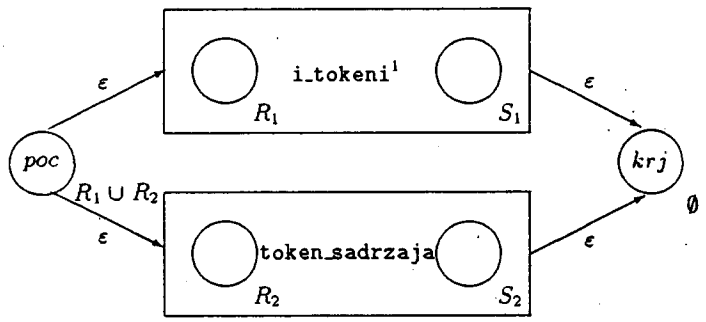
$$
\begin{aligned}
\texttt{i\_tokeni}.eps &= \texttt{token\_sadrzaja}^1.eps \wedge \texttt{token\_sadrzaja}^2.eps \\
\texttt{i\_tokeni}.obav\_prol &= \texttt{token\_sadrzaja}^1.obav\_prol \cup \\
&\quad \texttt{token\_sadrzaja}^2.obav\_prol \\
\texttt{i\_tokeni}.opc\_kraj &= \texttt{token\_sadrzaja}^1.opc\_kraj \cup \\
&\quad \texttt{token\_sadrzaja}^2.opc\_kraj \\
\texttt{i\_tokeni}.prvi &= \texttt{token\_sadrzaja}^1.prvi \cup \\
&\quad \texttt{token\_sadrzaja}^2.prvi \\
\texttt{token\_sadrzaja}^1.poc.kakvo\_st &= (\texttt{token\_sadrzaja}^1.eps) \\
&\quad ? \ \texttt{Ima\_eps} : \ \texttt{Nema\_eps} \\
\texttt{token\_sadrzaja}^2.poc.kakvo\_st &= (\texttt{token\_sadrzaja}^2.eps) \\
&\quad ? \ \texttt{Ima\_eps} : \ \texttt{Nema\_eps}
\end{aligned}
$$



$$\text{if}(\texttt{token\_sadrzaja}^1.prvi \cap \texttt{token\_sadrzaja}^2.prvi \neq \emptyset)$$
"Model sadržaja je dvosmislen"
$$\text{if}(\texttt{token\_sadrzaja}^1.prvi \cap \texttt{token\_sadrzaja}^2.opc\_kraj \neq \emptyset)$$
"Model sadržaja je dvosmislen"
$$\text{if}(\texttt{token\_sadrzaja}^1.opc\_kraj \cap \texttt{token\_sadrzaja}^2.prvi \neq \emptyset)$$
"Model sadržaja je dvosmislen"

227 i_tokeni : i_tokeni[1] And token_sadrzaja

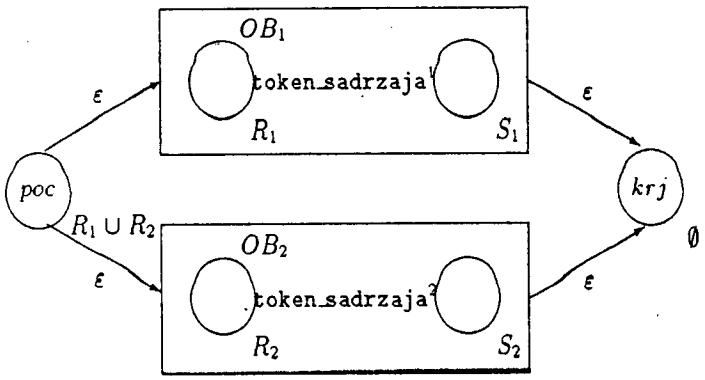
$$\begin{aligned}
\texttt{i_tokeni}.eps &= \texttt{i_tokeni}^1.eps \wedge \texttt{token_sadrzaja}.eps \\
\texttt{i_tokeni}.obav_prol &= \texttt{i_tokeni}^1.obav_prol \cup \\
&\quad \texttt{token_sadrzaja}.obav_prol \\
\texttt{i_tokeni}.opc_kraj &= \texttt{i_tokeni}^1.opc_kraj \cup \\
&\quad \texttt{token_sadrzaja}.opc_kraj \\
\texttt{i_tokeni}.prvi &= \texttt{i_tokeni}^1.prvi \cup \\
&\quad \texttt{token_sadrzaja}.prvi \\
\texttt{token_sadrzaja}.poc.kakvo_st &= (\texttt{token_sadrzaja}.eps) \\
&\quad ? \ \texttt{Ima_eps} \\
&\quad : \ \texttt{Nema_eps}
\end{aligned}$$



$$\text{if}(\texttt{i\_tokeni}^1.prvi \cap \texttt{token\_sadrzaja}.prvi \neq \emptyset)$$
"Model sadržaja je dvosmislen"
$$\text{if}(\texttt{i\_tokeni}^1.prvi \cap \texttt{token\_sadrzaja}.opc\_kraj \neq \emptyset)$$
"Model sadržaja je dvosmislen"
$$\text{if}(\texttt{i\_tokeni}^1.opc\_kraj \cap \texttt{token\_sadrzaja}.prvi \neq \emptyset)$$
"Model sadržaja je dvosmislen"

228 ili_tokeni : token_sadrzaja[1] Or token_sadrzaja[2]

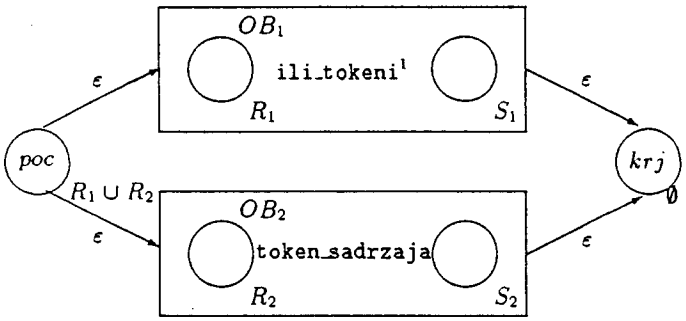
$$\begin{aligned}
\texttt{ili_tokeni}.eps &= \texttt{token_sadrzaja}^1.eps \vee \texttt{token_sadrzaja}^2.eps \\
\texttt{ili_tokeni}.obav_prol &= \texttt{token_sadrzaja}^1.obav_prol \cap \\
&\quad \texttt{token_sadrzaja}^2.obav_prol \\
\texttt{ili_tokeni}.opc_kraj &= \texttt{token_sadrzaja}^1.opc_kraj \cup \\
&\quad \texttt{token_sadrzaja}^2.opc_kraj \\
\texttt{ili_tokeni}.prvi &= \texttt{token_sadrzaja}^1.prvi \cup \\
&\quad \texttt{token_sadrzaja}^2.prvi
\end{aligned}$$



if(token_sadrzaja^1.*prvi* $\cap$ token_sadrzaja^2.*prvi* $\neq \emptyset$)
"Model sadržaja je dvosmislen"

229 ili_tokeni : ili_tokeni^1 Or token_sadrzaja

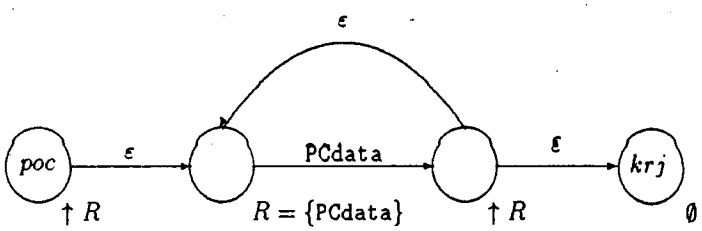
$$
\begin{aligned}
\text{ili\_tokeni}.\mathit{eps} &= \text{ili\_tokeni}^1.\mathit{eps} \vee \text{token\_sadrzaja}.\mathit{eps} \\
\text{ili\_tokeni}.\mathit{obav\_prol} &= \text{ili\_tokeni}^1.\mathit{obav\_prol} \cap \\
&\quad \text{token\_sadrzaja}.\mathit{obav\_prol} \\
\text{ili\_tokeni}.\mathit{opc\_kraj} &= \text{ili\_tokeni}^1.\mathit{opc\_kraj} \cup \\
&\quad \text{token\_sadrzaja}.\mathit{opc\_kraj} \\
\text{ili\_tokeni}.\mathit{prvi} &= \text{ili\_tokeni}^1.\mathit{prvi} \cup \\
&\quad \text{token\_sadrzaja}.\mathit{prvi}
\end{aligned}
$$



if(ili_tokeni^1.*prvi* $\cap$ token_sadrzaja.*prvi* $\neq \emptyset$)
"Model sadržaja je dvosmislen"

230 token_sadrzaja : Rni PCDATA

$$
\begin{aligned}
\text{token\_sadrzaja}.\mathit{eps} &= \bot \\
\text{token\_sadrzaja}.\mathit{obav\_prol} &= \text{PCdata} \\
\text{token\_sadrzaja}.\mathit{opc\_kraj}. &= \emptyset \\
\text{token\_sadrzaja}.\mathit{prvi} &= \text{PCdata}
\end{aligned}
$$

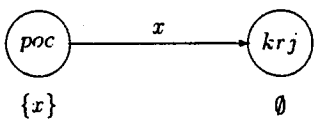


Definicija iz standarda precizira raščlanjive karakterske podatke kao sve podatkovne karaktere koji se pojavljuju između dve susedne etikete. Osim toga, za `#PCDATA` token sadržaja se smatra da ima indikator pojavljivanja **rep**. Na osnovu ovoga se može zaključiti, premda to nije eksplicitno rečeno, da reference entiteta koje se razrešavaju podatkovnim karakterima ulaze u `#PCDATA` token sadržaja. Isto tako, na primer, instrukcija obrade koja se pojavljuje između dve uzastopne etikete nije deo raščlanjivih karakterskih podataka (što se razlikuje od tumačenja u [150]), ali podatkovni karakteri pre instrukcije obrade kao i oni posle nje su deo istog `#PCDATA` token sadržaja. To je sasvim u skladu sa radom leksičkog analizatora koji u ovim slučajevima vraća deo po deo `#PCDATA` tokena sadržaja koji se zatim prepoznaje gornjim automatom.

231 `token_sadrzaja : Ime pojavljivanje`

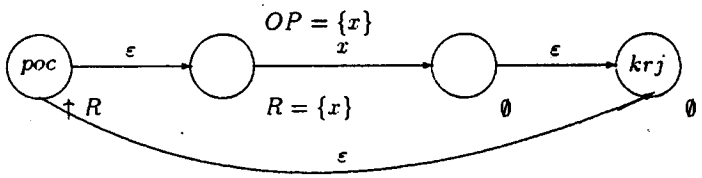
`pojavljivanje` = ε

$$\begin{aligned} \texttt{token\_sadrzaja}.\mathit{eps} &= \bot \\ \texttt{token\_sadrzaja}.\mathit{obav\_prol} &= \texttt{Ime} \\ \texttt{token\_sadrzaja}.\mathit{opc\_kraj} &= \emptyset \\ \texttt{token\_sadrzaja}.\mathit{prvi} &= \texttt{Ime} \end{aligned}$$



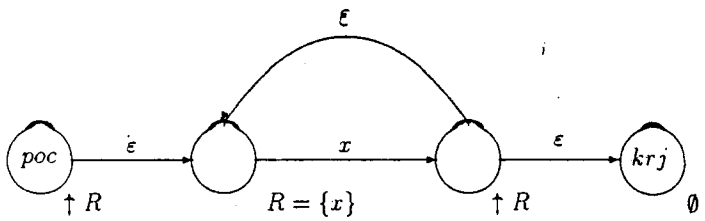
`pojavljivanje` = `Opt`

$$\begin{aligned} \texttt{token\_sadrzaja}.\mathit{eps} &= \top \\ \texttt{token\_sadrzaja}.\mathit{obav\_prol} &= \emptyset \\ \texttt{token\_sadrzaja}.\mathit{opc\_kraj} &= \texttt{Ime} \\ \texttt{token\_sadrzaja}.\mathit{prvi} &= \texttt{Ime} \end{aligned}$$



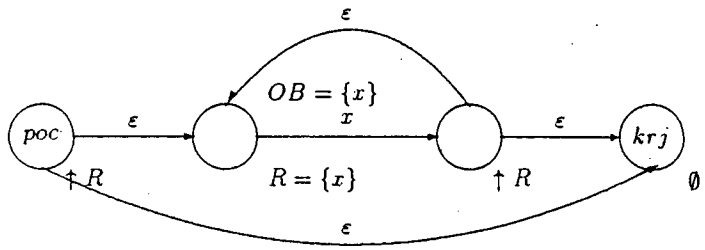
pojavljivanje = Plus

$$\begin{aligned} \texttt{token\_sadrzaja}.\mathit{eps} &= \bot \\ \texttt{token\_sadrzaja}.\mathit{obav\_prol} &= \texttt{Ime} \\ \texttt{token\_sadrzaja}.\mathit{opc\_kraj} &= \emptyset \\ \texttt{token\_sadrzaja}.\mathit{prvi} &= \texttt{Ime} \end{aligned}$$



pojavljivanje = Rep

$$\begin{aligned} \texttt{token\_sadrzaja}.\mathit{eps} &= \top \\ \texttt{token\_sadrzaja}.\mathit{obav\_prol} &= \emptyset \\ \texttt{token\_sadrzaja}.\mathit{opc\_kraj} &= \texttt{Ime} \\ \texttt{token\_sadrzaja}.\mathit{prvi} &= \texttt{Ime} \end{aligned}$$



232 token_sadrzaja : model

$$\mathcal{A}(\texttt{token\_sadrzaja}) = \mathcal{A}(\texttt{model})$$

Stanje u NKA koji je konstruisan Tompsonovim algoritmom ima dva izlazna luka samo ako su oba obeležena sa ε. Takav je slučaj i sa NKA koji se konstruišu postupkom koji je ovde opisan. Osim toga, ovakvim stanjima je skup $PRVI$ uvek pridružen. U slučaju ponovljivih tokena, ponavljanje obezbeđuje drugi izlazni ε luk, a skup $PRVI$ se upravo odnosi na ponovljivi token. Drugim rečima, samo stanju koje ima samo jedan ε izlazni luk ne mora biti pridružen skup $PRVI$.

4.5 Obilazak automata

Raščlanjavanje instancije dokumenta obavlja se na sličan način kao i raščlanjavanje SGML deklaracije i DTD-a, kroz saradnju leksičkog i sintaksičkog analizatora. Kao što je već rečeno, analiza sadržaja svakog elementa zavisi od toga kako je deklarisan. Za elemenat čiji je sadržaj

deklarisan modelom sadržaja to znači da sadržaj tog elementa mora prepoznavati njemu pridruženi NKA. Kako se u sadržaju tog elementa mogu pojavljivati drugi elementi, njihov sadržaj će prepoznavati njima pridruženi NKA. To praktično znači da se u toku obilaska jednog automata prelazi na obilazak drugog automata, pri čemu treba zapamtiti stanje starog automata da bi se moglo nastaviti sa njegovim obilaskom kada se stigne u završno stanje novog automata.

Prirodna struktura za čuvanje stanja automata je stog čija je maksimalna veličina određena vrednošću veličine TAGLVL. Kako se konstruisani NKA obilaze deterministički, što će biti opisano u 4.5.1, na stog uvek treba smestiti samo jedno stanje automata koje se može predstaviti indeksom u nizu u kome su smeštena sva stanja svih automata.

Obilazak automata odvija se u toku prepoznavanja početnih etiketa, raščlanjivih karakterskih podataka i krajnjih etiketa, i to na sledeći način:

- neka se NKA nalazi u stanju *tek_st* i neka je prepoznata početna etiketa koja odgovara tokenu *c*; obavljaju se koraci sledećeg algoritma (Alg. 4.1):

```
Ako postoji sledeće stanje s iz stanja tek_st za token c onda
  zapamti stanje s na stogu stanja automata
  tek_st postaje početno stanje NKA pridruženog tokenu c
inače
  GREŠKA: Nedozvoljen sadržaj za tekući element!
```

- neka se NKA nalazi u stanju *tek_st* i neka su prepoznati raščlanjivi karakterski podaci (tekući element ima mešoviti sadržaj ili sadržaj od elemenata); obavljaju se koraci sledećeg algoritma (Alg. 4.2):

```
Ako postoji sledeće stanje s iz stanja tek_st za token PCdata onda
  tekuće stanje postaje s
inače
  GREŠKA: Nedozvoljen sadržaj za tekući element!
```

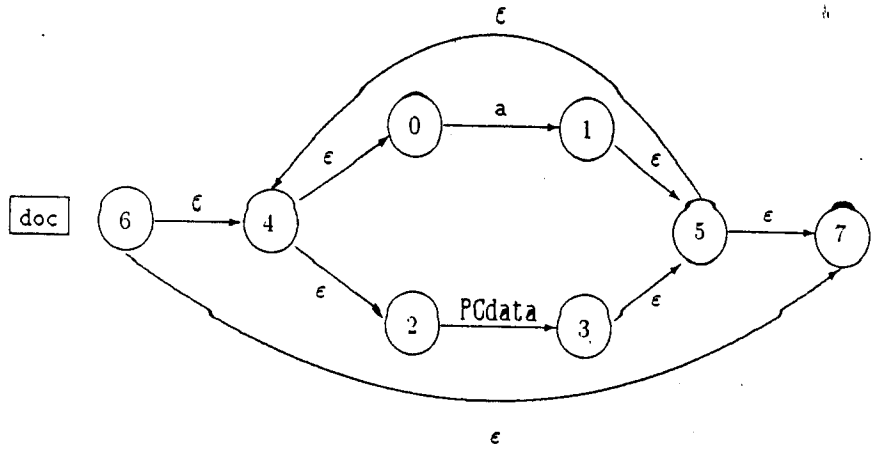
- neka se NKA nalazi u stanju *tek_st* i neka je prepoznata krajnja etiketa tekućeg elementa; obavljaju se koraci sledećeg algoritma (Alg. 4.3):

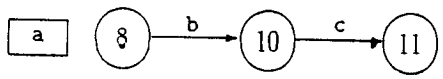
```
Sve dok tek_st nije završno stanje i postoji ε luk iz tek_st ponavljaj
  Ako je tek_st filter stanje onda
    Ako su prepoznati svi članovi and-sekvencije ili
    neprepoznati članovi imaju ε prolaz onda
      Ponovo se uspostavlja skup prvi filter stanja
      tek_st postaje završno stanje podautomata and-sekvencije
    inače
      GREŠKA: Nedostaje obavezni sadržaj za tekući element
  inače
    tek_st je sledeće stanje po prvom ε luku
Ako je tek_st završno stanje onda
  skini sa stoga stanje prethodnog automata koje postaje novo tek_st
inače
  GREŠKA: Nedostaje obavezni sadržaj za tekući element
```

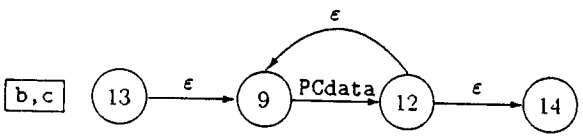
Ovaj postupak može se ilustrovati na sledećem primeru. Neka je DTD dokumenta deklarisan na sledeći način:

```
<!ELEMENT doc   - - ( a | #PCDATA )*          >
<!ELEMENT a     - - ( b , c )                 >
<!ELEMENT (b,c) - - ( #PCDATA )               >
```

Za ovaj dokument konstruišu se sledeći automati:







Prepoznavanje sledeće instancije dokumenta odvija se na sledeći način:

	tekuće stanje	stog	novo stanje
<doc>	6	6	
Neki tekst	3	6	3
<a>	1	1, 6	8
	10	10, 1, 6	13
Deo b	12	10, 1, 6	12
	14=zav	1, 6	10
<c>	11	11, 1, 6	13
Deo c	12	11, 1, 6	13
</c>	14=zav	1, 6	11
	11=zav	6	1
Kraj dokumenta	3	6	3
</doc>	7=zav	∅	

4.5.1 Funkcija prelaska

Pretvaranje nedeterminističkog u deterministički konačni automat nije neophodno već se nedeterministički konačni automat može direktno obilaziti. Uobičajena strategija koristi činjenicu da ako se NKA nalazi u stanju s on se takođe nalazi i u svim stanjima do kojih se može doći prelazeći ε lukove iz stanja s, odnosno, nalazi se u svim stanjima koja se nalaze u skupu $S = \varepsilon$-zatvorenje(s). Ako se u ulaznoj struji nalazi token t, onda automat NKA iz skupa stanja S prelazi u skup stanja M koji sadrži sva takva stanja m u koja se može stići lukom obeleženim sa t iz nekog stanja $s \in S$.

NKA konstruisani za SGML modele sadržaja mogu se obilaziti deterministički, odnosno tako da se za svako stanje s i token t može nedvosmisleno odrediti sledeće stanje. Naime, na osnovu definicije nedvosmislenih modela sadržaja (videti 2.2.4 i 4.4.3) sledi da skup M može za svaki ulazni token sadržati najviše jedan element (tvrđenje 1), a to znači da se prilikom obilaska NKA koristeći skup $PRVI$ pridružen uz njegova stanja uvek može za svako stanje s i ulazni token t izabrati sledeće stanje. Izbor puta koji ulazi u stanje s kao i kraćeg puta (tvrđenja 2 i 3) obezbeđuje se algoritamski, kao što će biti prikazano u ovom odeljku.

Na osnovu definicije skupa $PRVI$ iz 4.4.1 i njegove konstrukcije opisane u 4.4.3 sledi da za svako stanje s automata i ulazni token t, za vrednost skupa $PRVI$ postoje tri mogućnosti:

- $PRVI = \emptyset$;
- $t \notin PRVI$;
- $t \in PRVI$.

U prva dva slučaja, prvim ε lukovima treba prelaziti u naredna stanja, sve dok se ne dođe do stanja u kome je $t \in PRVI$. Ovi koraci su opisani u narednom algoritmu (Alg. 4.4):

```
Pomeranje po ε lukovima do onog stanja čiji skup PRVI sadrži token c
Sve dok iz stanja tek_st vodi ε luk i
skup PRVI uz tek_st je prazan ili
token t ∉ PRVI, ponavljaj
  Ako tek_st nije filter stanje onda
    tek_st postaje novo stanje po prvom ε luku
  inače
    Ako je moguć izlazak iz filtera
      tek_st postaje novo stanje po drugom ε luku
      Ponovo se uspostavlja skup PRVI filter stanja
    inače
      GREŠKA: Nema pomaka iz stanja tek_st sa tokenom c
```

Jednom kada se stigne u stanje u kome je $t \in PRVI$, obilazak se nastavlja ε lukovima sve dok se ne stigne u stanje u kome je luk obeležen tokenom t. U naredno stanje se prelazi prvim ili drugim ε lukom u zavisnosti od toga kom skupu $PRVI$ pripada token t. Ova dva skupa, zbog nedvosmislenosti modela, su u najvećem broju slučajeva disjunktna. U slučajevima kada to ne važi (videti 4.4.1), bira se drugi ε luk jer on obezbeđuje ponavljanje tokena čemu SGML daje prednost. Ovi koraci su opisani u narednom algoritmu (Alg. 4.5):

Pomeranje po ε lukovima sve dok je token t element skupa PRVI
Sve dok iz stanja tek_st vodi ε luk i token $t \in PRVI$, ponavljaj
  Ako je *tek_st* filter stanje onda
    Ako je moguć prolazak kroz filter stanje onda
      tek_st postaje novo stanje po prvom ε luku
    inače
      GREŠKA: Nema pomaka iz stanja *tek_st* sa tokenom c
  inače
    Ako je *tek_st* početno stanje člana **and**-sekvencije onda
      Modifikuj skup $PRVI$ filtera oduzimanjem mogućnosti od kojih će samo jedna biti upotrebljena
    Ako je t element skupa $PRVI$ uz stanje u koje se stiže drugim ε lukom
      tek_st postaje novo stanje po drugom ε luku
    inače
      tek_st postaje novo stanje po prvom ε luku

Kada se stigne u stanje iz koga vodi luk obeležen tokenom t, prelazi se u to stanje, i to je stanje u kome se čeka na naredni ulazni token. Ovi koraci su opisani sledećim algoritmom (Alg. 4.6):

Iz tekućeg stanja vodi luk obeležen tokenom
Ako je *tek_st* početno stanje člana **and**-sekvencije onda
  Modifikuj skup $PRVI$ filtera oduzimanjem mogućnosti od kojih će samo jedna biti upotrebljena
Ako je luk iz stanja *tek_st* obeležen tokenom t
  tek_st postaje novo stanje po tom luku
inače
  GREŠKA: Nema pomaka iz stanja *tek_st* sa tokenom c

Prilikom obilaska automata posebnu obradu zahtevaju filter stanja. Ako je f filter stanje a z početno stanje nekog tokena odgovarajuće **and**-sekvencije, onda su moguće sledeće obrade nad filter stanjem:

- *prolazak kroz filter stanje.* Obavljaju se sledeći koraci (Alg. 4.7):

  Ako je token $t \in PRVI(f)$ onda
    tek_st postaje sledeće stanje po prvom ε luku
  inače
    Ako je moguć izlazak iz filter stanja onda
      tek_st postaje sledeće stanje po drugom ε luku

- *modifikacija filter stanja.* Obavljaju se sledeći koraci (Alg. 4.8):

  Ako je z početno stanje člana **and**-sekvencije takvo da je $t \in PRVI(z)$ onda
    $PRVI(f)$ dobija vrednost $PRVI(f) \backslash PRVI(z)$

- *restauracija filter stanja.* Obavljaju se sledeći koraci (Alg. 4.9):

  Neka je s sledeće stanje po prvom ε luku stanja f
  $PRVI(f)$ dobija vrednost $PRVI(s)$

- *da li je moguć izlazak iz filter stanja.* Obavljaju se koraci opisani sledećim algoritmom (Alg. 4.10):

```
f postaje tekuće stanje tek_st
Ako je skup PRVI uz stanje f prazan onda
  izlazak iz filter stanje je moguć
inače
  Za svaki token t ∈ PRVI ponavljaj
    Sve dok iz stanja tek_st vodi ε luk i
    tek_st nije početno stanje člana and-sekvencije ponavljaj
      Ako je t element skupa PRVI uz stanje u koje se stiže drugim ε lukom
        tek_st postaje novo stanje po drugom ε luku
      inače
        tek_st postaje novo stanje po prvom ε luku
      Ako ne postoji ε prolaz kroz token and-sekvencije
        izlazak iz filter stanja nije moguć
  izlazak iz filter stanja je moguć
```

4.5.2 Realizacija izuzetaka

Kao što je rečeno u 2.2.2, izuzeci su po svojoj prirodi dinamički što znači da se ne mogu ni obraditi niti se može proveriti njihova primenljivost u fazi konstrukcije automata, odnosno u fazi analize DTD-a. Koji se izuzeci na neki element mogu primeniti zavisi ne samo od deklaracije tog elementa već i od pojavljivanja tog elementa u instanciji dokumenta, odnosno od toga u sadržaju kojih se sve elemenata on nalazi.

Realizacija uključivanja i isključivanja, premda se oni uvek pominju zajedno, pod nazivom izuzeci, potpuno je različita. U standardu je formalno opisano kako primenljivo uključenje modifikuje model sadržaja elementa. Taj formalni opis nije, međutim, od praktičnog značaja za realizaciju raščlanjivača instancije dokumenta jer bi zahtevao konstrukciju NKA za svako pojavljivanje elementa u instanciji dokumenta što bi bilo neprihvatljivo neefikasno. S druge strane, s obzirom da je za svaki element iz DTD konstruisan NKA koji prepoznaje njegov model sadržaja, implementacija uključenja je izuzetno jednostavna. Potrebno je samo modifikovati algoritam 4.1 na sledeći način: pre nego što se objavi greška zbog pojave nedozvoljenog sadržaja za tekući element, treba proveriti da li je tekući token primenljivo uključenje (Alg. 4.1′):

```
..... (Alg. 4.1)
inače Ako je c u skupu primenljivih uključenja onda
  zapamti stanje tek_st na stogu stanja automata
  tek_st postaje početno stanje NKA pridruženog tokenu c
inače
  GREŠKA: Nedozvoljen sadržaj za tekući element!
```

Implementacija isključenja je potpuno drukčija i nešto složenija. Ona se svodi na proveru uslova postavljenog u standardu da isključenja koja su primenljiva na neko pojavljivanje elementa u instanciji dokumenta ni na koji način ne smeju menjati status obaveznosti tokena iz njegovog modela. Da bi se proverio taj uslov koristi se skup *obav_prol* koji se pridružuje svakom elementu iz DTD i nekim stanjima NKA koji je konstruisan za njegov model sadržaja. Izračunavanje ovog skupa je opisano u 4.4.3. Skup *obav_prol* pridružen uz element sadrži sve

one tokene koji se bar jednom moraju pojaviti u sadržaju svakog pojavljivanja tog elementa. Skup *obav_prol* pridružen uz stanje NKA sadrži sve tokene čije je pojavljivanje neophodno za prolazak do prvog stanja za koje je $PRVI = \emptyset$, tj. do završnog stanja nekog podautomata.

U MSGML-u provera ovog uslova odvija se na dva nivoa čiji je cilj utvrđivanje eventualne neprimenljivosti nekog isključenja što je pre moguće:

1. Prva provera je formalna i odvija se u fazi raščlanjavanja DTD-a. Za svaki element a i svaki token c u njegovoj listi isključenja proverava se ispunjenost uslova $c \notin obav\_prol$, gde je *obav_prol* skup pridružen uz element a;

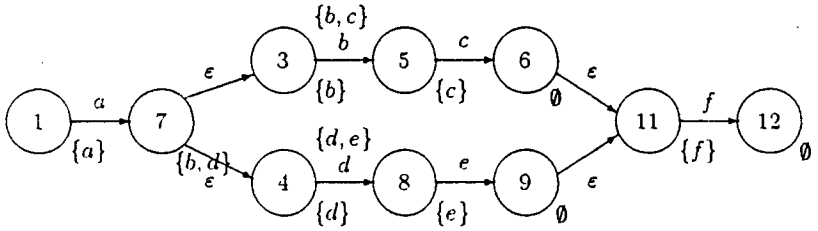
2. Druga provera se odvija u fazi raščlanjavanja instancije dokumenta. Neka se NKA nalazi u stanju `tek_st` i neka je prepoznata početna etiketa elementa c i neka je *isklj* skup isključenja svih otvorenih elemenata, do elementa c, ne uključujući ga. Tada se pre izvršavanja algortima 4.1′ obavljaju koraci sledećeg algoritma (Alg. 4.11):

Neka je OP skup *obav_prol* uz sledeće stanje od *tek_st* po prvom ili drugom ε luku koji se bira u zavisnosti od c
Ako je $isklj \cap OP \neq \emptyset$ onda
  GREŠKA: Isključenja se ne mogu primeniti
inače Ako je $isklj \cap c.obav\_prol \neq \emptyset$ onda
  GREŠKA: Isključenja se ne mogu primeniti
inače /* Sva isključenja su primenljiva; počinje prolaz kroz NKA (Alg. 4.1′) */

Određivanje skupa OP objašnjava se činjenicom da se u raščlanjivaču MSGML, radi uštede memorijskog prostora, skup *obav_prol* dodeljuje samo početnom stanju nekog opcionog puta, kako je opisano u 4.4.1. Time se utvrđuje nevaljanost nekih isključenja koja su u principu dozvoljena za model sadržaja kao celinu. To se može ilustrovati na sledećem primeru. Neka je element.doc deklarisan na sledeći način:

```
<!ELEMENT tekst          - - ( doc | #PCDATA )+          -( c ) >
<!ELEMENT doc            - - ( a , ((b , c) | (d , e)), f )      >
<!ELEMENT (a, b, c, d, e, f) - - ( #PCDATA )                     >
```

Skup *obav_prol* uz element doc je {a, f} jer se ovi elementi obavezno moraju pojaviti u sadržaju elementa doc, dok se grupe (b , c) i (d , e) alternativno pojavljuju. Automat konstruisan za model sadržaja elementa doc prikazan je na sledećoj slici:



Ispitivanje isključenja sledeće instancije dokumenta odvija se u sledećim koracima:

	važeća isključenja	obavezni prolaz uz element	obavezni prolaz uz stanje	presek
`<tekst>Nešto`	$\{c\}$	$\emptyset$	$\{doc\}$	$\emptyset$
`<doc>`	$\{c\}$	$\{a, f\}$	$\emptyset$	$\emptyset$
`<a>deo A</a>`	$\{c\}$	$\emptyset$	$\emptyset$	$\emptyset$
`<b>deo B</b>`	$\{c\}$	$\emptyset$	$\{b, c\}$	$\{c\}$
				Stop! Greška

Redosled ispitivanja uslova primenljivosti tokena c je uslovljen SGML redosledom prioriteta koji je dat u 2.2.2: najvišeg prioriteta su tokeni koji se mogu ponavljati, zatim slede ostali tokeni iz modela sadržaja eventualno modifikovani isključenjima a najnižeg prioriteta su uključenja. Token koji je istovremeno važeće uključenje i važeće isključenje primenjuje se kao isključenje.

Ostaje pitanje izračunavanja skupa *isklj*. Jedna mogućnost je izračunavanje tog skupa za svaki novi token uključivanjem u njega isključenja iz svih otvorenih elemenata što je neefikasno ali ne i neostvarivo jer je broj otvorenih elemenata ograničen malim brojem TAGLVL. Druga mogućnost, koja izgleda vrlo prihvatljiva, je računanje novog *isklj* na osnovu tekućeg skupovnim operacijama unija i razlika. Kako više elemenata može imati ista isključenja, skupovna operacija razlika nije adekvatna. U mSGML se, stoga, skupovi *isklj* čuvaju na stogu: svaka početna etiketa puni stog (posle algoritma 4.11) a svaka krajnja etiketa skida skup sa stoga.

4.5.3 Izostavljanje etiketa

Od tehnika minimizacije u mSGML-u je primenjena tehnika izostavljanja etiketa. Za izostavljanje krajnje etikete potrebno je, pre svega, proveriti svaku krajnju etiketu iz ulazne struje, da li odgovara tekućem elementu, pa ako ne odgovara treba pokušati zatvoriti sve elemente koji su ugnježdeni u onaj čija se krajnja etiketa pojavila. Taj se postupak može opisati sledećim algoritmom (Alg. 4.12). Neka je u ulaznoj struji krajnja etiketa elementa k i neka je tekući element a.

```
Sve dok je k ≠ a ponavljaj
  Ako nije dozvoljeno izostavljanje krajnje etikete elementa a onda
    GREŠKA: Nedostaje obavezna krajnja etiketa
  inače
    Ako stog aktivnih elemenata nije prazan onda
      Skini element a sa stoga aktivnih elemenata
      /* Provera da li je automat elementa a u završnom stanju */
      ..... (Alg. 4.3) .....
      /* Jeste. Sada je aktivno stanje automata obuhvatnog elementa */
      Element sa vrha stoga aktivnih elemenata postaje novi tekući element a
    inače
      GREŠKA: Element k nije otvoren
..... (Alg. 4.3) .....
```

Za implementaciju izostavljanja krajnje etikete ovo nije dovoljno jer se krajnja etiketa nekog elementa može izostaviti i ako iza njega sledi neki element ili SGML raščlanjivi podaci koji su na tom mestu nedozvoljeni u njegovom sadržaju. Prema tome potrebno je modifikovati algoritam 4.1′ uvođenjem još jednog uslova u slučaju kada nema pomaka iz stanja *tek_st* sa tokenom *c*. Taj uslov se proverava pre provere uključenja. Modifikacijom se dobija sledeći algoritam 4.1″:

Da li je element c isključen?
..... (Alg. 4.11) .....
Ima li prolaza kroz automat sa tokenom c?
..... (Alg. 4.1) .....
inače /* *Da li je izostavljena neka početna etiketa?* */
  Ako je pomeranje iz *tek_st* moguće samo sa tokenom *d* i
  ako je dozvoljeno izostavljanje početne etikete elementa *d* onda
    zapamti stanje *tek_st* na stogu stanja automata
    zapamti element *d* na stogu aktivnih elemenata
    tek_st postaje početno stanje NKA pridruženog tokenu *d*
    d postaje nova vrednost od *c*
    ..... (Rekurzivno izvršavanje alg. 4.1″) .....
  inače /* *Da li je izostavljena krajnja etiketa od c?* */
    ..... (Alg.4.3) .....
    ..... (Rekurzivno izvršavanje alg. 4.1″) .....
inače /* *Da li je element c uključenje?* */
  ..... (Alg. 4.1′)

Ukoliko nema prolaza kroz automat sa tokenom *c*, automat se već nalazi u stanju kome pridruženi skup $PRVI$ nije prazan ali ne sadrži token *c* (videti 4.4 i 4.5). Za utvrđivanje da je u stanju *tek_st*, odnosno na određenom mestu u modelu sadržaja, obevezno pojavljivanje određenog elementa, potrebno je samo utvrditi da skup $PRVI$ sadrži samo jedan element i to je onda element čija je početna etiketa izostavljena.

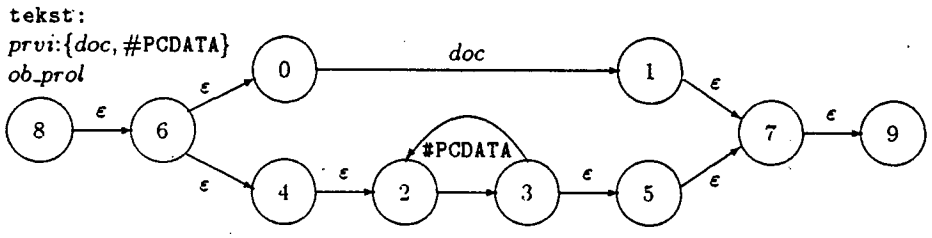
Kao što je rečeno u tački o izostavljanju etiketa u 2.2.3, na osnovu pojavljivanja elementa koji je u određenom kontekstu nedozvoljen jer je isključen zaključuje se da su prisutne krajnje etikete svih elemenata koji obuhvataju tekući a u kojima je taj element isključen. To znači da je potrebno kada se u ulaznoj struji pojavi početna etiketa nekog elementa, pre provere primenljivosti isključenja, mogućnosti prolaza kroz automat i primenljivosti uključenja (algoritam 4.1″), primeniti sledeći postupak (Alg. 4.13):

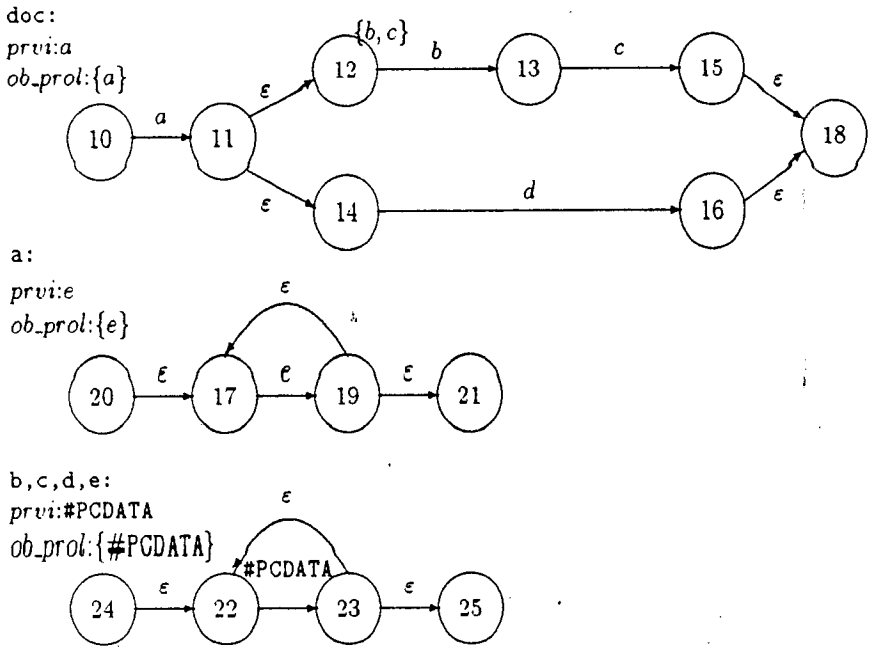
```
Sve dok je c ∈ isklj ponavljaj
  Ako nije dozvoljeno izostavljanje krajnje etikete tekućeg elementa a onda
    GREŠKA: Pojava isključenog elementa ili nedostaje obavezna krajnja etiketa
  inače
    Ako stog aktivnih elemenata nije prazan onda
      Skini element a sa stoga aktivnih elemenata
      /* Provera da li je automat elementa a u završnom stanju */
         ..... (Alg. 4.3) .....
      /* Jeste. Sada je aktivno stanje automata obuhvatnog elementa */
      Element sa vrha stoga aktivnih elemenata postaje novi tekući element a
      Skini sa stoga skup isklj
    inače
      GREŠKA: Nedozvoljen element
/* Provera da li se isključenje može primeniti i
   prolaz kroz algoritam sa tokenom c */
   ..... (Alg. 4.1'') .....
```

Interakcija izostavljanja etiketa i isključivanja može se razmotriti na sledećem primeru. Neka je deklarisan sledeći DTD:

```
<!ELEMENT tekst          - - ( doc | #PCDATA )+      -( c )+( b ) >
<!ELEMENT doc            - - ( a , ((b , c) | d ) )               >
<!ELEMENT a              0 0 ( e+ )                  -( b )       >
<!ELEMENT (b, c, d, e)   0 0 ( #PCDATA )                          >
```

NKA konstruisani za elemente **tekst**, **doc**, **a**, **b**, **c**, **d** i **e** kao i skupovi *prvi* i *obav_prol* pridruženi elementima i stanjima automata prikazani su na sledećoj slici:





Raščlanjavanje jedne instancije dokumenta odvija se u sledećim koracima:

	važeća isključenja	izostavljena etiketa	obavezni prolaz uz stanje ili element	primenjen algoritam
`<tekst>`	c		$\emptyset$	4.1
`Neki tekst`	c		$\emptyset$	4.2
`<doc>`	c,b		$\{a\}$	4.1
`<e>`	c,b	`<a>`	$\{e\}$	4.1″ i 4.1
`deo E, 1`	c,b		{#PCDATA}	4.2
`<e>`	c,b	`</e>`	$\{e\}$	4.1″ i 4.1
`deo E, 1`	c,b		{#PCDATA}	4.2
`<b>`	c	`</e>,</a>`	$\{b, c\}$	4.13 i 4.11
				Stop! c je isključen, ni b se ne može primeniti

Glava 5

Elektronski rečnik srpskog jezika

Termin *elektronski rečnik* koristi se u dva značenja: u prvom značenju to je svaki rečnik predstavljen u elektronskom obliku i namenjen nekoj informatičkoj obradi dok u drugom značenju on određuje sistem elektronskih rečnika DELA kako ga je definisao Moris Gros u [47] i [48]. Tako se, na primer, pod elektronskim rečnikom u prvom značenju ponekad smatra i sistem namenjen otkrivanju tipografskih grešaka u tekstu koji se sastoji samo od liste reči organizovane na način koji omogućava da se za svaku reč u tekstu proveri da li je „ispravna", odnosno, da li pripada rečniku. U ovom radu će se termin elektronski rečnik koristiti u drugom značenju.

Elektronski rečnik se suštinski razlikuje od tradicionalnih rečnika. Premda se rečnici namenjeni čoveku, s obzirom na danas uobičajen postupak uređivanja i slaganja uz pomoć računara, mogu naći u mašinski čitljivom obliku, taj oblik nije nimalo bliži formi elektronskog rečnika od njegovog uobičajenog, papirnog oblika. U njemu su informacije predstavljene na način koji odgovara čoveku — čitaocu, a taj oblik je najčešće nepogodan za računarsku obradu.

Iako tradicionalni rečnici, uz ostale izvore informacija, kao što su gramatike i korpusi, predstavljaju temelj za izradu elektronskog rečnika, automatska transformacija tradicionalnog u elektronski rečnik nije moguća. Mnoge informacije u tradicionalnom rečniku nisu eksplicitno predstavljene već se u tumačenju teksta koji prati rečničku odrednicu pretpostavlja određeno znanje čitaoca o jeziku. Tako se, na primer, u rečnicima srpskog jezika uz imenice muškog roda na konsonant (I vrsta promene) ne navodi informacija o animatnosti koja je od značaja za utvrđivanje oblika akuzativa.

Informacije u elektronskom rečniku moraju biti potpune, eksplicitne i kodirane na odgovarajući način, o čemu tradicionalna leksikografija obično ne vodi računa. Tako je, na primer, u [116] gramatička informacija uz glagol obično predstavljena na sledeći način:

stàjati, stòjīm nesvrš.

Pretpostavlja se, naime, da čitalac na osnovu oblika infinitiva i prvog lica jednine prezenta glagola može da zaključi kakvi su i svi drugi oblici tog glagola. Međutim, sa stanovišta izrade elektronskog rečnika ove informacije nisu eksplicitne jer su navedena samo dva od mogućih tridesetak prostih oblika. U [117], je, na primer, kao karakterističan naveden oblik drugog lica jednine imperativa *grej* glagola *grejati* dok to nije učinjeno za glagole *zagrejati* i *nagrejati*. U istom rečniku, oblici glagola *viđeti* koji je označen kao dijalekatizam ijekavskog izgovora, dati su uz odrednicu *videti* gde je kao oblik drugog lica jednine imperativa naveden oblik *vidi*,

dok je oblik *vidi* označen kao pokrajinski.[1]

Ovi nasumično izabrani primeri pokazuju da tradicionalan prilaz predstavljanju gramatičkih informacija nije podesan u informatičkoj obradi. Za nju je mnogo prihvatljivija neka vrsta numeričkog kôda koji bi potpuno i jednoznačno određivao konjugaciju svakog pojedinačnog glagola (videti 5.3.1).

Tradicionalni rečnici nisu dovoljno precizni ni sa stanovišta fonda reči koje obuhvataju. Tako, na primer, u [116] odrednica **vrijed-**, ek. vred- čitaocu govori da potraži objašnjenje svih reči koje počinju niskom *vrijed-* uz njihove ekavske varijante koje počinju niskom *vred-*. Sa stanovišta rečničkog fonda to znači da u jeziku postoje uz 17 reči koje počinju niskom *vred-* još 17 reči koje počinju niskom *vrijed-* i koje predstavljaju njihovu ijekavsku varijantu. Međutim, uz 7 odrednica koje počinju niskom *vred-* takva mogućnost nije potvrđena i u nekim slučajevima i ne postoji (na primer, *vrednota*, *vrednoća*). Iscrpna analiza odnosa informacija u tradicionalnom i elektronskom rečniku data je u [144].

Kako se elektronski rečnik u značenju sistema rečnika DELA gradi na osnovu tradicionalnih rečnika, ovakve neeksplicirane informacije u njima moraju na neki način biti dopunjene. Metodologija izrade elektronskih rečnika razvijena je u LADL/CERIL i opisana u [48], [24] i [144]. Pod sistemom elektronskih rečnika podrazumeva se više rečnika između kojih su uspostavljene odgovarajuće veze. On se sastoji od rečnika prostih reči DELAS, rečnika takozvanih složenih reči DELAC i fonoloških rečnika DELAP. U ovom tekstu biće govora samo o rečnicima prostih reči DELAS, njihovoj izradi za srpski jezik i njihovom korišćenju u obradi elektronskih tekstova.

5.1 Elektronski rečnik prostih reči

Svi rečnici sistema DELA mogu se predstaviti u obliku uređene liste objekata, pri čemu se ti objekti mogu formalno predstaviti u obliku uređenog para (M, I) gde je M formalna reč[2] koja predstavlja ulaz u rečnik a I informacija pridružena formalnoj reči M. Formalne reči, kao i njima pridružene informacije, zavise od vrste rečnika u sistemu DELA. Ono što je za sve ove podrečnike zajedničko to je da ni u jednom od njih ne postoje dva ista uređena para. Naime, ako u rečniku postoje dva para (M_1, I_1) i (M_2, I_2) i važi $M_1 = M_2$ onda obavezno važi $I_1 \neq I_2$.

Ulaz u prvu listu koja se naziva DELAS čine one formalne reči koje odgovaraju formi rečničke odrednice. Informacija koja prati svaku formalnu reč sadrži kôd morfološke klase koji određuje njena paradigmatska svojstva. Taj kôd mora biti takav da se za datu formalnu reč može odrediti tačan skup svih njenih oblika. Na primer, za francuski jezik morfološki kôd imenica opisuje jednu uređenu četvorku kojom su opisani „nastavci" u jednini i množini muškog i ženskog roda, ako postoje. Jedan takav kôd je, na primer, $N36 = [eur, rice, eurs, rices]$ koji opisuje promenu imenica kao što su *ambasadeur* i *acteur*, odnosno promenu imenica koje se u nominativu singulara muškog roda završavaju na *-eur*, za koje postoji ženski rod koji se u nominativu singulara završava na *-rice* i koje imaju množinu.[3] Zbog bogate morfologije srpskog jezika, predstavljanje morfološkog koda promenljivih reči pomoću uređenih n-torki nije pogodno, već će se koristiti drugačija sredstva, o čemu će kasnije biti reči.

[1] Autor indeksa [72] je, na primer, smatrao da oblik *vidi* odgovara glagolu *videti*. Videti poslovicu 5668, *Tu ga čuj, tu ga vidi.*

[2] Za definiciju pojma *formalna reč* videti [144].

[3] Kod morfološke klase za francuski sadrži ponekada elemente derivacione paradigme.

Osim kôda morfološke klase, formalnoj reči u DELAS-u može biti pridružen i sintaksički kôd koji predstavlja sintezu skupa bulovskih svojstava koja odgovaraju određenim sintaksičkim kategorijama [44]. Odnos između ova dva koda može se ilustrovati na primeru francuskog glagola *voler* čija su značenja *leteti* i *krasti*. Ovi homonimni glagoli imaju isti morfološki kôd ali se njihovi sintaksički kodovi razlikuju jer je glagol *voler* u značenju leteti neprelazan i nema dopune dok je u značenju *krasti* prelazan i dozvoljava dve dopune. Otuda sama formalna reč *voler* ima dva ulaza u rečniku DELAS, jer im se razlikuju pridružene informacije.

U srpskom jeziku, formalna reč *valjati* odgovara glagolu *váljati* sa značenjem *koturati, kotrljati* i glagolu *vàljati* sa značenjem *vredeti, biti vredan*[4]. Oba glagola imaju istu promenu te im je pridružen isti morfološki kod V01.00.2 (videti tabelu 5.4). Njihove sintaksičke osobine se pak razlikuju, ali kako sintaksički kôd u DELAS srpskog jezika nije uključen, oni imaju jedan ulaz u DELAS: formalnu reč *valjati*.

Ulaz u drugu listu koja se naziva DELAF čine formalne reči koje predstavljaju sve oblike flektivne paradigme formalnih reči iz DELAS. Lista DELAF može biti izračunata polazeći od ulaza u DELAS-u, bilo generisanjem bilo na neki drugi način (na osnovu različitih osnova [24]). Na taj način, par (*ambasadeur*, *N*36) u francuskom DELAS proizvodi sledeća četiri ulaza u listi DELAF:

```
ambasadeur,ambasaduer.N36:Nms
ambasadrice,ambasaduer.N36:Nfs
ambasadeurs,ambasaduer.N36:Nmp
ambasadrices,ambasaduer.N36:Nms
```

Jedan red u rečniku DELAF se sastoji od četiri elementa:

- flektivni oblik, odnosno, tekstualna reč;
- formalna reč iz DELAS-a, odnosno odgovarajuća leksička reč;
- morfološki kôd (koji može služiti i za razlikovanje formalnih reči u DELAS-u);
- vrednosti realizovanih morfosintaksičkih kategorija tekstualne reči.

Gornji primer prikazuje jednostavan slučaj kada svaka od formalnih reči: *ambasadeur*, *ambasadrice*, *ambasadeurs* i *ambasadrices* ima jedan ulaz u DELAF, što znači da svakoj od njih odgovara jedna leksička reč iz DELAS sa jednim morfološkim kodom i da je svakoj od tekstualnih reči pridružena samo jedna realizacija morfosintaksičke kategorije. U slučaju srpskog jezika, ulaz u DELAF je obično složeniji kao što to pokazuje sledeći primer [144]:

```
babi,baba.N71.1.01:Nfsd+
babi,baba.N71.1.01:Nfsl+
babi,baba.N75.01:Nmsd+
babi,baba.N75.01:Nmsl+
babi,babo.N78.01:Nmsd+
babi,babo.N78.01:Nmsd+
babi,baba.N70.01:Nfsd-
babi,baba.N70.01:Nfsl-
```

[4] Naglasci se kod ova dva glagola razlikuju, ali kako su rečnici DELAS namenjeni isključivo obradi pisanih tekstova, u njima akcenatski znaci nisu kodirani.

Formalna reč *babi* ima osam ulaza u DELAF. Njoj se mogu pridružiti dve formalne reči u DELAS-u: *baba* i *babo*. Prva od njih se, pri tom, opisuje sa tri različita morfološka koda: sa značenjem *starica* kodom N71.1.01, sa značenjem *slavski kolač* kodom N70.01 a sa značenjem *otac* kodom N75.01. Svaki od tih kodova generiše različite skupove dozvoljenih oblika. Na taj način formalnoj reči *babi* odgovaraju četiri ulaza u DELAS-u: *baba.N71.1.01*, *baba.N75.01*, *babo.N78.01* i *baba.N70.01*. a svaki od njih može biti realizacija dativa ili lokative jednine.

Alternativno, formalna reč *babi* se može prikazivati u rečniku DELAF i kao jedan ulaz:

`babi,baba.N71.1.01:Nfsd+:Nfsl+,baba.N75.01:Nmsd+:Nmsl+`
`,babo.N78.01:Nmsd+:Nmsd+,baba.N70.01:Nfsd-:Nfsl-` < *krajreda* >

Treba napomenuti da su navedena dva oblika predstavljanja ekvivalentna i da se mogu generisati jedan iz drugog. Korišćenje jednog ili drugog oblika zavisi prevashodno od informatičke aplikacije u kojoj će biti primenjen.

U narednim odeljcima biće prikazano formiranje navedene dve liste rečnika DELAS za srpski jezik[5].

5.2 Alfabet teksta i rečnika

Kada se govori o azbuci u kojoj se tekst na prirodnom jeziku zapisuje, neophodno je razlikovati tri tipa azbuka [144]:

- *pravopisna azbuka* koja je opisana u [107] i koja aproksimira grafemsku ravan SJ;
- *kodne azbuke* koje omogućavaju unos teksta sa standardizovane tastature i koje su definisane standardnim kodnim shemama: međunarodnim standardima, kao što su [56] i [59], i nad njima propisanim jugoslovenskim standardima kao što je [122];
- *tipografska azbuka* koja se sastoji od kolekcije grafičkih znakova u vizuelizovanoj formi teksta.

Ova tri nivoa mogu biti izvor različitih zabuna, a posebno imajući u vidu da azbuka SJ prema [107] ravnopravno koristi dva pisma: ćirilično i latinično. Tekst se po pravilu zapisuje jednim od ova dva pisma, premda je dozvoljeno pojavljivanje elemenata zapisanih alternativnim pismom. Karakteristični su primeri „PC- рачунар“, „YU- лига“ ili „ аутомобил марке BMW“. Osim skraćenica, koje su često u ćiriličnom tekstu zapisane latiničnim pismom, javlja se i umetanje teksta zapisanog na nekom stranom jeziku, na primer „public relations менаџер“ i „off shore компанија“. U takvim se situacijama zanemaruje činjenica da delovi teksta zapisani latiničnim pismom i umetnuti u ćirilični tekst, najčešće ne koriste latinično pismo SJ već nekog drugog jezika: engleskog, francuskog i slično. U datim primerima, W i Y, prema [107] nisu slova latiničnog pisma SJ.

Sličnu zabunu unosi i pojava rimskih brojeva u zapisu teksta. Oni se lako mogu interpretirati kao deo teksta zapisan latiničnim alfabetom. Međutim oni predstavljaju deo teksta zapisan korišćenjem specifičnog ograničenog alfabeta $\{I, V, X, L, C, D, M\}$. Da bi se takve zabune izbegle u standardu ISO 10646 predviđena su posebna kodna mesta za cifre rimskih brojeva [64].

Akcenatski znaci se ne beleže u pisanom tekstu na SJ osim u izuzetnim slučajevima kada pisac teksta oseća da može doći do teškoća u razumevanju teksta zbog neobeleženog naglaska.

[5] U radu se koristi skraćnica SJ podjednako za srpski i za srpskohrvatski jezik. Razlike između ova dva naziva nisu predmet ovog rada.

Tada se oni mogu javiti na tipografskom nivou, kao u „Odlučio sam sâm ..." ili „Bez pârâ ni u crkvu". Akcenatski znaci se u pravopisu i gramatikama SJ posmatraju kao zasebni grafički objekti koji ne čine celinu sa samoglasnikom, tako da, recimo, *â* ima, u formalnom smislu, nedefinisan status (na primer, nerešeno je pitanje poretka, i slično).

Pored znakova azbuke SJ koji učestvuju u formiranju reči, u pisanju teksta koriste se i znaci čija je funkcija da razdvoje pojavljivanje pojedinih reči u zapisu teksta. Ti znaci su najvećim delom interpunkcijski znaci čija je uloga prema [107] „jasnije prikazivanje onoga što hoće da se kaže", i u njih se tradicionalno ubrajaju tačka, zarez, upitnik i drugo. S obzirom na ulogu koju ovi znaci imaju u postupku izdvajanja formalnih reči iz elektronskog teksta, njima se pridružuju i znaci o kojima Pravopis ne govori, kao što su *blanko* (znak za razmak), *kraj reda* ili *tabulator*. Ovako proširen skup interpunkcijskih znakova određuje *skup separatora*.

Strogo definisanje ova dva skupa, azbuke i skupa separatora, je od osnovnog značaja za izdvajanje formalne reči iz elektronskog teksta, što je pak prvi korak u svakoj njegovoj daljoj obradi. Ovaj zadatak ne može se lako rešiti. Tako je, na primer, u [127] navedeno sedam različitih uloga navodnika i dvanaest različitih uloga crtice u francuskom pisanom tekstu a sličan je slučaj i sa tekstom na SJ. Premda navodnik i crtica nisu deo azbuke već skupa separatora u francuskom ili SJ, česti su slučajevi njihovog korišćenja unutar reči: *aujourd'hui*, *bjež'te* i *Đur'đ*, odnosno, *bas-relief* i *đul-devojka* i *naliv-pero*. Crtica se, takođe, često koristi i kao separator, kao što pokazuju primeri iz [107]: *nazovi-Srbin*, *pod Lovćen-planinom*, *budem-budeš-bude*. Tretiranje separatora (posebno navodnika i crtice), mora biti usklađeno sa azbukom rečnika DELAS.

Zapisivanje teksta u elektronskom obliku ne sme da bude takvo da na bilo koji način menja pravopisne zahteve niti da sputava uobičajenu praksu. Zapis teksta, međutim, mora biti takav da omogućava njegovu efikasnu i preciznu obradu. Kao najjednostavniji primer može se navesti problem korišćenja dva pisma u SJ. Za kodiranje ćiriličnog pisma se, po pravilu, koristi 30 različitih kodova dok se za kodiranje latiničnog pisma koristi 27 različitih kodova [122]. Naime, ćirilični znaci *љ*, *њ* i *џ* se zamenjuju u latiničnom pismu digrafima *lj*, *nj* i *dž* a zamena ćiriličnog *ђ* latiničnim digrafom *dj* je takođe prisutna. Na taj način ne postoji jednoznačna korespondencija između kodova za zapis ćiriličnog i latiničnog pisma. U slučajevima kada je poželjno isti tekst vizuelizovati korišćenjem ćiriličnog i latiničnog pisma, treba tačno znati kako je tekst originalno kodiran. Nedostatak ove informacije dovodi do degradiranja jezičke poruke. Tako, na primer u ćiriličnoj vizuelizaciji latinično kodiranog teksta javljaju se sledeće formalne reči (primer iz [144]):

...на Саветовању, на стручним предавањима
са стручњацима Бирострој ...

Analiza ovako kodiranog teksta nije moguća u prethodno razmatranom konceptu elektronskog rečnika, osim ako se greške koje se sistematski mogu javljati i same ne uključe u rečnik.

Otuda se nameće potreba za kodiranjem koje će učiniti tekst nezavisnim od moguće vizuelizacije. Korišćenjem na primer SGML jezika, tekst se snabdeva informacijom o originalnom kodiranju koju bi program za vizuelizaciju mogao koristiti i prilikom ćirilične vizuelizacije latinično kodiranog teksta primeniti postupak automatske transliteracije kakav je opisan u [80] i [144] (problem razlikovanja *nj* kao konsonantske grupe u *konjunkcija* i *nj* kao oznake digrafa u *savetovanje*). Gornji tekst je dovoljno kodirati na sledeći način:

```
<language id=SCO wsd='JUS.I.B1'>
```

```
...na Savetovanju, na stručnim predavanjima
sa stručnjacima Birostroja ...
```

gde `<language>` etiketa, prisutna u zaglavlju TEI aplikacije govori da je tekst koji sledi napisan (najvećim delom) na srpskohrvatskom jeziku, korišćenjem jezički specifičnog latiničnog koda JUS.I.B1.

Slične situacije se pojavljuju kada se ne vodi računa o privremenoj promeni pisma unutar teksta, kao u primeru (iz „Naše Borbe", decembar 1995):

Довезла се у белом БМЊ-у, у пратњи свог личног обезбеђења.

Adekvatnim kodiranjem, u kome je svaka promena azbučnog sistema na adekvatan način zabeležena, onemogućila bi ovakve pojave. Korišćenjem SGML-a, gornji tekst bi se mogao zapisati ovako:

```
<language id=SCO wsd=srb???>
.....................................................
Dovezla se u belom
<w lang=ENG>BMW</w>-u,
u pratwi svog ličnog obezbeđewa.
```

(Gornji primer je vizuelizovan latiničnim pismom i neproporcionalnim fontom da bi se jasnije predstavilo o kakvim se stvarno kodovima, u smislu bitovskih sekvencija, radi.) Brojni primeri prenebregavanja razlika između pravopisne azbuke, kodne azbuke i tipografske azbuke dati su u [144] i [94].

Azbuka rečnika DELAS zavisi od konkretnog prirodnog jezika koji rečnik opisuje. Prilikom definisanja azbuke rečnika DELAS za SJ uzeti su u obzir sledeći zahtevi:

- rečnik treba da bude tako kodiran da izražava grafemski a ne grafički sastav azbuke, tako da se može koristiti za obradu tekstova kodiranih latiničnim ili ćiriličnim pismom, odnosno, ne treba da postoje posebne varijante rečnika za ćirilično, odnosno, latinično pismo;
- rečnik treba da bude tako kodiran da je moguća njegova upotreba na raznovrsnim računarskim sistemima, što, recimo, podrazumeva da se prevođenje iz ASCII u EBCIDIC kôd može obaviti bez gubitka informacija.

Azubuka SJ koja se koristi u rečnicima DELAS je sledeća:

```
Ž A B C D Dy E F G H I J K L Lx M N Nx O P R S T U V Z Š Đ Ć Č
ž a b c d dy e f g h i j k l lx m n nx o p r s t u v z š đ ć č -
```

Prednost ove azbuke je da je ona najbliža onoj kodnoj azbuci za latinično pismo [122] koja se najčešće koristi za kodiranje tekstova na SJ. Jedina razlika je zamena digrafa `lj`, `nj` i `dž` digrafima `lx`, `nx` i `dy` čime se ukida dvoznačnost ovih sekvencija i njihova interpretacija kao digrafa ili kao konsonantskih grupa. Tako je u rečnicima DELAS *nadžnjeti* predstavljena formalnom reči `nadžnxeti` dok je *nadžvrljati* predstavljena sa `nadyvrlxati`.

Važna osobina ove azbuke je da se ona može kodirati 7-bitnim kodom ISO 646 za koji se sa velikom verovatnoćom može pretpostaviti da se bez gubitka ili deformacije može prenositi sa jednog na drugi računarski sistem te da se može koristiti na računarskim mrežama. Redosled slova u ovoj azbuci je sa namerom prikazan u redosledu koji određuje kolaciona sekvencija

ovog koda. Rečnici DELAS ma kako da su organizovani, o čemu će biti reči u sledećem odeljku, moraju biti uređeni preko poretka slova po formalnoj reči da bi se efikasno implementirala funkcija traženja u rečniku. Primenjujući ovakav redosled, sekvencija **žaba** prethodi sekvenciji **aba** a sekvencija **lxš** prethodi sekvenciji **lš** (premda je teško naći primer na SJ u kome bi do ovog drugog slučaja došlo). Uređivanje rečnika u redosledu kolacione sekvencije ima višestruki značaj:

- uređivanje rečnika i funkcija traženja mogu se efikasno implementirati.
- uređivanje rečnika po latiničnom ili ćirilićnom redosledu bi učinilo rečnik podesnijim za rad sa jednim od dva pisma što se htelo prvim od navedena dva zahteva izbeći a dobitak bi, na uštrb efikasnosti, bio mali.

Ne treba gubiti iz vida da su rečnici DELAS namenjeni isključivo informatičkoj obradi te da se za potrebe vizuelizacije mogu urediti u nekom drugom redosledu, ćiriličnom ili latiničnom.

Za razliku od azbuke francuskog DELAS-a, u azbuku rečnika DELAS za SJ uključena su, pored malih, i velika slova. Prema pravopisu, reč počinje velikim slovom na početku rečenice, a unutar rečenice se koristi za vlastita imena i reči iz poštovanja. U ovom drugom slučaju, upotreba velikog slova svojstvena je samoj reči. Prema [47], vlastita imena ne ulaze u sastav francuskog DELAS-a, pa samim tim ne postoji ni potreba za uvođenjem velikih slova u azbuku rečnika. Kako su vlastita imena promenljive reči SJ, njihovo isključivanje iz rečnika ne bi bilo opravdano.

Kao jedna moguća primena ličnih imena iz rečnika DELAS može se posmatrati sistem za izradu cirkularnih pisama, u kojima se pisac obraća svaki put nekoj drugoj osobi sa liste, a svako takvo obraćanje zahteva određeni oblik imena (na primer, *Dragi Vlado*, *Na ruke gospodina Petrovića* ili *za Zoricu*).

Ovo ne bi bio dovoljan razlog za uvođenje velikih slova u azbuku, da veliko slovo nema distinktivnu ulogu u reprezentaciji reči. Tako je *Vlada* lično ime muškog roda, dok je *vlada* imenica ženskog roda. Takođe, *rak*, u značenju *životinja* je imenica muškog roda koja ima množinu, dok u značenju *bolest* nema množinu. *Rak* kao ime *sazvežda* nema množinu dok u značenju *osobe rođene u zodijačkom znaku* ima množinu. Ovo znači da bi ovim dvema formalnim rečima, *rak* i *Rak*, bila pridružena četiri različita koda morfološke klase u rečniku DELAS. O mogućem proširivanju azbuke rečnika DELAS biće govora u odeljku 5.4.

5.3 Rečnici DELAS i DELAF za SJ

Ulaz u listu DELAS čine formalne reči koje odgovaraju pojmu uobičajenih rečničkih odrednica. Takva odluka je razumljiva imajući u vidu da se ova lista formira prvenstveno na osnovu tradicionalnih rečnika. U prilog ovakvom opredeljenju idu i uobičajene primene elektronskih rečnika, kao što su lematizacija (to jest postupak svođenja oblika reči iz pisanog teksta na „osnovni“ oblik), te pretraživanje teksta u kome se u tekstu pronalaze sva pojavljivanja (bez obzira na oblik) nekih reči koje se po pravilu zadaju u obliku rečničke odrednice.

Sintaksa jednog ulaza u listu DELAS može se predstaviti regularnim izrazom:

```
{azb}+(","{vrs}{kls}*izg)*
```

U zapisu ovog izraza koristi se sintaksa LEX-a [86], pri čemu je značenje imena:

```
azb  [\x45Ž-Čž-č]                              azbuka
vrs  ["N" | "Adj" | "V" | "Adv" | "Con" ...]   vrsta reči
kls  [^,:%Ž-Čž-č]                              morfološka klasa
izg  [ "*" | "E" | "J" ]                       izgovor
```

Kao što se iz ovog zapisa vidi, formalna reč i informacija o njoj razdavajaju se zarezom. Kôd morfološke klase sastoji se od oznake vrste reči, čiji je potpuni spisak dat u tabeli 5.1, i oznake klase koja definiše promenu formalne reči. Za imeničke i pridevske zamenice, kao i za različite vrste brojeva, uvedene su posebne oznake vrste reči zbog njihove specifične uloge. Za nepromenljive reči se navodi samo oznaka klase. Za označavanje klase koriste se cifre i proizvoljni specijalni znaci, izuzev zareza, dvotačke i znaka procenta koji imaju separatorsko značenje u listama DELAS i DELAF. Blanko karakter se u ulazu liste DELAS ne pojavljuje. Izvod iz liste DELAS/SJ dat je u dodatku D.1.

Vrsta reči	Kod	Primer
Imenice	N	*usta*
Pridevi	Adj	*pijan*
Glagoli	V	*znati*
Prilozi	Adv	*večeras*
Veznici	Con	*ali*
Uzvici	Int	*ej*
Rečce	Par	*međutim*
Predlozi	Pre	*uz*
Imeničke zamenice	ProN	*ona*
Pridevske zamenice	ProA	*njen*
Osnovni brojevi	Num	*trinaest*
Redni brojevi	Ord	*dvestoti*
Zbirni Brojevi	Col	*pedesetoro*
Brojčane imenice	NumN	*tridesetorica*
Brojčani pridevi	NumA	*dvoji*

Tabela 5.1: Oznake vrste reči u elektronskom rečniku DELAS

Uz informacije koje opisuju paradigmatska svojstva odrednice, formalnoj reči u DELAS-u mogu se načelno pridružiti i druge informacije, kao što je sintaksički kôd, kôd hijerarhija, leksičke crte i drugo [21]. Svakom ulazu u DELAS/SJ pridružen je samo kôd koji određuje atribut izgovora za formalnu reč. U ovom trenutku, u DELAS su uključene odrednice koje pripadaju ekavskom i ijekavskom izgovoru, i one su na odgovarajući način kodirane korišćenjem parametra izg. Pri tome treba naglasiti da pripadnost određenom izgovoru može uticati na formiranje oblika i imati distinktivnu ulogu u preciziranju ulaza u DELAS. Razmotrimo slučaj broja *dva*, koji se u takvom obliku javlja u oba izgovora i koristi uz imenice muškog i srednjeg roda, dok se uz imenice ženskog roda u ekavskom izgovoru koristi oblik *dve* a u ijekavskom oblik *dvije*. U DELAS-u se, stoga, za broj *dva* nalazi sledeći ulaza:

```
dva,Num020E,Num021J
```

dok se u DELAF-u oblici ekavskog izgovora pridružuju „ekavskoj odrednici" `dva.Num020E` a oblici ijekavskog izgovora „ijekavskoj odrednici" `dva.Num021J`. Oblici koji se koriste u oba izgovora pridružuju se obema formalnim rečima:

```
dvaju,dva.Num020E:mpg*:npg*,dva.Num021J:mpg*:npg*
dve,dva.Num020E:fpn*:fpa*:fpv*
dveju,dva.Num020E:fpg*
dvije,dva.Num021J:fpn*:fpa*:fpv*
dviju,dva.Num021J:fpg*
```

Sličan je slučaj i sa brojevima *oba*, *obadva* ili sa glagolom *biti*. O mogućnosti spajanja odrednica različitog izgovora biće govora u odeljku 5.4.

Lista DELAF sastoji se od svih oblika formalnih reči navednih u listi DELAS. Sintaksa ulaza u listi DELAF, koristeći iste definicije imena uvedene u definiciji ulaza liste DELAS, može se predstaviti sledećim regularnim izrazom:

```
{azb}+(","{azb}*"."{vrs}{kls}*(":"{grm}+)*)*
```

pri čemu se ime grm definiše na sledeći način:

```
grm  ["*" | "m" | "n" | "f" | "s" | "p" | "1" ... ]
```

Kao što se iz ovih regularnih izraza vidi, DELAS i DELAF za SJ se formiraju tako da se sve informacije o jednoj formalnoj reči okupljaju oko jednog ulaza. Iz tog razloga, u DELAS-u i DELAF-u za SJ ne postoje dva ulaza (M_1, I_1) i (M_2, I_2) takva da važi $M_1 = M_2$ (primer broja *dva*). Jedan izvod iz liste DELAF za SJ dat je u dodatku D.2.

Gramatičke informacije koje opisuju formalnu reč iz DELAF-a zavise od vrste reči kojoj se ta formalna reč pridružuje. Na primer, kodovi koji se koriste za opisivanje glagola dati su u tabeli 5.2. Gramatičke informacije, međutim, ne zavise samo od vrste reči kojoj se formalna reč pridružuje već i od samog oblika: ako se oblik iz DELAF-a pridružuje glagolu, onda će se jednim gramatičkim informacijama opisivati oblik glagola u prezentu, drugim radni pridev glagola, dok se, recimo, za prilog sadašnji neće koristiti nikakve dodatne gramatičke informacije. Na primer, oblik *pevam* glagola *pevati* opisuje se kodovima `P1s`, oblik *pevao* kodovima `PPsm` dok se za oblik *pevajući* navodi samo kod `AdvPr`.

Od značaja je takođe i redosled navođenja informacija i on je unapred utvrđen. Takav postupak ne samo što olakšava korišćenje rečnika već dopušta upotrebu džoker-karaketra (engl. *wild character*) u slučajevima kada formalna reč iz DELAF-a odgovara svim dozvoljenim vrednostima na određenoj poziciji u gramatičkom kodu. Upotreba džoker-karaktera može se ilustrovati na sledećem ulazu iz DELAF-a za SJ:

```
kojim.koji.ProA07:msi*:nsi*:*pd*:*pi*:*pl*
```

Na četvrtoj poziciji u gramatičkom kodu zamenica je oznaka animatnosti. U svih 5 navedenih gramatičkih kodova na ovoj poziciji je džoker * kao oznaka da se oblik `kojim` koristi uz imenice bilo da su označene kao živo bilo kao neživo. Kod poslednja tri kôda se i na prvoj poziciji pojavljuje zvezdica što znači da se oblikom `kojim` u dativu, instrumentalu i lokativu plurala mogu predstaviti oblici za sva tri roda: muški, ženski i srednji.

Primenom koda morfološke klase na formalnu reč određuje se precizan skup svih oblika te formalne reči. Za većinu evropskih jezika generisanje flektivnih oblika se obavlja po broju osnova. Ovde su primenjena dva suštinski drukčija prilaza:

- Prvi, primenjen na imenske reči (imenice i prideve), definiše takav kôd koji omogućava morfološko izračunavanje svih oblika reči, a koji se zasniva na modelu segmentacije reči na njen nepromenljivi i promenljivi deo;
- Drugi, koji je primenjen na ostale promenljive vrste reči, opisuje putem regularnog izraza (ili automata) sve promene koje se dešavaju u promenljivom delu formalne reči.

5.3.1 Glagoli

Morfološka klasa glagola treba da obuhvata one glagolske oblike koji se u tradicionalnim gramatikama nazivaju prostim glagolskim oblicima i koji se dobijaju dodavanjem odgovarajućih nastavaka (za lice i oblik) na odgovarajuću osnovu (infinitivnu ili prezentsku). U ove glagolske oblike spadaju: infinitiv, prezent, aorist, imperfekat, imperativ, glagolski prilog sadašnji, glagolski prilog prošli, radni glagolski pridev i trpni glagolski pridev.

Ovi su oblici od značaja za rečnik DELA jer separatorski karakteri (pre svega blanko karakter) ili, pak, druge proste reči ne učestvuju u njihovom građenju kao što je slučaj sa složenim oblicima. Futur I ima lik prostog glagolskog oblika kada se enklitički oblik pomoćnog glagola *hteti* koristi iza infinitiva glagola koji se menja pa se, stoga, i njegovi oblici uključuju u opis morfološke klase glagola.

Trpni glagolski pridev nije uključen u opis morfološke klase već ima zaseban ulaz u listi DELAS i obeležen je kodom vrste reči `Adj`. Razlozi su isključivo praktične prirode a motivisani su velikim brojem oblika pridevske flektivne paradigme (videti 5.3.2), što prevazilazi trenutno raspoložive računarske resurse za obradu rečnika. O problemima koji potiču od dimenzija rečnika biće više govora u 5.5. Osim toga, određivanje koda morfološke klase glagola uključivalo bi, potencijalno, i klasifikaciju trpnih prideva. Adekvatna obrada trpnih glagolskih prideva u sistemu DELA će biti naknadno urađena.

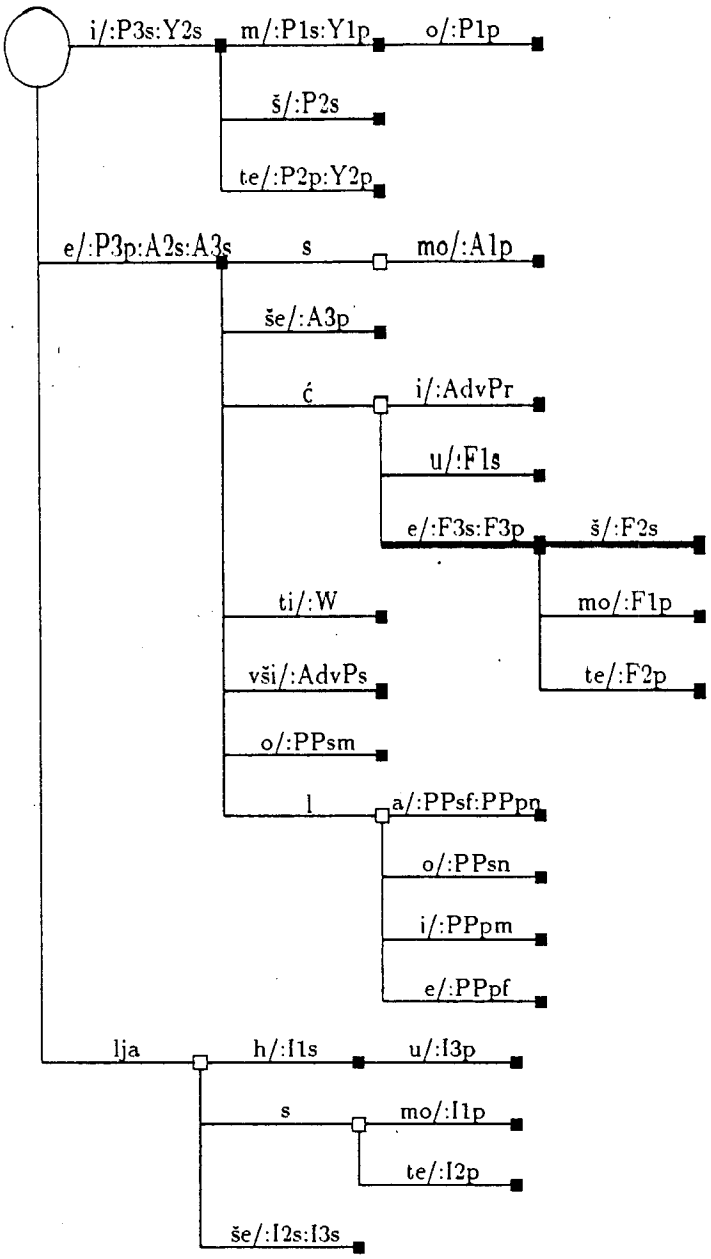
Gramatička informacija	Kod
muški, ženski, srednji rod	m, f, n
lice	1, 2, 3
jednina, množina	s, p
infinitiv	W
prezent	P
aorist	A
imperfekat	I
imperativ	Y
glagolski prilog sadašnji	AdvPr
glagolski prilog prošli	AdvPs
radni glagolski pridev	PP
futur I	F

Tabela 5.2: Gramatičke informacije za opis glagola u elektronskom rečniku DELAF

U tabeli 5.2 navedene su gramatičke informacije koje se u rečniku DELAF koriste za opisivanje oblika glagola. Ove informacije obuhvataju gramatičke kategorije lica, vremena, načina, gramatičkog broja i gramatičkog roda. Lični glagolski oblici, a to su oblici glagola u prezentu, aoristu, imperfektu, futuru I i imeprativu, određuju se kategorijama vremena, odnosno načina, lica i gramatičkog broja. Od neličnih glagolskih oblika, radni glagolski pridev određuje se u kategorijama gramatičkog broja i gramatičkog roda, dok oblici infinitiva, glagolskog priloga sadašnjeg i prošlog ne zahtevaju druge informacije.

Klasifikacija glagola prema vrsti konjugacije opisuje se obično u gramatičkim priručnicima, i ređe u dodacima rečnika. Ovakve klasifikacije se predlažu iz istraživačkih razloga sa ciljem izučavanja glagolskog sistema određenog jezika ([108], [106]), ali i iz čisto praktičnih razloga,

priručnik [17]). Za SJ opisano je u literaturi više takvih klasifikacija za koje je zajedničko da polaze od toga da se glagolski oblici dobijaju od dve osnove: prezentske i infinitivne (ili aoristne). U daljoj subklasifikaciji one se međusobno razlikuju. Klasifikacije opisane u [132] i [131] uzimaju u obzir odnos ove dve osnove, dok se klasifikacija opisana u [12] zasniva samo na infinitivnoj osnovi. Sve ove klasifikacije teže da opišu promenu što većeg broja glagola sa pravilnom promenom, dok određeni broj glagola uvek ostaje izvan sistema klasifikacije, tvoreći klasu nepravilnih glagola.



Slika 5.1: Transduktor za opis svih oblika glagola *živeti*

Klasifikacija opisana u [68] namenjena je informatičkim obradama i imala je za cilj automatsko generisanje glagolskih oblika. Ona je, takođe, polazila od postojanje dve osnove,

koriste različite osnove. Ipak, sa stanovišta računarske obrade ovaj pristup nije podesan, jer tradicionalni rečnici vode glagol pod infinitiv.

Klasifikacija koja se ovde predlaže sačinjena je isključivo za potrebe informatičke obrade. Ona polazi od sledećih pretpostavki: formalna reč koja je ulaz u rečnik DELAS, a koja predstavlja infinitiv glagola u SJ, može se posmatrati kao niska karaktera. Posmatranjem oblika flektivne paradigme, iz te niske se može formalno izdvojiti njen početni deo ili prefiks[6] koji ćemo nazvati *nepromenljivim delom formalne reči* a koji predstavlja zajednički faktor u svim glagolskim oblicima. Taj nepromenljivi deo može biti i prazna niska $\emptyset$ kao što je slučaj kod glagola *slati, šaljem* i *gnati, ženem*. Kôd jedne klase glagola je definisan listom završnih delova ili sufiksa koji operacijom konkatenacije sa nepromenljivim delom opisuju sve glagolske oblike. Svakom elementu liste završnih delova pridružene su gramatičke informacije oblika koji se tim završnim delom grade.

Na slici 5.1 predstavljena je u obliku konačnog transduktora ovakva lista sufiksa za glagol *živeti*. Nepromenljivi deo formalne reči *živeti* je *živ* a istim transduktorom bili bi predstavljeni i sufiksi glagola *grmeti* i *svrbeti*. Ova lista sadrži ukupno 27 sufiksa za opis 35 glagolskih oblika i mogla je biti predstavljena i u obliku uređene 35-torke. Predstavljanje sufiksa pomoću transduktora je daleko pogodnije kako zbog glagola koji nemaju sve oblike — na primer, glagol *moći* nema oblike imperativa — tako i zbog glagola koji imaju višestruke oblike — na primer, glagol *dati, dam* i *dadem* i *dadnem*. Nepravilni glagoli, u ovakvom pristupu, su svedeni na posebne klase sa malim brojem pridruženih glagola: klasa kojoj takvi glagoli pripadaju sadržaće mali broj glagola, kao što se vidi iz dodatka D.3.

Postupak klasifikacije se sastoji u navođenju svih mogućih transduktora. Za klasifikaciju francuskih imenica i glagola [127], kôd deklinacije, odnosno, konjugacije dobijen je njihovim numerisanjem brojevima od 1 pa naviše. Radi lakšeg snalaženja među brojnim različitim transduktorima za SJ, uz svaku klasu glagola uveden je i pojam podklase, na slična način na koji je to učinjeno i za imenske reči. Taj pristup će ovde biti predstavljen i ilustrovan u tabelama 5.4, 5.5 i 5.6.

Uočava se da se sufiksi u nekim klasama poklapaju sa nastavcima glagolskih oblika: takav je slučaj sa glagolima *pevati* iz klase 01.01 i *tresti* iz klase 08.01 (videti tabelu 5.4). Za većinu drugih klasa važi da se sufiksi takođe mogu formalno razdvojiti na dva dela: početni deo, nazvan *infiks*, i završni deo koji predstavlja nastavak glagolskog oblika. U velikom broju klasa dva infiksa se na različite načine kombinuju sa nastavcima glagolskih oblika: na primer, takve su klase 20, 21, 24 itd. Konkatenacija nepromenljivog dela formalne reči i infiksa kao rezultat može dati niske koje se poklapaju sa pojmom infinitivne i prezentske osnove u tradicionalnoj gramatici. Neke klase, pak, imaju tri infiksa (klasa 34, na primer) (posledica jotovanja osnove u imperfektu). Cilj ove klasifikacije nije da opiše morfološke aspekte ovih fenomena, premda se u nekim slučajevima može pojaviti saglasnost ove klasifikacije sa nekom od tradicionalnih klasifikacija. S druge strane, u ovoj podeli, glagoli *goreti* i *trošiti* pripadaju različitim podklasama iste klase 33, dok su u tradicionalnim klasifikacijama [131] i [12] oni u različitim klasama.

Ostaje da se precizira šta su nastavci glagolskih oblika i kako se oni izdvajaju iz formalne reči — glagolskog oblika jer se tek tako može utvrditi broj i oblik infiksa. U tabeli 5.3 dati su mogući nastavci svih glagolskih oblika koji su od značaja za klasifikaciju. Nastavci su izabrani tako da se: (1) izabrani nastavci jednog glagolskog oblika mogu dopisati na jedan infiks; (2) ako se za neki glagolski oblik može primeniti više nastavaka, bira se onaj koji daje

[6] *prefiks, sufiks* i *infiks* se ovde koriste u značenju koje imaju u teoriji formalnih jezika a ne u lingvistici.

Glagolski oblici	Kod	Nastavci
1. Infinitiv	1	-ti
	2	-ći
2. Prezent	1	-m, -š, ∅, -mo, -te, -ju
	2	-em, -eš, -e, -emo, -ete, -u
	3	-im, -iš, -i, -imo, -ite, -e
	4	-im, -iš, -i, -imo, -ite, -u
3. Aorist	1	-h, ∅, ∅, -smo, -ste, -še
	2	-oh, -e, -e, -osmo, -oste, -oše
	3	-uh, -u, -u, -usmo, -uste, -uše
[redacted]kat	1	-h, -še, -še, -smo, -ste, -še
	2	-ah, -aše, aše, -asmo, -aste, -ahu
	3	-ijah, -ijaše, -ijaše, -ijasmo, -ijaste, -ijahu
5. Imperativ	1	∅, -mo, -te
	2	-i, -imo, -ite
	3	-j, -jmo, -jte
6. Glagolski prilog sadašnji		/-P3s/ći
7. Glagolski prilog prošli	1	-vši
	2	-avši
	3	-uvši
8. Radni glagolski pridev	1	-o, -la, -lo, -li, -le, la
	2	-ao, -la, -lo, -li, -le, la
	3	∅, -la, -lo, -li, -le, la

Tabela 5.3: Mogući nastavci glagolskih oblika

manji broj infiksa klase. Radi zadovoljenja prvog zahteva, uvedeni su nastavci 3 i 4 za oblike prezenta. Gornji zahtev nije se mogao u potpunosti ispoštovati, posebno zbog oblika radnog glagolskog prideva glagola ijekavskog izgovora (na primer, radni pridev glagola *besnjeti* glasi *besnio*, *besnjela*, ...)

Zadovoljavanje drugog zahteva može se ilustrovati na primeru glagola *pevati*. Za oblike imperativa izabran je nastavak 3 (-j, -jmo, -jte) premda je mogao biti izabran i nastavak 1 (∅, -mo, -te) čime bi bio uveden infiks j koji se ne koristi u gradnji ni jednog drugog oblika a klasa bi koristila dva infiksa umesto jednog, praznog. S druge strane, za klasu glagola *pljuvati*, izabran je nastavak 3 za oblike imperativa jer se infiks j koristi i u gradnji oblika prezenta i priloga sadašnjeg dok se prazni infiks ne koristi.

Klasifikacija je sprovedena na osnovu tipa promene koji je u tabeli 5.4 predstavljen nizom od osam elemenata od kojih svaki redom odgovara glagolskom obliku iz tabele 5.3. Svaki element u nizu predstavljen je u obliku X/Y gde X označava koji infiks se koristi a Y koji nastavci se koriste za građenje odgovarajućih oblika. Samo za oblike priloga sadašnjeg element u nizu je oblika X, jer nastavak za ovaj oblik zavisi od tipa prezentskog nastavka. Crtica (—) u nizu znači da odgovarajući oblik glagola te klase ne postoji dok je element niza tamo gde ima više dozvoljenih oblika ("$X/Y+$") + "X/Y"

Kod klase	Tip promene	Promenljivi deo	Primer
01.00.2	1/1 1/1 1/1 1/1 1/3 1 1/1 1/1		*pevati dospeti*
02.00.2	1/1 1/1+ 1/1+ 1/1+ 1/3+ 1 + 1/1 1/1 2/2 2/2 2/3 2/2 2	/Ø/d/	*znati*
03.00.2	1/1 1/1+ 1/1+ 1/1 — 1 1/1 1/1 2/2 3/2	/Ø/dn/d/	*morati*
04.00.2	1/1 1/1+ 1/1+ 1/1+ 1/3 1 1/1 1/1 2/2 2/2 2/3+ 4/2 3/2	/Ø/d/đ/dn/	*imati*
05.00.2	1/1 1/1+ 1/1+ 1/1+ — 1 1/1 1/1 2/2 2/2 3/2	/Ø/d/đ/	*nemati*
07.00.3	1/1 1/1+ 1/1 1/1 1/3 1+ 1/1 1/1 2/2 2	/a/Ø/	*tkati*
08.00.2	1/1 1/2 1/2 1/3 1/2 1 1/2 1/2		*tresti*
09.00.2	1/1 2/2 2/2 2/3 2/2 2 2/2 2(1)/2	/Ø/t/	*rasti*
11.00.3	1/1 2/2 2/2 2/3 2/2 2 2/2 2/2	/s/z/	*gristi*
12.00.4	1/1 2/2 2/2 2/3+ 2/2 2 2/2 2/2 2/2	/ps/b/	*grepsti*
13.00.3	1/1 2/2 2/2 2/3 2/2 2 2/2 3/1	/s/d/Ø/	*krasti*
13.01.3		/s/t/Ø/	*mesti*
15.00.4	1/1 2/2+ 2/2+ 2/3 2/2+ 2 + 2/2+ 4/1 3/2 4/1 3/2 3 4/1	/es/eđ/Ø/e/	*gresti*
16.00.3	1/1 2/2 2/2 3/2 2/2 2 2/2 4/1	/s/d/đ/Ø/	*jesti*
19.00.3	1/1 2/2 2/2 2/3 2/2 2 2/2 3/3	/s/d/Ø/	*bosti*
20.00.3	1/1 2/2 1/1 1/1 2/1 2 1/1 1/1	/a/Ø/	*grejati*
20.01.4		/va/j/	*pljuvati*
20.02.5		/ova/uj/	*gladovati*
20.03.5		/iva/uj/	*dosađivati*
20.04.5		/eva/uj/	*vojevati*
21.00.3	1/1 2/2 1/1 1/1 2/2 2 1/1 1/1	/a/Ø/	*orati*
21.01.3		/a/lj/	*zobati*
21.02.4		/sa/š/	*pisati*
21.03.4		/ha/š/	*jahati*
21.04.4		/ga/ž/	*lagati*
21.05.4		/za/ž/	*vezati*
21.06.4		/ca/č/	*micati*
21.07.4		/ka/č/	*plakati*
21.08.4		/ta/ć/	*kakotati*
21.09.4		/da/đ/	*glodati*
21.10.5		/ska/št/	*iskati*
21.11.5		/ska/šć/	*stiskati*
21.12.5		/hta/hć/	*drhtati*
21.13.5		/la/olj/	*klati*
22.00.4	1/1 2/2 1/1 1/1+ 2/2 2 1/1 1/1 2/2+ 2/3	/ra/er/	*brati*
22.01.4		/va/ov/	*zvati*
22.02.5		/sla/šalj/	*slati*

5.4 Klasifikacija nesvršenih glagola

Subklasifikacija unutar jedne klase vrši se na osnovu konkretnih realizacija infiksa. Na primer, glagoli *krasti* i *mesti* pripadaju istoj klasi 13 a razlikuju se samo po realizaciji infikasa: prvi koristi infikse /s/d/Ø/ (*kra-s-ti, kra-d-em, kra-Ø-o*) dok drugi koristi infikse /s/t/Ø/ (*me-s-ti, me-t-em, me-Ø-o*).

Kod klase	Tip promene	Promenljivi deo	Primer
23.00.4	1/1 2/2 1/1 3/1 2/2 2 1/1 1/1	/nu/n/nja/	*brinuti*
23.01.4		/snu/sn/šnja/	*kisnuti*
23.02.4		/znu/zn/žnja/	*mrznuti*
24.00.2	1/1 2/2 1/1 2/2 2/1 2 1/1 1/1	/∅/j/	*piti*
25.00.4	1/1 2/2 1/1 2/2 2/2 2 1/1 1/1	/le/elj/	*mleti*
26.00.3	1/1 2/2 1/1 2/2 2/2 2 1/1+ 2/1 2/1	/e/∅/	*mreti*
27.00.3	1/1 2/2 1/1 2/2+ 2/2 2 1/1 1/1 2/4	/e/m/	*žeti*
28.00.4	1/1 2/2 1/1 2/3+ 2/2 2 1/1 1/1 3/2	/le/un/unj/	*kleti*
30.00.3	1/1 2/2+ 1/1+ 2/2+ 2/2+ 2+ 4/1+ 4/1 3/1 4/1 3/2 3/2 3 1/1	/e/nj/anj/nje/	*žeti*
31.00.3	1/1 2/3 1/1 1/1 2/2 2 1/1 1/1	/a/∅/	*bežati*
32.00.3	1/1 2/3 1/1 1/1 2/1 2 1/1 1/1	/a/∅/	*brojati*
32.01.5		/aja/oj/	*stajati*
33.00.3	1/1 2/3 1/1 2/2 2/2 2 1/1 1/1	/e/∅/	*goreti*
33.01.3		/i/∅/	*trošiti*
34.00.3	1/1 2/3 1/1 3/2 2/2 2 1/1 1/1	/e/∅/lj/	*živeti*
34.01.3		/i/∅/lj/	*gubiti*
34.02.4		/le/l/lj/	*želeti*
34.03.4		/li/l/lj/	*paliti*
34.04.4		/ne/n/nj/	*crveneti*
34.05.4		/ni/n/nj/	*zvoniti*
34.06.4		/ze/z/ž/	*mrzeti*
34.07.4		/zi/z/ž/	*paziti*
34.08.4		/te/t/ć/	*leteti*
34.09.4		/ti/t/ć/	*slutiti*
34.10.4		/de/d/đ/	*bledeti*
34.11.4		/di/d/đ/	*saditi*
34.12.4		/si/s/š/	*kositi*
34.13.5		/sne/sn/šnj/	*besneti*
34.14.5		/sti/st/šć/	*postiti*
34.15.5		/zni/zn/žnj/	*prazniti*
34.16.5		/sli/sl/šlj/	*misliti*
34.17.5		/ci/cl/č/	*kititi*
35.00.5	1/1 2/3 1/1 3/2+ 2/2 2 1/1 1/1 4/2	/sti/st/šć/št/	*krstiti*
36.00.3	1/1 2/4 1/1 2/2 2/2 2 1/1 1/1	/e/∅/	vreti
50.00.2	1/2 2(3)/1 3/2 4/3+ 4/2 3 3/2 3/2 2/2	/∅/š/h/s/	*vrći*
50.01.2		/∅/č/k/c/	*seći*
50.02.2		/∅/ž/g/z/	*leći*
52.00.2	1/2 2(3)/N 2/2 2/2 — 2 2/2 2/2	/∅/g/ž/	*moći*
54.00.2	1/2 2/2 2/2 3/2 2/2 2 4/2 4/2	/∅/d/đš/	*ići*

Tabela 5.4: Klasifikacija nesvršenih glagola — nastavak

Gramatičkom kategorijom glagolskog vida označava se trajanje radnje, stanja ili zbivanja koje glagol označava u rečenici. U SJ, glagoli se prema vidu dele na nesvršene, svršene i dvovidske. U tradicionalnim gramatikama, glagolski vid nije od značaja za klasifikaciju glagola po vrstama. Kako svršeni glagoli nemaju oblike imperfekta i priloga sadašnjeg, gramatički vid je od značaja za predloženu klasifikaciju, jer ti glagoli pripadaju klasama koje opisuju automati u kojima ne postoje putevi za ove oblike. Većini klasa nesvršenih glagola

odgovara podklasa svršenih glagola. Na primer, klasi 01.00 (*pevati*) odgovara klasa svršenih glagola 01.50 (*propevati*), a klasi 19.00 (*bosti*) odgovara klasa svršenih glagola 19.50 (*zabosti*). Kako su neke klase i podklase svršenih glagola nastale kao posledica alternacije u imperfektu, za odgovarajuće svršene glagole ta klasa ili podklasa ne postoji. Na primer, glagol *jesti* je u klasi 16.00 koja je nastala iz klase 13.00 (*krasti*) zbog jotovanja u imperfektu. Svršeni glagol *izjesti* je, s toga, u klasi 13.50 (kao i *ukrasti*).

Nekim glagolima, kao što je *povèsti*, kome odgovara formalna reč povesti pridružuju se dve klase: 11.50 za paradigmu *povèsti, povèzēm, povezao* koja odgovara značenju *uzeti koga u prevozno sredstvo*, i 13.50 za paradigmu *povèsti, povèdēm, poveo* koja odgovara značenju *uzeti koga sa sobom*. S druge strane, formalna reč *valjati* odgovara i glagolu *váljati* i glagolu *vàljati* i oni pripadaju istoj klasi, 01.00. Neki glagoli, kao na primer, *njíhati* se može menjati na dva potpuno različita načina: *njíhati, njíšēm* i *njíhati, njíhām* i stoga se formalnoj reči njihati pridružuju dva koda morfološke klase, 21.03 i 01.00. Tamo gde se u glagolskoj paradigmi pojavljuju samo neki višestruki oblici uvodi se nova klasa koja te oblike obuhvata. Takav je slučaj glagola *tkati* koji ima dvostruke oblike prezenta: *tkam* i *tkem* i, s toga, dvostruke oblike priloga sadašnjeg: *tkajući* i *tkući*, što formira klasu 07.00 izdvojenu iz klase 01.00.

Ekavski i ijekavski izgovor nastali su kao rezultat zamene starog glasa „jat" (ćirilično ѣ, latinično ě) glasom *e*, odnosno, glasovnim grupama *ije* i *je*. Tabele 5.4 i 5.5 ilustruju klasifikaciju glagola ekavskog izgovora kao i onih u kojima refleksa starog glasa „jat" nema. Ukoliko se refleks starog glasa „jat" nalazi u prefiksu formalne reči, to jest u njenom nepromenljivom delu, glagol ijekavskog izgovora nalazi se, najčešće, u istoj klasi kao i odgovarajući glagol ekavskog izgovora. Tako su u klasi 01.00 glagoli *pevati* i *pjevati* a u klasi 34.09 glagoli *pretiti* i *prijetiti*. Oni glagoli ijekavskog izgovora čiji ekavski par u infiksu, odnosno promenljivom delu, ima glas *e* kao refleks starog „jat" svrstavaju se u zasebne klase. Njihovi infiksi se razlikuju od infiksa odgovarajućeg glagola ekavskog izgovora: na primer, infiksi glagola *mreti* su /e/∅/ (*mr-e-ti, mr-∅em, mr-e-h, mr-∅-ah,...*), a infiksi glagola *mrijeti* su /ije/∅/ (*mr-ije-ti, mr-∅-em, mr-e-h, mr-∅-ah,...*) pa prvi pripada klasi 26.00 a drugi klasi 26.25.

Glagoli ijekavskog izgovora u nekim slučajevima imaju više infiksa od odgovarajućeg glagola ekavskog izgovora kao rezultat zamene starog „jat" različitim glasovnim grupama u ijekavskom izgovoru. Na primer, infiksi glagola *goreti* su /e/∅/ (*gor-e-ti, gor-∅-im, gor-e-h, gor-∅-ah, gor-∅-i, gor-∅-ećі, gor-e-vši, gor-e-o*) dok su infiksi glagola *gorjeti* /je/∅/i/ (*gor-je-ti, gor-∅-im, gor-je-h, gor-∅-ah, gor-∅-i, gor-∅-ećі, gor-je-vši, gor-i-o, gor-je-la*).

Ponekad glagoli ijekavskog izgovora pripadaju zasebnoj klasi u odnosu na odgovarajući glagol ekavskog izgovora premda se e kao refleks starog „jat" ne nalazi u infiksu glagola ekavskog izgovora, do čega takođe dolazi usled zamene „jat" različitim grupama u ijekavskom izgovoru. Takav je slučaj glagola *odseći* iz klase 50.51 i glagola *odsjeći* iz klase 50.76. O mogućnosti sistematskog rešavanje pitanja dijalekatskih i grafemskih varijacija kroz rečnik DELA biće govora u odeljku 5.4.

Pomoćni glagoli *jesam, biti* i *hteti*, kao i glagoli nepravilnog građenja kao *velim*, razvrstani su u podklase klase 99. Kako se u odričnim oblicima glagola *jesam* odrična rečca *ne* spaja sa glagolskim oblikom dajući: *nisam, nisi, ...* ovi oblici su takođe uključeni u automat za prepoznavanje oblika glagola *jesam*. Za odrične oblike glagola *hteti*, gde takođe dolazi do spajanja rečce sa glagolom to nije učinjeno jer se niska ne- u funkciji odrične rečce *ne* prepoznaje u fazi obrade teksta što je od značaja i za druge vrste reči, pre svega prideve i priloge.

Kod klase	Tip promene	Promenljivi deo	Primer
01.50.2	1/1 1/1 1/1 — 1/3 — 1/1 1/1		*propevati*
02.50.2	1/1 1/1+ 1/1+ — 1/3+ — 1/1 1/1 2/2 2/2 2/2	/Ø/d/	*saznati*
06.50.2	1/1 1/1+ 1/1+ — 1/3 — 1/1 1/1 2/2+ 2/2 3/2 3/1	/Ø/d/dn/	*dati*
07.50.3	1/1 1/1+ 1/1 — 1/3 — 1/1 1/1 2/2	/a/Ø/	*natkati*
08.50.2	1/1 1/2 1/2 — 1/2 — 1/2 1/2		*otresti*
09.50.2	1/1 2/2 2/2 — 2/2 — 2/2 2(1)/2	/Ø/t/	*zarasti*
10.50.2	1/1 2/2 2/2 — 2/2 — 2/2+ 1/1 1/1	/Ø/s/	*doneti*
11.50.3	1/1 2/2 2/2 — 2/2 — 2/2 2/2	/s/z/	*odgristi* *ozepsti*
13.50.3	1/1 2/2 2/2 — 2/2 — 2/2 3/1	/s/d/Ø/	*ukrasti* *izjesti*
13.51.3		/s/t/Ø/	*omesti*
14.50.3	1/1 2/2+ 2/2 — 2/2+ — 2/2+ 4/1 3/1 3/2 3/3	/s/d/dn/Ø/	*sesti*
17.50.3	1/1 2/2+ 2/2+ — 3/2 — 2/2+ 4/1 3/1 3/3 3/3+ 4/1	/s/t/tn/Ø/	*srèsti*
18.50.3	1/1 2/2+ 4/2+ — 2/2+ — 2/3+ 5/1 3/1 3/3 3/2 3/3	/s/n/dn/d/Ø/	*pasti*
19.50.3	1/1 2/2 2/2 — 2/2 — 2/2 3/3	/s/d/Ø/	*zabosti*
20.50.3	1/1 2/2 1/1 — 2/1 — 1/1 1/1	/a/Ø/	*zagrejati*
20.51.4		/va/j/	*popljuvati*
24.50.2		/Ø/j/	*piti*
21.50.3	1/1 2/2 1/1 — 2/2 — 1/1 1/1	/a/Ø/	*razorati*
21.51.3		/a/lj/	*nazobati*
21.52.4		/sa/š/	*zapisati*
21.53.4		/ha/š/	*odjahati*
21.54.4		/ga/ž/	*slagati*
21.55.4		/za/ž/	*povezati*
21.56.4		/ca/č/	*uzmicati*
21.57.4		/ka/č/	*zaplakati*
21.58.4		/ta/ć/	*otkakotati*
21.59.4		/da/đ/	*oglodati*
21.60.5		/ska/št/	*zaiskati*
21.61.5		/ska/šć/	*poljeskati*
21.62.5		/hta/hć/	*zadrhtati*
21.63.4		/la/olj/	*zaklati*
22.50.4		/ra/er/	*obrati*
22.51.4		/va/ov/	*odazvati*
22.52.5		/sla/šalj/	*poslati*
22.53.5		/gna/žen/	*izagnati*
23.50.3		/u/Ø/	*granuti* *vaskrsnuti* *omrznuti*
25.50.4		/le/elj/	*mleti*
27.50.3		/e/m/	*uzeti*
27.51.3		/u/p/	*zasuti*
27.52.3		/u/m/	*naduti*
27.53.3		/e/n/	*zapeti*
28.50.4		/le/un/	*prokleti*

Kod klase	Tip promene	Promenljivi deo	Primer
26.50.3	1/1 2/2 1/1 — 2/2 — 1/1+ 2/1 2/1	/e/∅/	*umreti*
29.50.2	1/1 2/2 1/1+ — 2/2 — 1/1 1/1 3/2	/∅/n/d/	*stati*
30.50.3	1/1 2/2+ 1/1+ — 2/2+ — 4/1+ 4/1 3/1 4/1 3/2 1/1	/e/nj/anj/nje/	*požeti*
31.50.3	1/1 2/3 1/1 — 2/2 — 1/1 1/1	/a/∅	*natrčati*
33.50.3		/e/∅/	*ogoreti* *naživeti*
33.51.3		/i/∅/	*zaželeti* *potrošiti* *izgubiti* *zapaliti*
32.50.3	1/1 2/3 1/1 — 2/1 — 1/1 1/1	/a/∅/	*prebrojati*
32.51.5		/aja/oj/	*ustajati*
36.50.3	1/1 2/4 1/1 — 2/2 — 1/1 1/1	/e/∅/	*dodreti*
50.51.2	1/2 2(3)/1 3/2 — 4/2 — 3/2 3/2	/∅/č/k/c/	*odseći*
50.52.2		/∅/ž/g/z/	*izleći*
51.50.2	1/2 2(3)/2+ 3/2 — 5/2+ — 3/2 3/2 4/2 4/2	/∅/č/k/kn/c/	*reći*
53.50.2	1/2 2/2 3(4)/2 — 5/2 — 3/2 3/2	/∅/gn/g/ž/z/	*pomoći*
55.50.2	1/2 2/2+ 2/2+ — 3/2+ — 5/2 5/2 3/2+ 3/2+ 4/2 4/2 4/2	/∅/č/k/kn/c/	*otići*
56.50.2	1/2 2/2 2/2 — 2/2 — 3/2 3/2	/∅/d/š/	*ući*
57.50.5	1/1+ 3/2 1/1+ — 3/2 — 1/1+ 1/1+ 2/2 4(5)/2 4/2 4/2	/gnu/∅/gn/g/ž/	*dignuti* *(dići)*
57.51.5		/knu/∅/kn/k/č/	*zataknuti* *(zataći)*

Tabela 5.5: Klasifikacija svršenih glagola — nastavak

Ovde predstavljena klasifikacija nastala je kao rezultat analize paradigmatskih svojstava 1927 glagola SJ koji se pojavljuju u tekstu poslovica iz [72] u postupku povezivanja elektronskog teksta sa elektronskim rečnikom o čemu će biti reči u glavi 6. Ulazi u DELAF za glagole dobijeni su primenom izgrađenih transduktora na klasifikovane glagole iz ove ekcerpirane grupe.

5.3.2 Imenske reči

Kôd morfološke klase imenskih reči koji se pridružuje odgovarajućim formalnim rečima u rečniku DELAF zasniva se na njihovoj klasifikaciji koja je opisana u [144]. Ova se klasifikacija razlikuje od opisane klasifikacije glagola po tome što se ona zasniva na određenom morfološkom računu nad formalnom reči. Jednom kodu morfološke klase glagola pridružuje se automat nalik na onaj prikazan na slici 5.1 pomoću koga se mogu generisati svi njegovi oblici. Morfološka klasa imenskih reči označava skup morfografemskih i morfološki uslovljenih informacija na osnovu koga se primenom morfološkog računa mogu generisti svi oblici polazeći od oblika odrednice. Skup morfografemskih binarnih veličina (u formi **tačno/netačno**), njih ukupno petnaest, precizira prisustvo ili odsustvo određene glasovne alternacije u imenskoj promeni. Te glasovne alternacije su, na primer, nepostojano *a* u nominativu singulara, palatalizacija, jednačenje po mestu tvorbe i druge. Na osnovu grafemskog sastava reči i pridruženog skupa morfografemskih informacija nije, u principu, moguće izračunati flektivnu

parametara, njih osamnaest za imenice a šesnaest za prideve, pomoću kojih se, uz primenu prisutnih alternacija mogu izračunati svi traženi oblici. Ti parametri za imenice su, recimo, gramatički rod, animatnost, egzistencija plurala, promena roda u pluralu i slično dok su karakteristične veličine za prideve egzistencija određenog vida, egzistencija komparativa, produženi nastavak za genitiv singulara, itd.

Kod klase	Tip promene	Promenljivi deo	Primer
01.25.4	1/1 2/1 1/1 1/1 2/1 2 1/1 1(3)/1	/je/ij/i/	*dospjeti*
25.25.5	1/1 2/2 1/1 2/2 2/2 2 1/1 1/1	/lje/elj/	*mljeti*
26.25.5	1/1 2/2 1/1 2/2 2/2 2 1/1+ 2/1 2/1	/ije/∅/	*mrijeti*
33.25.4	1/1 2/3 1/1 2/2 2/2 2 1/1 1(3)/1	/je/∅/i/	*gorjeti*
34.25.4	1/1 2/3 1/1 3/2 2/2 2 1/1 1(4)/1	/je/∅/lj/i/	*živjeti*
34.27.4		/lje/l/lj/li/	*željeti*
34.29.5		/nje/n/nj/ni/	*crvenjeti*
34.31.5		/zje/z/ž/zi/	*mrzjeti*
34.33.5		/tje/t/ć/ti/	*letjeti*
34.35.5		/dje/d/đ/di/	*bledjeti*
34.38.5		/snje/sn/šnj/sni/	*besnjeti*
37.25.4	1/1 2/3 1/1+ 3/2 3/2+ 2 1/1 1/1 3/2 2/2	/de/d/đ/	*videti*
50.26.4	1/2 2(3)/1 3/2 4/3+ 4/2 3 5/2 5/2 2/2	/je/iječ/ijek/ijec/jek/	*sjeći*
10.75.5	1/1 2/2 2/2+ — 2/2 — 2/2+ 1(3)/1 1/1 1/1	/ije/es/i/	*donijeti*
25.75.5	1/1 2/2 1/1 — 2/2 — 1/1 1/1	/lje/elj/	*samljeti*
26.75.5	1/1 2/2 1/1 — 2/2 — 1/1+ 2/1 2/1	/ije/∅/	*umrijeti*
33.75.4	1/1 2/3 1/1 — 2/2 — 1/1 1(3)/1	/je/∅/i/	*gorjeti* *živjeti* *pomrzeti*
34.77.5		/lje/l/li/	*zaželjeti*
34.79.5		/nje/n/ni/	*zacrvenjeti* *pobjesnjeti*
50.76.4	1/2 2(3)/1 3/2 — 4/2 — 5/2 5/2	/je/iječ/ijek/ijec/jek/	*odsjeći*

Tabela 5.6: Dodatna klasifikacija glagola ijekavskog izgovora

Morfološki račun, zasnovan na ovim informacijama može se ilustrovati na primeru imenice *kolevka* a detaljno je opisan u [139] i [144].

```
kolevka
1 2 2 0 1 1 1 0 0 0 0 0 0 1 1 2 0 0
F F F F F F T F F F F F F F
```

Za izračunavanje oblika formalne reči **kolevka** koriste se sledeći imenički parametri (navedeni u redu iza formalne reči): **kolevka** je imenica (1) ženskog roda (3) koja se menja po e-promeni (2), ima i singular (5) i plural (6) i ne menja rod u pluralu. Imenica ima atipičan nastavak *-i* u genitivu plurala za e-promenu (16). Od glasovnih alternacija na formiranje oblika utiče sibilarizacija (*k*, *g*, *h* ispred *i* prelaze redom u *c*, *z*, *s*), što je navedeno kao vrednost T (od TRUE) devetog pramatera u niza morfografemskih binarnih veličina (drugi red iza formalne

kolevka	kolevke
kolevke	kolevki
kolevci	kolevkama
kolevku	kolevke
kolevko	kolevke
kolevci	kolevkama
kolevkom	kolevkama

Promenom jednog imeničkog parametra (postavljanje tipičnog nastavka za genitiv plurala) i jedne morfografemske binarne veličine (nepostojano *a* u genitivu plurala) izračunava se drugi skup oblika u kome je genitiv plurala **kolevaka** (obe mogućnosti dozvoljavaju rečnici [117] i [116]).

Klasifikacija imenskih reči izvršena je na osnovu skupa morfološki uslovljenih parametara, analizom različitih realizovanih skupova. Ove morfološki uslovljene veličine grupisane su, za imenice, u pet opštih parametara: klasa, deklinacioni tip, rod, broj i infiks. Pobrojavanjem različitih kombinacija ovih veličina dobijene su klase imenica. Tako, na primer, klasu N17 čine opšte imenice muškog roda koje imaju množinu istog roda, obeležene su živo i petog su deklinacionog tipa, što znači da u nominativu singulara nemaju nastavak, da u genitivu singulara imaju nastavak *a*, u vokativu singulara nastavak *e*, u instrumentalu singulara nastavak *em* a u genitivu plurala nastavak *a*. Tipičan predstavnik ove klase je *šišmiš*. Svaka klasa se

podklasa	alternacija	primer
.11	palatalizacija	*mrtvac*
.61	nepostojanao *a* u ns	*jedinac*
	nepostojanao *a* u gp	*vranac*
	palatalizacija	*begunac*
.12	nepostojanao *a* u ns	*ubogac*
	nepostojanao *a* u gp	*vrabac*
	palatalizacija	*rožac*
	jednačenje po zvučnost	
.15	nepostojanao *a* u ns	*prasac*
	nepostojanao *a* u gp	*nosac*
	palatalizacija	
	jednačenje po mestu tvorbe	
.16	nepostojanao *a* u ns	*svetac*
	nepostojanao *a* u gp	*domorodac*
	palatalizacija	*sudac*
	gubljenje *t* ili *d*	*gladac*
.18	nepostojanao *a* u ns	*davalac*
	nepostojanao *a* u gp	*tužilac*
	prelazak ($l \rightarrow o$)	*ranoranilac*
	palatalizacija	

Tabela 5.7: Primer klasifikacije imenica na osnovu uključivanja različitih alternacija

dalje može razvrstavati prema vrednostima morfografemskih veličina koje su uključene pri-

se vidi da su u izračunavanju oblika imenica svake potklase uključene drukčije kombinacije alternacija.

Klasifikacija imenica mogla bi se sprovesti promenjujući isti postupak koji je korišćen kod klasifikacije glagola. Tada bi se, na primer, oblici imenice *šišmiš* mogli predstaviti sledećim transduktorom[7]:

$$šišmiš(\emptyset/_{ns} + a/_{gs,as,gp} + u/_{ds,ls} + e/_{vs,ap} + em/_{is} + i/_{np,vp} + ima/_{dp,lp,ip})$$

dok bi transduktori imenica *ubogac*, *vrabac* i *rožac* bili:

$$\begin{aligned} ubo \quad (\quad & gac/_{ns} + kca/_{gs,as,gp} + kcu/_{ds,ls} + gče/_{vs,ap} + \\ & gcem/_{is} + kci/_{np,vp} + gcima/_{dp,lp,ip}) \\ vra \quad (\quad & bac/_{ns} + pca/_{gs,as,gp} + pcu/_{ds,ls} + pče/_{vs,ap} + \\ & pcem/_{is} + pci/_{np,vp} + pcima/_{dp,lp,ip}) \\ ro \quad (\quad & žac/_{ns} + šća/_{gs,as,gp} + šcu/_{ds,ls} + gšče/_{vs,ap} + \\ & žcem/_{is} + šci/_{np,vp} + žcima/_{dp,lp,ip}) \end{aligned}$$

Na osnovu ovoga se može izvesti da je transduktor klase N17.12:

$$\begin{aligned} (K_1 a K_2/_{ns} + \mathcal{Z}(K_1,K_2)a/_{gs,as,gp} + \mathcal{Z}(K_1,K_2)u/_{ds,ls} + K_1\mathcal{P}(K_2)e/_{vs,ap} + \\ K_1K_2em/_{is} + \mathcal{Z}(K_1,K_2)i/_{np,vp} + K_1K_2ima/_{dp,lp,ip}) \end{aligned}$$

gde su K_1 i K_2 konsonanti a $\mathcal{Z}$ i $\mathcal{P}$ označavaju funkcije jednačenja po zvučnosti, odnosno, palatalizacije[8]. Iz ovoga se može zaključiti da jedna imenska klasa može da obuhvati više tražličitih transduktora (onih koji ne sadrže „promenljive"). S druge strane, imajući u vidu kako je sprovedena klasifikacija imenica, jedan transduktor može opisivati oblike imenica samo jedne klase.

Klasifikacija imenica data u [144], a ovde ukratko opisana zasniva se na pojmu *elementarne flektivne paradigme* $P(x)$ imenske reči x, pod čime se podrazumeva flektivna paradigma reči x koja se iz nje izračunava za određene fiksne vrednosti morfografemskih i morfološki uslovljenih veličina. Karakteristika elementarne flektivne paradigme je da u transduktoru koji je opisuje ne postoje dva izlaza sa istom vrednošću. Tako su navedene dve elementarne flektivne paradigme reči *kolevka*, N70.05 i N72.01, koje se razlikuju samo u obliku genitiva plurala. Kada varijacije morfografemskih i morfološki uslovljenih veličina nemaju distinktivnu ulogu, kao što je slučaj sa imenicom *kolevka*, onda se ove elementarne flektivne paradigme mogu okupiti u jednu složenu flektivnu paradigmu $S(x)$ koja se obrazuje kao unija odgovarajućih elementarnih paradigmi. Tada bi *kolevka* pripadala sledećoj složenoj klasi:

[7] *Transduktor* se koristi da označi konačni transduktor i regularni transduktorski izraz.

[8] Operacija jednačenja po zvučnosti može se ovako definisati:

$$\mathcal{Z}(K_1,K_2) = \begin{cases} K_1^z K_2, & \text{ako je } K_2 \in \{b,g,d,dj,ž,z,dž\} \\ K_1^b K_2, & \text{ako je } K_2 \in \{p,k,t,ć,š,s,č\} \\ K_1K_2, & \text{za ostale suglasnike} \end{cases}$$

Operacija palatalizacije može se ovako definisati:

$$\mathcal{P}(K) = \begin{cases} K^{pn}, & \text{ako je } K \in \{k,g,h\} \text{ (zadnjenepčani)} \\ K, & \text{za ostale suglasnike} \end{cases}$$

$$kolev(ka/_{ns}+ke/_{gs,np,ap,vp}+ci/_{ds,ls}+ku/_{as}+ko/_{vs}+kom/_{is}+ki/_{gp}+kama/_{dp,is,lp}+aka/_{gp})$$

Treba odmah uočiti da nije moguće proširivanje svih imenica iz klase N70.05 ili N72.02 u ovu opštu klasu: tako *igračka* iz klase N70.05 prema rečnicima [117] i [116] ne može imati oblik genitiva plurala **igrački* dok *vojska* iz klase N72.02 ne može imati oblik genitiva plurala **vojsaka*.

U nekim slučajevima različite vrednosti nekih morfografemskih i morfološki uslovljenih veličina imaju distinktivnu ulogu pa imenice koje pripadaju različitim klasama predstavljaju na semantičkom planu različite odrednice [143]. Tako *drvo* u značenju *biljka* ima genitiv singulara *drva* i *drveta* i pripada klasama N51.01 i N52.01 dok *drvo* u značenju *sasečenog drveta* ima genitiv singualara samo *drva* i pripada klasi N51.01. Drugi primer je imenica *otac* koja gradi tri elementarne flektivne paradigme karakterisane oblicima nominativa plurala *oci*, *ocevi* i *očevi* i stoga pripada trima klasama: N17.16, N19.15 i N19.16. U sintagmama, *gradski oci* i *duhovni oci* koristi se, međutim, samo imenica *otac* iz klase N17.16 dok se u značenju roditelja u savremenom jeziku koristi samo imenica *otac* iz klase N19.16.

Spajanje elementarnih flektivnih paradigmi u složene paradigme se, stoga, ne može obaviti na nivou klasa već za svaku imensku reč pojedinačno. Imenske reči su u rečniku DELA za SJ klasifikovane u skladu sa njihovim elementarnim flektivnim paradigmama a njihovo okupljanje sledi tek posle potvrde različitih oblika na korpusu.

Ovde predstavljena klasifikacija primenjena je na 2717 imenica i 630 prideva koji se javljaju u tekstu poslovica iz [72]. Ulaz u DELAF imenskih reči dobijen je pridruživanjem mofografemskih svojstava i morfološki uslovljenih veličina izdvojenim imenicama i pridevima na osnovu kojih su primenom programa MORF generisani oblici njihove flektivne paradigme [138] i [140].

5.3.3 Zamenice

Prema službi u rečenici, zamenice mogu biti *imeničke* i njima se u DELAS-u za SJ dodeljuje kôd vrste reči ProN i *pridevske* kojima se dodeljuje kôd reči ProA. Zamenice su u SJ promenljiva vrsta reči te se i one moraju na neki način klasifikovati. Za potrebe izrade rečnika DELA zamenice su klasifikovane koristeći isti pristup koji je korišćen i za klasifikaciju glagola a to je pobrojavanje različitih transduktora kojima se mogu generisati i prepoznavati različiti oblici zamenica. Svi transduktori zamenica dati su u tabeli 5.8. Neki od njih opisuju promenu samo jedne zamenice (na primer, transduktor ProN01 opisuje samo promene lične zamenice *ja*) dok neki opisuju promene više njih (na primer, ProN12 opisuje promenu upitne zamenice *ko* i neodređenih zamenica *neko*, *niko*, *iko*).

U DELAF-u se oblici zamenica opisuju istim gramatičkim kodom kojim se opisuju i imenice. Gramatički kôd često sadrži džokerski simbol *: na primer, jedan oblik lične zamenice svakog lica *sebe* koja se koristi za sva lica i oba broja u DELAF-u je predstavljen sa

```
sebe,.ProN11:**g*:**a*
```

U DELAS-u za SJ predstavljene su samo one zamenice koje mogu biti predstavljene kao formalna reč sastavljena od slovnih karaktera. Tako se u DELAS-u nalazi zamenica *kogod* ali ne i *ko god*, *ma ko* i slično. Kod prepoznavanja zamenica u tekstu, poseban slučaj koji se ne može obraditi korišćenjem samo DELAF-a predstavljaju zamenice sastavljene od *i* i *ni* i upitnih zamenica *ko* i *šta*. Naime, predlog se ne stavlja ispred cele zamenice već između

sastavnih delova: *ni od koga*, *ni u čemu* i slično, što zahteva prepoznavanje zamenica u lokalnom okruženju.

Klasa	primer	transduktor
ProN01	ja	$ja_{sn} + me(\emptyset/_{sg,sa} + ne/_{sg,sa} + ni/_{sd,sl}) + mnom(\emptyset/_{si} + e/_{si}) + mi/_{sd}$
ProN02	t(i)	$i/_{sn,sd,sv} + e(\emptyset/_{sg,sa} + be/_{sg,sa} + bi/_{sd,sl}) + obom/_{si}$
ProN03	mi	$mi/_{pn} + na(s/_{pg,pa} + m(\emptyset/_{pd} + a/_{pd,pi,pl}))$
ProN04	v(i)	$i/_{pn} + a(s/_{pg,pa} + m(\emptyset/_{pd} + a/_{pd,pi,pl}))$
ProN05	on	$on/_{msn} + nj(\emptyset/_{msa} + e(ga/_{msg,msa} + m(\emptyset/_{msl} + u/_{msd,msl})) +$ $im(\emptyset/_{msi} + e/_{msi}) + mu/_{msd*} + ga/_{msg,msa}$
ProN06	ono	$ono/_{nsn} + nj(\emptyset/_{nsa} + e(ga/_{nsg,nsa} + m(\emptyset/_{nsl} + u/_{nsd,nsl})) +$ $im(\emptyset/_{nsi} + e/_{nsi}) + mu/_{nsd*} + ga/_{nsg,nsa}$
ProN07	ona	$ona/_{fsn} + nj(e/_{fsg} + o(j/_{fsd,fsl} + jzi/_{fsd,fsl}) + u/_{fsa} + om(\emptyset/_{fsi} + e/_{fsi})) +$ $j(e/_{fsg,fsa} + oj/_{fsd} + u/_{fsa})$
ProN08	oni	$oni/_{pmn} + nji(h/_{msg,mpa} + ma/_{mpd,mpi,mpl}) + i(h/_{mpg,mpa} + m/_{mpd})$
ProN09	ona	$ona/_{npn} + nji(h/_{nsg,npa} + ma/_{npd,npi,npl}) + i(h/_{npg,npa} + m/_{npd})$
ProN10	one	$one/_{fmn} + nji(h/_{fsg,fpa} + ma/_{fpd,fpi,fpl}) + i(h/_{fpg,fpa} + m/_{fpd})$
ProN11	s(ebe)	$e(\emptyset/_{g,a} + be/_{g,a} + bi/_{d,l}) + i/_{d}$
ProN12	ko	$ko(\emptyset/_{n} + g(\emptyset/_{g,a} + a/_{g,a}) + m(\emptyset/_{d,l} + e/_{d,l} + u/_{d}) + kim(\emptyset/_{i} + e/_{i})$
ProN13	što	$što/_{n,a} + če(g(\emptyset/_{g} + ga/_{g}) + m(\emptyset/_{l} + u/_{d,l})) + čim(\emptyset/_{i} + e/_{i})$
ProA01	ov(aj)	$a(\emptyset/_{fsn,npn,npa} + j/_{msn,msa-}) + o(\emptyset/_{nsn,nsa} + g(\emptyset + a)/_{msg,nsg,msa+} +$ $m(\emptyset/_{msd,msl,nsd,nsl,fsi} + (e + u)/_{msd,msl,nsd,nsl}) + j/_{fsd,fsl}) + u/_{fsa} +$ $e/_{fsg,mpa,fpn,fpa} + i(\emptyset/_{mpn} + h/_{mpg,npg,fpg} + m(\emptyset/_{msi,nsi,pd,pi,pl} + e/_{msi,nsi} +$ $a/_{pd,pi,pl}))$
ProA02	njen	$\emptyset/_{msn,msa-} + o(\emptyset/_{nsn,nsa} + g(\emptyset + a)/_{msg,nsg,msa+} +$ $m(\emptyset/_{msd,msl,nsd,nsl,fsi} + (e + u)/_{msd,msl,nsd,nsl}) + j/_{fsd,fsl}) +$ $u/_{msd,msl,nsd,nsl,fsa} + e/_{fsg,mpa,fpn,fpa} + i(\emptyset/_{mpn} + h/_{mpg,npg,fpg} +$ $m(\emptyset/_{msi,nsi,pd,pi,pl} + e/_{msi,nsi} + a/_{pd,pi,pl})) + a/_{msg,nsg,fsn,npn,npa}$
ProA03	nek(i)	$o(\emptyset/_{nsn,nsa} + g(\emptyset + a)/_{msg,nsg,msa+} + m(\emptyset/_{msd,msl,nsd,nsl,fsi} +$ $(e + u)/_{msd,msl,nsd,nsl}) + j/_{fsd,fsl}) + u/_{msd,msl,nsd,nsl,fsa} +$ $e/_{fsg,mpa,fpn,fpa} + i(\emptyset/_{msn,msa-,mpn} + h/_{mpg,npg,fpg} + m(\emptyset/_{msi,nsi,pd,pi,pl} +$ $e/_{msi,nsi} + a/_{pd,pi,pl})) + a/_{msg,nsg,fsn,npn,npa}$
.01	kak(av)	$a \leftarrow \emptyset$
ProA04	naš	$\emptyset/_{msn,msa-} + e(\emptyset/_{nsn,nsa,fsg,mpa,fpn,fpa} + g(\emptyset + a)/_{msg,nsg,msa+} +$ $m(\emptyset + u)/_{msd,msl,nsd,nsl}) + oj/_{fsd,fsl}) + u/_{fsa} +$ $om/_{fsl} + i(\emptyset/_{mpn} + h/_{mpg,npg,fpg} + m(\emptyset/_{msi,nsi,pd,pi,pl} + a/_{pd,pi,pl} +$ $a/_{pd,pi,pl})) + a/_{fsn,npn,npa}$
.01 ProA05	s(av) čij(i)	$a \leftarrow \emptyset$ $e(\emptyset/_{nsn,nsa,fsg,mpa,fpn,fpa} + g(\emptyset + a)/_{msg,nsg,msa+} +$ $m(\emptyset + u)/_{msd,msl,nsd,nsl}) + oj/_{fsd,fsl}) + u/_{fsa} +$ $om/_{fsl} + i(\emptyset/_{msn,msa-,mpn} + h/_{mpg,npg,fpg} + m(\emptyset/_{msi,nsi,pd,pi,pl} + a/_{pd,pi,pl} +$ $a/_{pd,pi,pl})) + a/_{fsn,npn,npa}$
ProA06	m(oj)	$oj(\emptyset/_{msn,msa-} + e(\emptyset/_{nsn,nsa,fsg,mpa,fpn,fpa} + g(\emptyset + a)/_{msg,nsg,msa+} +$ $m(\emptyset + u)/_{msd,msl,nsd,nsl} + oj/_{fsd,fsl} + u/_{fsa} +$ $om/_{fsl} + i(\emptyset/_{mpn} + h/_{pg} + m(\emptyset/_{msi,nsi,pd,pi,pl}) + a/_{pd,pi,pl}) +$ $a/_{pd,pi,pl})) + a/_{fsn,npn,npa}) +$ $og(\emptyset + a)/_{msg,msa+,nsg} + om(\emptyset + e + u)/_{msd,msl,nsd,nsl}$
ProA06	k(oji)	$e(\emptyset/_{nsn,nsa,fsg,mpa,fpn,fpa} + g(\emptyset + a)/_{msg,nsg,msa+} +$ $m(\emptyset + u)/_{msd,msl,nsd,nsl} + oj/_{fsd,fsl} + u/_{fsa} +$ $om/_{fsl} + i(\emptyset/_{msn,msa-,mpn} + h/_{pg} + m(\emptyset/_{msi,nsi,pd,pi,pl}) + a/_{pd,pi,pl}) +$ $a/_{pd,pi,pl})) + a/_{fsn,npn,npa}) +$ $og(\emptyset + a)/_{msg,msa+,nsg} + om(\emptyset + e + u)/_{msd,msl,nsd,nsl}$

Tabela 5.8: Transduktori za opis oblika zamenica

5.3.4 Brojevi

Brojevi kao vrsta nesamostalnih reči koje označavaju koliko ima onoga što znače reči uz koje stoje, dele se na osnovne, redne i zbirne kojima se u rečniku DELA dodeljuju redom kodovi vrste reči Num, Ord i Col. Brojevima koji se menjaju dodeljuje se kôd klase kao oznaka transduktora koji njihovu promenu opisuje (videti 5.9).

Klasa	primer	transduktor
Num01	jed(an)	$an/_{msn,msa-,msv} + no(\emptyset/_{nsn,nsa,nsv} + g(\emptyset + a)/_{msg,nsg,msa+} + m(\emptyset/_{msd,msl,nsd,nsl,fsi} + (e+u)/_{msd,msl,nsd,nsl}) + j/_{fsd,fsl}) + nu/_{fsa} + e/_{fsg,mpa,fpn,fpa,fpv} + ni(\emptyset/_{mpn,mpv} + h/_{mpg,npg,fpg} + m(\emptyset/_{msi,nsi,pd,pi,pl} + +a/_{pd,pi,pl})) + na/_{msg,nsg,fsn,npn,npa}$
Num02	dv(a)	$a(\emptyset/_{mpn,mpa,mpv,npn,npa,npv} + ju/_{mpg,npg} + ma/_{mpd,mpi,mpl,npd,npi,npl}) + e(\emptyset/_{fpn,fpa,fpv} + je/_{fpg} + ma/_{fpd,fpi,fpl})$
Num03	tri	$\emptyset/_{pn,pa,pv} + i(ju/_{pg} + ma/_{pd,pi,pl})$
Num04	pet	*nepromenljivo*
Num05	nul(a)	$a(\emptyset/_{fsn,fsv,fpg} + ma/_{fpd,fpi,fpl}) + e/_{fsg,fpn,fpa,fpv} + i/_{fsd,fsl} + o(\emptyset/_{fsv} + m/_{fsi}) + u/_{fsa}$
Num06	milion	$\emptyset/_{msn,msa,msv} + a/_{msg,mpg} + e/_{mpa} + om/_{msi} + u/_{msd,msl} + i(\emptyset/_{mpn,mpv} + ma/_{mpd,mpi,mpl})$
Num07	prv(i)	$i(\emptyset/_{msn,msa-,msv,mpn,mpv} + h/_{pg} + m(\emptyset/_{msi,nsi,pd,pi,pl} + a/_{pd,pi,pl} + a/_{fsn,fsv,npn,npa,npv} + e/_{fsg,mpa,fpn,fpa,fpv} + o(\emptyset/_{nsn,nsa,nsv} + (\emptyset + a)/_{msg,msa,nsg}) + j/_{fsd,fsl} + m(\emptyset/_{msd,msl,nsd,nsl,fsi} + (e+u)/_{msd,msl,nsd,nsl})) + u/_{fsa}$
Num08	treć(i)	$i(\emptyset/_{msn,msa-,msv,mpn,mpv} + h/_{pg} + m(\emptyset/_{msi,nsi,pd,pi,pl} + a/_{pd,pi,pl}) + a/_{fsn,fsv,npn,npa,npv} + e(\emptyset/_{npn,npv,fsg,mpa,fpn,fpa,fpv} + (g+ga)/_{msg,msa+,nsg}) + (m+mu)/_{msd,msl,nsd,nsl}) + o(j/_{fsd,fsl} + m/_{fsi}) + u/_{fsa}$
Num09	dvo(je)	$j(e(\emptyset/_{sn,sa,sv,mpa,fpn,fpa} + g(\emptyset + a)/_{sg} + m(\emptyset + u)/_{sd,si,sl}) + a/_{npn,npa} + ga/_{sg} + i(\emptyset/_{mpn} + m/_{pd,pi,pd} + h/_{pg})) + (g+ga)/_{sg} + (m + ma + me)/_{sd,si,sl}$
Num10	četvor(o)	$o(\emptyset/_{sn,sa,sv,mpa,fpa} + ma/_{sd} + me/_{sd,sl} + a/_{npn,npa} + ga/_{sg} e/_{mpa,fpn,fpa} + i(\emptyset/_{mpn} + m/_{pd.pi.pl} + h/_{pg}) + (ma + me + mu)/_{sd,si,sl}$
Num11	četver(o)	$o(\emptyset/_{sn,sa,sv,mpa,fpa} + ma/_{sd,si,sl} + a/_{npn,npa} + ga/_{sg} e/_{mpa,fpn,fpa} + i(\emptyset/_{mpn} + m/_{pd.pi.pl} + h/_{pg}) + mu/_{sd,si,sl}$
N75.99	petorica	$\emptyset/_{msn} + e/_{msg} + i/_{msd,msl} + om/_{msi} + u/_{msa}$
N88	desetak	*nepromenljivo*

Tabela 5.9: Transduktori za opis oblika brojeva

Redni brojevi imaju oblike za sva tri roda i za oba broja. Od osnovnih brojeva, samo broj *jedan* ima oblike za sva tri roda i oba broja, dok broj *dva* ima iste oblike za muški i srednji rod, a broj *tri* iste oblike za sva tri roda. Brojevi *dva* i *tri* imaju samo oblike nekadašnje dvojine. Brojevi veći od *četiri* su nepromenljivi. Brojevi kao što su *hiljada* i *milion* menjaju se kao imenice ženskog, odnosno, muškog roda i predstavljaju brojeve samo po funkciji u jeziku. Zbirni brojevi imaju u jednini iste oblike za sva tri roda dok se u množini ti oblici razlikuju. Gramatičke informacije nekih karakterističnih ulaza u DELAF-u izgledaju, stoga, ovako:

```
jednog,jedan.Num01*:msg*:msa+:nsg*          Različiti oblici za sva tri roda i oba broja
dvama,dva.Num02E:mpd*:mpi*:mpl*:npd*:npi*:npl*
dve,dva.Num02E:fpn*:fpa*:fpv*
trima,tri.Num03*:*pd*:*pi*:*pl*                    Isti oblici za sva tri roda
```

```
dvadeset,.Num04*                                    Brojevi > 5 su nepromenljivi
nulu,nula.Num05*:fsa*                               Kao imenice klase N70.01
stotina,.Num05*:fsn*:fsv*:fpg*
milionu,milion.Num*06:msd*:msl*                     Kao imenice klase N04.01
bilion,.Num*06:msn*:msa*:msv*
prvu,prvi.Ord07*:fsa*                  Redni brojevi, oblici za sva tri roda i oba broja
drugi,.Ord07*:msn*:msa-:msv*:mpn*:mpv*
treću,treći.Ord08*:fsa*
dvoji,dvoje.Col09*:mpn*          Zbirni brojevi, različiti oblici za sva tri roda u množini
dvoje,.Col09*:*sn*:*sa*:*sv*:mpa*:fpn*:fpa*
četvore,četvoro.Col10*:mpa*:fpn*:fpa*
četvori,četvoro.Col10*:mpn
petorica,.NumN75.99*:msn+                           Brojčana imenica, promenljiva
petorice,petorica.NumN75.99*:msg+
dvadesetak,.NumN88*                                 Brojčana imenica, nepromenljiva
```

Kao što se vidi iz tabele 5.9, među brojeve su uključene i brojčane imenice sa sufiksom *-ica* koje se menjaju kao imenice klase N75.99 (imenice muškog roda koje se završavaju na *-a* i koje nemaju oblik vokativa ni oblike množine) i brojčane imenice sa sufiksom *-ak* koje su nepromenljive i pripadaju klasi N88. U sastav rečnika DELA ušli su samo oni složeni brojevi koji se prema Pravopisu [107] pišu zajedno (*dvanaest, dvesta*) a ne i višečlani brojevi koji se pišu odvojeno (*dvadest jedan, dve stotine*).

5.4 Normalizacija varijacija u ulazima rečnika DELA

Kada je u pitanju izbor ulaza u rečnik DELA, premda se oni suštinski razlikuju od tradicionalnih rečnika, oslanjaju se na njih u velikoj meri. U rečniku DELA se reprodukuju sve varijacije odrednica zabeležene u tradicionalnim rečnicima, a koje odražavaju nestabilnu (ili promenljivu) pravopisnu normu, varijacije u preuzimanju stranih reči i razlike u izgovoru. Takve varijacije su prisutne u većini evropskih jezika (na primer, varijacije koje postoje između britanskog i američkog engleskog jezika). Varijacije u odrednicama francuskog jezika opisane su u [47] i one potiču, prvenstveno, od nedovoljno precizne pravopisne norme te od preuzimanja stranih reči, što ilustruju naredna dva primera:

`Moyen('-' + ' ')(A + Â + â)ge + moyen('-' + ' ')âge` (srednji vek)	→	*Moyen-Age, Moyen Age Moyen-Âge, Moyen Âge Moyen-âge, Moyen âge moyen-âge, moyen âge*
`(k(ach + ash) + cash)(ère + er)` (košer)	→	*kachère, kacher kashère, kasher cashère, casher*

Slični izvori varijacija odrednica prisutni su i u SJ, što ilustruju naredni primeri nastali

`patišpa(lj + n + nj(a + ∅))`	→	*patišpalj m., patišpan m., patišpanja ž., patišpanj m.*
`patrijar((a + ∅) + ak + (a + ∅)h + ha)`	→	*patrijara,-rka m.; patrijar,-ara m. patrijarak,-rka m.; patrijarah,-rha m. patrijarh,-arha m.; patrijarha m.*
`baba('-' + ∅)zeman`	→	*baba-zeman, babazeman*

Kao što se iz ovih primera vidi, ovakve varijacije odrednica mogu se opisati regularnim izrazima i u tekstu prepoznavati odgovarajućim konačnim automatima.

Međutim, najveći broj izvora varijacija u SJ potiču od glasovnih promena, pri čemu su najčešće razlike u izgovorima koje su nastale na osnovu zamene starog samoglasnika „jat" (*ě*) samoglasnikom *e* u ekavici, samoglasnikom *i* u ikavici i zamenom glasovnim grupama *ije* i *je* u ijekavici[9]. Prema [107], odnos između ekavskih i ijekavskih zamena jata može se ovako sistematizovati:

- „neutralizovano jat" to jest odnos *e* – *e*, što znači da je i u ijekavskom izgovoru jat dalo *e* (u kratkim slogovima iza konsonantske grupe koja se završava sa *r*). Ova zamena nekad obuhvata sve oblike reči, na primer, *greška* i *breza* dok nekada obuhvata samo neke oblike reči, na primer, *brijeg, bregovi* i *ždrijebe, ždrebeta.* Neutralizovano jat se ponekad ogleda i kroz odnos *i* – *i*, kada se i u ekavici i u ijekavci jat razvilo u *i* (pod uticajem narednog *j*). Na primer, infiks komaprativa *-ij* u *noviji, moderniji* i infiks imperfekta *-ij* u *pletijah, tresijah.*
- dugo jat, to jest odnos *e* – *ije* (u ekavici je dugi slog a u ijekavici obično dva kratka). Na primer, takva je zamena u *grešiti* – *griješiti, dete* – *dijete, lep* – *lijep.*
- kratko jat, to jest odnos *e* – *je* (i u ekavici i u ijekavici je kratak slog). Takav je slučaj sa: *vera* – *vjera, vešt* – *vješt* i *pevati* – *pjevati.*
- produženo jat, to jest odnos *e* – *je* (i u ekavici i u ijekavici je kratak slog koji zahvati duljenje). Ovo je sporadična zamena koja je vezana za posebne uslove i pozicije. Takav je slučaj u: *ìzmeriti* – *izmjeriti* i *mèdved* – *mèdvjed* – *medvjēda.*
- suženo jat, to jest zamena *e* – *i* (kratko jat koje se u ijekavici zamenjuje zatvorenijim samoglasnikom *i* ispred *j* i ispred *o* nastalog od *l*). Na primer, *grejati* – *grijati* i *deo* – *dio, živeo* – *živio.*

Sa stanovišta opisa varijacija u SJ nisu od značaja zamene koje ne dovode do razlikovanja u izgovoru (na primer, neutralizovano jat). Takođe nije od značaja ni razlikovanje kratkog i produženog jata jer oni ne proizvode razlike koje se beleže u pisanom tekstu.

Kao što je i u [107] rečeno, realni odnosi u zameni jata su mnogo složeniji nego što je u ovakvoj sistematizaciji izloženo. To se, pre svega odnosi, na neujednačenost u primeni ijekavskih zamena jata što postepeno dovodi do diferencijacije ijekavice na „istočnu" (ili „južnu") i „zapadnu". Tako nastaju leksički dubleti u ijekavskoj zameni jata tipa: *zaliv* – *zaljev, uticaj* – *utjecaj, prelom* – *prijelom* i *sjedjeti* – *sjediti.* Treba uočiti da se varijante istočne ijekavice: *zaliv, uticaj* i *prelom* poklapaju sa odgovarajućom ekavskom varijantom. U ijekavici često dolazi do jotovanja suglasnika koji prethodi, premda se jotovanje u književnom jeziku

[9] Srpski književnojezički izraz obuhvata ekavicu i ijekavicu dok se hrvatski književni jezik zasniva na ijekavici. Niiedna književna standardizacija na srpskohrvatskom jezičkom prostoru nije obuhvatila ikavicu.

najčešće ne dopušta. Na nekoliko primera, ekscerpiranih iz [117] biće izložena kompleksnost zamene jata u realnoj jezičkoj praksi[10]. U tabeli je znakom pitanja označeno da o paradigmi određene varijante u rečniku nije ništa rečeno.

ekavski	živeti	živim, živeh, življah, živeo
ijekavski	živjeti	živim, živjeh, življah, živio
ikavski i	živiti	?
neknjiževno ekavski i		
neknjiževno ijekavski		
dijalekatski ijekavski	življeti	?
ekavski	kolevka	gp. kolevki, kolevaka
ijekavski	kolijevka	gp. kolijevki, kolijevaka, koljevaka
ikavski	kolivka	?
neknjiževno ijekavski	koljevka	?
ekavski	besneti	besnim, besneh, bešnjah, besneo
ijekavski	bjesnjeti	bjesnim, bjesnjeh,, bješnjah, bjesnio
ikavski	bisniti	?
neknjiževno ekavski	besniti	besnim, besnih, bešnjah, besnio
neknjiževno ijekavski	bjesniti	bjesnim, bjesnih, bješnjah, bjesnjeo (?)
	bješnjeti	bjesnim, bješnjeh, bješnjah, bješnjeo
	biješnjeti	bijesnim, biješnjeh, biješnjah, biješnjeo
	bijesnjeti	bijesnim, bijesnjeh, biješnjah, bijesnjeo
	bijesniti	bijesnim, bijesnih, biješnjah, bijesnio

Zamena jata nije jedini izvor varijacija u SJ. Drugi izraženi izvor varijacija je suglasnik h, njegovo izostavljanje ili zamena drugim suglasnicima. Premda je pravilo književnog jezika da se suglasnik h održava svugde gde je izvorno postojalo, za mnoge reči više ne postoji svest o tome, pa se ono izostavlja ili zamenjuje, najčešće sa j ili v. Tako nastaju sledeće varijacije od kojih sve ne pripadaju književnom jeziku:

h + ∅	$(h+\emptyset)at$, $(h+\emptyset)alva$, $gre(h+\emptyset)ota$, $(h+\emptyset)ladan$
h + v	$du(h+v)ati$, $mu(h+v)a$, $glu(h+v)$
h + j	$ki(h+j)ati$, $pro(h+j)a$
h + j + ∅	$me(h+j+\emptyset)ana$
h + j + v	$ažda(h+j+v)a$
(h + ∅)v + f	$((h+\emptyset)v+f)atati$, $za((h+\emptyset)v+f)ala$

Osim ovih, prisutne su i mnoge druge varijacije od kojih su samo neke priznate u književnom jeziku:

[10] U prikazanoj tabeli, oblici glagola su interpretacija autora. U rečniku su oni najčešće dati kao uređene liste oblika prezenta, imperfekta itd. bez jasne naznake koji od tih oblika pripada kojoj klasi.

i + ∅	$bož(i+\emptyset)ji$, $vraž(i+\emptyset)ji$
j + ∅	$ašči(j+\emptyset)nica$, $iz(j+\emptyset)elica$
j + v	$dobi(j+v)ati$
a + ∅	$afek(a+\emptyset)t$, $objek(a+\emptyset)t$
l + ∅	$raso(l+\emptyset)$, $sto(l+\emptyset)$, $bari(l+\emptyset)o$
o + l	$dě(o+l)$, $bě(o+l)$
f + v	$Či(f+v)utin$
o + e	$gospodar(o+e)v$
e + ∅	$zatr(e+\emptyset)ti$

5.4.1 Leksikografema kao sredstvo za opisivanje glasovnih varijacija

Varijacije prouzrokovane glasovnim promenama mogu se takođe opisati regularnim izrazima. Tako bi se, na primer, sve varijacije odrednica *babadevojka*, *hleb* i *besneti* mogle opisati sledećim regularnim izrazima:

`baba('-' + ∅)(d(e+je+i)+đe)vojka`	→	*baba-devojka, baba-djevojka, baba-divojka, baba-đevojka babadevojka, babadjevojka, babadivojka, babađevojka*
`(h + ∅)(l + lj)eb`	→	*hleb, hljeb, leb, ljeb*
`b(esn(e + i) + isni + ijes(nje + ni + šnje) + jes(nje + sni +šnje))ti`	→	*besneti, besnjiti bisniti, bijesnjeti bijesniti, biješnjeti bješnjeti, bjesniti, bješnjeti*

Predstavljanje pomoću regularnih izraza svih varijacija izazvanih glasovnim promenama u SJ ima više nedostataka. Ove varijacije, a posebno one prouzrokovane zamenom jata, su česte pa bi bio potreban veliki broj regularnih izraza za njihovo opisivanje. Osim toga, ovakvim izrazima ne mogu se sistematski opisati te promene. Naime, veliki broj zamena jata se sistematski može opisati. Jedna takva zamena je, na primer:

ekavski	$ě \rightarrow e$
ijekavski	$ě \rightarrow je$
ikavski	$ě \rightarrow i$
dijalekatski ijekavski	$Kě \rightarrow \mathcal{J}(K)e$

U tabeli K označava konsonant a $\mathcal{J}(K)$ operaciju jotovanja[11]. Ova zamena jata opisuje varijacije u: *devojka*, *dever*, *medved*, *nedelja*, *denuti*, *naterati*, *mesec* i mnoge druge.

Za opis ovakvih grafemskih varijacija u SJ predlaže se uopštenje pojma grafeme, *leksikografema*, koja predstavljaju apstraktnu jedinicu sistema kodiranja rečnika [146], [84]. Leksikografema ima sledeće osobine: u tekstu se ona realizuje kao jedan grafijski znak, niska grafijskih znakova ili kao prazna niska pri čemu izborom određene realizacije upravlja skup atributa i neobaveznih operatora.

[11] Operacija jotovanja može se ovako definisati:

$$\mathcal{J}(K) = \begin{cases} K\text{lj}, & \text{ako je } K \text{ usneni nenepčani suglasnik} \\ \text{prednjonepčani alternant}, & \text{ako je } K \text{ nenepčani suglasnik} \\ K, & \text{za ostale suglasnike} \end{cases}$$

Ako malim slovima grčkog alfabeta označimo leksikografeme, onda se sve varijacije odrednica *hleb* i *besneti* mogu u rečniku predstaviti odrednicama $\alpha l\beta b$ i $b\gamma sn\delta ti$. Treba uočiti da je $\beta \neq \gamma$ i $\gamma \neq \delta$ i $\delta \neq \beta$ premda sve tri leksikografeme u ekavici imaju istu grafijsku reprezentaciju u tekstu: *e*. Ove leksikografeme mogu se opisati sledećim grafijskim reprezentacijama, atributima i operatorima:

α	→	*h*	
	→	$\emptyset$	dijalektizam
β	→	*e*	ekavski
	→	*e*, jotovanje	ijekavski
γ	→	*e*	ekavski
	→	*je*	ijekavski
	→	*i*	ikavski
	→	*ije*	ijekavski, neknjiževno
δ	→	*e*	ekavski
	→	*e*, jotovanje	ijekavski
	→	*i*	ikavski
	→	*i*	ekavski, neknjiževno
	→	*i*	ijekavski, neknjiževno
	→	*e*, jotovanje, jednačenje po mestu	ijekavski, neknjiževno

Definisanje leksikografeme pomoću atributa omogućava sistematsko opisivanje svih varijanti. Tako rečnička odrednica $b\gamma sn\delta ti$ ne uključuje kao moguću varijantu koju rečnik [117] ne beleži **bešnjeti* jer realizaciju leksikografeme γ kao grafijskog znaka *e* omogućava atribut „ekavski" dok realizaciju leksikografeme δ kao grafijskog znaka *e* uz jotovanje prethodnog suglasnika i jednačenje po mestu tvorbe omogućava atribut „ijekavski, neknjiževno".

Neki atributi se mogu definisati kao *inkluzivni* što znači da za realizaciju varijante koja sadrži više leksikografema nije potrebno doslovno slaganje atributa, već jedan atribut može podrazumevati drugi. Na gornjem primeru, varijantu *bješnjeti* realizuje u slučaju leksikografeme γ atribut „ijekavski" a u slučaju leksikografeme δ atribut „ijekavski, neknjiževno" (realizovana varijanta se, naravno, određuje kao neknjiževna ijekavska varijanta). U ovom slučaju je atribut „ijekavski" inkluzivan i uključuje atribut „ijekavski, neknjiževni".

Sličan je slučaj i sa leksikografemama α i β. U ovom slučaju se atribut „dijalektizam" leksikografeme α može slobodno kombinovati sa atributima „ijekavski" i „ekavski" leksikografeme β čime rečnički oblik $\alpha l\beta b$ opisuje četiri varijante — sve moguće kombinacije grafijskih reprezentacija.

Korišćenjem leksikografeme okupljaju se pod jednu odrednicu varijante koje su sve opisane istom morfografemskom klasom [147]. Na primer, rečničkom obliku $glu\varphi$ pridružuje se morfografemska klasa A07.01, gde je φ leksikografema koja se realizuje bilo kao grafijska niska *v* bilo kao grafijska niska *h*, bez posebne naznake atributa (u rečnicima [117] i [116] su ta dva oblika opisana sa **gluv = gluh**). Klasa A07.01 obuhvata prideve koji imaju i određeni i neodređeni vid, koji imaju komparativ koji se gradi pomoću infiksa *j* i koji jotuje suglasnik koji mu prethodi (tako je komparativ prideva *gluv, gluvlji* a komparativ prideva *gluh, glušći*).

Da bi leksikografema u potpunosti opisivala njome obuhvaćene varijante, potrebno je precizirati i njeno ponašanje i evetualne različite realizacije u oblicima paradigme. Ovo je od značaja samo za leksikografeme koje opisuju zamenu jata u ijekavici. Do ovih razlika u

realizaciji dolazi pod uticajem „vrste" jata, promeni naglaska i okruženja u paradigmi. Za imenice su moguće sledeće različite zamene jata:

0. *ije*	nepromenjeno		ždr*ije*lo
1. *ije*	*ije* i	plural	gr*ije*h, gr*ije*hovi
	e		gr*ije*h, grehovi
2. *ije*	*ije* i	plural	v*ije*k, v*ije*kovi
	je		v*ije*k, v*je*kovi
3. *ije*	*ije* i	plural	kor*ije*n, kor*je*novi
	e		kor*ije*n, korenovi
4. *ije*	*e*	plural	br*ije*g, bregovi
5. *ije*	*je*	plural	v*ije*nac, v*je*nčevi
6. *ije*	*ije* i	gen.plu.	kol*ije*vka, kol*ije*vaka
	e, jot.		kol*ije*vka, kol*je*vaka
7. *ije*	*je*	gen.sin.	zv*ije*re, zv*je*reta
8. *ije*	*e*	gen.sin.	vr*ije*me, vremena
9. *ije*	*e*, jot.	gen.sin.	*dije*te, *de*teta
10. *i*	*ije*	gen.sin	d*io*, d*ije*la

Karakteristično je za imenice da je njihov nepromenljivi deo, onaj deo na koji paradigmatske promene nemaju uticaja, podložan glasovnim promena, to jest u ijekavici su moguće različite zamena jata u oblicima paradigme. S druge strane, glasovne promene izazvane zamenom jata nemaju uticaja na promenljivi deo imeničke paradigme. To se može ilustrovati na primeru imenice *koren* koja može biti pridružena dvema morfološkim klasama: N13.01 i N15.01. Korišćenjem leksikografeme od ove imenice se mogu oformiti dve odrednice: $kor\varphi_1 en$.N13.01 i $kor\varphi_2 en$.N15.01. Paradigme ove dve odrednice mogu se opisati sledećim transduktorima:

$$kor\varphi_1 en(\emptyset/_{ns,as} + a/_{gs,gp} + u/_{ds,vs,ls} + om/_{is} + i/_{np,vp} + e/_{ap} + nima/_{dp,ip,lp})$$
$$kor\varphi_2 en(\emptyset/_{ns,as} + a/_{gs} + u/_{ds,vs,ls} + om/_{as} + ovi/_{np,vp} + ova/_{gp} + ove/_{ap} + ovima/_{dp,ip,lp})$$

Razlika ove dve paradigme je u oblicima množine koji se u slučaju klase N15.01 grade sa infiksom *-ov-*. Ovim paradigmama opisane su promene obe dozvoljene varijante: *koren* i *korijen*. Kao što se vidi i iz ovog primera, leksikografeme koje opisuju zamenu jata pojavljuju se uvek u nepromenljivom delu imenice. Leksikografema φ iz ovog primera može se opisati sledećim atributima i operatorima:

φ	$\rightarrow$	*e*	ekavski
	$\rightarrow$	*ije*	ijekavski

Razlika između leksikografema φ_1 i φ_2 je u tome što prva znači da „*ije*" zamena jata ostaje ista u svim oblicima nepromenjena dok druga znači da se u oblicima množine jat zamenjuje sa *e* ili sa *je*. Odrednice $kor\varphi_1 en$.N13.01 i $kor\varphi_2 en$.N15.01 opisuju na taj način sledeće paradigme:

korφ_1*en*.N13.01		*kor*φ_2*en*.N15.01		
ekavski	ijekavski	ekavski	ijekavski	
			ije → *e*, *pl*	*ije* → *je*, *pl*
koren	korijen	koren	korijen	korijen
korena	korijena	korena	korijena	korijena
korenu	korijenu	korenu	korijenu	korijenu
koren	korijen	koren	korijen	korijen
korenu	korijenu	korenu	korijenu	korijenu
korenom	korijenom	korenom	korijenom	korijenom
korenu	korijenu	korenu	korijenu	korijenu
koreni	korijeni	korenovi	korenovi	korjenovi
korena	korijena	korenova	korenova	korjenova
korenima	korijenima	korenovima	korenovima	korjenovima
korene	korijene	korenove	korenove	korjenove
koreni	korijeni	korenovi	korenovi	korjenovi
korenima	korijenima	korenovima	korenovima	korjenovima
korenima	korijenima	korenovima	korenovima	korjenovima

U slučaju glagola, leksikografema koja opisuje zamenu jata može se javljati i u nepromenljivom delu glagola i u njegovom promenljivom delu (infiksu) (videti 5.3.1). Na primer, glagol *mreti* pripada klasi V26.00 i za građenje oblika koristi infikse */e/∅/* dok glagol *mrijeti* pripada klasi V26.25 i za građenje oblika koristi infikse */ije/∅/*. Sve varijante glagola mogle bi se okupiti u jedan oblik, *mr*φ*ti*.V'26.00, korišćenjem leksikografeme φ, pri čemu bi klasa V'26.00 za građenje oblika koristila infikse */φ/∅/*. Leksikografema φ u ovom slučaju ima sledeće značenje:

φ	→	*e*	ekavski
	→	*ije*	ijekavski
	→	*i*	ikavski

Na sličan način bi se mogle redefinisati i klase V33 i V34. Tako se varijante *goreti*, *gorjeti* i *goriti* mogu opisati korišćenjem leksikografeme sa *gor*ψ_1*ti*.V'33.00, pri čemu klasa V'33.00 za građenje oblika koristi infikse */ψ_1/∅/*. Varijante *želeti*, *željeti* i *želiti* se mogu opisati sa *žel*ψ_2*ti*.V'34.02, pričemu klasa V'34.02 za građenje oblika koristi infikse */lψ_2/l/lj/*. Leksikografeme ψ_1 i ψ_2 imaju sledeće značenje:

ψ_1	→	*e*	ekavski
	→	*je*	ijekavski
	→	*i*	ikavski + ek.dij. + ijek.dij.
ψ_2	→	*e*	ekavski
	→	*e*, jotovanje	ijekavski
	→	*i*	ikavski + ek.dij. + ijek.dij.

Treba napomenuti da se u oba slučaja ijekavska zamena jata u obliku jednine muškog roda glagolskog prideva radnog zamenjuje sa *i*. Uvođenjem leksikografema, potklase V26.25, V33.25 i V34.35 više nisu potrebne, kao i druge podklase klase V33. Treba, međutim uočiti da je moguće uvođenje novih podklasa klasa V26, V33, V34 s obzirom na moguće pojavljivanje različitih leksikografema u infiksu. Na primer, *boleti* i *želeti* su u klasi V34.02 i sve njihove varijante mogle bi se opisati leksikografemom ψ_2. Prema [117] varijante glagola *boleti* morale bi se, međutim, opisati sledećom leksikografemom:

ψ_3	→	e	ekavski
	→	e, jotovanje	ijekavski
	→	i	ikavski + ek.neknj. + ijek.neknj.

Na taj način bi glagoli *boleti* i *želeti* pripadali različitim podklasama klase V'34: jedna bi koristila infikse $/l\psi_3/l/lj/$ a druge infikse $/l\psi_2/l/lj/$.

Od zamena jata koje se pojavljuju u nepromenljivom delu glagola, promenama u ijekavici su podložne samo one zamene koje se nalaze na spoju nepromenljivog dela i infiksa.

1. *ije*	→	*ije*, W, A, AdvPs, PP	*donijeti*
		e, P, A, Y, AdvPs	
		i, PPms	
2. *je*	→	*je*, W, AdvPs, PP	*sjeći*
		ije, P, A, I, Y, AdvPr	
3. *je*	→	*je*, W, A, I, AdvPs, PP	*dospjeti*
		ije, P	
		i, Pmp3, Y, AdvPr, PPms	

Neka ω označava sledeću leksikografemu:

ω	→	e	ekavski
	→	je	ijekavski

Tada se varijante glagola *verovati* i *seći* mogu predstaviti sa $v\omega_1 rovati$ i $s\omega_2 ći$. Razlika između leksikogarfema ω_1 i ω_2 je u tome što prva ne menja ijekavsku zamenu jata u oblicima dok je druga menja prema drugoj shemi iz gornje tabele. Oblici ova dva glagola prikazani su u donjoj tabeli.

$v\omega_1 rovati$		$s\omega_2 ći$	
verovati	vjerovati	seći	sjeći
verujem	vjerujem	sečem	siječem
verovah	vjerovah	sekoh	sjekoh
verovah	vjerovah	secijah	sijecijah
		sečah	siječah
veruj	vjeruj	seci	sijeci
verujući	vjerujući	sekući	sijekući
verovavši	vjerovavši	sekavši	sjekavši
verovao	vjerovao	sekao	sjekao
verovala	vjerovala	sekla	sjekla

Uvođenjem ovakvih leksikografema i redefinisanjem podklasa V01.00, V50.01 i V10.50, podklase V01.25, V50.26 i V10.75 više nisu potrebne. Za razliku od prethodnog slučaja, uvođenje leksikografema u nepromenljivi deo glagola ne stvara potrebu za uvođenjem novih podklasa.

5.4.2 Leksikografema u rečnicima DELA

Azbuka za kodiranje rečnika DELA može se proširiti uvođenjem leksikografema. Ako se postupi na taj način, rečnici DELA se transformišu u meta-rečnike koji se sastoje od meta DELAS-a (skraćeno, mDELAS) i meta DELAF-a (skraćeno, mDELAF). Na primer, u DELAS-u se nalaze sledeći ulazi za lekseme *koren* i *korijen*:

koren,N13.01E,N15.01E
korijen,N13.01J,N15.01J

Uvođenjem leksikografema φ_1 i φ_2 u azbuku za kodiranje rečnika koje su opisane u 5.4.1, ova dva ulaza iz DELAS-a zamenjuju se sledećim ulazima u mDELAS-u:

korφ_1n,N13.01
korφ_2n,N15.01

Kao što se vidi, u mDELAS-u ekavska i ijekavska varijanta više nisu razdvojene. No, s druge strane, ulazi koji odgovaraju različitim morfološkim klasama postaju odvojeni ulazi u mDELAS-u, jer korišćenjem proširene azbuke za kodiranje rečnika formalne reči korφ_1n i korφ_2n više nisu jednake.

Ovim dvama ulazima iz DELAS-a odgovaraju sledeći ulazi u DELAF-u:

koren,.N13.01:msn-:msa-,.N15.01:msn-:msa-
korena,koren.N13.01:msg-:mpg-,koren.N15.01:msg-
korene,koren.N13.01:mpa-
koreni,koren.N13.01:mpn-:mpv-
korenima,koren.N13.01:mpd-:mpi-:mpl-
korenom,koren.N13.01:msi-,koren.N15.01:msi-
korenova,koren.N15.01:mpg-,korijen.N15.01:mpg-
korenove,koren.N15.01:mpa-,korijen.N15.01:mpa-
korenovi,koren.N15.01:mpn-:mpv-,korijen.N15.01:mpn-:mpv-
korenovima,koren.N15.01:mpd-:mpi-:mpl-,korijen.N15.01:mpd-:mpi-:mpl-
korenu,koren.N13.01:msd-:msv-:msl-,koren.N15.01::msd-:msv-:msl-
korijen,.N13.01:msn-:msa-,.N15.01:msn-:msa-
korijena,korijen.N13.01:msg-:mpg-;korijen.N15.01:msg-
korijene,korijen.N13.01:mpa-
korijeni,korijen.N13.01:mpn-:mpv-
korijenima,korijen.N13.01:mpd-:mpi-:mpl-
korijenom,korijen.N13.01:msi-,korijen.N15.01:msi-
korijenu,korijen.N13.01:msd-:msv-:msl-,korijen.N15.01:msd-:msv-:msl-
korjenova,korijen.N15.01:mpg-
korjenove,korijen.N15.01:mpa-
korjenovi,korijen.N15.01:mpn-:mpv-
korijenovima,korijen.N15.01:mpd-:mpi-:mpl-

Korišćenjem leksikografema korφ_1n i korφ_2n odgovarajući ulazi u mDELAF-u postaju:

korφ_1n,.N13.01:msn-:msa-
kor$\varphi_1$na,korφ_1n.N13.01:msg-:mpg-
kor$\varphi_1$ne,korφn.N13.01:mpa-
kor$\varphi_1$ni,korφ_1n.N13.01:mpn-:mpv-
kor$\varphi_1$nima,korφ_1n.N13.01:mpd-:mpi-:mpl-
kor$\varphi_1$nom,korφ_1n.N13.01:msi-
kor$\varphi_1$nu,korφ_1n.N13.01:msd-:msv-:msl-
korφ_2n,.N15.01:msn-:msa-
kor$\varphi_2$na,korφ_2n.N15.01:msg-

```
korφ2nu,korφ2n.N15.01:msd-:msv-:msl-
korφ2nova,korφ2n.N15.01:mpg-
korφ2nove,korφ2n.N15.01:mpa-
korφ2novi,korφ2n.N15.01:mpn-:mpv-
korφ2novima,korφ2n.N15.01:mpd-:mpi-:mpl-
```

mDELA rečnici mogu se upotrebiti na dva načina. S jedne strane, mDELA rečnici su generatorni na osnovu kojih se mogu generisati rečnici tipa DELAS i DELAF koji koriste uobičajenu azbuku datu u 5.2 i koji sadrže sve varijante odrednica. Korišćenjem atributa leksikografema, iz mDELA rečnika mogu se generisati i posebni rečnici, na primer, rečnici DELAS i DELAF ekavskog izgovora, ili DELAS i DELAF rečnici ekavskog i ijekavskog izgovora bez neknjiževnih oblika i slično.

S druge strane, mDELA rečnici mogu se koristiti i direktno, bilo za generisanje varijanti i njihovih oblika iz ulaza u mDELAS-u, na primer pri pretraživanju teksta, bilo za pronalaženje tekstualne reči u rečniku mDELAF i pridruživanje toj reči mogućih odrednica iz rečnika mDELAS. O generisanju varijanti na osnovu leksikografema i oblika na osnovu koda morfološke klase bilo je govora u 5.4.1 a ovde će biti naznačena mogućnost direktnog korišćenja mDELA rečnika za pronalaženje tekstualnih reči. Kao što je bilo rečeno u 5.2, azbuka zapisa teksta i rečnika se po pravilu razlikuju i dok je azbuka zapisa rečnika, jednom izabrana i stabilna, tekstovi u čijoj će se analizi taj rečnik koristiti mogu biti zapisani korišćenjem različitih azbuka ili sistema kodiranja. Pronalaženje tekstualne reči u rečniku DELAF stoga zahteva usklađivanje ove dve azbuke, o čemu će biti više reči u glavi 6. Ovo usklađivanje je u najvećem broju slučajeva jednostavno, jer jednom kodu (ili slovu) azbuke teksta najčešće odgovara jedan, isti ili različit, kôd (ili slovo) azbuke rečnika.

Usklađivanje azbuke za zapis teksta sa azbukom rečnika mDELA je, međutim, složenije. Ne samo što većem broju kodova azbuke teksta može da odgovara jedan kod azbuke rečnika (na primer, sekvenciji „ije" iz teksta može odgovarati jedan kod azbuke rečnika — leksikografema), već jednom kodu iz azbuke teksta mogu odgovarati različiti kodovi azbuke rečnika, u zavisnosti od lokalnog okruženja. Na primer, tekstualne reči *đever* i *đeram* prevode se na sledeći način u azbuku rečnika mDELA:

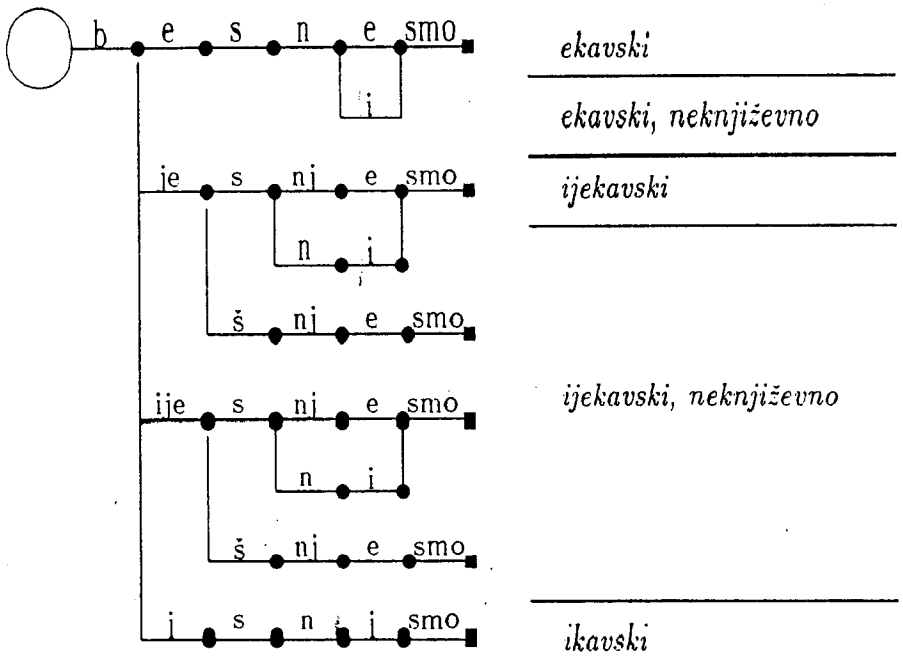
đ	→	d	*đ*	→	đ
e	→	α	*e*	→	e
v	→	v	*r*	→	r
e	→	e	*a*	→	a
r	→	r	*m*	→	m

Dalje usložnjavanje potiče otuda što se neke leksikografeme realizuju kao prazne niske, što dovodi do toga da prilikom prevođenja tekstualne reči u azbuku rečnika na određenim pozicijama leksikografemu treba umetnuti. To se može pokazati na primeru tekstualne reći *alva*, čije preslikavanje u azbuku rečnika zahteva umetanje leksikografeme χ:

∅	→	χ
a	→	a
l	→	l
v	→	v
a	→	a

Umesto da se tekstualna reč prevodi iz azbuke teksta u azbuku rečnika, ulazi u rečnicima mDELA mogu se shvatiti kao generatori konačnih automata, pri čemu atributi leksikografema

upravljaju njihovim generisanjem. Primera radi, ulaz *bγsnδsmo,bγsnδti*.V34.13.5, ukoliko se ne ograniči upotreba ni jednog atributa, generiše sledeći automat:

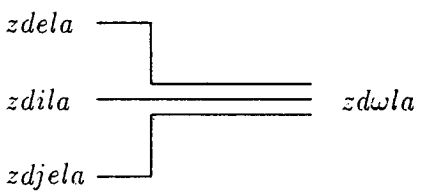


Kombinovanjem jednog ili više atributa mogu se generisati podautomati jednog ovakvog opšteg automata. U svakom slučaju, prepoznavanje tekstualne reči sada se svodi na njeno prepoznavanje jednim od automata iz generisanog skupa.

I jedan i drugi pristup komplikuju postupak pretraživanja teksta i zahtevaju složeniju organizaciju DELA rečnika od uređene liste ulaza. Kako prvi podaci u izgradnji rečnika pokazuju (videti 5.5), uvođenje leksikografeme u rečnike DELA smanjuje broj ulaza za manje od 10%. Naime, u do sada izgrađenom DELA rečniku, približno 5% imenica i prideva treba opisati leksikografemom koja opisuje zamene jata, dok je za glagole taj broj oko 9%. Od svih glasovnih promena, najveći broj se odnosi upravo na promene izazvane zamenom jata. Ovaj podatak govori da uvođenje leksikografeme ne bi znatno doprinelo smanjenju memorijskog zauzeća. Šta više, složenija struktura rečnika mogla bi dovesti i do povećanja memorijskih zahteva. Stoga je, u ovom trenutku, leksikografema ugrađena kao dodatna informacija uz ulaze rečnika DELAS.

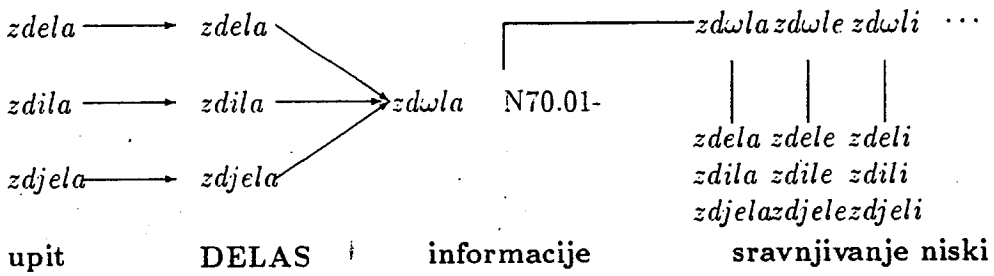
5.4.3 Informacije o glasovnim promenama u rečniku DELAS

Podaci o glasovnim promenama ugrađeni su kao dodatne informacije u rečnik DELAS uz sve varijantne oblike sa ciljem da se povežu oblici koji nastaju usled određenih glasovnih promena. Njima se, naime, pridružuje „normalizovani“ oblik zapisan korišćenjem leksikografema. Tako se, na primer, varijantnim oblicima *zdela*, *zdjela* i *zdila* pridružuje isti normalizovani oblik:

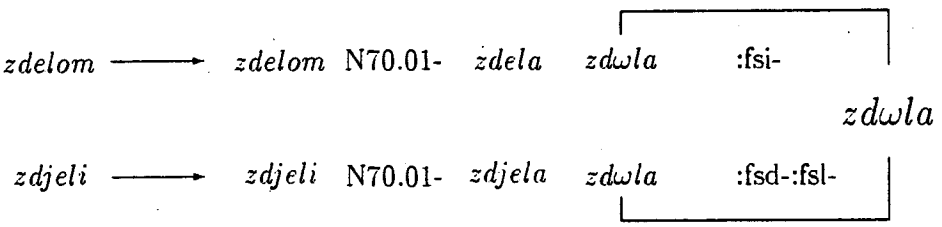


U ovom odeljku biće prikazano kako se korišćenjem ovako koncipiranih informacija u DELAS-u mogu realizovati osnovne operacije nad tekstom, kao što su pretraživanje teksta, svođenje tekstualnih reči na leksičke reči i prevođenje teksta iz jednog u drugi varijantski oblik.

Pretraživanje teksta može se ostvariti navođenjem bilo kog varijantskog oblika leksičke reči. Kao što je prikazano na narednoj slici, informacija iz DELAS-a za zadati oblik leksičke reči omogućava određivanje svih varijantskih oblika kao i svih oblika njihove flektivne paradigme. Moguće je i generisanje oblika izabranih na osnovu određenih atributa na način koji je opisan u 5.4.2. Skup svih generisanih niski zatim predstavlja obrazac za sravnjivanje sa tekstom ([6], [22], [11], [82]).

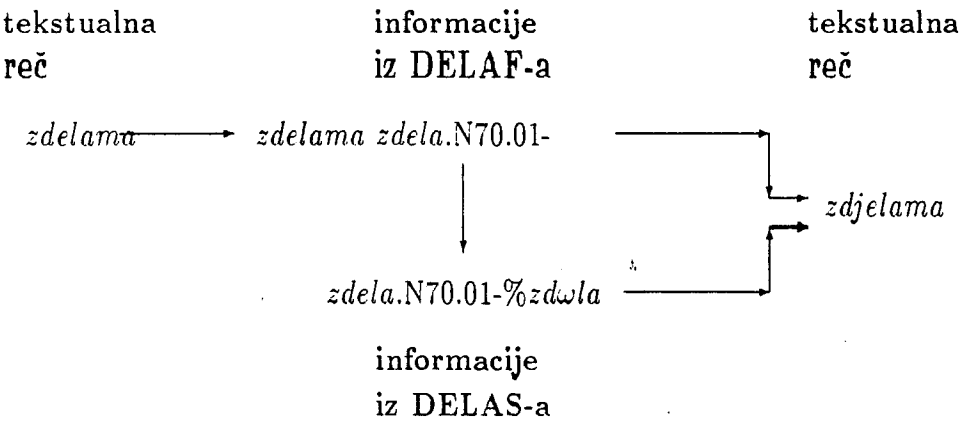


Svođenje tekstualne reči na leksičku reč svodi se na pronalaženje te reči u DELAF-u gde je kao informacija zabeležena leksička reč čiji je ona oblik. Zbog homografije oblika, ovaj postupak nije uvek dovoljan i bez razmatranja šireg konteksta se u principu ne može razrešiti. Ponekad je poželjno da se tekstualna reč ne svodi na leksičku reč već na „normalizovani" oblik leksičke reči koji obuhvata sve varijante leksičke reči, kako je prikazano i na sledećoj slici:[12]



Ponekad je potrebno transformisati tekst u kome preovlađuju oblici jednog izgovora, ili u kome su izmešani oblici raznih izgovora, tako da se u njemu pojavljuju oblici samo jednog, izabranog, izgovora. Na narednoj slici je prikazano kako se takav postupak može sprovesti. Za njegovu realizaciju koriste se sve informacije iz DELAS-a i DELAF-a: morfološki kôd, kôd gramatičkih kategorija i leksikografeme. Primer jednog takvog postupka biće opisan u glavi 6 i dodatku E.4.1.

[12]Prilikom izrade registra ključnih reči u [72], svi oblici glagola *bolξti* (*boleti*, *boljeti*, *boliti*) svedeni su na leksičku reč *boleti* jer se u svih 25 pojavljivnja ovaj glagol javlja isključivo u oblicima prezenta (*boli* i *bole*) koji su isti u svim izgovorima. S obzirom da u [72] preovlađuje ijekavski izgovor, izbor leksičke reči *boljeti* bio bi možda pogodniji.



S obzirom na potencijalno neograničeni broj leksikografema, odnosno na potrebu da se njihov broj unapred ne ograničava, kao i na potrebu da se u zapisu rečnika DELA ostane u okvirima kodiranja 7-bitnim kodom propisanim ISO 646 standardom, za realizaciju leksikografeme izabran je sledeći postupak (videti i [121]): od svih varijantnih oblika bira se jedan, a zatim se preciziraju karakterske pozicije u njemu koje se u varijantama razlikuju i vrsta i tip glasovne promene koja do razlikovanja dovodi. Na primer, leksička reč $\alpha l \beta b$ u čijem zapisu se koriste dve leksikografeme α i β se u DELAS-u zapisuje na sledeći način:

```
hleb#H1.05#E3.26
```

Od svih varijantnih oblika — *hleb*, *leb*, *hljeb* i *ljeb* — izabran je kao reprezentativan oblik *hleb*. Specijalnim karakterom `'#'` označen je opis leksikografeme koji se sastoji iz tri dela:

- vrste glasovne promene (`H`, varijacije vezane uz suglasnik *h*; `E`, varijacije vezane za zamenu jata);
- karakterska pozicija na kojoj se odražavaju glasovne promene (na prvoj poziciji promene vezane za suglasnik *h* a na trećoj poziciji promene vezane za zamenu jata);
- tip glasovne promene (`H`-varijacija je tipa 5, što znači da izostavljanje suglasnika *h* označava dijalekatske oblike a `E`-varijacija je tipa 26, što znači da se u ijekavskom izgovoru jat zamenjuje grupom *je* i pri tome dolazi do jotovanja prethodnog suglasnika).

Opis leksikografeme ponekad sadrži i dodatne informacije koje govore kako se glasovna promena ponaša u okviru flektivne paradigme. Na primer, leksička reč *dpte* se u DELAS-u zapisuje na sledeći način:

```
dete#E2.03.09
```

Leksikografema `E2.03.09` govori da se radi o glasovnim promenama izazvanim promenama jata, da se te promene odražavaju na drugoj karakterskoj poziciji, glasovna promena je tipa 03 (jat se zamenjuje sa *e* u ekavskom, grupom *ije* u ijekavskom a sa *i* u ikavskom izgovoru). Poslednji podatak 09 govori da se u paradigmatskoj promeni ijekavskog izgovora *ije* zamenjuje sa *je* a da u dijalekatskim varijantama dolazi do jotovanja.

U DELAS/SJ su do sada ugrađene informacije o glasovnim promenama izazvanim zamenama jata i izostavljanjem, odnosno zamenom, suglasnika *h*. Kao reprezentativni oblik za leksikografemski opis zamena jata izabran je ekavski oblik, premda je mogao biti izabran i neki drugi. Ipak, ekavski oblik je nešto pogodniji od ijekavskog jer se u njemu jat po pravilu zamenjuje sa *e*, dakle jednim karakterom. Opravdanje za eksplicitno označavanje karaktera

```
bledeti#E3.03#E5.41
crveneti#E6.30
uticaj#E3.50
```

U prvom primeru karakteri *e* predstavljaju dve različite zamene jata, u drugom samo jedan od karaktera *e* predstavlja zamenu jata a u trećem je jat u ekavskom izgovoru zamenjeno sa *i*.

Informacije o varijantama se u DELAS-u ne vezuju za jedan ulaz, već za ulaz i morfografemski kôd, kao što pokazuje naredni primer:

```
beg,N07.02+*,N01.02+*,N08.02-E%#E2.03
```

Ulazu beg pridružena su tri morfografemska koda: prva dva odgovaraju dvema različitim paradigmama imenice *beg* sa značenjem *veliki posednik u turskom carstvu*, dok treći odgovara imenici *beg* sa značenjem *bežanje, bekstvo*. Samo ulaz `beg,N08.02-E` ima varijacije i one su opisane leksikografemom `E2.03`.

Za opis varijanti u DELAS- u koristi se ukupno 50 leksikografema za opis zamena jata i 10 leksikografema za opis izostavljanja ili zamene suglasnika *h*. Njihov repertoar je prikazan u sledećoj tabeli.

vrsta	kod	tip	primer
E	1	*e*; *i*	*grejati*
	2	*e*; *ije*	*levi*
	3–7	*e*; *ije*; *i*	*ždrelo*
	8	*e*; *je*	*dospeti*
	9–17	*e*; *je*; *i*	*nevernik*
	18	*e*; *ije*; *je*	*koren*
	19–22	*e*; *ije*; *je*; *i*	*zapovediti*
	23–25,36–41,43	*e*; *je*; *je*, jot.; *i*	*živeti*
	26–27	*e*; *je*, jot.	*hleb*
	28–31	*e*; *je*, jot.; *i*	*lepota*
	32–34	*e*; *ije*; *je*, jot.; *i*	*leska*
	35	*e*; *je*; *je*, jot.	*deverak*
	42	*e*; *je*; *je*, jot.; *ije*; *i*	*ded*
	44	*e*; *je*, jot.; *je*, jot., jedn.	*izlesti*
	45	*e*; *ije*; *je*, jot.; *i*, jot.; *ije*, jot.	*gnev*
	46	*e*; *ije*; *i*; *je*, jot.; *i*, jot.; *ije*, jot.	*gnezdo*
	47	*e*; *i*; *je*, jot; *je* jot., jedn.	*besneti*
	48	*e*; *je*; *i*; *je* jot., jedn.	*zdela*
	50	*i*; *je*	*uticaj*
H	1–2	*h*; *v*	pastuh
	3	*h*; *v*; ∅	muha
	4	*h*; *j*; *v*; ∅	aždaha
	5–7	*h*; ∅	hladan
	8	*h*; *j*; ∅	mehana
	9–10	*h*; *j*	kihati

5.5 Izgradnja rečnika DELA za SJ

U rečniku DELA za SJ se javljaju sve vrste reči. Fond zamenica, brojeva, predloga, uzvika, veznika, partikula i, delimično, priloga popunjen je konsultovanjem postojećih gramatika ([14], [132], [131] i [12]) i rečnika ([117] i [116]) i sadrži oblike ekavskog i ijekavskog izgovora, a u određenoj meri i oblike južnog ijekavskog izgovora. Od priloga u rečnik su uvršteni oni koji nisu nastali od prideva. DELAS sadrži 1268 ulaza za ove vrste reči i zahteva dalje dopunjavanje, posebno prilozima.

Fond imenica, glagola, prideva i priloga nastalih od prideva popunjen je onim rečima koje se pojavljuju u Vukovim narodnim poslovicama [72] i koje su ekscerpirane u fazi izrade registra ključnih reči novog izdanja. Ekscerpciju su izvršili leksikografi Instituta za srpski jezik na osnovu konkordancija izrađenih programom AURORA [141]. DELAS/SJ sadrži 2720 imenica, 1884 glagola, 630 prideva i 111 priloga koji su postali od prideva. Oblika specifičnih ekavskom, odnosno, ijekavskom izgovoru je među imenicama 147, glagolima 190 a među pridevima 30. Svi ovi ulazi iz DELAS-a, a posebno oni koju su specifični ekavskom ili ijekavskom izgovoru, ne pojavaljuju se obavezno u tekstu poslovica već se u registru ključnih reči javljaju kao uputnice.

DELAF je konstruisan iz ulaza za DELAS i to za imenice i prideve izračunavanjem svih oblika korišćenjem programa MORF ([139] i [140]) a za glagole, zamenice i brojeve generisanjem svih oblika korišćenjem konačnih transduktora.

Generisani rečnik se sastoji od dve tekstualne datoteke koje sadrže samo kodove predviđene ISO 646 standardom [56]. Izvodi iz ovih rečnika predstavljeni su u dodacima D.1 i D.2 dodatka. Broj redova u rečniku prema vrstama reči kao i memorijsko zauzeće predstavljeno je u narednoj tabeli.

Vrsta reči	DELAS		DELAF	
	KByte	Broj ulaza	KByte	Broj ulaza
imenica	50K	2720	627K	18241
pridevi	11K	630	565K	10956
glagoli	42K	1884	1693K	49076
ostali	21K	1378	133K	3849
ukupno	124K	6569	3009K	81152

Poređenja radi, prema [24] za francuski jezik rečnik DELAS ima 80000 ulaza dok rečnik oblika DELAF ima 550000 ulaza. Odnos broja ulaza u DELAF-u prema DELAS-u je za francuski jezik 1 : 7 dok je za srpski jezik taj odnos 1 : 12. Kada se govori o odnosu broja ulaza u DELAF-u prema DELAS-u treba imati u vidu da nisu za sve prideve iz DELAS-a generisani oblici komparativa i superlativa, čak i za one prideve gde se oni potencijalno mogu realizovati. Takođe su generisani oblici trpnog glagolskog prideva samo za one glagole za koje se trpni pridev u javlja u tekstu poslovica. Ova ograničenja su sprovedena radi ograničavanja dimenizije rečnika DELAF.

Za rukovanje rečnicima razvijen je skup programa na jezicima C i Awk. Pretraživanje rečnika obavlja se preko melanžnih tabela korišćenjem skupa melanžnih procedura na jeziku C opisanih u [53].

Glava 6

Povezivanje elektronskog teksta i elektronskog rečnika

U glavama 2 i 3 predstavljen je jedan način za obeležavanje teksta upotrebom SGML jezika kao i mogućnosti koje taj način pruža u izradi elektronskog teksta. U glavi 4 prikazan je jedan sistem za raščlanjavanje formalnog zapisa ovakvog elektronskog teksta. U glavi 5 izložen je koncept elektronskog rečnika i opisana je izgradnja tog rečnika za srpski jezik. U ovoj glavi biće predstavljen model elektronskog teksta koji integriše koncept obeležavanja formalne strukture teksta i obeležavanja teksta leksičkim i gramatičkim informacijama. Primena ovakvog modela biće ilustrovana na primeru izrade elektronskog teksta Vukovih narodnih poslovica [72].

6.1 Integrisani model elektronskog teksta

Pretpostavke za izgradnju integrisanog modela elektronskog teksta su:

1. postojanje formalnog zapisa njegove strukture u obliku SGML definicije tipa dokumenta;
2. tekst obeležen u skladu sa tom definicijom;
3. elektronski rečnik DELA jezika na kome je tekst zapisan.

U integrisanom modelu svakoj tekstualnoj reči iz teksta, ili podskupu tekstualnih reči koji je određen prema nekom unapred zadatom kriterijumu, pridružuju se potencijalne leksičke reči i njihove potencijalne leksičke kategorije.

Izdvajanje tekstualnih reči iz teksta obeleženog SGML odnosi se na onaj deo njegovog sadržaja koji se opisuje kao raščlanjivi karakterski podaci, odnosno, onaj deo koji se opisuje tokenom `#PCDATA` (videti 2.2.1). Ostale vrste sadržaja, kao što su karakterski podaci, a to su zamenljivi karakterski podaci `RCDATA` i karakterski podaci `CDATA`, se u najvećem broju slučajeva koriste za zapisivanje sadržaja koji ne podleže jezičkoj obradi (na primer, matematičke formule, i slično).

Eksplicitno kodiranje strukture dokumenta SGML etiketama omogućava izbor sadržaja koji će biti podvrgnut jezičkoj obradi u zavisnosti od elementa i njegove pozicije u strukturi dokumenta kao i od vrednosti atributa. Tako, na primer, kao element jezičke obrade mogu se pojaviti samo oni elementi čiji sadržaj predstavlja [illegible]

atributa lang odgovara izabranom jeziku[1]. Moguće je obrađivati samo sadržaj izabranih elemenata.

Prilikom izrade elektronskog teksta Vukovih narodnih poslovica obrađivan je sadržaj sledećih elemenata (videti 3.3.1 i dodatak B):

$$(pv \wedge \neg n.i.prov) \vee (pexp \wedge c.r.s)$$

Ovo znači da je obrađen sadržaj elementa pv kao i svih u njemu ugnježdenih elemenata, osim elementa n.i.prov, kao i sadržaj elementa c.r.s koji je ugnježden u element pexp.

Izdvajanje tekstualnih reči zavisi od načina kodiranja samog teksta i mora biti usklađeno sa azbukom elektronskog rečnika. Tekstualna reč se definiše kao niska karakatera između dva separatora. Odavde sledi da je za opis tekstualne reči neophodno utvrditi azbuku teksta i skup separatora. Da bi se toj reči mogle pridružiti gramatičke informacije, zapis tekstualne reči treba prevesti iz azbuke teksta u azbuku rečnika da bi bilo moguće pronalaženje tih informacija u elektronskom rečniku DELA. U narednom odeljku biće prikazana azbuka i skup separatora elektronskog teksta Vukovih narodnih poslovica kao i procedura prevođenja izdvojene tekstualne reči u azbuku rečnika koja je primenjena prilikom izrade elektronskog teksta zasnovanog na integrisanom modelu.

Tekstualna reč transkribovana u azbuku rečnika se traži u elektronskom rečniku DELAF da bi se u njemu pronašle informacije koje se zatim dodeljuju tekstualnoj reči kao vrednosti njenih atributa *vrsta reči*, *morfološki kod*, *leksička reč*, *gramatičke kategorije*. Na osnovu informacija ekstrahovanih iz rečnika DELAF moguća su i pretraživanja rečnika DELAS radi pronalaženja dodatnih informacija kakva je, na primer, veza sa varijantskim oblicima (videti odeljke E.1 i E.2 dodatka). Sve informacije koje se na ovaj način pridružuju tekstualnoj reči predstavljaju potencijalne vrednosti njenih atributa. Naime, tekstualna reči može biti homoformni ili homografijski oblik više reči, kao što je prikazano na primerima u glavi 5. Redukovanje skupa potencijalnih vrednosti se može obaviti automatskim procedurama koje testiraju lokalno okruženje dok se konačna dodela vrednosti može ostvariti tek posle potpune sintaksne analize (videti [85], [93], [118]).

Elektronski tekst zasnovan na integrisanom modelu predstavlja i sam SGML dokument. U njegovu definiciju tipa dokumenta uveden je samo jedan novi element w koji menja strukturu postojećeg elektronskog teksta kao uključenje na nivou celog dokumenta. Ako je, na primer, element rad koren u hijerahijskoj strukturi kojom se predstavlja DTD polaznog dokumenta, njegova deklaracija se menja na sledeći način:

```
<!ELEMENT rad   - -  ( sadrzaj_rad )  +( w )  >
<!ELEMENT w     - o  ( #PCDATA )      -( w )  >
<!ATTLIST w       a    CDATA          #REQUIRED >
```

Uključivanje elementa w na najvišem hijerarhijskom nivou znači da se on može uključiti u sadržaj svih u njega ugnježdenih elemenata. Navođenjem elementa w u listi isključenja deklaracije tog istog elementa znači da ugnježdavanje elementa w nije dozvoljeno (videti 2.2.2). Bilo bi moguće uključiti element w u samo u one elemente čiji sadržaj se jezički obrađuje. U datom primeru iz obrade Vukovih narodnih poslovica to bi značilo uključivanje elementa w u sadržaj elementa pv, a isključivanje iz elementa n.i.prov. Ovakvo rešenje bi zahtevalo izmenu DTD-a za svaku jezičku obradu i zato nije pogodno.

[1] U TEI DTD-u globalni atribut lang koji se nasleđuje od obuhvatnih elemenata definiše jezik na kome je sadržaj tog elementa zapisan.

Novi element w ima samo jedan atribut koji je obavezan. Vrednost tog atributa je deklarisana kao karakterski podaci, što znači da može sadržati dozvoljene SGML karaktere. Sve informacije ekstrahovane iz rečnika DELAF i DELAS postaju vrednost atributa a. Alternativna mogućnost bi bila da se deklariše više atributa elementa w: na primer, vr za vrstu reči, lr za leksičku reč i tako dalje. Ovakvo rešenje bi zahtevalo da unapred bude poznato koliko i koje se sve informacije ekstrahuju iz rečnika. Odabrano rešenje ne nameće nikakva ograničenja, a kako su ekstrahovane vrednosti zapisane korišćenjem iste notacije koja se koristi za zapis rečnika DELA, dostupne su sve potrebne vrednosti.

6.2 Primer izrade elektronskog teksta

U ovom odeljku biće prikazana izrada elektronskog izdanja teksta Vukovih narodnih poslovica zasnovanog na integrisanom modelu predstavljenom u prethodnom odeljku. Izvori za njegovu izradu su SGML zapis Vukovih narodnih poslovica koji koristi prošireni TEI DTD (videti dodatak B) i elektronski rečnik DELA za srpski jezik.

Tekst Vukovih narodnih poslovica je zapisan korišćenjem standarda [122] za 7-bitno kodiranje srpskohrvatskog latiničnog pisma kojim je definisana azbuka teksta. Separatori u ovom tekstu su svi 7-bitno kodirani karakteri osim slova engleske abecede (slova A-Z i a-z), specifičnih slova srpskohrvatske latinice (slova $\check{Z}$, $\check{S}$, $\check{D}$, $\acute{C}$, $\check{C}$ i odgovarajuća mala slova $\check{z}$, $\check{s}$, $\check{d}$, $\acute{c}$, $\check{c}$) i, imajući u vidu njihove specifične uloge, apostrofa ' i crtice -. Cifre su, dakle, takođe separatori što je opravdano s obzirom na prirodu analiziranog teksta. Ipak, u drugim tekstovima, cifre se mogu posmatrati i kao deo azbuke (na primer, zbog datuma).

Azbukom teksta i skupom separatora definisanim na gornji način izdvajane su tekstualne reči iz teksta kao niske karaktera (ne-separatora) između separatora. Obrada tekstualnih reči se vrši procedurom *obradi_reč*:

```
Neka je w izdvojena tekstualna reč
obradi_reč(w)
wl = prevedi_u_azbuku_rečnika(w)
Ako je tr = pronadji_u_DELAF(wl) onda
  at = tr
Ako wl počinje velikim slovom onda
  wv = pretvori_u_mala(wl)
  Ako je tr = pronadji_u_DELAF(wv) onda
    at = dopiši_na(at, tr)
Ako wl sadrži crticu i do sada reč nije pronađena, onda
  w = do_crtice(wl)
  at = obradi_reč(w)
Ako je pretraga bila upsešna onda
  Vrednost atributa a reči w je at
inače
  Vrednost atributa a reči w je '?'
Pomeri pokazivač iza reči w
```

Ova procedura vodi računa o različitim ulogama koje veliko slovo može imati (videti 5.2). Ako je u nekoj reči veliko slovo upotrebljeno na početku rečenice tada ta reč neće biti pronađena u rečniku i stoga početno veliko slovo treba konvertovati u malo i ponoviti pretragu rečnika. Međutim, ako je reč sa početnim velikim slovom pronađena u rečniku, to

ne znači da ne treba pokušati i sa malim početnim slovom zbog onih leksičkih reči u kojima je operacija veliko/malo slovo distinktivna: *Ruža* i *ruža*, *Rok* i *rok* i tako dalje.

U slučaju crtice, primenjen je suprotan postupak. Crtica se prvo isključuje iz skupa separatora i samo ako pretraga rečnika sa tako izdvojenom tekstualnom reči nije bila uspešna, crtica se uključuje u skup separatora i postupak se ponavlja sa novim tekstualnim rečima. Apostrof je bezuslovno isključen iz skupa separatora. Pretpostavka je, koja je u slučaju konkretnog teksta zadovoljena, da apostrof nije korišćen kao zamena za znak navoda[2]. Kao što će se videti, niti njegovo uključivanje u skup separatora, niti njegovo isključivanje iz tog skupa na sadašnjem nivou razvoja rečnika DELA/SJ ne daje zadovoljavajuće rezultate.

Procedura *prevedi_u_azbuku_rečnika* je, imajući u vidu sličnost azbuke konkretnog teksta i azbuke rečnika DELA/SJ vrlo jednostavna. Jedini problem predstavlja razlikovanje digrafa od odgovarajućih konsonantskih grupa, koje je u ovoj proceduri u potpunosti rešen prema postupku detaljno opisanom u [80] i [144].

Primenom opisanog postupka dobijeno je elektronsko izdanje teksta Vukovih narodnih poslovica zasnovano na integrisanom modelu. Iz ovog teksta je izdvojeno 44243 tekstualnih reči.

Pridruženu vrednost nema 2078 tekstualnih reči, odnosno, manje od 5% ukupnog broja reči. Reči koje nemaju pridruženu vrednost potiču otuda što za izdvojenu tekstualnu reč nema ulaza u rečniku DELAF/SJ. Svim preostalim rečima pridružen je skup potencijalnih vrednosti. Treba reći da u određenom broju slučajeva pridruženi skup vrednosti ne sadrži stvarnu vrednost tekstualne reči. Na primer, tekstualnoj reči *ja* pridružuje se samo jedna vrednost: `.ProNO1:*sn*`. Ta tekstualna reč može osim lične zamenice prvog lica jednine u nominativu predstavljati i veznik[3], te bi joj trebalo pridružiti skup vrednosti `.Con,.ProNO1:*sn*`. Ovakve situacije su posledica nepotpunosti rečnika DELA/SJ. Izvod iz izveštaja o nepridruženim potencijalnim vrednostima dat je u dodatku E.3.

Uzroci promašaja u pretraživanju do kojih dolazi usled grešaka mogu se razvrstati u tri grupe:

- *greške u tekstu*, odnosno tipografske greške — Premda je dva puta pre obrade izvršena korektura teksta poslovica, ovakve grešake su ipak prisutne, premda u malom broju. Na primer, u poslovici *Bolje se navrh vinograda dogovarati nego se nadno njaga dogovarati* tekstualna reč njaga stoji umesto njega;
- *greške u indeksu* — Rečnik DELA/SJ je, za potrebe izrade elektronskog teksta Vukovih narodnih poslovica, sastavljen nad leksikom iz registra ključnih reči [72]. Propusti učinjeni prilikom izrade registra izražavaju se kao promašaji u elektronskom izdanju. Na primer, u registru se pojavljuju glagoli *zapovediti* i *zapovijediti*, što je pogrešna zamena jata pa tekstualna reč *zapovjedih* iz poslovice *Ja zapovjedih đaku a đak crkvenjaku* nema pridruženu vrednost. Sličnim propustom se u registru ne javlja imenica *korijenje* (niti *korenje*) već *korijen* ukazuje na poslovicu *Ako je trava pokošena, ostalo je korijenje.* Tekstualna reč *korijenje* stoga nema pridruženu vrednost. Posledica ove vrste grešaka može biti i da dodeljeni skup potencijalnih vrednosti ne bude potpun. Na primer, na poslovicu *Ja kad videh zelen drijen, predadoh mu vas moj drijem i lijen* u registru ključnih reči ukazuje pridev *lijen* pa je tekstualnoj reči *lijen* pridružena samo vrednost `,.A06.51%len#E2.03:p#msn*:p#msa-` dok je vrednost `,.N04.01-J%len#E2.03:msa-` izostala. Sličan je slučaj i sa *volij* u poslovicama 610–613;

[2] Navodi su u tekstu SGML TEI obeleženih Vukovih narodnih poslovica sadržaj elementa q.

[3] Na primer, u poslovici *Ja ja, ja on.*

- *greške u izradi rečnika* — Do ove vrste grešaka dolazi zbog propusta u izradi rečnika. Tako je omaškom izostao broj *šest* dok je broj *hiljadu* pogrešno kodiran kao `hiljadu` umesto `hilxadu`. Takođe je omaškom ispušten određen broj imenica (*mati*), glagola (*prtiti*) i prideva. Do ove vrste grešaka u najvećoj meri dolazi zbog nedovoljno razrađenih programskih alata za izradu i održavanje rečnika.

Promašaji koji nastaju usled grešaka se lako mogu otkloniti korekcijom teksta i dopunom rečnika. Na postojanje promašaja koji nastaju usled ovih uzroka treba, međutim, uvek računati.

Druga vrsta promašaja je prouzrokovana nedostacima postupka za prevođenje iz azbuke teksta u azbuku rečnika kao i određenih propusta u definisanju flektivnih skupova promenljivih vrsta reči. Uzroci nastanka ove vrste promašaja se mogu razvrstati u sledeće grupe:

- *problem apostrofa.* Apostrof se u srpskom jeziku koristi na mestima gde su u pisanju izostala slova a u govoru glasovi, pri čemu on obično zamenjuje jedno slovo, odnosno, glas. Apostrof, kao deo azbuke teksta, nije separator, te su kao tekstualne reči iz Vukovih narodnih poslovica izdvojene: *al'* (umesto *ali*), *ka'* (umesto *kao*), *stić'* (umesto *stići*), *ogr'ješiti* (umesto *ogriješiti*), *mog'o* (umesto *mogao*) i mnoge druge kojima je dodeljen prazan skup vrednosti. Uključivanje apostrofa u skup separatora ne bi bilo uspešnije jer bi, recimo, umesto jedne tekstualne reči *ogr'ješiti* bile prepoznate dve: *ogr* i *ješiti*. Kako je upotreba apostrofa posebno u pesničkim i literarnim tekstovima česta treba razviti procedure za adekvatnu obradu apostrofa (na primer, za supstituciju apostrofa izostavljenim slovom).

- *problem pravopisa.* Neke tekstualne reči su pogrešno izdvojene zbog upotrebe pravila o spojenom i odvojenom pisanju reči zastarelog pravopisa. Tako je, na primer, iz poslovice *Ako ga Bog zna koliko ja, zlo ponj*, izdvojena tekstualna reč *ponj*, umesto dve tekstualne reči *po* i *nj*[4]. Na ovakvu vrstu propusta, koja može nastati i usled nedovoljnog poznavanja pravopisa, uvek treba računati.

- *arhaični oblici.* Imajući u vidu vrstu obrađivanog teksta u njemu se javlja i određen broj zastarelih, neknjiževnih i dijalekatskih oblika. Ilustracije radi, ovde će biti navedeni neki od tih oblika uz redni broj poslovice u kojoj se oni javljaju:
  - zamenice: *svačij* (34) umesto *svačiji*, *svak* i *nek* (774) umesto *svaki* i *neki*, *togaj* (2464) umesto *toga*, *takijeh* (1425) umesto *takvih*;
  - pridevi: *učinjenijeh* (326) umesto *učinjenih*, *malijema* (1156) umesto *malima*;
  - imenice: *dnevi* (285) umesto *dana*, *ljudma* (5280) umesto *ljudima*, *zubma* (60[3]) umesto *zubima*;
  - glagoli: *izio* (41) umesto *izjeo*, *ije* (4938) umesto *jede*.

  Neki od ovih slučajeva mogu se obraditi upotrebom leksikografema. U svakom slučaju, uključivanje ovakvih oblika u rečnik DELA/SJ zahtevalo bi da se u rečniku eksplicitno označi kom registru ti oblici pripadaju.

- *arhaične leksičke reči.* U registru ključnih reči Vukovih narodnih poslovica mnoge zastarele reči zamenjene su književnim oblicima. Tako se, na primer, imenica *čovek* uvek pojavljuje u oblicima imenice *čoek* a *pčela* u oblicima imenice *ćela*. Kako je rečnik

[4] Elektronski tekst Vukovih narodnih poslovica nastao je na osnovu prethodnog izdanja [71]. U novom izdanju [72] tekst je prilagođen savremenoj pravopisnoj normi.

DELA/SJ nastao na osnovu registra ključnih reči svim tekstualnim rečima koje odgovaraju ovakvim leksičkim rečima pridružen je prazan skup vrednosti. Takvih primera ima još:

- pridevi: *Božij* (2190) umesto *Božiji*, *ljucki* (2931) umesto *ljudski*, *masan* (2972) umesto *mastan*, *devojački* (1313) umesto *djevojački*;
- imenice: *zeitin* (512) umesto *zejtin*, *šćer* (1287) umesto *kćer*, *brastvo* (1775) umesto *bratstvo*, *đevojka* (1158–1162) umesto *devojka*;
- glagoli: *drktati* (oblik *drkće* u 1124) umesto *drhtati*, *ćerati* (oblik *ćerajući* u 5716) umesto *tjerati*, **sideti* (oblik *sidela* u 5425) umesto *sedeti*[5], *zađesti* (oblik *zađelo* u 5188) umesto *zadjenuti* odnosno *zadjesti*.

Veći broj ovih slučajeva može se obraditi uvođenjem leksikografeme. Šta više, mnogi od ulaza u DELAS-u već koriste leksikografeme koje ove slučajeve razrešavaju, kao na primer, `terati,V01.00.2E%#E2.39`. Međutim, ulazi DELAS-a su nastali iz registra ključnih reči i nisu dopunjavani uvođenjem leksikografema. Iz ovih razloga treba usavršiti programsku podršku izradi rečnika.

- *morfološki kôd.* U izvesnom broju slučajeva, promenljivim vrstama reči nije dodeljen adekvatan morfološki kôd. Do takvih propusta je najčešće dolazilo tamo gde su mogući višestruki oblici reči za izabrane vrednosti gramatičkih kategorija. Primeri su:
  - imenice: *miš* je u klasi N23.01+* (nominativ množine *miševi*) a treba da bude i u klasi N21.01+* (nominativ množine *miši*). Stoga, leksička reč *miš* nije prepoznata, na primer, u poslovici 1219. Slično, *puška* je u klasi N70.02-* (dativ/lokativ jednine *puški*) a treba da bude u klasi N70.05-* (dativ/lokativ jednine *pušci*) (4592), *svat* je u klasi N07.01+* (nominativ množine *svatovi*) a treba da bude i u klasi N01.01+* (nominativ množine *svati*) (4872), *molitva* je u klasi N72.01-* (genitiv množine *molitvi*) a treba da bude i u klasi N70.02-* (genitiv množine *molitava*) (5363);
  - glagoli: *predati* je u klasi V01.50.2* a treba da bude u klasi V06.50.2* (prvo lice jednine aorista, *predadoh* uz *predah*) (1769);
- *duži nastavci prideva.* Duži nastavci prideva koji završavaju na *-a*, *-e* i *-u* javljaju se za oblike neodređenog vida prideva u množini u dativu, instrumentalu i lokativu a za oblike određenog vida i u jednini u genitivu, dativu i lokativu za muški i srednji rod. Programski paket MORF [140] je generisao pridevske oblike korišćenjem samo kraćih nastavaka pa stoga određenom broju pridevskih oblika nije pridružena vrednost:
  - nastavak *-oga*: *pravoga* (173), *bolesnoga* (2808), *staroga* (3453);
  - nastavak *-ega*: *vražijega* (814), *tuđega* (2575), *večernjega* (3305);
  - nastavak *-omu*: *zdravomu* (2808), *ludomu* (3884), *mudromu* (5530);
  - nastavak *-ome*: *zdravome* (317);
  - nastavak *-emu*: *tuđemu* (2595), *mrzećemu* (3215), *zadnjemu* (4458);
  - nastavak *-ima*: *zdravima* (439), *mlađima* (3214);

  Prilikom generisanje pridevskih oblika, treba uključiti i oblike koji koriste duže nastavke u svim onim slučajevima gde su tako generisani oblici potpuno ravnopravni.
- *trpni pridev.* U registru ključnih reči u [72] naveden je izvestan broj prideva koji su istovremeno trpni glagolski pridevi odgovarajućih glagola: *blagosloven* od *blagosloviti*, *izjeden* od *izjesti*, *kršten* od *krstiti*, *nadut* od *naduti* i mnogi drugi. U uverenju da je

[5] Leksička reč *sedeti* navedena je u registru ključnih reči. Leksička reč *sjedjeti* bi možda bila bolje rešenje.

najveći deo trpnih glagolskih prideva naveden u registru kao zasebna ključna reč, a u želji da dimenzije konstruisanog rečnika budu takve da se on može koristiti na skromnim računarskim resursima koji su bili na raspolaganju, transduktori za opis glagolskih oblika nisu obuhvatali trpni glagolski pridev (videti 5.3.1). Izrada elektronskog teksta zasnovanog na integrisanom modelu je, međutim, pokazala da izvestan broj trpnih glagolskih prideva nije u registru prikazan kao ključna reč, pa njihovim oblicima nije dodeljena vrednost. Bez dodeljene vrednosti su ostali, na primer, oblici: *izglubljena* (5948) od *izgubiti*, *zaboravljeno* (5947) od *zaboravljati*, *hvaljene* (6017) od *hvaliti*, *oderano* (6067) od *oderati*, *naduvana* (6149) od *naduvati*, *udata* (6368) od *udati*. Koliko je pitanje izdvajanja trpnih glagolskih prideva u zasebne leksičke reči osetljivo, pokazuje primer poslovice 6149: *Čoek je kao naduta* (ili *naduvena* ili *napuhana*) *mješina*. U registru ključnih reči na tekstualnu reč *naduta* ukazuje pridev *nadut* a na tekstualnu reč *naduven* glagol *naduvati*. U [117] su i *nadut* i *naduven* trpni glagolski pridevi glagola *naduti*. Adekvatnoj obradi trpnih glagolskih prideva treba posvetiti dalja istraživanja.

Ranije je ukazano da se tekstualnim rečima, pretraživanjem rečnika DELAF, dodeljuje skup potencijalnih leksičkih reči koje se mogu realizovati kao ta tekstualna reč. Izrada elektronskog izdanja teksta Vukovih narodnih poslovica zasnovanog na integrisanom modelu je pokazala da od 42164 tekstualne reči kojima je pridružen skup vrednosti, 25468 ima pridruženu samo jednu leksičku reč, što čini 60% ukupnog broja reči. Preostale tekstualne reči predstavljaju slučajeve homografija. Tekstualne reči sa najvećim brojem pridruženih leksičkih reči su:

tekstualna reč	leksička reč i gramatičke kategorije
bega	bego.N40.01+*:msg+:msa+:mpg+
	beg.N07.02+*:msg+:msa+
	beg.N08.02-E:msg-
	beg.N01.02+*:msg+:msa+:mpg+
	begati.V01.00.2E:P3s:A2s:A3s
gore	gore.Adv*
	gora.N70.01-*:fsg-:fpn-:fpa-:fpv-
	goreti.V33.00.3E:P3p:A2s:A3s
	gorjeti.V33.25.4J:P3p
	rđav.A14.06*:k@mpa*:k@fsg*:k@fpn*:k@fpa*
	:k@fpv*:k@nsn*:k@nsa*:k@nsv*
	zao.A14.04*:k@mpa*:k@fsg*:k@fpn*:k@fpa*
	:k@fpv*:k@nsn*:k@nsa*:k@nsv*

Na broj leksičkih reči utiče broj morfoloških klasa koji se leksičkoj reči mogu pridružiti. Takav je primer leksičke reči *beg* sa značenjem *posednik u turskom carstvu* kojoj se mogu pridružiti morfološki kodovi N01.02+* i N07.02+* u zavisnosti od oblika množine. Takođe se tekstualnoj reči može pridružiti više varijanti leksičke reči. Takav je primer glagola *goreti* i *gorjeti*. Može se pretpostaviti da će dalje popunjavanje rečnika DELA/SJ uticati na smanjenje broja tekstualnih reči kojima se pridružuje samo jedna leksička reč.

U narednoj tabeli su predstavljeni svi skupovi potencijalnih vrednosti prema vrstama reči koji su dodeljeni tekstualnim rečima prilikom izrade elektronskog izdanja teksta Vukovih narodnih poslovica zajedno sa učestalošću njihovog pojavljivanja. Ovi podaci su od značaja za izradu postupaka automatskog sužavanja skupa potencijalnih vrednosti primenom lokalnih gramatika, pri čemu se sužavanje skupa vrednosti ne odnosi samo na broj leksičkih reči u njima

već i na pridružene gramatičke kategorije. U [95] je predložen jedan takav postupak koji se zasniva na slaganju predloga sa određenim padežima i taj postupak je primenjen na korpusu matematičkih tekstova. Da bi jedan takav postupak mogao biti primenjen, tekstualna reč mora biti prepoznata kao predlog. Podaci iz naredne tabele, koji pokazuju homografiju predloga sa oblicima drugih vrsta reči, pokazuju da je taj zadatak daleko složeniji za literarne nego matematičke tekstove.

A	2162	Adv	1251	Con Par	1934	N V	917
A A	10	Adv Adv	2	Con Par ProA	27	N V V	118
A A Adv	13	Adv Con	1143	Con Par ProA ProN	146	N V V V	24
A A Adv N V V	36	Adv Con Int	736	Con Par V	545	N V V V V	1
A A Adv V	13	Adv Con N	2	Con V V	14	Num	310
A A N	19	Adv Con Par ProA	27	Con V V V	42	Num Num	58
A A N V V	21	Adv Con ProA	90	Int	56	NumN	
A A V	1	Adv Con ProN	328	Int N	8	Par	1536
A Adv	477	Adv Con ProN ProN	8	Int N N	1	Par ProA	46
A Adv Int	9	Adv Int	37	Int N V	1	Par V	54
A Adv N	190	Adv N	115	Int Par	5	Pre	1497
A Adv Pre	23	Adv N Pre	139	Int Pre	1652	Pre V	38
A Adv Pre V	61	Adv N Pre V	2	Int ProN	25	Pre V V	14
A Adv ProA	1	Adv Par ProN	61	Int ProN ProN ProN	6	ProA	1037
A Adv V	129	Adv Pre	304	Int V	4	ProA ProA	36
A Col	1	Adv Pre V	3	N	8458	ProA ProA ProN ProN	50
A N	259	Adv ProA	287	N N	1263	ProA ProN	378
A N N	1	Adv ProA ProN	5	N N N	54	ProA ProN ProN	146
A N V	10	Adv ProA ProN ProN	8	N N N N V	2	ProA V	140
A N V V	10	Adv ProN	62	N N N Ord	26	ProN	2293
A NA	21	Adv V	5	N N N V	49	ProN ProN	653
A Num	2	Adv V V V	14	N N Ord	3	ProN ProN ProN	12
A Par	1	Col	12	N N Pre	46	ProN ProN V	370
A ProA	2	Con	409	N N V	67	ProN V	1415
A V	303	Con Int	1	N N V V	19	V	6276
A V V	25	Con Int	66	N Num	39	V V	781
NA	9	Con N N	101	N Ord	24	V V V	453
Ord	94	Con N N N	1	N Par	5	V V V V	10
		Con N N Par	2	N Pre	32		
		Con N Par	2	N ProN	8		

U tekstu Vukovih narodnih poslovica 3811 tekstualnih reči su potencijalno označene kao predlozi a od njih 1497 su dobile samo tu vrednost. Među njima su predlozi: *s*, *sa*, *k*, *za*, *zbog*, *od*, *kod*, *pred*, *vrh*[6], *svrh*, *savrh*, *navrh*, *dovrh*, *udno*, *nadno* i drugi. Preostalih 2314 tekstualnih reči su potencijalno označene, osim kao predlog, i kao pridev, prilog, imenica, glagol ili uzvik. Kao predlog i pridev je označena tekstualna reč *preko* a kao predlog i prilog *mjesto*, *naokolo*, *po*, *poslije*, *preko*, *put*, *pređe*, *kroz* i druge. Kao predlog i imenica označene su: *do*, *dno*, *među*, *pređe*, *oko*, *mjesto*, *čelo* i *kraj*. Kao predlog i glagol označene su tekstualne reči: *pređe*, *više*, *radi*, *nada* i *preda*. Kao predlog i uzvik označeni su *u*, *na* i *o*.

6.3 Primeri primene elektronskog teksta

U ovom odeljku biće predstavljeni neki primeri primene elektronskog teksta zasnovanog na integrisanom modelu. Svi primeri primene izrađeni su nad delom elektronskog teksta Vukovih

[6] Imenica *vrh* nije za sada u rečniku DĖLA/SJ.

narodnih poslovica koji sadrži poslovice koje počinju slovom **J**. U ovom delu je skup potencijalnih vrednosti pridružen tekstualnim rečima sveden na jednu realizovanu vrednost, što nije uvek moguće. Na primer, u poslovici *Ja ga ljubim, a on se utire*, tekstualna reč *ga* može se pridružiti i leksičkim rečima *on* i *ono*. Takođe, u poslovici *Jedna šugava ovca cijelo stado ošuga*, tekstualna reč *ošuga* može predstavljati oblik trećeg lica jednine i prezenta i aorista glagola *ošugati*. U nekim slučajevima nije se moglo razrešiti o kakvim se gramatičkim kategorijama radi, kao na primer u poslovici *Jaku brastvu jaka i pravdu* i *Ječam trče a rakija viče*. Osim toga, dodeljena je vrednost i onim tekstualnim rečima kojima nije bila pridružena nikakva vrednost iz razloga analiziranih u prethodnom odeljku.

- *podešavanje teksta.* Integrisani model elektronskog teksta omogućava podešavanje teksta prema unapred zadatim atributima jer je leksička reč koja je pridružena tekstualnoj reči zapisana korišćenjem leksikografema. Upotrebom postupka koji je vizuelno prikazan na slici u pododeljku 5.4.3, jedan varijantni oblik može se zameniti drugim. Taj postupak se može algoritamski formalizovati na sledeći način:

Neka je a atribut izabrane varijante
Neka je w izdvojena tekstualna reč
Neka je l pridružena leksička reč zapisana korišćenjem leksikografema
Neka je k morfološki kod leksičke reči
Neka je s skup pridruženih vrednosti gramatičkih kategorija
$w' = varij\_obl\_lg(l, k, s)$
w' je tekstualna reč zapisana korišćenjem leksikografeme
$w'' = varij\_obl(w', a)$
w'' je novi varijantni oblik tekstualne reči

U odeljku E.4.1 dodatka dat je tekst poslovica u kome su varijantni oblici koji potiču od zamene jata podešeni na ekavski izgovor a varijantni oblici koji potiču od izostavljanja ili zamene suglasnika *h* podešeni na suglasnik *h*. Moguće primene ovakvog postupka podešavanja teksta date su u [147].

- *lematizovane konkordancije.* Pojam konkordancija poznat je iz računarske literature ([141], [51], [52], [27]) u kojoj se koristi i alternativni termin KWIC — skraćenica od engleskog termina *keywords in context*. Premda su moguće različite realizacije i vizualne prezentacije [148], one se sve u osnovi svode na istu ideju: ključna reč je tekstualna reč uz koju se vezuju sva njena pojavljivanja u tekstu. Ključne reči se prikazuju u izabranom redosledu a svaka realizacija tekstualne reči u izabranom kontekstu (određen broj karaktera ili reči ulevo i udesno od ključne reči). Programi za izradu konkordancija predstavljaju tipičan primer programa koji su korisni za filološka i lingvistička istraživanja a koji tekst ne tretiraju kao jezički podatak već kao nisku karaktera.

  Da bi se sastavljene konkordancije mogle efikasno koristiti često se mora izvršiti njihova „lematizacija“, koja predstavlja postupak svođenja ključne reči na odgovarajuću leksičku reč. Bez pomoći odgovarajućih jezičkih programskih alata taj postupak se mora ručno obaviti. Tako su u prvoj fazi izrade registra ključnih reči lematizaciju konkordancija teksta Vukovih narodnih poslovica obavila dva leksikografa. Na osnovu elektronskog teksta zasnovanog na integrisanom modelu moguće je izraditi konkordancije u kojima ključna reč nije tekstualna već leksička reč. U pododeljku E.4.2 dodatka date su konkordancije dela Vukovih narodnih poslovica u kojima se ključna reč sastoji

od dva podatka: leksičke reči zapisane korišćenjem leksikografeme i vrste reči. Ovako redefinisana ključna reč obuhvata pojavljivanja tekstualnih reči koje bi inače mogle biti veoma udaljene u klasično izrađenim konkordancijama. Na primer, tekstualne reči *ja*, *mene*, *mi* okupljaju se oko zajedničke leksičke reči *ja*.

- *realizacija gramatičkih kategorija.* U [144] je ukazano na probleme izrade elektronskog rečnika na osnovu informacija pribavljenih iz tradicionalnih rečnika. Koncept elektronskog rečnika, kao što je rečeno u uvodu glave 5, zahteva ekspliciranje svih informacija, pre svega o oblicima promenljivih vrsta reči. Takve informacije u tradicionalnim rečnicima često nisu prisutne. Primer je genitiv množine imenice *molitva* koji je u tekstu poslovica realizovana kao *molitava*. Rečnik [117] ne govori ništa o oblicima ove imenice. Ovakva potreba za ekspliciranjem informacija o flektivnim oblicima može da dovede do unošenja nepotvrđenih oblika u elektronski rečnik. Sledeći eksperiment je pokušaj da se istraži koji se oblici leksičkih reči realizuju.

  Na osnovu onog dela elektronskog teksta Vukovih narodnih poslovica koji sadrži poslovice koje počinju slovom **J** izrađena je lista svih lema, koje predstavljaju leksičku reč sa pridruženim morfosintaksičkim kodom, kojima su pridružene realizovane vrednosti gramatičkih kategorija. Ovakav postupak primenjen na većem korpusu omogućio bi proveru realizacije pojedinih oblika i unošenje u rečnik DELA samo realizovanih oblika.

- *sintaksički obrasci.* Izgrađeni registar ključnih reči omogućava da se poslovice u zbirci od blizu 7000 poslovica pronađu prema ključnim rečima, a ne isključivo prema azbučnom redosledu. Proučavanje poslovica pokazuje da mnoge među njima imaju sličnu sintaksičku strukturu, a nekad i značenje, a da se razlikuju leksički. Stoga je zanimljiv eksperiment popisa sintaksičkih obrazaca. Oni su sastavljeni za Vukove narodne poslovice koje počinju slovom **J**. U obrascima su, osim vrste reči, navedene i gramatičke kategorije, i to:
  - za imenice, zamenice i brojeve, kategorije broja i padeža;
  - za glagole, kategorije vremena, lica i broja;
  - za prideve, kategorije broja, padeža, vida i stepen poređenja;

  U obrascima su, osim toga, navedeni i interpunkcijski znaci. Uz obrasce su navedeni i redni brojevi poslovica u kojima su realizovani. Na uzorku na kome je postupak primenjen, jedan obrazac se realizuje najviše tri puta. Ponavljanje obrasca može označavati sličnost leksike. Na primer:
  - 1772 Jak kao vjedogonja;
  - 1773 Jak kao zemlja;
  - 1774 Jak kao međed.

  Ove poslovice se zbog leksičke sličnosti i navode jedna do druge u tekstu poslovica. Poslovice mogu imati isti obrazac a leksički se razlikovati:
  - 1833 Jedna glava hiljada jezika;
  - 1841 Jedna smrt trista uzroka;
  - 1843 Jedna šteta sto grijeha.

Lista obrazaca ukazuje i na sličnost sintaksičkih obrazaca nekih poslovica:

- 1781 Ja mu tamo kolača ne spremam;
- 1807 Ja ti tamo kolač ne spremam.

Dalje primene elektronskog teksta izgrađenog na integrisanom modelu, kao što prethodni primeri pokazuju, obuhvataju indeksiranje teksta, transformaciju ili redukovanje njegovog sadržaja, recimo za potrebe formiranja baze apstrakata, kao i konstrukciju hiperteksta zasnovanog na dinamički formiranim vezama koje se, pak, zasnivaju na logičkoj strukturi teksta i njegovom leksičkom sadržaju.

Z A K LJ U ČA K

U radu je detaljno opisan i ispitan integrisani model teksta koji predstavlja pokušaj normalizovanja internog informatičkog zapisa teksta. Osnovni razlog za predloženu normalizaciju proističe iz potrebe objedinjavanja postupaka manipulacije, pretraživanja i formalnih transformcija zapisa teksta u različitim informatičkim primenama. Integrisani model teksta objedinjuje različite informatičke definicije teksta (odnosno, njegovog zapisa) sagledane iz ugla različitih obrada kojima ovakav objekt može biti podvrgnut. Modelom su, s jedne strane, obuhvaćena precizna sredtsva za opis logičke i grafičke strukture zapisa teksta, kao formalnog aparata pogodnog za izraz njegovog sadržaja (u smislu postojećih međunarodnih normi). S druge strane, predloženi model omogućava aproksimiranje jednog nivoa jezičke supstancije zapisa teksta uvođenjem posebnih formalnih sredstva za izraz njegovog grafemskog i leksičkog sadržaja uvođenjem sistema rečnika i formalnih gramatika različitog nivoa. Ovi rečnici i gramatike čine sastavni deo zapisa teksta.

Opis logičke i grafičke strukture teksta se ostvaruje kroz upotrebu jezika za obeležavanje izraženih SGML meta-jezikom. Opis grafemskog sastava se temelji na uvođenju pojma leksikografeme kao minimalne apstraktne jedinice kodiranja rečnika. Okvir za reprezentaciju leksičkih struktura pružila je metodologija višeslojnih sistema elektronskih rečnika DELA. Ovako modeliran elektronski zapis teksta zasniva se na metodama obrade karakterskih niski kao i na različitim metodama obrade regularnih izraza.

Opisani integrisani model elektronskog teksta omogućava da se reše raznovrsni problemi u obradi srpskog jezika, i na nivou obrade zapisa teksta, i na nivou obrade leksičkih jedinica. Osnovna crta koja povezuje ova dva nivoa se zasniva na mogućnosti da se istim mehanizmom konačnih automata (bez eksplicitnog opisa gramatičkih pravila) eksplicira strukturiranje zapisa teksta na suštinski različitim nivoima. Ova činjenica ne smanjuje značaj modeliranja sintaksičkih relacija: naprotiv, pojedini problemi razmatrani na nivou leksičke analize jasno ukazuju na potrebu za metodološki sličnim sistemom enkodiranja sintaksičkih informacija. U odnosu na metode koje se koriste za druge jezike, obrada zapisa teksta na srpskom jeziku ima svoje specifičnosti koje su ugrađene u predloženi model, sa mogućnošću da se primene i u nekim drugim situacijama (starofrancuski, i slično).

Kao eksperimentalan materijal za testiranje modela elektronskog teksta izabran je kompletna zbirka narodnih poslovica Vuka Stefanovića Karadžića. U prvoj fazi izrade elektronskog teksta, po prvi put, skoro 150 godina posle prvog izdanja, sastavljen je registar ključnih reči koji sadrži opis celokupanog leksičkog fonda poslovica. U drugoj fazi je sastavljen elektronski tekst poslovica u skladu sa integrisanim modelom koji predstavlja polaznu osnovu za njihovo dalje izučavanje, pre svega na sintaksičkom planu.

Potreba za profinjavanjem izloženog modela kao i za proširivanjem mogućnosti njegove primene dalje određuje pravce razvoja. Deo programske podrške koja obezbeđuje analizu logičke i grafičke strukture teksta se može dopuniti tako da prihvati punu SGML specifikaciju koja obuhvata i sve njegove dodatne mogućnosti.

Jedan od pravaca daljnjeg rada se sastoji u dopunjavanju leksičkog fonda sistema elektronskih rečnika dodatnim informacijama. U ovoj tački otvara se plodno područje računarsko-lingvističkih istraživanja: tradicionalni rečnici srpskog, koji su jedan od osnovnih izvora građe za sastavljanje elektronskih rečnika, ne sadrže potrebne informacije u formalizovanom obliku. Formiranje elektronskog korpusa savremenog jezika polazeći od integrisanog modela elektronskog teksta bi omogućio značajno drukčije prilaze tumačenju i obradi leksičke građe omogućavajući istovremeno značajne rezultate na području lingvističkih istraživanja, kao i neophodne instrumnente za efikasnu obradu ulaza na srpskom.

U predloženom modelu su leksičke informacije kao skup potencijalnih vrednosti pridružene tekstualnim rečima. Redukovanjem ovog skupa se znatno olakšava zadatak sintakičke analize. rezultat ovog istraživanja se prirodno može proširiti na područje lokalnih gramatika, kao podesnog informatičkog instrumenta za ekstenzivni opis i registrovanje sintaksičkih fenomena.

Iz informatičkog ugla izgradnja sistema lektronskih rečnika i njihova interakcija sa automatizovanim korpusima može, pred izstraživača, postaviti niz interesantnih i inspirativnih problema, počevši od pitanja integriteta ovakvog sistema, preko problema njegove efikasne orghanizacij i naročito, skladištenja i pretraživanja. Dalji informatički razvoj bi mogao doprineti da odgovori na ova pitanja bude od presudnog značaja za očuvanje upotrebne vrednosti pojedinih jezika, a time i srpskog, u okruženju informacionih tehnologija.

Literatura

[1] Adobe System Incorporated, *Postscript language tutorial and cookbook*, Addison-Wesley, Reading, MA, (1985)

[2] Association of American Publishers, *Standard for Electronic Manuscript Preparation and Markup* Version 2.0, Electronic Manuscript Porject, Washington, D.C., (1987)

[3] Aho, A. V., Ullman, J. D., *The Theory of Parsing, Translation, and Compiling*, Prentice-Hall, New Jersey, (1972)

[4] Aho, A. V., Corasick, M. J., *Efficient String Matching: An Aid to Bibliographic Search*, Communications of the ACM, Vol. 18, No. 6, pp. 333–340, (1975)

[5] Aho, A. V., Kernighan, B. W., Weinberger, P. J. *Awk — A Pattern Scanning and Processing Language*, Software—Practice and Experiance, Vol. 9, pp. 267–279, (1979)

[6] Aho, A. V., *Pattern Matching in Strings* in "Formal Language Theory", Academic Press, New York, pp. 325–347, (1980)

[7] Aho, A.V., Sethi, R., Ullman, J.D., *Compilers—Principles, Techniques, and Tools*, Addison-Wesley, Reading, Massachusetts, (1986)

[8] Aho, A. V., Kernighan, B. W., Weinberger, P. J. *The Awk Programming Language*, Addison-Wesley, Reading, Massachusetts, (1988)

[9] Amsler, R. A., Tompa, F. Wm., *An SGML-based Standard for English Monolingual Dictionaries*, in the Proc. of th Fourth Annual Conference of the UW Centre for the New Oxford English Dictionary "Information in Text", Waterloo, pp. 61–80, (1988)

[10] Atkins, S.B.T., *Tools for Computer-aided Corpus lexicography: The Hector Project* Papers in Computational Lexicography COMPLEX'92, (ed. by Kiefer, F.; Kiss, G.; Pajsz, J.), Linguistic Institut, Hungarian Academy of Sciences, Budapest, pp. 1–60, (1992)

[11] Baeza-Yates, R.; Gonnet, G. H., *A New Approach to Text Searching*, Communications of the ACM, Vol. 35, No. 10, pp. 74–82, (1992)

[12] Barić, E. i grupa autora, *Priručna gramatika hrvatskoga književnoga jezika*, Školaska knjiga, Zagreb, (1979)

[13] Barnard, D.T., Fraser, Ch.A., Logan, G.M., *Generalized Markup for Literary Texts*, Literary and Linguistic Computing, Oxford University Press, Vol. 3, No. 1, pp. 26–31, (1988)

[14] Belić, A., *Gramatika srpskohrvatskog jezika*, II izdanje, Izdavačko i knjižarsko preduzeće Geca Kon A.D., Beograd, (1934)

[15] Berg, D. L., Gonnet, G. H., Tompa, F. W., *The New Oxford Dictionary Project at the University of Waterloo*, in "Computational Lexicology and Lexicography: Special Issue Dedicated to Bernard Quemada", eds. A. Zampolli, L. Cignoni, and C. Peters, series Linguistica Computazionale Vol. VII, Giardini Editori, Pisa, pp. 29–44, (1991)

[16] Berghel, H., *The Web*, Communications of the ACM, Vol. 39, No. 1, pp. 33–40, (1996)

[17] *Le nouveau Bescherelle — l'art de conjuguer*, Hatier, Paris, (1966)

[18] Birnbaum, D.J., *Enkodiranje nelatiničnih alfabeta*, (prevod sa engleskog C. Krstev), ZADUŽBINA, godina III, broj 9, pp. 10, Beograd, (1990)

[19] Blake, G.E., Bray, T., Tompa, F.Wm., *Shortening the OED: Experience with a Grammar-Defined Database* ACM Transactions on Information Systems, Vol. 10, No. 3, (1992)

[20] Blake, G.E., Consens, M.P., Kilpeläinen, P., Larson, P.-Å., Snider, T., Tompa, F. W. *Text / Relational Database Management Systems: Harmonizing SQL and SGML*, Proc. Applications od Databases (ADB-94), Vadstena, pp. 267–280, (1994)

[21] Bonan-Garrigues, M., *Méthode de parametrage des dictionnaires et grammaires électroniques*, Thèse de doctorat, Université Paris 7, Laboratoire d'automatiques documentaire et linguistique, (1993)

[22] Boyer, R. S., Moore, J. S., *A Fast String Searching Algorithm*, Communications of the ACM, Vol. 20, No. 10, pp. 762–772, (1977)

[23] Bryan, M., *SGML — An Author's Guide to the Standard Generalized Markup Language*, Addison-Wesley, Workingham, England, (1988)

[24] Courtois, B., *Un systèm de dictionnaires électroniques pour les mots simples du français*, in: Dictionnaires électroniques du français, Langues française, No. 87, Larousse, Paris, (1990)

[25] Crochemore, M., Rytter, W., *Text Algorithms* Oxford University Press, New York Oxford, (1994)

[26] Day, C.A., *Using Ventura Publisher*, Oxford University Press, Oxford, (1988)

[27] Dej, K.A., *Računarska obrada teksta*, (prevod s engleskog C. Krstev), Nolit, Beograd, (1990)

[28] Erjavec, T., *Public Domain Generic Tools: An Overview* TELRI Proceedings of the First European Seminar "Language Resources for Language Technology" Tihany, Hungary, pp. 37–47, (1995)

[29] Farber, D. J., Griswold, R. E., Polonsky, I. P., *SNOBOL, A String Manipulation Language*, Journal of the Computing Machinery, Vol. 11, No. 2, pp. 21–30, (1964)

[30] Garside, R., Leech, G., Sampson, G., *The computational analysis of English: a corpus-based approach*, Longman, London & New York, (1987)

[31] Gimpel, J. F., *A Theory of Discrete Patterns and Their Implementation in SNOBOL4*, Communications of the ACM, Vol. 16, No. 2, pp. 91–100, (1973)

[32] Gimpel, J. F., *Algorithms in SNOBOL4*, John Wiley & Sons, New York, (1976)

[33] *GOEDEL*, (man pages), University of Waterloo, Center for the New Oxford English Dictionary

[34] Goldfarb, Ch.F., *A generalized approach to document markup*, Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, Oregon 1981, ACM SIGPLAN Notices, Vol. 16, Np. 6, pp. 68–73, (1981)

[35] Goldfarb, C.F., *Hytime: A standard for structured hypermedia interchange*, IEEE Computer, vol.24, No. 8, pp. 81–84, (1991)

[36] Gonnet, G. H., Tompa, F. W., *A Constructive Approachg to the Design of Algorithms and Their Data Structures*, Communications of the ACM, Vol. 26, No. 11, pp. 912–922, (1983)

[37] Gonnet, G. H., Tompa, F. W., *Mind Your Grammar: a New Approach to Modelling Text*, Proceedings of the 13th International Conference on Very Large Data Bases, pp. 339–346, Brighton, England, (1987)

[38] Gonnet, G.H., Baeza-Yates, R.A., Snider, T., *Lexicographic Indices for Text: Inverted files vs.* PAT *trees*, Tech. Rept. OED-91-01, UW Centre for the New OED and Text research, University of Waterloo, (1991)

[39] Gortan-Premk, D., *O gramatičkoj informaciji i semantičkoj identifikaciji u velikom opisnom rečniku*, Našjezik, Institut za srpskohrvatski jezik. Beograd, Vol. XXIV, No. 3, 107–114, (1980)

[40] Griswold, R. E., Poage, J. F., Polonsky, I. P., *The SNOBOL4 Programming Language*, Prentice-Hall, New Jersey, (1971)

[41] Griswold, R. E., Griswold, M. T., *A SNOBOL4 Primer*. Prentice-Hall, New Jersey, (1973)

[42] Griswold, R. E., *String and List Processing in SNOBOL4: Techniques and Applications*, Prentice-Hall, New Jersey, (1975)

[43] Griswold, R. E., Griswold, M. T., *The Icon Programming Language*, Prentice-Hall, New Jersey, (1983)

[44] Gross, M., *M/'ethodes en syntaxe*, Hermann, Paris, (1975)

[45] Gross, M., *Methods and Tactics in the Construction of Lexicon-grammar*, Selected papers from SICOL-86, Hanshin Publishing Company, Seoul, (1988)

[46] Gross, M.; Perrin, D. (eds.), *Electronic Dictionaries and Automata in Computational Linguistics*, Lecture Notes in Computer Science, no. 377, Springer-Verlag, Berlin (1989)

[47] Gross, M., *The use of Finite Automata in the Lexical Representation of Natural Languages*, in: [Gro89a], pp. 34–50, 1989

[48] Gross, M., *La construction de dictionnaires électroniques*, Annales des télécommunications, 44(1–2), pp. 4–19, (1989)

[49] Hall, P. A. V., Dowling, G. R., *Approximate String Matching* Computing Surveys, Vol. 12, No., 4, pp. 381–402, (1980)

[50] Herwijnen, E. van, *Practical SGML*, Kluwer Academic Publishers, Dordrecht/Boston/London, (1990)

[51] Hockey, S., *Programming for the Humanities* Clarendon Press, Oxford, (1985)

[52] Hockey, S., Martin, J., *The Oxford Concordance Program Version 2*, Literary and Linguistic Computing, Oxford University Press, Vol. 2, No. 2, pp. 125–131, (1987)

[53] Holub, A.I. *Compiler Design in C*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, (1990)

[54] Ide, N., *Encoding Standards for Linguistic Corpora*, TELRI Proceedings of the First European Seminar "Language Resources for Language Technology" Tihany, Hungary, pp. 65–78, (1995)

[55] Ide, N., Sperberg-McQueen, C.M., *The TEI: History, Goals, and Future*, Computers and the Humanities, Kluwer Academic Press, Vol. 29, pp. 5–15, (1995)

[56] International Organization for Standardization, *Information processing — ISO 7-bit coded character set for information interchange*, ISO 646, (1983)

[57] International Organization for Standardization, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, ISO 8879, (1986)

[58] International Organization for Standardization, *Information processing — Text and office systems — Office Document Architecture (ODA) and Interchange Formats*, ISO 8613, (1988)

[59] International Organization for Standardization, *Information processing — 8-bit Single Byte Coded Graphic Character sets — Part 5: Cyrillic Aphabet*, ISO 8859-5, (1987)

[60] International Organization for Standardization, *Information Processing — SGML nad Text-entry Systems — Guidelines for SGML Syntax-Directed Editing Systems*, ISO/IEC 10037, (1989).

[61] International Organization for Standardization, PUBLISHING SYSTEM, *SGML Application — Specification for Standards and Technical Reports*, ISO/PS 2, (1990)

[62] International Organization for Standardization, *Information Processing — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*, ISO/IEC 8824, (1990)

[63] International Organization for Standardization, *Information Technology — Hypermedia/Time-based structuring language (HyTime)* , ISO/IEC 10744, (1992)

[64] International Organization for Standardization, *Information Technology — Universal Multiple-octet Coded Character Set — (UCS), Part 1, Architecture and Basic Multilingual Plane*, ISO/IEC 10646-1, (1993)

[65] International Organization for Standardization, *Information Technology — Standard Music Description Language (SMDL)*, ISO/IEC 10743 JTC1 DIS

[66] International Organization for Standardization, *Information Processing — Text Composition — Standard Page Description Language*, ISO/IEC 10180, (1995)

[67] International Organization for Standardization, *Information Processing — Text Composition — Document Style Semantics and Specification Language* ISO/IEC 10179, (1996)

[68] Janković, Đ., Janković, N., *Generisanje glagolskih oblika u srpskohrvatskom jeziku*, Zbornik radova sa II naučnog skupa „Računarska obrada lingvističkih podataka", Institut „Jožef Stefan", Bled, pp. 223–232, (1982)

[69] Johnson, S.C., Yacc—Yet Another Compiler Compiler, Computer Science Technical Report 32, AT&T Bell Laboratories, Murray Hill, New Jersey, (1975)

[70] Jones, S., *Text and context: document storage and processing*, Springer-Verlag, Berlin (1991)

[71] Karadžić, V.S., *Srpske narodne poslovice*, PROSVETA—NOLIT, (1987)

[72] Karadžić, V.S., *Vukove narodne poslovice s registrom ključnih reči*, NOLIT, (1996)

[73] Kazman, R.N., *Strucutring the Text of the OED Through Finite State Transductions*, Tech. Report CS-86-20, Departement of Computer Science, University of Waterloo, Waterloo, Ontario, (1986)

[74] Klint, P., *A Study in String Processing Languages*, Lecture Notes in Computer Science, no. 205, Springer-Verlag, Berlin (1985)

[75] Knuth, D.E., *The Art of Computer Programming: Sorting and Searching*, vol. 3, Addison-Waseley, Reading, Mass., (1973)

[76] Knuth, D. E., Morris, J. H., Pratt, V. R, *Fast Pattern Matching in Strings*, SIAM J. Computing, Vol. 6, Num. 2, pp. 323–350, (1977)

[77] Knuth, D. E.: *TeXbook*, Addison-Wesley, Reading, Massachusetts, (1984)

[78] Krstev, C., *Frekvencijski rečnik konsonantskih grupa u srpskohrvatskom jeziku i problem rastavljanja na slogove*, Zbornik radova sa II naučnog skupa „Računarska obrada lingvističkih podataka", Institut „Jožef Stefan", Bled, pp. 389–404, (1982)

[79] Krstev, C., *Rastavljanje reči srpskohrvatskog jezika na kraju retka*, Zbornik sa III naučnog skupa „Računarska obrada jezičkih podataka", Institut „Jožef Stefan", Bled, pp. 289–301, (1985)

[80] Krstev, C., *Programski sistemi za obradu teksta*, Magistarska teza, Matematički fakultet, Univerzitet u Beogradu, (1989)

[81] Krstev, C., *Serbo-Croatian hyphenation: A TEXpoint of view*, TUGboat, Vol. 12, No. 2, pp. 215–223, (1991)

[82] Krstev, C., *Algoritmi za sravnjivanje niski karaktera*, RAČUNARSTVO, Broj 2, Matematički institut, Beograd, (1993)

[83] Krstev, C., Vitas, D., *Konkordancije paralelizovanih tekstova*, Zbornik radova sa XXXVIII Konferencije ETRAN-a, pp. 229–230, Niš, (1994)

[84] Krstev, C., Pavlović-Lažetić, G., Vitas, D., *Neutralization of Variations in the Structure of a Dictionary Entry in Serbo-Croatian*, Proceedings of the First European Conference on Formal Description of Slavic Languages, Universität Leipzig, u štampi, (1995)

[85] Laport, E., *Experiances with Lexical Disamiguation Using Local Grammars* Papers in Computational Lexicography COMPLEX'94, (ed. by Kiefer, F.; Kiss, G.; Pajsz, J.), Research Institut for Linguistics, Hungarian Academy of Sciences, Budapest, pp. 163–172, (1994)

[86] Lesk, M.E., *Lex—a lexical analyzer generator*, Computer Science Technical Report 39, AT&T Bell Laboratories, Murray Hill, N.J., (1975)

[87] Liang, F. M., *Word Hy-phen-a-tion by Com-put-er*, Ph. D. Thesis, Department of Computer Science, Stanford University, Report No. STAN-CS-83-977, (1983)

[88] Logan, H. M., Logan, G, *An Inquiry into Inquiry System: A Discussion of Some Applications of Data Rerieval to the New OED Database*, in the Proc. of th Fourth Annual Conference of the UW Centre for the New Oxford English Dictionary "Information in Text", Waterloo, pp. 81–95, (1988)

[89] Mair, Ch., *Machine-Readable Text Corpora and the Linguistic Description of Language*, Proceedings of the Conference "Text Analysis and Computers", Mannheim, pp. 65–76, (1995)

[90] Mergenthaler, E., *Computer-Assisted Content Analysis*, Proceedings of the Conference "Text Analysis and Computers", Mannheim, pp. 3–32, (1995)

[91] Maurer, W. D., *The Programmer's Introduction to SNOBOL*, Elsevier, New York, (1976)

[92] Malkov, S., *Prepoznavanje berojeva*, seminarski rad na specijalnom postdiplomskom kursu „Otvorena pitanja u obradi srposkog jezika"kod D. Vitasa, Matematički fakultet, Beograd, (1996)

[93] Mohri, M., *Syntactic Analysis by Local Grammars Automata: an Efficient Algorithm* Papers in Computational Lexicography COMPLEX'94, (ed. by Kiefer, F.; Kiss, G.; Pajsz, J.), Research Institut for Linguistics, Hungarian Academy of Sciences, Budapest, pp. 179–192, (1994)

[94] Nenadić, G., Vitas, D., *Pravopis kao okvir za informatičku standardizaciju jezika*, Zbornik radova sa skupa „Standardizacija i kvalitet u informacionim tehnologijama", Beograd, (1995)

[95] Nenadić, G., *Algoritmi prepoznavanja složenih reči u matematičkom tekstu i njihove primene*, Magistarska teza, Matematički fakultet, Beograd, (1997)

[96] Newcomb, S.R., Kipp, N.A., Newcomb, V.T., *The HyTime Hypermedia/Time-based Document Structuring Language*, Communications of the ACM, Vol. 34, No. 11, (1991)

[97] Newsted, P. R., *SNOBOL — An Introduction to Programming*, Hayden Book Company, New Jersey, (1975)

[98] Oracle Corporation *SQL*TextRetrieval, Developer's Guide*, Version 1.1, (1989)

[99] Parezanović, N., *Fortran 90*, IP Nauka, Modularna biblioteka računarstva, Beograd, (1993)

[100] Paterson, J.L., *Computer Programs for Spelling Correction*, Lecture Notes in Computer Science, no. 96, Springer-Verlag, Berlin, (1983)

[101] Pavlović-Lažetić, G., *Baze podataka i ekspertni sistemi u upravljanju tekstom*, doktorska disertacija, Prirodno-matematički fakultet, Univerzitet u Beogradu, Beograd, (1987)

[102] Pavlović-Lažetić, G., *Morfološka analiza vođena pravilima*, u zborniku sa naučnog skupa „Matematička i računarska lingvistika", urednik D. Vitas, Društvo za primenjenu lingvistiku Srbije, Beograd, pp. 136–141, (1990)

[103] Pavlović-Lažetić, G., *Osnove relacionih baza podataka*, VESTA — Matematički fakultet, Beograd, (1996)

[104] PC Magazine, special issue on Electronic Publishing, February 7, (1995)

[105] Perrin, D., *Automates et algorithmes sur les mots*, Annales des télécommunications, Vol. 44, No.1–2, pp. 20–33, (1989)

[106] Pešikan, M., *Još o klasifikaciji srpskohrvatskih glagola*, Našjezik, Institut za srpskohrvatski jezik, Beograd, Vol. XXIV, No. 3, 139–146, (1980)

[107] Pešikan, M., Jerković, J., Pižurica, M. Pravopis srpskoga jezika Matica srpska, Novi Sad, (1993)

[108] Popova, T.P., Neka razmišljanja o klasifikaciji i o promeni glagola u savremenom srpskohrvatskom jeziku, Našjezik, Institut za srpskohrvatski jezik, Beograd, Vol. XXIV, No. 3, 129–138, (1980)

[109] Platon, *Država* Beogradski izdavačko-grafički zavod, Beograd, (1983)

[110] *Pravopis srpskohrvatskog književnog jezika sa pravopisnim rečnikom*, izradila Pravopisna komisija, Matica srpska — Matica hrvatska, Novi Sad — Zagreb, (1960)

[111] Prešić, M., *Formalne gramatike*, Računarstvo 1(2), Matematički institut, Beograd, pp. 1–24, (1992)

[112] Raggett, D., *Hypertext Markup Language Specification, version 3.0*, working document of Internet Engineering Task Force (IETF), (1995)

[113] Raymond R.D., Tompa, F.Wm., *Hypertext and the Oxford English Dictionary*, Communications of the ACM, Vol. 31, No. 7, pp. 871–879, (1988)

[114] Raymond R.D., Tompa, F.Wm., Wood, D., *Markup Reconsidered*, 1st Inetrnational Workshop on Principles of Document Processing, Washington, D.C., (1992)

[115] Raymond R.D., Tompa, F.Wm., Wood, D., *From Data Representation to Data Model: Meta-Semantic Issues in the Evolution of SGML*, Computer Standards & Interfaces, Vol. 18, pp. 25–36, (1996)

[116] — *Rečnik srpskohrvatskoga književnog jezika*, Matica srpska—Matica Hrvatska, Novi Sad—Zagreb, (1967)

[117] — *Rečnik srpskohrvatskog književnog i narodnog jezika*, vol. 1–14 (A–N), Srpska akademija nauka i umetnosti, Institut za srpskohrvatski jezik, Beograd, 1959–1990

[118] Roche, E., *Looking for Syntactic Patterns in Texts*, Papers in Computational Lexicography COMPLEX'94, (ed. by Kiefer, F.; Kiss, G.; Pajsz, J.), Research Institut for Linguistics, Hungarian Academy of Sciences, Budapest, pp. 279–288, (1994)

[119] Salminen, A., Tompa, F., *PAT expressions: an algebra for text searching*, Acta Linguistica Hungarica, Vol. 41, No. 1–4, pp. 309–332, (1994)

[120] Salminen, A., Tompa, F.,Wm., *Grammars++ for Modelling Information in Text*, University of Waterloo, Computer Science Department, CS–96–40, (1996)

[121] Sampson, G., *How Fully Does a Machine-Usable Dictionary Cover English Text?*, Literary and Linguistic Computing, vol. 4, No. 1, pp.29–35, (1989)

[122] Savezni zavod za standardizaciju, *Skup znakova za razmenu podataka kodiranih sa 7 bitova za srpskohrvatsko latinično pismo*, JUS I.B1.002, (1986)

[123] Savezni zavod za standardizaciju, *Jedinice za unos podataka — Tastatura sa 47 tipki za slovenačko i hrvatsko latinično pismo*, JUS I.K1.002, (1986)

[124] Sedgewick, R., *Algorithms*, Addison-Wesley, Reading, Massachusetts, (1983)

[125] Sema Group Belgium S.A., *The MARK-IT Manual*, (1990)

[126] Sema Group S.A., *The WRITE-IT Manual*, (1990)

[127] Silberztein, M. D., *The Lexical Analysis of French*, in: [Gro89a], pp. 93–110, (1989)

[128] Silberztein, M. D., *Dictionnaires électronique et analyse automatique de textes (Le système INTEX)*, MASSON, Paris, (1993)

[129] Sperberg-McQueen, C.M., Burnard, L., *Guidelines for Electronic Text Encoding and Interchange*, Text Encoding Initiative, Chicago–Oxford, (1994)

[130] Sperberg-McQueen, C.M., Burnard, L. *The Design of the TEI Encoding Scheme*, Computers and the Humanities, Kluwer Academic Press, Vol. 29, pp. 17–39, (1995)

[131] Stanojčić, Ž., Popović, Lj., Micić, S., *Savremeni Srpskohrvatski jezik i kultura izražavanja*, Zavod za udžbenike i nastavna sredstva, Beograd, Zavod za izdavanje udžbenika, Novi Sad, (1989)

[132] Stevanović, M., *Savremeni srpskohrvatski jezik*, Naučna knjiga, Beograd, (1969)

[133] Teubert, W., *Language Resources: The Foundations of a Pan-European Information Society*, TELRI Proceedings of the First European Seminar "Language Resources for Language Technology" Tihany, Hungary, pp. 105–128, (1995)

[134] Tompa, F. Wm., *What is (Tagged) Text?*, "Dictionaries in the Electroinc Age" Proc. 5th Conf. of Univ. of Waterloo Centre for the New OED, Oxford, England, pp. 81–93, (1989)

[135] Tompa, F. Wm., Raymond, D. R., *Database design for a Dynamic Dictionary*, Research in Humanities Computing I: Papers from the 1989 ACH-ALLC Conference, Ian Lancashire (ed.), Oxford University Press, pp. 257–272, (1991)

[136] Tompa, F. Wm., *Recovering Information from Stored Text*, Transactions of the Royal Society of Canada, Series VI, Vol. V, pp. 105–109, (1994)

[137] Tompa, F. Wm., *Not Just Another Database Project: Developments at UW*, Proc. 10th Conf. of Univ. of Waterloo Centre for the New OED, pp. 82–89, (1994)

[138] Vitas, D., *Podela na slogove srpskohrvatskih reči*, Informatika, Ljubljana, 3(1981)

[139] Vitas, D., *Generisanje imeničkih oblika u srpskohrvatskom jeziku*, Informatica, Ljubljana, pp. 49–55, 3(1981)

[140] Vitas, D., *Generisanje pridevskih oblika*, Informatica, Ljubljana, pp.54–58, 1(1982)

[141] Vitas, D., *Prikaz jednog programskog sistema za automatsku obradu teksta*, Zbornik radova sa II naučnog skupa „Računarska obrada lingvističkih podataka", Institut „Jožef Stefan", Bled, pp. 457–465, (1982)

[142] Vitas, D., *Ka programu za automatsko menjanje i dopunjavanje zakonskih tekstova*, Zbornik radova sa II naučnog skupa „Računarska obrada lingvističkih podataka", Institut „Jožef Stefan", Bled, pp. 467–478, (1982)

[143] Vitas, D., Krstev, C., *Interaction between Dictionary and Text in Serbo-Croatian*, Papers in Computational Lexicography COMPLEX'92, (ed. by Kiefer, F.; Kiss, G.; Pajsz, J.), Linguistic Institut, Hungarian Academy of Sciences, Budapest, pp. 333–342, (1992)

[144] Vitas, D., *Matematički model morfologije srpskohrvatskog jezika (imenska fleksija)*, doktorska disertacija, Matematički fakultet, Univerzitet u Beogradu, Beograd, (1993)

[145] Vitas, D., Pavlović-Lažetić, G., Krstev, C., *Electronic Dictionary and Text Processing in Serbo-Croatian*, Linguistische Arbeiten, Max Niemeyer Verlag, pp. 225–231, Tübingen, (1993)

[146] Vitas, D., Krstev, C., Pavlović-Lažetić, G., *Dictionary Layers Underlying Electronic Texts*, Proceedings of the Conference "Text Analysis and Computers", Mannheim, (1995)

[147] Vitas, D., Krstev, C., *Tuning the Text with an Electronic Dictionary*, Papers in Computational Lexicography COMPLEX'96, (ed. by Kiefer, F.; Kiss, G.; Pajsz, J.), Linguistic Institut, Hungarian Academy of Sciences, Budapest, pp. 267–276, (1996)

[148] Vučković V., Krstev, C., Lalović, I., Uporedna analiza programskih paketa za izradu konkordanci, Zbornik sa IV naučnog skupa „Računarska obrada jezičkih podataka", Institut „Jožef Stefan", Portorož, pp. 245–248, (1988)

[149] Wall,L., Schwartz, R.L., *Programming Perl*, O'Reilly & Associates, Sebastopol, California, (1991)

[150] Warmer, J., van Egmond, S., *The Implementation of the Amsterdam SGML Parser*, Electronic Publishing, Origination, Dissemination and Design, J. Wiley & Sons, Ltd., Chichester, pp. 65–90, Vol. 2, No. 2, (1989)

[151] Wu, S., Manber, U., *Fast Text Searching Allowing Errors*, Communications of the ACM, Vol. 35, No. 10, pp. 83–91, (1992)

[152] Wu, S., Manber, U., Myers, E., *A Subquadratic Algorithm for Approximate Regular Expression Matching*, Journal of Algorithms, Vol. 19, pp. 346–360, (1995)

[153] Cover, R., *SGML Web Page*, http: //www.sil.org/sgml/sgml.html

[154] Zhang, J. *OODB and SGML Techniques in Text Database: An Electronic Dictionary System*, SIGMOD RECORD, Vol. 24, No. 1, pp. 3–8, (1995)

Dodatak A

Primer SGML dokumenta

```
<!-- Početak SGML deklaracije                                          -->
<!SGML "ISO 8879:1986"
CHARSET
  -- Dokument koristi dva karakterska skupa                           --
BASESET
  "ISO 646-1983//CHARSET International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET  0       9     UNUSED
         9       2     9
         11      2     UNUSED
         13      1     13
         14      18    UNUSED
         32      95    32
         127     1     UNUSED
BASESET
  "ISO Registration Number 100//
   CHARSET ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/13 4/1"
DESCSET  128     32    UNUSED
         160     95    32
         255     1     UNUSED
  -- Kapaciteti su iz referentne konkretne sintakse                   --
CAPACITY PUBLIC "ISO 8879:1986//CAPACITY Reference//EN"
  -- Sintaksa se odnosi deklaraciju tipa dokumenta i
     na instanciju dokumenta                                          --
SCOPE    DOCUMENT
  -- Definicija varijantne sintakse                                   --
SYNTAX
  -- Kontrolni i ostali ne-SGML karakteri                             --
SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
         16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
         127 255
  -- Konkretna sintaksa za označavanje koristi
     karakterski skup ISO 646                                         --
BASESET
  "ISO 646-1983//
   CHARSET International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET   0      128      0
  -- Funkcionalni karakteri su kraj sloga, početak sloga,
     blanko i tabulator (separator)                                   --
```

```
FUNCTION RE      13
         RS      10
         SPACE   32
         TAB     SEPCHAR 9
  -- Dozvoljenim karakterima imena dodaju se crtica i tačka        --
NAMING   LCNMSTRT ""
         UCNMSTRT ""
         LCNMCHAR "-."
         UCNMCHAR "-."
  -- Mala i velika slova u imenima su ekvivalentna                 --
         NAMECASE GENERAL YES
  -- osim imena entiteta                                           --
                  ENTITY  NO
  -- Opšti graničnici i graničnici kratkih referenci
     su kao u referentnoj konkretnoj sintaksi                      --
DELIM    GENERAL SGMLREF
         SHORTREF SGMLREF
  -- Rezervisana imena su kao u referentnoj konkretnoj sintaksi    --
NAMES    SGMLREF
  -- Veličine su kao u referentnoj konkretnoj sintaksi, osim
     sledećih šest koje se povečavaju                          --
QUANTITY SGMLREF
         ENTLVL   32
         NAMELEN  32
         LITLEN   2048
         GRPCNT   100
         GRPGTCNT 180
         TAGLVL   24
  -- U dokumentu su dozvoljena sledeća opciona svojstva:
     izostavljanje etiketa, skraćivanje etiketa, i
     poddokumenta (najviše 10)                                   --
FEATURES
MINIMIZE DATATAG NO     OMITTAG  YES    RANK     NO     SHORTTAG YES
LINK     SIMPLE  NO     IMPLICIT NO     EXPLICIT NO
OTHER    CONCUR  NO     SUBDOC   YES 10 FORMAL   NO
APPINFO NONE>
<!-- Kraj SGML deklaracije                                        -->

<!-- Poč etak definicije tipa dokumenta DTD                        -->
<!DOCTYPE SNTPI.prilog [
<!-- Notacija koja se korisiti za slike u tekstu                   -->
<!NOTATION bit.map SYSTEM "skanirana slika"                        >
<!-- Deklaracije parametarskih entiteta                            -->
<!-- Fraze koje su delovi paragrafa                                -->
<!ENTITY % fraza                "#PCDATA | istaknuto | termin  "   >
<!ENTITY % INHERITED            "#IMPLIED"                         >
<!-- Atributi koji se mogu pridružiti (skoro svim) elementima      -->
<!ENTITY % opsti.atributi       "
           jezik           CDATA          %INHERITED
           izgled          CDATA          #IMPLIED
           identifikacija  ID             #IMPLIED        "        >
<!-- Deklaracije opštih entiteta                                   -->
<!-- Referisanje datoteka koje sadrže slike                        -->
```

```
<!ENTITY slika1  SYSTEM          "c:slika.bmp"  SDATA   bit.map          >
<!-- Fraze sadrže parsibilne karakterske podatke                          -->
<!ELEMENT (istaknuto, termin)
                              - -  (#PCDATA)                              >
<!-- Fusnota se sastoji iz fraza; ne može sadržati druge fusnote          -->
<!ELEMENT fusnota             - -  ((%fraza;)*)       -(fusnota)          >
<!-- Sadržaj rada se sastoji od paragrafa, tabela i slika                 -->
<!ELEMENT sadrzaj             - o  ((paragraf | tabela | slika)*)         >
<!-- Paragraf se sastoji od fraza, lista i referenci literature           -->
<!ELEMENT paragraf            o o  (%fraza; | lista |
                                    ref.literature)+                      >
<!-- Lista se sastoji od bar jednog člana liste                           -->
<!ELEMENT lista               - -  ((clan.liste)+)                        >
<!-- Elementi liste mogu biti automatski oznaceni slovima ili brojevima-->
<!-- označeni nekim istim specijalnim znakom ili neoznačeni              -->
<!ATTLIST lista               %opsti.atributi;
          tip                     (alfa, numer, nenumer, jednostavna)
                                  jednostavna                             >
<!-- Tabela se sastoji od neobaveznog naslova i bar jedne vrste          -->
<!ELEMENT tabela              - o  (naslov?, (vrsta.tabele)+)             >
<!-- Tabele se obavezno numerišu, a može im se unapred zadati            -->
<!-- za potrebe aplikacije broj vrsta i kolona                           -->
<!ATTLIST tabela                   %opsti.atributi;
          n                  NUMBER          #REQUIRED
          broj.vrsta         NUMBER          #IMPLIED
          broj.kolona        NUMBER          #IMPLIED        >
<!-- Vrsta tabele se sastoji od jednostavne ćelije ili tabele            -->
<!ELEMENT vrsta.tabele        - o  ((celija | tabela)+)                   >
<!-- ćelija i član liste mogu sadržati fraze                             -->
<!ELEMENT (celija,clan.liste) - o  ((%fraza;)*)                           >
<!-- Za ćeliju tabele može se zadati na koliko vrsta odn.                -->
<!-- kolona se prostire                                                  -->
<!ATTLIST celija              %opsti.atributi;
          br.vrsta            NUMBER          1
          br.kolona           NUMBER          1               >
<!-- Slika se satoji od neobaveznog naslova i sadržaja slike             -->
<!ELEMENT slika               - o  (naslov?, sadrzaj.slike)               >
<!ATTLIST slika                    %opsti.atributi;
          n                   NUMBER          #REQUIRED       >
<!-- Slike se obavezno numerišu                                          -->
<!-- Sadržaj slike je u nekoj datoteci čije je ime preko entiteta        -->
<!-- pridruženo atributu teka                                            -->
<!ELEMENT sadrzaj.slike       - o  EMPTY                                  >
<!-- Za sliku se može zadati prostor koji ona zauzima                    -->
<!ATTLIST sadrzaj.slike       x.kor    NMTOKEN sirina.teksta
                              y.kor    NUTOKEN #IMPLIED
                              teka     ENTITY  #REQUIRED                  >
<!-- Struktura rada VI naučnom skupu SNTPI je sledeća                    -->
<!ELEMENT SNTPI.prilog        - -  (o.autoru & rad)                       >
<!-- Podaci o autoru sadrži ime autora, belešku o njemu i                -->
<!-- i njegovoj instituciji                                              -->
<!ELEMENT o.autoru            - o  (autor, beleska, institucija)          >
<!ELEMENT autor               o o  (titula?, ime)                         >
```

```
<!-- Beleška i prvi apstrakt sadrže bar jedan paragraf                    -->
<!ELEMENT (beleska,apstrakt1)  - o  ((paragraf)+)                          >
<!-- Podaci o instituciji sadrže njen naziv i adresu                      -->
<!ELEMENT institucija          - o  (naziv, adresa)                        >
<!-- Adresa sadrži poštanski kod, ime grada, ulicu i neobavezno           -->
<!-- državu navedene u proizvoljnom redosledu                             -->
<!ELEMENT adresa               - o  (kod & grad & ulica & drzava?)         >
<!-- Rad se sastoji od imena autora, naslova, prvog i drugog apstrakta,   -->
<!-- teksta rada, neobavezne literature. Fusnote se u okviru rada         -->
<!-- mogu ma gde pojaviti                                                 -->
<!ELEMENT rad                  - o  (autor, naslov, apstrakt1,
                                     apstrakt2, tekst.rada, literatura?)
                                     +(fusnota)                            >
<!-- Drugi apstrakt se sastoji od naslova i bar jednog paragrafa          -->
<!ELEMENT apstrakt2            - o  (naslov, (paragraf)+)                  >
<!-- Tekst rada se sastoji od bar jednog odeljka                          -->
<!ELEMENT tekst.rada           - o  ((odeljak)+)                           >
<!-- Odeljak se sastoji od naslova iza koga sledi neobavezni sadržaj      -->
<!-- praćen bar jednim pododeljkom                                        -->
<!ELEMENT odeljak              - o  (naslov,
                                     ((sadrzaj)?, (pododeljak)*))          >
<!-- Pododeljak se sastoji od naslova iza koga sledi sadržaj              -->
<!ELEMENT pododeljak           - o  (naslov, (sadrzaj) )                   >
<!-- Odeljak, pododeljak i fusnota se obavezno numerišu                   -->
<!ATTLIST (odeljak, pododeljak,fusnota)     %opsti.atributi;
                  n                         NUMBER              #REQUIRED   >
<!-- Literatura se sastoji od bar jednog člana                            -->
<!ELEMENT literatura           - o  ((clan.literature)+)                   >
<!-- član literature se sastoji od neobaveznog jednog ili više autora,    -->
<!-- naslova, vrste publikacije i neobaveznih podataka o izdavaču,        -->
<!-- mestu izdavanja, referentnim stranicama i vremenu izdavanja          -->
<!ELEMENT clan.literature      - o  ((autor)*, naslov, publikacija,
                                     izdavac?, mesto.izdavanja?,
                                     stranice?, vreme.izdavanja?)          >
<!-- Č lan literature mora obavezno da bude označen jedinstvenim          -->
<!-- identifikatorom                                                      -->
<!ATTLIST clan.literature   jezik           CDATA           %INHERITED
                            izgled          CDATA           #IMPLIED
                            identifikacija  ID              #REQUIRED   >
<!-- Preko tog identifikatora se literatura referiše u radu              -->
<!ELEMENT ref.literature       - o  EMPTY                                  >
<!ATTLIST ref.literature    vidi    IDREF       #REQUIRED                  >
<!ELEMENT mesto.izdavanja      - o  (grad, drzava?)                        >
<!ELEMENT publikacija          - o  (knjiga | zbornik | casopis)           >
<!ELEMENT casopis              - o  (naslov, casopis.tom, casopis.broj)    >
<!-- Svi ovi elementi sadrže parsibilne karakterske podatke, a           -->
<!-- početna i završna etiketa im se mogu u zavisnoti od konteksta       -->
<!-- izostaviti                                                           -->
<!ELEMENT (titula, ime, naziv, kod, grad, ulica, drzava, naslov,
           izdavac, stranice, vreme.izdavanja, knjiga, zbornik,
           casopis.tom, casopis.broj)
    o o  (#PCDATA)     >
<!-- Svi ovi elementi imaju opšte atribute                                -->
```

```
<!ATTLIST (SNTPI.prilog, o.autoru, rad, autor, beleska, institucija,
          adresa, titula, ime, naziv, kod, grad, ulica, drzava,
          naslov, apstrakt1, apstrakt2, tekst.rada, literatura,
          paragraf, istaknuto, termin, clan.liste,
          publikacija, izdavac, mesto.izdavanja,
          stranice, vreme.izdavanja, knjiga, zbornik, casopis,
          casopis.tom, casopis.broj, vrsta.tabele)
   %opsti.atributi;    >
]>
<!-- Kraj definicije tipa dokumenta                                    -->

<!-- Poč etak instancije dokumenta                                     -->
<SNTPI.prilog>
<rad><autor><titula>Prof.dr<ime>Petar Petrović
<naslov>NASLOV RADA NA SRPSKOM JEZIKU
<apstrakt1>U ovom uputstvu opisano je na koji način bi trebalo pripremiti
   radove za Zbronik.
<apstrakt2 jezik=engleski><naslov>NASLOV RADA NA ENGLESKOM JEZIKU</naslov>
We suggest your papaers to be prepared in form of this tamplate. The abtsract
in English could take about 15 rows.
<tekst.rada><odeljak n=1><naslov>Uvod
<sadrzaj><paragraf>Radove pisati na srpskom ili engleskom jeziku na listu ...
... Radove dostaviti: <lista tip=alfa><clan.liste>na disketi, otkucane na
<istaknuto>WORD 6.0 for WINDOWS</istaknuto> i</clan.liste>
<clan.liste><istaknuto>2 (dva)</istaknuto> odštampana primerka rada;
najkasnije do <istaknuto>15.04.1996.g.</istaknuto></lista>
<odeljak n=2><naslov>Prvi i drugi podnaslov
<sadrzaj><paragraf>Na sredini prve stranice, nakon dva prazna reda (posle
zaglavlja) napisati titulu, ime i prezime autora<fusnota n=1>Kratku belešku
o autoru sa ....na posebnoo listu u prilogu</fusnota>. Nakon dva prazna reda...
<paragraf>Za imena autora koristiti...
<pododeljak n=1><naslov>Apstrakti
<sadrzaj><paragraf>Ispod naslova rada...
<odeljak n=2>Zaglavlje, fusnote i naglaseni delovi teksta
<sadrzaj><paragraf>U zaglavlju treba da stoji naziv rada...
<pododeljak n=1>Tabele
<sadrzaj><paragraf>U tekstu se mogu koristiti tabele...
<tabela n=2 broj.vrsta=6 broj.kolona=5>
<naslov>Upore|enje tehnologija sanajznačajnijim
<vrsta.tabele><celija>Sopstvene tehnologije su:
              <celija>Prijektovanje i priprema
              <celija>Proizvodnja
              ...
<vrsta.tabele><celija></celija>
              <celija>MPI UZR
              <celija>MPI UZR
              ...
<slika n=1><naslov>Primer koriscenja slike koje u uputstvu nema
<sadrzaj.slike teka=slika1>
<odeljak n=3>Literatura
<sadrzaj><paragraf>Navesti samo naslove koji su direktno...
...Na primer: u <ref.literature vidi=MrNOBODY> se koristi...
<literatura>
```

```
<clan.literature identifikacija=MrNOBODY>
<naslov>ACM Code of Ethics and Professional Conduct
<publikacija><casopis>Communications of the ACM
<casopis.tom>vol. 35<casopis.broj>no. 5/95
<stranice>pp. 94-100
<vreme.izdavanja>1992
<clan.literature identifikacija=Forester92>
<autor>T. Forester
<autor>P. Morrison
<naslov>Computer Ethics, Cautrionary Tales and Ethical Dilemmas in Computing
<publikacija><knjiga>
<izdavac>The MIT Press
<mesto.izdavanja>Cambridge
<vreme.izdavanja>1994
<o.autoru>
<titula>mr<ime>Jovan J. Jovanović
<beleska>Ovaj autor je veoma vredan.
<institucija>Mocna Firma
<adresa><kod>11000<grad>Beograd
        <ulica>Fina
</SNTPI.prilog>
<!-- Kraj instancije dokumenta                                    -->
```

Dodatak B

Jedna TEI aplikacija

U ovom odeljku je predstavljena jedna TEI aplikacija koja je razvijena za potrebe elektronskog izdanja Vukovih narodnih poslovica, koje je nastalo na osnovu izdanja [72].

B.1 Definicija DTD

```
<!DOCTYPE TEI.2 system 'tei2.dtd' [
<!-- Deklaracija sistema pisanja                                    -->
     <!ENTITY srbcyr system 'serbloc.wsd' SUBDOC >
<!-- Modifikacija TEI DTD-a za kolekciju Vukovih poslovica          -->
     <!ENTITY % TEI.extensions.ent system 'e-vuk.ent' >
     <!ENTITY % TEI.extensions.dtd system 'e-vuk.dtd' >
<!-- Bazni skup etiketa                                  -->
     <!ENTITY % TEI.prose 'INCLUDE' >
<!-- Dodatni skup etiketa                                -->
     <!ENTITY % TEI.names.dates 'INCLUDE' >
     <!ENTITY % TEI.linking 'INCLUDE' >
]>
<TEI.2>
<!-- Srpske narodne poslovice Vuka Stefanovića Karadžića        -->
<!-- Zaglavlje elektronskog izdanja                             -->
<teiHeader>
<fileDesc>
<titleStmt><title>Srpske narodne poslovice - elektronski izdanje</title>
   <author>Vuk Stefanović Karadžić</author>
```

B.2 Deklaracija lokalnih entiteta

```
<!-- Menjaju se definicije sledećih ELEMENATA                          -->
<!ENTITY % body 'IGNORE' >
<!ENTITY % placeName 'IGNORE' >

<!-- Novi modeli sadržaja                                               -->
<!ENTITY % m.content.prov 'opt.ph | n.i.prov | obs.ph | f.gap |
   c.b.r | c.r.s  '  >
<!ENTITY % m.content.exp  's.a | c.r.s ' >
<!ENTITY % m.name.date    'persName | placeName | dataStruct ' >
```

```
<!-- Dodaju se elementi za imena, mesta i datume i u TEI-inter        -->
<!ENTITY % x.inter ' %m.name.date; | '                                 >

<!-- Akcentovana ćirilična slova                                      -->
<!ENTITY adsil SDATA "[adsil ]" -- = malo a, dugi silazni             -->
<!ENTITY Adsil SDATA "[Adsil ]" -- = veliko A, dugi silazni           -->
.....................................................................
<!-- Tekstovi su u sledećim tekama                                    -->
<!ENTITY aslovo SYSTEM "e-vuk-a.txt" >
<!ENTITY bslovo SYSTEM "e-vuk-b.txt" >
.....................................................................
<!-- Njima odgovaraju sledeći označeni odeljci                        -->
<!--ENTITY % aodelj 'INCLUDE' -->
<!ENTITY % aodelj 'IGNORE'>
<!--ENTITY % bodelj 'INCLUDE' -->
<!ENTITY % bodelj 'IGNORE' >
.....................................................................
```

B.3 Deklaracija lokalnih elemenata

```
<!-- Novi modeli sadržaja za Vukove poslovice                         -->
<!-- Ne može da se smesti u *.ent teku jer koristi modele sadržaja
     koji će tek biti definisani u teiclas2.ent                      -->
<!-- Tekst poslovice se opisuje sa specialPara.prov                  -->
<!-- nove fraze u njemu su: opcioni termin                           -->
<!--                        u poslovici je ali nije njen deo         -->
<!--                        može se zameniti nečim                    -->
<!--                        može zameniti nešto                       -->

<!ENTITY % m.phrase.prov '%m.data; | %m.edit; | %m.hqphrase; |
  %m.name.date; |
  %m.loc; | %m.seg; | %m.content.prov; ' >
<!ENTITY % phrase.seq.prov '(#PCDATA | %m.phrase.prov; )* '      >
<!--U paragrafu poslovicea ostale su samo: karakteri, fraze a od
    inter-level elemenata samo dijalozi                             -->
<!--U specijalnom paragrafu su od chank-elemenata ostali samo: paragraf,
    i grupa stihova, odn. stihovi poslovice, a zajedničkih elemenata
    nema                                                            -->
<!ENTITY % specialPara.prov '(( p | lg.prov ) |
                             (#PCDATA | %m.phrase.prov; | q)* ) ' >

<!-- Tekst objašnjenja se opisuje sa specialPara.exp                -->
<!-- nove fraze u njemu su: može zameniti nešto                     -->
<!--                        vidi i - veza sa nekom poslovicom       -->
<!ENTITY % m.phrase.exp  '%m.data; | %m.edit; | %m.hqphrase; |
                          %m.name.date; |
                          %m.loc; | %m.seg; | %m.content.exp '       >
<!--U paragrafu objašnjenja ostale su samo: karakteri, fraze a od
    inter-level elemenata samo liste, dijalozi i citati             -->
<!--U specijalnom paragrafu su od chank-elemenata ostali samo: paragraf,
    i grupa stihova, odn. stihovi poslovice, od zajedničkih elemenata
```

```
                                                                    -->
   samo liste
<!ENTITY % specialPara.exp '((( p | lg.prov ) ,
                          ( list | lg.prov | p | q )* ) |
                          (#PCDATA | %m.phrase.exp; |
                          list | q | quote )* ) '                    >

<!-- ELEMENT opt.ph - poslovica stoji i sa njom i bez nje           -->
<!ELEMENT opt.ph     - -  (%phrase.seq.prov;)                        >
<!ATTLIST opt.ph          %a.global
                          resp      CDATA    'Vuk'
                          original  (Y | N)  Y                      >

<!-- ELEMENT n.i.prov - fraza navedena u tekst poslovice ali
     nije njen deo                                                  -->
<!ELEMENT n.i.prov   - -  (%phrase.seq.prov;)                       >
<!ATTLIST n.i.prov        %a.global
                          resp      CDATA    'Vuk'
                          original  (Y | N)  Y
                          type      CDATA    #IMPLIED               >

<!-- ELEMENT obs.ph - nepristojna fraza koja nije navedena u tekst  -->
<!-- u potpunosti nego tačkicama                                    -->
<!ELEMENT obs.ph     - -  (%phrase.seq;)                            >
<!ATTLIST obs.ph          %a.global
                          value     CDATA    #IMPLIED
                          resp      CDATA    #IMPLIED               >

<!-- ELEMENT f.gap - dopuna - fraza koja je navedena u tekstu poslovice  -->
<!-- ali nju treba nečim popuniti                                   -->
<!ELEMENT f.gap      - -  (%phrase.seq;)                            >
<!ATTLIST f.gap           %a.global
          type      CDATA    #IMPLIED
          resp      CDATA    #IMPLIED                  >
<!-- ELEMENT - c.b.r - canBeReplaced - fraza navedena u tekstu
     poslovice koja se moze nečim i zameniti                        -->
<!-- atributi su %a.global ali je id-atribut obavezan               -->
<!ELEMENT c.b.r      - -  (%phrase.seq.prov;)                       >
<!ATTLIST c.b.r           n         CDATA    #IMPLIED
          rend      CDATA    #IMPLIED
          resp      CDATA    'Jasmina'
          id        ID       #REQUIRED                 >

<!-- ELEMENT - c.r.s - canReplace - fraza koja moze da zameni neku
     drugu u tekstu poslovice ili objašnjenu                        -->
<!ELEMENT c.r.s           - -  (%phrase.seq.prov;)                  >
<!-- atributi su %a.global ali je id-atribut obavezan               -->
<!ATTLIST c.r.s           n         CDATA    #IMPLIED
          rend      CDATA    #IMPLIED
          resp      CDATA    'Jasmina'
          id        ID       #REQUIRED                 >

<!-- ELEMENT - s.a - seeAlso - tekst koji opisuje vezu s nekom
     poslovicom u tekstu objašnjenja                                -->
```

```
<!ELEMENT s.a          - -  (%specialPara;)                              >
<!-- atributi su %a.global ali je id-atribut obavezan                    -->
<!ATTLIST s.a              %a.global;
                           %a.pointer;
                           target    IDREF    #REQUIRED                 >
<!-- Stihovi kao deo poslovica                                          -->
<!ELEMENT l.prov       - o  (%phrase.seq.prov;)                         >
<!ATTLIST l.prov           %a.global;
          part                (Y | N | I | M | F) n
          met                 CDATA              #IMPLIED              >

<!ELEMENT lg.prov      - o  ((%n.head)?,
                           (( l.prov ) | ( lg.prov ))+)                 >
<!ATTLIST lg.prov          %a.global;
          type                CDATA              #IMPLIED
          rhyme               CDATA              '*INHERITED'
          part                (y | n | i | m | f) n                    >

<!-- Menja se deklaracija za element body                               -->
<!-- Element body u kolekciji Vukovih poslovica sadrži samo             -->
<!-- element koji je kolekcija poslovica                                -->

<!ELEMENT body         - O  ( divproverb )                               >
<!ATTLIST body             %a.global;                                    >

<!-- Element divproverb sadrži samo naslov iza koga slede              -->
<!-- elementi divletter koji sadrže poslovice na neko slovo             -->
<!ELEMENT divproverb   - O  ((%n.head)?, ( divletter )+)                 >
<!ATTLIST divproverb       %a.global;                                    >

<!-- Element divletter sadrži samo opcioni naslov iza koga slede       -->
<!-- poslovice: elementi divp                                           -->
<!-- ili grupe od dve poslovice (dele isto objašnjenje)                 -->
<!ELEMENT divletter    - O  ((%n.head;)?, ( divp | group.divp )+)        >
<!ATTLIST divletter        %a.global;
                           %a.divn;
                           letter    CDATA    #IMPLIED                 >

<!-- Element group.divp sadrži dve poslovice koje dele isto             -->
<!-- objašnjene (eventualno)                                            -->
<!ELEMENT group.divp   - - ( divp , divp )                               >
<!ATTLIST group.divp       %a.global;
                           type      CDATA    #IMPLIED                 >

<!-- Element divp sadrži samo poslovicu i eventualno                    -->
<!-- objašnjenje                                                        -->
<!ELEMENT divp         - O  ((( pv ), ( pexp )?)* ,
                            (%m.loc;)* )                                >
<!ATTLIST divp             %a.global;
                           type      CDATA    #IMPLIED
                           resp      CDATA    'Vuk'
                           original  (Y | N)   Y                        >
<!-- Element pv (poslovica) je paragraf                                 -->
```

```
<!-- koji može da sadrži sledeće fraze:                                    -->
<!--     opphrase: fraza koja je deo poslovice a moze se i izbaciti        -->
<!-- Element pexp (objašnjenje poslovice) je paragraf                      -->
<!-- Poslovica                                                             -->
<!ELEMENT pv            - O  (%specialPara.prov;)                          >
<!ATTLIST pv                 %a.global;                                    >

<!-- Objašnjenje poslovice                                                 -->
<!ELEMENT pexp          - O  (%specialPara.exp;)                           >
<!ATTLIST pexp               %a.global;                                    >

<!-- Menja se lista atributa za placeName                                  -->
<!-- Dodaje se atribut koji govori o tome da mesto odredjuje               -->
<!-- u kom kraju se poslovica koristi                                      -->
<!-- Pretpostavljena vrednost je Y(es)                                     -->
<!ELEMENT placeName       - - ((%m.placePart; | %m.phrase; |
                               #PCDATA)+)                                  >
<!ATTLIST placeName           %a.global;
        %a.names;
        type                  CDATA                  #IMPLIED
        full                  (yes | abb | init)     yes
        use.of.prov           (Y | N)                Y                   >
```


B.4 Kodirani tekst Vukovih poslovica

```
<divletter letter=G type='odeljak'>
<head>G.</head>
<divp id=P658 n=658>
<pv>Gad ovoga svijeta!</pv>
<pexp>
    Reče se za ružno čeljade.
</pexp>
</divp>
<divp id=P659 n=659>
<pv>Galija jednoga ne čeka.</pv>
</divp>
<divp id=P660 n=660>
<pv>Garbin ljuti, koji more do dna muti.</pv>
<pexp>
    U <placeName type='mesto'>Dubrovniku</placeName>
    <term rend='italic'>Garbin</term> se zove
    nekakav vjetar, čini mi se zapadni.
</pexp>
</divp>
<divp id=P661 n=661>
<pv>Gatala baba da nije mraza, pa osvanuo snijeg do
    <obs.ph value='guzice' resp='CV'>g....e</obs.ph>.</pv>
</divp>
<divp id=P662 n=662>
<pv>Gvožđe valja kovati dok je vruće.</pv>
<pexp>
    Ili: <s.a target=P663 resp='Vuk'></s.a>
```

```
</pexp>
</divp>
<divp id=P663 n=663>
<pv>Gvožđe se kuje dok je vruće.</pv>
<ptr target=P662 resp='CV'>
</divp>
<divp id=P664 n=664>
<pv>Gde je cvet tu je med.</pv>
<pexp>
    Čele bez cvijeta ne mogu meda skupiti.
</pexp>
</divp>
<divp id=P665 n=665>
<pv>Gizdavoj nevi ne moš ovrlj oko glave obmotati.</pv>
<pexp>
    Kad je kome teško ugoditi.
</pexp>
</divp>
<divp id=P666 n=666>
<pv>Gizda lomi, a glad mori.</pv>
<pexp>
    Gledaj: <s.a target=P2963 resp='Vuk'>Malo jede, al' se lijepo
    nosi.</s.a>
</pexp>
</divp>
<divp id=P667 n=667>
<pv>Glava a! <n.i.prov type='pokret'>(podignuvši ruke i raširivši šake
    prema glavi)</n.i.prov> a u glavi na!</pv>
<pexp>
    Kao uhvativši se palcem i kažiprstom za zube, i znači: glava velika,
    ali prazna.
</pexp>
</divp>
<divp id=P668 n=668>
<pv>Glava je skuplja s jezikom nego bez jezika.</pv>
<pexp>
    Vredniji je čoek koji umije govoriti nego onaj koji ne umije.
</pexp>
</divp>
<divp id=P669 n=669>
<pv>Glava je starija od knjige.</pv>
</divp>
<divp id=P670 n=670>
<pv>Glava li ga boli?</pv>
<pexp>
    Kad se kazuje da ko nema uzroka s čime biti nezadovoljan.
</pexp>
</divp>
<divp id=P671 n=671>
<pv>Glava pred glavu.</pv>
<pexp>
    Gledaj: <s.a target=P504 resp='Vuk'>Varka pred <persName>Marka</persName>
    </s.a>
```

Dodatak C

Formalna gramatika podskupa SGML-a

U ovom odeljku data je formalna gramatika SGML-a kakva je u neznatno izmenjenom obliku korišćena kao ulazna YACC-ova datoteka sistema mSGML. Kao što je rečeno u 4.2, ova gramatika ne pokriva potpuni SGML. Nisu pokrivena sledeća svojstva:

- rang,
- podatkovne etikete,
- spone.

Neka svojstva koja za sada nisu realizovana u mSGML-u su u gramatici prisutna i biće realizovana u kasnijima fazama razvoja sistema. To su kratke reference i konkurentne strukture.

U gramatici su korišćene sledeće konvencije:

- malim slovima su označeni neterminali;
- velikim slovima su označeni tokeni koji predstavljaju ključne reči SGML-a;
- početnim velikim slovom su označeni ostali tokeni, kao što su graničnici, imena i slično. Tako je, na primer, MDO token koji predstavlja ključnu reč SGML-a kojom se označava graničnik „otvaranje deklaracije oznaka“ dok je Mdo token koji predstavlja sam taj graničnik.

```
0  $accept : SGMLdoc $end

1  SGMLdoc : SGMLdekl prolog dok_instancija

2  SGMLdekl : Mdo SGML Min_literal CHARSET opis_skupa_karaktera
              CAPACITY skup_kapaciteta
              SCOPE domen_konkretne_sintakse
              SYNTAX konkretna_sintaksa
              FEATURES MINIMIZE DATATAG dane OMITTAG dane
                 RANK dane SHORTTAG dane
                 LINK SIMPLE dane_broj IMPLICIT dane EXPLICIT dane_broj
                 OTHER CONCUR dane_broj SUBDOC dane_broj FORMAL dane
              APPINFO informacije_o_aplikaciji Mdc
```

```
 3  opis_skupa_karaktera : osnovni_skup opisani_skup
 4                       | opis_skupa_karaktera osnovni_skup opisani_skup

 5  osnovni_skup : BASESET javni_identifikator

 6  opisani_skup : DESCSET opis_karaktera
 7               | opisani_skup opis_karaktera

 8  opis_karaktera : Broj Broj Broj
 9                 | Broj Broj Min_literal
10                 | Broj Broj UNUSED

11  skup_kapaciteta : PUBLICC javni_identifikator
12                  | SGMLREF lista_kapaciteta

13  lista_kapaciteta : ime_kapaciteta Broj
14                   | lista_kapaciteta ime_kapaciteta Broj

15  ime_kapaciteta : TOTALCAP | ENTCAP | ENTCHCAP | ELEMCAP | GRPCAP
20                 | EXGRPCAP | EXGRPCAP | EXNMCAP | ATTCAP | ATTCHCAP
24                 | AVGRPCAP | NOTCAP | NOTCHCAP | IDCAP | IDREFCAP
29                 | MAPCAP | LKSETCAP | LKNMCAP

32  domen_konkretne_sintakse : DOCUMENT
33                           | INSTANCE

34  konkretna_sintaksa : PUBLICC javni_identifikator
35                     | PUBLICC javni_identifikator lista_prekidaca
36                     | SHUNCHAR nepozeljni_karakteri opis_skupa_karaktera
                           FUNCTION RE Broj RS Broj SPACE Broj
                               lista_dodatnih_funkcija
                           NAMING LCNMSTRT parametarski_literal
                                  UCNMSTRT parametarski_literal
                                  LCNMCHAR parametarski_literal
                                  UCNMCHAR parametarski_literal
                           NAMECASE GENERAL dane ENTITY dane
                           DELIM GENERAL SGMLREF lista_opstih_granicnika
                                 SHORTREF granicnici_kratkih_referenci
                           NAMES SGMLREF lista_rezervisanih_imena
                           QUANTITY SGMLREF lista_velicina

37  lista_prekidaca : SWITCHES Broj Broj
38                  | lista_prekidaca Broj Broj

39  nepozeljni_karakteri : NONE
40                       | lista_nepozeljnih_karaktera

41  lista_nepozeljnih_karaktera : CONTROLS
42                              | Broj
43                              | lista_nepozeljnih_karaktera Broj

44  lista_dodatnih_funkcija :
45                          | lista_dodatnih_funkcija
```

```
                                Ime klasa_funkcije Broj

46  klasa_funkcije : FUNCHAR | MSICHAR | MSOCHAR | MSSCHAR | SEPCHAR

51  lista_opstih_granicnika :
52                          | lista_opstih_granicnika
                                ime_opsteg_granicnika parametarski_literal

53  ime_opsteg_granicnika : AND | COM | CRO | DSC | DSO | DTGC | DTGO
60                        | ERO | ETAGO | GRPC | GRPO | LIT | LITA
66                        | MDC | MDO | MINUS | MSC | NET | OPT | OR
73                        | PERO | PIC | PIO | PLUS | REFC | REP | RNI
80                        | SEQ | SHORTREF | STAGO | TAGC | VI

85  granicnici_kratkih_referenci : SGMLREF
86                               | NONE
87                               | granicnici_kratkih_referenci
                                     parametarski_literal

88  lista_rezervisanih_imena :
89                           | lista_rezervisanih_imena
                                 rezervisano_ime Ime

90  rezervisano_ime : ANY | ATTLIST | CAPACITY | CDATA | CHARSET
95                  | CONREF | CURRENT | DEFAULT | DOCTYPE | DOCUMENT
100                 | DTD | ELEMENT | ELEMENTS | EMPTY | ENDTAG
105                 | ENTITIES | ENTITY | FIXED | ID | IDREF | IDREFS
111                 | IGNORE | IMPLIED | INCLUDE | INITIALL | LINK
116                 | LINKTYPE | LPD | MD | MSS | NAME | NAMES | NDATA
123                 | NMTOKEN | NMTOKENS | NONSGML | NOTATION | NUMBER
128                 | NUMBERS | NUTOKEN | NUTOKENS | O | PCDATA | PI
134                 | POSTLINK | PUBLICC | RCDATA | RE | SDATA | SHORTREF
140                 | SIMPLE | SPACE | STARTTAG | SUBDOC | SYNTAX
145                 | SYSTEM | TEMP | TEXT | USELINK | USEMAP

150 lista_velicina :
151                | lista_velicina velicina Broj

152 velicina : ATTCNT | ATTSPLEN | BSEQLEN | DTAGLEN | DTEMPLEN
157          | ENTLVL | GRPCNT | GRPGTCNT | GRPLVL | LITLEN
162          | NAMELEN | NORMSEP | PILEN | TAGLEN | TAGLVL

167 dane : YES
168      | NO

169 dane_broj : NO
170           | YES Broj

171 informacije_o_aplikaciji : NONE
172                          | Min_literal

173 javni_identifikator : Min_literal
```

```
174  prolog : prolog_razno sekvencija_dtd

175  prolog_razno :
176               | prolog_razno deklaracija_komentara
177               | prolog_razno instrukcija_obrade

178  deklaracija_komentara : Mdo Mdc

179  instrukcija_obrade : Pio Kar_podaci Pic

180  sekvencija_dtd : dtd prolog_razno
181                 | sekvencija_dtd dtd prolog_razno

182  dtd : Mdo DOCTYPE Ime spoljasnji_identifikator
           Dso podskup_dtd Dsc Mdc
183      | Mdo DOCTYPE Ime Dso podskup_dtd Dsc Mdc
184      | Mdo DOCTYPE Ime spoljasnji_identifikator Mdc
185      | Mdo DOCTYPE Ime Mdc

186  spoljasnji_identifikator : SYSTEM sistemski_podaci
187                           | SYSTEM
188                           | PUBLICC Min_literal sistemski_podaci
189                           | PUBLICC Min_literal

190  podskup_dtd :
191               | podskup_dtd deklaracija_elementa
192               | podskup_dtd deklaracija_liste_atributa
193               | podskup_dtd deklaracija_notacije
194               | podskup_dtd deklaracija_entiteta
195               | podskup_dtd deklaracija_komentara
196               | podskup_dtd instrukcija_obrade

197  deklaracija_elementa : Mdo ELEMENT tip_elementa minimizacija
                            deklaracija_sadrzaja Mdc

198  tip_elementa : Ime
199               | Grpo grupa_imena Grpc

200  grupa_imena : sek_imena
201              | ili_imena
202              | i_imena

203  sek_imena : Ime
204            | sek_imena Seq Ime

205  i_imena : Ime And Ime
206          | i_imena And Ime

207  ili_imena : Ime Or Ime
208            | ili_imena Or Ime

209  minimizacija : min min
```

```
210  min : 0
211      | Minus

212  deklaracija_sadrzaja : CDATA
213                       | RCDATA
214                       | EMPTY
215                       | ANY izuzeci
216                       | model izuzeci

217  model : Grpo grupa_tokena Grpc Opt
218        | Grpo grupa_tokena Grpc Plus
219        | Grpo grupa_tokena Grpc Rep
220        | Grpo grupa_tokena Grpc

221  grupa_tokena : sek_tokeni
222               | ili_tokeni
223               | i_tokeni

224  sek_tokeni : token_sadrzaja
225             | sek_tokeni Seq token_sadrzaja

226  i_tokeni : token_sadrzaja And token_sadrzaja
227           | i_tokeni And token_sadrzaja

228  ili_tokeni : token_sadrzaja Or token_sadrzaja
229             | ili_tokeni Or token_sadrzaja

230  token_sadrzaja : Rni PCDATA
231                 | Ime pojavljivanje
232                 | model

233  pojavljivanje :
234                | Opt
235                | Plus
236                | Rep

237  izuzeci : iskljucivanja ukljucivanja

238  ukljucivanja :
239               | Plus1 Grpo grupa_imena Grpc

240  iskljucivanja :
241                | Minus Grpo grupa_imena Grpc

242  deklaracija_liste_atributa : Mdo ATTLIST pridruzeni
                                      lista_def_atributa Mdc

243  pridruzeni : tip_elementa
244             | pridruzena_notacija

245  pridruzena_notacija : Rni NOTATION Ime
246                      | Rni NOTATION Grpo grupa_imena Grpc
```

```
247 lista_def_atributa : definicija_atributa
248                    | lista_def_atributa definicija_atributa

249 definicija_atributa : Ime deklarisana_vrednost
                             pretpostavljena_vrednost

250 deklarisana_vrednost : CDATA | ENTITY | ENTITIES | ID | IDREF
255                      | IDREFS | NAME | NAMES | NMTOKEN | NMTOKENS
260                      | NUMBER | NUMBERS | NUTOKEN | NUTOKENS
264                      | NOTATION
265                      | Grpo grupa_imena Grpc
266                      | Grpo grupa_tokena_imena Grpc

267 grupa_tokena_imena : sek_tokeni_imena
268                   | ili_tokeni_imena
269                   | i_tokeni_imena

270 token_imena : Ime
271             | Broj
272             | Brtoken

273 sek_tokeni_imena : token_imena
274                  | sek_tokeni_imena Seq token_imena

275 i_tokeni_imena : token_imena And token_imena
276                | i_tokeni_imena And token_imena

277 ili_tokeni_imena : token_imena Or token_imena
278                  | ili_tokeni_imena Or token_imena

279 pretpostavljena_vrednost : Rni FIXED spec_vrednosti_atributa
280                         | spec_vrednosti_atributa
281                         | Rni REQUIRED
282                         | Rni CURRENT
283                         | Rni CONREF
284                         | Rni IMPLIED

285 spec_vrednosti_atributa : zamenljivi_literal
286                        | Ime
287                        | Broj
288                        | Brtoken

289 deklaracija_notacije : Mdo NOTATION Ime spoljasnji_identifikator Mdc

290 deklaracija_entiteta : Mdo ENTITY Ime tekst_opsteg_entiteta Mdc
291                      | Mdo ENTITY Rni DEFAULT
                             tekst_opsteg_entiteta Mdc
292                      | Mdo ENTITY Pero Ime tekst_entiteta Mdc

293 tekst_opsteg_entiteta : CDATA parametarski_literal
294                      | SDATA parametarski_literal
295                      | spoljasnji_identifikator tip_entiteta
296                      | tekst_entiteta
```

```
297  tekst_entiteta : parametarski_literal
298                 | PI parametarski_literal
299                 | STARTTAG parametarski_literal
300                 | ENDTAG parametarski_literal
301                 | MSS parametarski_literal
302                 | MD parametarski_literal
303                 | spoljasnji_identifikator

304  tip_entiteta : SUBDOC
305               | CDATA Ime
306               | CDATA Ime Dsc skup_specif_atributa
307               | NDATA Ime
308               | NDATA Ime Dsc skup_specif_atributa
309               | SDATA Ime
310               | SDATA Ime Dsc skup_specif_atributa

311  skup_specif_atributa :
312                       | skup_specif_atributa specif_atributa

313  specif_atributa : Ime Vi spec_vrednosti_atributa
314                  | spec_vrednosti_atributa

315  dok_instancija : jedan_element prolog_razno

316  jedan_element : poc_etiketa sadrzaj krj_etiketa

317  poc_etiketa : Stago Ime skup_specif_atributa Tagc
318              | Stago Ime Tagc
319              | Stago Grpo grupa_imena Grpc Ime
                      skup_specif_atributa Tagc
320              | Stago Grpo grupa_imena Grpc Ime Tagc

321  sadrzaj :
322          | sadrzaj Kar_podaci
323          | sadrzaj jedan_element
324          | sadrzaj sadrzaj_razno

325  sadrzaj_razno : deklaracija_komentara
326                | instrukcija_obrade

327  krj_etiketa : Etago Ime Tagc
328              | Etago Grpo grupa_imena Grpc Ime Tagc

329  sistemski_podaci : Min_literal
330                   | Sis_literal

331  parametarski_literal : sistemski_podaci
332                       | Kar_literal
333                       | Par_literal

334  zamenljivi_literal : sistemski_podaci
335                     | Kar_literal
```

```
336                 | Zam_literal
```

Dodatak D

Izvod iz elektronskog rečnika SJ

D.1 Izvod iz DELAS-a za SJ

```
žaba,N70.01+*
žaliti,V34.03.4*
žalosnica,N71.01+*
žalosno,Adv*
žalost,N82.04-*
žalostan,A08.03*
žarilo,N51.01-*
žariti,V33.01.3*
žbun,N08.01-*
žderati,V21.00.3*
ždral,N07.01+*
ždralx,N23.01+*
ždrebe,N65.01+E%#E4.03
ždrelo,N51.01-E%#E4.03
ždrijebe,N65.01+J%ždrebe#E4.03
ždrijelo,N51.01-J%E4.03
ždrmanxi,N06.01-*
ždrmnxan,A08.01*
žeželx,N22.01-*
žežen,A06.01*
žedan,A08.02*
žedneti,V34.04.4E%#E5.30
žednxeti,V34.29.5J%žedneti#E5.30
želeti,V34.02.4E%#E4.30
želxa,N70.01-*
želxeti,V34.27.4J%želeti#E4.30
žena,N70.01+*
ženin,A02.01*
ženiti,V34.05.4*
ženski,A03.01*
žensko,N51.01-*
ženstvo,N51.02-*
žerava,N70.01-*
žeravica,N71.01-*
žestok,A12.02*
žeti,V30.00.3*,V27.00.3*
žetva,N70.02-*
žeđ,N82.01-*
```

```
žeđa,N70.01-*
žeći,V50.02.2*
žica,N70.01-*
žila,N70.01-*
žir,N08.01-*
žitak,N13.06-*
žitan,A08.02*
žito,N51.01-*
živ,A08.01*
živac,N18.61-*
živeti,V34.00.3E%#E4.23
živina,N70.01+*
živjeti,V34.25.4J%živeti#E4.23
živlxeti,V34.27.4J%živeti#E4.23
život,N04.01-*
životinxa,N70.01+*
žičica,N71.01-*
žlica,N70.01-*
žlje,Adv*
žmirke,Adv*
žnxeti,V26.00.3*
žrvanx,N22.51-*
žurba,N72.01-*
žuriti,V33.01.3*
žut,A07.01*
žuč,N22.01-*
žvakati,V21.07.4*
a,Con*,Adv*,Int*
aždaha,N70.01+*%#H5.04
aždaja,N70.01+*%aždaha#H5.04
aba,N70.01-*
acal,N04.01-*
adamski,A03.01*
adumac,N17.61+*%hadumac#H1.07
aferim,Int*
aga,N75.01+*
ah,Int*
aha,Int*
ahar,N04.01-*
ajmo,Int*
ajnc,N18.11-*
ajns,N08.01-*
ak,N15.01-*%hak#H1.07
ako,Con*,Par*
akobogda,Adv*
akov,N04.01-*
ala,N70.01-*,N70.01+*%hala#H1.07,Par*,Con*
alat,N04.01-*
ali,Con*
alva,N72.01-*%halva#H1.07
alvaluk,N04.04-*%halvaluk#H1.07
am,N08.01-*%ham#H1.07
ama,Con*,Par*
amanat,N04.01-*
amanet,N04.01-*
amat,N04.01-*
ambar,N04.01-*
ambisati,V21.02.4*
```

amen,Int*,Adv*
ametice,Adv*
ametom,Adv*
amin,Int*,Adv*
amo,Adv*
anđeo,N01.08+*
ao,Int*
aoh,Int*
aps,N08.01-*
arap,N01.01+*
aratos,Adv*
arbanaški,A03.01*
arište,N60.01-*
aršin,N04.01-*
arčiti,V33.01.3*%harčiti#H1.07
aspra,N72.01-*
astal,N04.01-*
ašikovanxe,N60.01-*
at,N07.01+*%hat#H1.07
au,Int*
avan,N04.01-*
avra,N72.01-*%havra#H1.07
aščija,N75.01+*
aščijnica,N71.01-*
baba,N70.01+*
babazeman,N04.01-*
babetina,N70.01+*
babica,N71.01+*
babin,A02.01*
babine,N60.01-*
babo,N40.01+*
bacati,V01.00.2*
baciti,V33.51.3*
badanx,N22.51-*
badava,Adv*
badavad,Adv*
bagav,A08.01*
baj,N08.01-*
baka,N70.01+*
bakalaj,N17.01+*
bakalar,N01.01+*
bakrač,N22.01-*
bakuštaja,N70.01-*
balav,A08.01*
balega,N70.01-*
balota,N70.01-*
bambat,A08.01*
ban,N07.01+*
banak,N13.54-*
banica,N71.01-*
bar,Par*
bara,N70.01-*
barabar,Adv*
barak,N04.04-*
bardak,N04.04-*
barem,Adv*
bareta,N70.01-*
barilo,N51.01-*

```
bario,N04.08-*
barjak,N04.04-*
bat,N08.01-*
batalxka,N72.02-*
batina,N70.01-*
batrgati,V01.00.2*
baš,Par*
baška,Adv*
baština,N70.01-*
baća,N75.01+*
bača,N75.01+*
bačina,N70.01-*
bačva,N72.01-*
bežati,V31.00.3E%#E2.12
beda,N70.01-E#%E2.03
bedra,N70.02-*
beg,N07.02+*,N01.02+*,N08.02-E%#E2.03
begati,V01.00.2E%#E2.12
begenisati,V21.02.4*
begluk,N04.04-*
bego,N40.01+*
begovac,N17.61+*
begunac,N17.61+E%#E2.12
beleg,N04.02-E%#E2.10#E4.26
beličast,A06.01*
belx,N22.01-E%#E2.06
belxa,N70.01+*
bena,N75.01+*
benetati,V01.00.2*,V21.08.4*
beo,A13.01E
berat,N04.01-*
berba,N72.01-*
berbernica,N71.01-*
beričet,N04.01-*
bes,N08.01-E%#E2.03.02
besan,A08.52E%#E2.21
beseda,N70.01-E%#E4.12
besediti,V34.11.4E%#E4.12
besjeda,N70.01-J%beseda#E4.12
besjediti,V34.11.4J%besediti#E4.12
besneti,V34.13.5E%#E2.20#E5.47
besposlen,A06.01*
besposličiti,V33.01.3*
besprestance,Adv*
bestija,N70.01+*
bestražice,Adv*
bestraga,Adv*
bez,Pre*
bezadnxica,N71.01-*
bezobrazan,A08.02*
bezobzirce,Adv*
bezrep,A06.01*
bezuman,A08.02*
beškot,N04.01-*
beštija,N70.01+*
bečar,N01.01+*
bečiti,V33.01.3*
bečka,N72.01-*
```

D.2 Izvod iz DELAF-a za SJ


```
kojekud,.Adv*
kojekuda,.Adv*
kojem,koji.ProA07:msd*:msl*:nsd*:nsl*
kojemu,koji.ProA07:msd*:msl*:nsd*:nsl*
kojeotkud,.Adv*
kojeotkuda,.Adv*
kojetko,.ProN12:**n+
koješta,.Int*,.ProN13:**n-:**a-
koječeg,koješta.ProN13:**g-
koječega,koješta.ProN13:**g-
koječem,koješta.ProN13:**l-
koječemu,koješta.ProN13:**d-:**l-
koječim,koješta.ProN13:**i-
koječime,koješta.ProN13:**i-
koji,.ProA07:msn*:msa-:mpn*
kojih,koji.ProA07:mpg*:npg*:fpg*
kojim,koji.ProA07:msi*:nsi*:*pd*:*pi*:*pl*
kojima,koji.ProA07:*pd*:*pi*:*pl*
kojoj,koji.ProA07:fsd*:fsl*
kojom,koji.ProA07:fsi*
koju,koji.ProA07:fsa*
koka,.N70.01+*:fsn+:fpg+
kokala,kokalo.N51.01-*:nsg-:npn-:npg-:npa-:npv-
kokalima,kokalo.N51.01-*:npd-:npi-:npl-
kokalo,.N51.01-*:nsn-:nsa-:nsv-
kokalom,kokalo.N51.01-*:nsi-
kokalu,kokalo.N51.01-*:nsd-:nsl-
kokama,koka.N70.01+*:fpd+:fpi+:fpl+
koke,koka.N70.01+*:fsg+:fpn+:fpa+:fpv+
koki,koka.N70.01+*:fsd+:fsl+
koko,koka.N70.01+*:fsv+
kokom,koka.N70.01+*:fsi+
kokot,.N01.01+*:msn+
kokota,kokot.N01.01+*:msg+:msa+:mpg+
kokote,kokot.N01.01+*:msv+:mpa+
kokoti,kokot.N01.01+*:mpn+:mpv+
kokotima,kokot.N01.01+*:mpd+:mpi+:mpl+
kokotom,kokot.N01.01+*:msi+
kokotu,kokot.N01.01+*:msd+:msl+
kokoš,.N21.01+*:fsn+:fsa+
kokoši,kokoš.N21.01+*:fsg+:fsd+:fsv+:fsl+:fpn+:fpg+:fpa+:fpv+
kokošima,kokoš.N21.01+*:fpd+:fpi+:fpl+
kokošju,kokoš.N21.01+*:fsi+
koku,koka.N70.01+*:fsa+
kokša,.N72.01+*:fsn+
kokšama,kokša.N72.01+*:fpd+:fpi+:fpl+
kokše,kokša.N72.01+*:fsg+:fpn+:fpa+:fpv+
kokši,kokša.N72.01+*:fsd+:fsl+:fpg+
kokšo,kokša.N72.01+*:fsv+
kokšom,kokša.N72.01+*:fsi+
kokšu,kokša.N72.01+*:fsa+
kola,kolo.N51.01-*:nsg-:npn-:npg-:npa-:npv-,.N62.01-*:npn-:npg-:npa-:npv-
kolac,.N18.01-*:msn-:msa-,.N20.61-*:msn-:msa-
kolaca,kolac.N18.01-*:mpg-
kolaci,kolak.N04.04-*:mpn-:mpv-
```

```
kolacima,kolak.N04.04-*:mpd-:mpi-:mpl-
kolak,.N04.04-*:msn-:msa-
kolaka,kolak.N04.04-*:msg-:mpg-
kolake,kolak.N04.04-*:mpa-
kolakom,kolak.N04.04-*:msi-
kolaku,kolak.N04.04-*:msd-:msl-
kolač,.N22.01-*:msn-:msa-
kolača,kolač.N22.01-*:msg-:mpg-
kolače,kolak.N04.04-*:msv-,kolač.N22.01-*:mpa-
kolačem,kolač.N22.01-*:msi-
kolači,kolač.N22.01-*:mpn-:mpv-
kolačima,kolač.N22.01-*:mpd-:mpi-:mpl-
kolaču,kolač.N22.01-*:msd-:msv-:msl-
kolca,kolac.N20.61-*:msg-,kolac.N18.01-*:msg-
kolce,kolac.N18.01-*:mpa-
kolcem,kolac.N18.01-*:msi-,kolac.N20.61-*:msi-
kolci,kolac.N18.01-*:mpn-:mpv-
kolcima,kolac.N18.01-*:mpd-:mpi-:mpl-
kolcu,kolac.N18.01-*:msd-:msl-,kolac.N20.61-*:msd-:msl-
koleda,.N70.01+*:fsn+:fpg+
koledama,koleda.N70.01+*:fpd+:fpi+:fpl+
kolede,koleda.N70.01+*:fsg+:fpn+:fpa+:fpv+
koledi,koleda.N70.01+*:fsd+:fsl+
koledo,koleda.N70.01+*:fsv+
koledom,koleda.N70.01+*:fsi+
koledu,koleda.N70.01+*:fsa+
kolena,koleno.N51.01-E:nsg-:npn-:npg-:npa-:npv-
kolenima,koleno.N51.01-E:npd-:npi-:npl-
koleno,.N51.01-E:nsn-:nsa-:nsv-
kolenom,koleno.N51.01-E:nsi-
kolenu,koleno.N51.01-E:nsd-:nsl-
kolevaka,kolevka.N70.05-E:fpg-
kolevci,kolevka.N70.05-E:fsd-:fsl-,kolevka.N72.02-E:fsd-:fsl-
kolevka,.N72.02-E:fsn-,.N70.05-E:fsn-
kolevkama,kolevka.N70.05-E:fpd-:fpi-:fpl-,kolevka.N72.02-E:fpd-:fpi-:fpl-
kolevke,kolevka.N70.05-E:fsg-:fpn-:fpa-:fpv-,kolevka.N72.02-E:fsg-:fpn-:fpa-:fpv-
kolevki,kolevka.N72.02-E:fpg-
kolevko,kolevka.N70.05-E:fsv-,kolevka.N72.02-E:fsv-
kolevkom,kolevka.N70.05-E:fsi-,kolevka.N72.02-E:fsi-
kolevku,kolevka.N72.02-E:fsa-,kolevka.N70.05-E:fsa-
kolijevaka,kolijevka.N70.05-J:fpg-
kolijevci,kolijevka.N70.05-J:fsd-:fsl-,kolijevka.N72.02-J:fsd-:fsl-
kolijevka,.N70.05-J:fsn-,.N72.02-J:fsn-
kolijevkama,kolijevka.N70.05-J:fpd-:fpi-:fpl-,kolijevka.N72.02-J:fpd-:fpi-:fpl-
kolijevke,kolijevka.N70.05-J:fsg-:fpn-:fpa-:fpv-,kolijevka.N72.02-J:fsg-:fpn-:fpa-:fpv-
kolijevki,kolijevka.N72.02-J:fpg-
kolijevko,kolijevka.N70.05-J:fsv-,kolijevka.N72.02-J:fsv-
kolijevkom,kolijevka.N70.05-J:fsi-,kolijevka.N72.02-J:fsi-
kolijevku,kolijevka.N70.05-J:fsa-,kolijevka.N72.02-J:fsa-
kolik,.ProA02:msn*:msa-
kolika,kolik.ProA02:msg*:nsg*:fsn*:npn*:npa*
kolike,kolik.ProA02:fsg*:mpa*:fpn*:fpa*
koliki,kolik.ProA02:mpn*
kolikih,kolik.ProA02:mpg*:npg*:fpg*
kolikim,kolik.ProA02:msi*:nsi*:*pd*:*pi*:*pl*
kolikima,kolik.ProA02:*pd*:*pi*:*pl*
kolikime,kolik.ProA02:msi*:nsi*
koliko,.Adv*,kolik.ProA02:nsn*:nsa*:
```

kolikog,kolik.ProA02:msg*:nsg*:msa+
kolikoga,kolik.ProA02:msg*:nsg*:msa+
kolikogod,.Adv*
kolikoj,kolik.ProA02:fsd*:fsl*
kolikom,kolik.ProA02:msd*:msl*:nsd*:nsl*:fsi*
kolikome,kolik.ProA02:msd*:msl*:nsd*:nsl*
kolikomu,kolik.ProA02:msd*:msl*:nsd*:nsl*
koliku,kolik.ProA02:msd*:msl*:nsd*:nsl*:fsa*
kolima,kolo.N51.01-*:npd-:npi-:npl-,kola.N62.01-*:npd-:npi-:npl-
kolo,.N51.01-*:nsn-:nsa-:nsv-
kolom,kolo.N51.01-*:nsi-
kolovođa,.N75.01+*:msn+:mpg+
kolovođama,kolovođa.N75.01+*:mpd+:mpi+:mpl+
kolovođe,kolovođa.N75.01+*:msg+:mpn+:mpa+:mpv+
kolovođi,kolovođa.N75.01+*:msd+:msl+
kolovođo,kolovođa.N75.01+*:msv+
kolovođom,kolovođa.N75.01+*:msi+
kolovođu,kolovođa.N75.01+*:msa+
kolu,kolo.N51.01-*:nsd-:nsl-
kolxa,kolxe.N60.01-*:nsg-:npn-:npg-:npa-:npv-
kolxe,.N60.01-*:nsn-:nsa-:nsv-,klati.V21.13.4*:P3s
kolxem,kolxe.N60.01-*:nsi-,klati.V21.13.4*:P1s
kolxemo,klati.V21.13.4*:P1p
kolxena,kolxeno.N51.01-J:nsg-:npn-:npg-:npa-:npv-
kolxenima,kolxeno.N51.01-J:npd-:npi-:npl-
kolxeno,.N51.01-J:nsn-:nsa-:nsv-
kolxenom,kolxeno.N51.01-J:nsi-
kolxenu,kolxeno.N51.01-J:nsd-:nsl-
kolxete,klati.V21.13.4*:P2p
kolxeš,klati.V21.13.4*:P2s
kolxi,klati.V21.13.4*:Y2s
kolxima,kolxe.N60.01-*:npd-:npi-:npl-
kolximo,klati.V21.13.4*:Y1p
kolxite,klati.V21.13.4*:Y2p
kolxiva,kolxivo.N51.01-*:nsg-:npn-:npg-:npa-:npv-
kolxivima,kolxivo.N51.01-*:npd-:npi-:npl-
kolxivo,.N51.01-*:nsn-:nsa-:nsv-
kolxivom,kolxivo.N51.01-*:nsi-
kolxivu,kolxivo.N51.01-*:nsd-:nsl-
kolxu,kolxe.N60.01-*:nsd-:nsl-,klati.V21.13.4*:P3p
kolxući,klati.V21.13.4*:AdvPr
kolče,kolac.N20.61-*:msv-,kolac.N18.01-*:msv-
kolčeva,kolac.N20.61-*:mpg-
kolčeve,kolac.N20.61-*:mpa-
kolčevi,kolac.N20.61-*:mpn-:mpv-
kolčevima,kolac.N20.61-*:mpd-:mpi-:mpl-
kom,tko.ProN12:**d+:**l+,ko.ProN12:**d+:**l+,koji.ProA07:msd*:msl*:nsd*:nsl*
komad,.N04.01-*:msn-:msa-
komada,komad.N04.01-*:msg-:mpg-
komade,komad.N04.01-*:msv-:mpa-
komadi,komad.N04.01-*:mpn-:mpv-
komadima,komad.N04.01-*:mpd-:mpi-:mpl-
komadom,komad.N04.01-*:msi-
komadu,komad.N04.01-*:msd-:msl-
komar,.N01.01+*:msn+
komara,komar.N01.01+*:msg+:msa+:mpg+
komarac,.N17.61+*:msn+
komaraca,komarac.N17.61+*:mpg+

```
komarca,komarac.N17.61+*:msg+:msa+
komarce,komarac.N17.61+*:mpa+
komarcem,komarac.N17.61+*:msi+
komarci,komarac.N17.61+*:mpn+:mpv+
komarcima,komarac.N17.61+*:mpd+:mpi+:mpl+
komarcu,komarac.N17.61+*:msd+:msl+
komare,komar.N01.01+*:msv+:mpa+
komari,komar.N01.01+*:mpn+:mpv+
komarima,komar.N01.01+*:mpd+:mpi+:mpl+
komarom,komar.N01.01+*:msi+
komaru,komar.N01.01+*:msd+:msl+
komarče,komarac.N17.61+*:msv+
kome,ko.ProN12:**d+:**l+,tko.ProN12:**d+:**l+,koji.ProA07:msd*:msl*:nsd*:nsl*
komegod,kogod.ProN12:**d+:**l+,tkogod.ProN12:**d+:**l+
komgod,kogod.ProN12:**d+:**l+,tkogod.ProN12:**d+:**l+
komod,.N04.01-*:msn-:msa-
komoda,komod.N04.01-*:msg-:mpg-
komode,komod.N04.01-*:msv-:mpa-
komodi,komod.N04.01-*:mpn-:mpv-
komodima,komod.N04.01-*:mpd-:mpi-:mpl-
komodom,komod.N04.01-*:msi-
komodu,komod.N04.01-*:msd-:msl-
komu,tko.ProN12:**d+,ko.ProN12:**d+,koji.ProA07:msd*:msl*:nsd*:nsl*
komugod,kogod.ProN12:**d+,tkogod.ProN12:**d+
komšija,.N77.01+*:msn+:msv+:mpg+
komšijama,komšija.N77.01+*:mpd+:mpi+:mpl+
komšije,komšija.N77.01+*:msg+:mpn+:mpa+:mpv+
komšiji,komšija.N77.01+*:msd+:msl+
komšijom,komšija.N77.01+*:msi+
komšijska,komšijski.A03.01*:p@fsn*:p@fsv*:p@npn*:p@npa*:p@npv*
komšijske,komšijski.A03.01*:p@mpa*:p@fsg*:p@fpn*:p@fpa*:p@fpv*
komšijski,.A03.01*:p@msn*:p@msa-:p@msv*:p@mpn*:p@mpv*
komšijskih,komšijski.A03.01*:p@*pg*
komšijskim,komšijski.A03.01*:p@msi*:p@nsi*:p@*pd*:p@*pi*:p@*pl*
komšijsko,komšijski.A03.01*:p@nsn*:p@nsa*:p@nsv*
komšijskog,komšijski.A03.01*:p@msg*:p@msa+:p@nsg*
komšijskoj,komšijski.A03.01*:p@fsd*:p@fsl*
komšijskom,komšijski.A03.01*:p@msd*:p@msl*:p@fsi*:p@nsd*:p@nsl*
komšijsku,komšijski.A03.01*:p@fsa*
komšiju,komšija.N77.01+*:msa+
konac,.N18.61-*:msn-:msa-
konaca,konac.N18.61-*:mpg-
konaci,konak.N04.04-*:mpn-:mpv-
konacima,konak.N04.04-*:mpd-:mpi-:mpl-
konak,.N04.04-*:msn-:msa-
konaka,konak.N04.04-*:msg-:mpg-
konake,konak.N04.04-*:mpa-
konakom,konak.N04.04-*:msi-
konaku,konak.N04.04-*:msd-:msl-
konat,.N04.51-*:msn-:msa-
konata,konat.N04.51-*:mpg-
konače,konak.N04.04-*:msv-
konca,konac.N18.61-*:msg-
konce,konac.N18.61-*:mpa-
koncem,konac.N18.61-*:msi-
konci,konac.N18.61-*:mpn-:mpv-
koncima,konac.N18.61-*:mpd-:mpi-:mpl-
koncu,konac.N18.61-*:msd-:msl-
```

D.3 Raspored imenskih reči i glagola po klasama

D.3.1 Imenice iz rečnika DELA po morfografemskim klasama

N01.01+

arap	bakalar	bećar	biskup	brav
crv	dever	dizdar	djever	domaćin
dorat	elefant	filosof	garov	gavran
golub	gospod	gospodar	govedar	gušter
horjat	iguman	jaran	jastreb	jastrijeb
kaluđer	kapetan	kaplar	kaur	kokot
komar	korman	krčmar	kuvar	lopov
majmun	majstor	mazgov	mađed	medved
mrav	murtat	narod	nerast	nečastiv
nizam	obad	orjat	pandur	pelivan
pobratim	pogrkos	posrbos	ratar	sirotan
sused	susjed	telal	tetreb	tetrijeb
troskot	tupan	vampir	varvar	vezir
vilan	vodičar	vran	zlotvor	zver
zvijer	šarov	čikov	čoban	

N01.02+

beg	drug	pjevidrug	plačidrug	vrag

N01.04+

bolesnik	crkvenxak	dedak	divlxak	dužnik
duhovnik	grešnik	hajduk	izdajnik	jatak
junak	kamatnik	konxanik	konxik	krivokletnik
krvnik	kurjak	linxak	lovnik	navađenik
nemotnxik	nemoćnik	nevernik	nevjernik	novlxak
pauk	pešak	pješak	pokajnik	poturčenxak
prosjak	putnik	rodnik	sirak	težak
ujak	utoplxenik	utopnik	uškoplxenik	vojak
vojnik	vuk	zemlxoradnik	zloguk	đak

N01.07+

duh	patrijarh

N01.08+

anđeo	đavo

N01.51+

fratar

N01.54+

deverak	djeverak	jedinak	krmak	momak
mravak	sinak			

N02.01+

brat	gospodin

N02.04+

čovek	čovjek

N03.01+

jarići	juniči	kozlići	kumčići	kučići
magarići	mačići	paščići	pačići	prasići
ptičići	siročići	telići	tičići	unučići
ćurići				

N03.11+

lxudi

N04.01-

život	acal	ahar	akov	alat
amanat	amanet	amat	ambar	aršin
astal	avan	babazeman	berat	beričet
beškot	biber	bir	bircauz	biričet

birov	bisag	biser	blagoslov	bostan
bunar	bungur	cekin	celiv	celov
dan	dinar	dizgust	dogovor	dom
domuz	donos	dram	dud	dukat
duvan	duvar	dyenet	dyevap	dyombos
esap	espap	golen	golijen	gontunar
govor	grklxan	grmen	grum	grumen
guber	hair	hajat	hajgir	hajvar
hatar	hater	hesap	inat	interes
inćar	iskop	iver	izgled	izvor
jad	jemin	jordan	kal	kalauz
kantar	kantun	kapot	kaput	karantan
karar	kesten	kijamet	kitab	kivot
klet	klijet	kob	komad	komod
kov	kremen	kupus	kuskun	laz
lokot	lopar	lov	majdonos	mal
manastir	mart	mehur	miris	mjehur
most	nakot	način	nedoďin	negled
nepovrat	obor	ocjed	odzdrav	oglav
okrp	oltar	otrov	pargal	patlidyan
perčin	pečat	pilav	pipun	pohod
pojas	pojat	poklon	pokrov	poplat
poprd	porod	potkov	potplat	pozdrav
procep	procijep	prst	prt	prud
razgovor	račun	rezil	rukav	sahat
samun	sapun	savet	savjet	selam
sevap	sijaset	skut	somun	spas
sugreb	taban	talambas	tavan	teret
trun	uglxen	vagan	vakup	vašar
vinober	vinograd	zaklad	zakon	zanat
zatar	zater	zauz	zavet	zavjet
začin	zejtin	zeman	zub	šaltor
šapat	šejtan	ďerdan	ćitab	ćivot
ćud	ćumur	čivit	čokot	

N04.02-

beleg	bilxeg	breg	brijeg	sneg
snijeg				

N04.04-

alvaluk	barak	bardak	barjak	begluk
brzak	buk	bulxuk	dimnxak	jezik
kijak	klobuk	kolak	konak	kvak
lešnik	lxešnik	mertik	mizdrak	narok
nijek	oblak	odyak	ponedelxnik	ponedjelxnik
potok	praznik	prednxak	puk	strok
svetnxak	svijetnxak	težatnik	tukoluk	urok
utrenik	zdravlxak	zrak		

N04.07-

greh	grijeh

N04.08-

bario	presto	prijesto

N04.51-

godimenat	ječam	konat	litar	nokat
ocat				

N04.54-

brabonxak	obojak	opanak	rastanak	sastanak
uranak	vazdanak			

N04.58-

kotao	čavao

N05.01-

grad	islam	kamen	led	petrusin
tamjan	tamnxan			

N05.08-

pepeo				

N06.01-

ždrmanxi	burići	lončići	ćebići	

N07.01+

ždral	at	ban	brav	ded
djed	dyin	gad	gem	hat
hrt	kmet	kos	kum	lav
pan	paun	pop	ris	rob
rod	sin	skot	slon	svat
zet				

N07.02+

beg	drug	vrag	ćor-beg	

N07.04+

vuk				

N07.07+

duh				

N07.08+

soko	vo			

N07.51+

ovan				

N08.01-

žbun	žir	ajns	am	aps
baj	bat	bes	bijes	bob
bol	bor	brod	brus	bun
cvet	cvijet	dar	dim	dlan
drem	dren	drijem	drijen	dub
dvor	dyep	glas	gnev	gnxat
grad	grm	grob	grom	grozd
gust	guz	ham	haps	hlad
hleb	hlxeb	hod	hrbat	hreb
hren	jed	kam	kar	kip
klin	klxun	krst	kus	kut
kvas	lan	leb	led	list
lxeb	med	mlin	nos	pas
plast	plen	plot	post	prsten
prut	rast	rat	red	rep
rit	rt	sat	slom	sram
srp	stan	stid	stvor	sud
svet	svijet	svrab	top	tor
trem	tres	trijes	trn	trs
trud	trup	um	vez	vid
vrat	vrt	zbor	zev	zid
zijev	štap	đem	đon	ćup
čap	čas	čep	čin	čun

N08.02-

beg	bijeg	breg	brijeg	sneg
snijeg				

N08.04-

puk	tuk			

N08.12-

rbat				

N08.51-

jaram	lakat	najam	petar	san
ujam	vetar	vjetar	zajam	

N09.09+
zec
N11.01+
dušmanin hrišćanin hrsuzin kasapin lužanin
čifutin čivutin čobanin
N11.01-
turin
N12.01+
mornar mrnar oficir pastir patrijar
pivar podrumar poklisar ugor čuvar
N12.04+
bonik ležak svedok svjedok
N12.07+
pastuh siromah
N12.51+
pas svekar
N12.54+
gusak kučak svračak volak
N13.01-
cvancik dyeferdar dyever koren korijen
muz nemir obraz oprez otpor
pancir papar pazar pendyer peškir
pir pogovor povraz prekor prigovor
prijekor prkos proces psaltir raf
saltir samar sir stožer sunđer
talijer talir tanxir tefter topuz
tovar ular umuz šićar ćar
čemer
N13.02-
bubreg lug polog razlog sapog
tarčug zalog
N13.04-
branik brk fišek istok jek
jošik kajmak krak kukurek kukurijek
mrzluk muštuluk mučnxak pamuk petak
sokak tandrk uzmak uzrok šenluk
šiblxak čabrenik čekrk čibuk
N13.06-
žitak lupatak napitak naplatak napredak
nazadak nestadak početak srditak težatak
užitak ustuk četvrtak
N13.07-
kožuh orah pazuh trbuh
N13.08-
počeok
N13.1.51-
ovas
N13.12-
priložak
N13.51-
čabar
N13.54+
masak
N13.54-
banak dalak danak domak klipak
nalivak nxisak oplečak papak sirak
smrčak ugarak upočak uštipak varićak
zastorak zatilak zatilxak šilxak šupak

čanak čarak
N14.01+
bik rak
N14.51+
orao tetak
N14.58+
detao djetao petao pijetao
N15.01-
ak bok brk buk dug
frk grah greh grijeh koren
korijen krak krug kruh lek
lijek lik lug luk mak
meh mijeh mir mrak plug
prag prah rog rok skok
smeh smijeh smok stog strah
trag trk tur uk vek
vijek zvek špag štrk čir
N15.08-
deo dio do kotao sto
N15.51-
mučak pesak pijesak prčak pupak
ručak točak vosak šipak ćošak
ćurak čičak
N15.58-
kabao pakao posao ugao uzao
N17.01+
bakalaj
N17.11+
mrtac mrtvac
N17.12+
bogac dušogubac kobac rožac ubogac
vrabac
N17.15+
nosac prasac
N17.16+
domorodac gladac otac platac sudac
svetac svitac
N17.18+
davalac dobivalac hvalilac ranilac ranoranilac
sedilac sjedilac tužilac
N17.61+
adumac begovac begunac bjegunac bratac
bunxevac carevac dizdarac docnolegalac jaganxac
jarac jedinac jemac kičac klinac
komarac kradlxivac krivac kupac lažac
lažlxivac lakomac lovac magarac našinac
parac pevac pijanac pijevac pjanac
poganac sinovac sivac skakavac skupac
slepac slijepac starac strašlxivac svirac
telac tenac trgovac udovac ujac
vranac vukovac znanac čudotvorac
N18.01-
grij kolac zalogaj
N18.11-
ajnc kotlac mesec mjesec prijesnac
prnxic vrbopuc
N18.12-
hlebac hlxebac podrobac zubac

N18.16-

kotac koštac

N18.18-

kotalac tobolac

N18.61-

živac	glogovac	gubac	jadac	katanac
klanac	konac	konopac	krastavac	krušac
lonac	mlinac	novac	pokrovac	pomolac
prdavac	sirac	slinac	stanac	tobolac
trupac	udarac	vijenac	zaklopac	škripac
ďurkovac				

N19.01+

car

N19.11+

stric

N19.16+

otac

N19.61+

jarac

N20.01-

boj	broj	vaj	znoj

N20.1.11-

koštac

N20.11-

lxuc

N20.61-

kolac

N21.01+

birtaš	bratić	cecelx	detlić	deveričić
djetlić	djeveričić	golić	govnović	grabancijaš
grďević	hajkač	hranitelx	hrsuz	izbirač
kikoš	kokoš	konx	kormoš	kovač
kralxević	krpelx	lupež	navičaj	našiš
neprijatelx	obirač	odyaković	ološ	plivač
pogaďač	potutkač	poštipač	prebirač	preprijatelx
prijatelx	probirač	pudar	roditelx	sinčić
slavuj	strelxač	strugač	stvoritelx	svirač
vranx	ďetić			

N22.01-

žeželx	žuč	bakrač	belx	bijelx
bulumać	cangurij	doboš	dupčić	egeduš
grabež	grošić	hajtaš	harač	ispokoj
jemlxeš	kaiš	kamiš	klxučić	kolač
kontentaj	košić	krcelx	krpež	kukolx
kupikrastavčić	lončić	luč	marač	marjaš
marč	milx	nefalx	nožić	običaj
opasač	otirač	ovrlx	očenaš	pestiš
pilx	pištolx	podlogač	pokoj	pokolx
ponxiš	potočić	povoj	pozder	prdež
tiganx	tonx	trpež	varoš	višt
vršaj	zavoj	zduhač	čuvalduz	

N22.51-

žrvanx	badanx	odar	pedalx	ražanx

N23.01+

ždralx	jež	kralx	miš	muž
pregalx	puž	spuž	vrač	

N23.01-

broć	dažd	groš	gunx	klxuč

koš kraj križ krš loj
malx mač mraz nož palx
panx plač raj

N23.51-
bubanx oganx pikalx ugalx šušanx
češalx

N24.51-
kašalx

N25.01+
gost

N26.01-
nokat prst zub

N26.1.01+
lxudi

N31.01+
braco hoćo nećo

N40.01+
babo bego doro jeko kunxado
ćako ćoro ćuko

N40.03-
kutao

N41.01-
konto

N42.01+
brato hadyo dūro

N51.00-
blago

N51.01+
govedo pseto

N51.01-
žarilo ždrelo ždrijelo žensko žito
barilo blato brašno brdo brvno
cedilo cjedilo crevo crijevo crtalo
davalo delo dizalo djelo dno
držalo drvo gnezdo gnijezdo grabilo
grizalo grlo gudalo gudilo izdiralo
jato jecalo jelo jezero kamkalo
kandilo kokalo koleno kolo kolxeno
kolxivo konxado kopito korito kormilo
krilo krčalo kudilxevo kusalo leto
liko lxeto meso mesto mito
mjesto mleko mlivo močilo mrzilo
nazivalo nebo ocilo ogledalo ognxilo
olovo opelo opijelo pazuho pecivo
pero pluto proso puto ralo
ramo rešeto rilo ruho runo
sadno salo sedalo selo seno
sečivo sijeno sito sjedalo sječivo
slovo solilo sočivo stado strašilo
tecivo telo testo tijelo tijesto
trkalo uzimalo vino vratilo vreteno
zlato zrcalo zrno zvono šilo
čekalo čelo čudo

N51.02-
ženstvo bogatstvo bratstvo carstvo deblo
dobro društvo gospodstvo govno grdilo
gumno horjatstvo junaštvo jutro kluvko
kumstvo maslo pismo platno prijatelxstvo

	pučanstvo	rebro	ropstvo	sedlo	siromaštvo
	srebro	staklo	trojstvo	vlasteostvo	zlo
	zlojutro	čojstvo	čoveštvo	čovještvo	
N51.03-					
	klupko				
N52.01+					
	drvo				
N52.01-					
	govno	uvo			
N53.01-					
	nebo				
N54.01-					
	mleko	mlijeko	oko	uho	uvo
N60.01-					
	arište	ašikovanxe	babine	bosilxe	breme
	brijeme	brčište	bunište	bunxište	dance
	davanxe	dugovanxe	godište	grebenište	grožđe
	groblxe	gvožđe	jaje	jevanđelije	jevanđelxe
	kajanxe	kolxe	koplxe	kovilxe	kusanxe
	kućište	lice	more	mučanxe	neimanxe
	obećanxe	odrište	ognxište	ome	oproštenije
	oproštenxe	oranxe	oružje	otmaštenije	ošće
	perje	piće	plakanxe	pleme	pleće
	pojanxe	polxe	pomirište	poslušanxe	posvadište
	poštenxe	poštipivanxe	prikazanije	prišestvije	prkance
	proleće	prošće	prtište	punxe	radovanxe
	raďanxe	rogulxe	roďenxe	sekirište	seme
	sirće	sjekirište	sjeme	sleme	slxeme
	smilxe	snoplxe	spasenije	srce	srebarce
	strnište	stvorenxe	sudište	sunce	suďenxe
	tečenxe	trnxe	ubijanxe	ubijenxe	uglxevlxe
	ulxe	umrće	uzdarje	učenxe	vaskrsenije
	vatrište	vavedenxe	venčanxe	veselxe	vime
	vjenčanxe	voće	vraćanxe	vreme	vrijeme
	vrtanxe	zdravlxe	zelxe	znanxe	zvonce
	šaraglxe	šetanxe	ševrdanxe	čatenije	
N60.02-					
	koplxe	rebarce			
N60.05-					
	drveće	drvlxe	granxe	kamenxe	klasje
N60.07+					
	teoce				
N61.01-					
	pleće				
N62.01-					
	klešta	kliješta	klxusa	kola	leďa
	nebesa	nosila	pleća	prsa	usta
	vešala	vješala	vrata		
N62.02-					
	nedra	nxedra			
N63.01-					
	vrata				
N64.01-					
	doba				
N65.01+					
	ždrebe	ždrijebe	dete	dijete	goveče
	jagnxe	jaje	jare	june	klxuse
	kopile	kozle	kumče	kuče	magare

mače	momče	nedonošče	nejače	pašče
pače	pile	prase	ptiče	rebarce
siroče	somče	srce	tane	tele
tiče	unuče	zvere	zvijere	štene
ďete	čavče	čelxade		

N65.01-

lonče	rebarce	sefte	sirće	uže
zvonce	ćebe			

N66.01+

iže	izjalocče	piple	trogoče	šiše

N66.01-

binxašče	birašče	bure	japundye	klxuse
mašče	vince	ďubre		

N67.01-

podne

N68.01-

ime	rame	teme	tjeme	

N69.01-

veče

N70.01+

žaba	žena	živina	životinxa	aždaha
aždaja	ala	baba	babetina	baka
belxa	bestija	beštija	bjelača	blebetuša
bubamara	budala	buha	bula	daša
devojčina	djevojčina	dobrulxa	dodola	dojkinxa
dovotkinxa	gera	gica	gospoďa	grmuša
guja	izješa	kamila	kesega	kobila
koka	koleda	konxina	kopriva	kornxača
koza	kozopaša	krava	krmača	kuja
kuma	lasta	leskovina	lija	lipa
loza	lxeskovina	lxuba	maca	mahovina
maja	mama	marva	mača	maćeha
mlečara	mlxečara	mnogulxa	moma	moruna
muha	muva	mušmula	nehteša	nehtješa
nemogoša	nena	neva	nevesta	nevjesta
ovčina	porodilxa	prelxa	psina	ptica
pčela	raja	reduša	repa	riba
roguša	ruža	sotona	sova	suseda
susjeda	svinxa	tanxa	teta	tetka
tica	tkalxa	trava	tuka	tuďiša
tučija	vedogonxa	vezilxa	vila	vrana
zeba	zličina	zlotkalxa	zmija	zolxa
ševa	šlxiva	šlxuka	štediša	štuka

N70.01-

želxa	žerava	žeďa	žica	žila
žlica	aba	ala	bakuštaja	balega
balota	bara	bareta	batina	baština
bačina	beda	beseda	besjeda	bijeda
bilxetina	bjelxina	boca	boja	brada
brana	brzina	bubota	budalaština	bukila
bundeva	bunika	bura	busija	carevina
cena	cicvara	cijena	cipela	ckvara
cvoka	dara	daća	dimiskija	dimiškinxa
dinxa	dobraklija	dobrota	dogovorna	dolama
dolina	držala	drača	družina	duvankesa
duša	dušina	dyamija	faca	fala
furuna	fučija	galija	gara	gazeta
glasina	glava	glavobolxa	glota	glxiva

goba	godina	gomila	gora	gotovina
gočobija	gradina	grana	grba	greda
grehota	greota	griža	griva	grmlxavina
guba	guza	guzobolxa	guša	hala
halxina	hrana	hudoba	hvala	ica
ikona	imovina	inokoština	istina	jagoda
jama	java	jeza	jeđa	juha
juva	kaca	kadifa	kafa	kajda
kalina	kalvarija	kandyija	kapa	kapulxača
karaboja	katana	kaša	kašika	keba
kera	kesa	kika	kila	kirija
kiša	klada	koža	koba	kora
korba	kosa	kotluša	košulxa	krajcara
krajina	kravlxača	kradža	kreda	kresta
krivica	krmetina	krtina	kruna	krupa
krčevina	kužina	kudelxa	kuga	kuka
kukulxica	kula	kupina	kuća	kućerina
kućiština	kučina	laža	lakoća	lazina
lađa	ledina	lepota	letina	leturđija
leća	lika	lima	liturgija	ličina
lopata	lubina	lula	lutrija	lxepota
lxetina	madyarija	mahrama	majstorija	mana
mandya	mangura	marama	maskara	masla
mašala	mehana	mera	mesečina	mešina
međa	mećava	milostinxa	mira	mjera
mjesečina	mješina	množina	mora	mraka
mreža	muža	munxa	nafaka	nafora
nagrda	namera	namjera	naplata	natraga
navaka	navora	nedaša	nedelxa	nedođija
nehara	neimaština	nemaština	nesreća	nestaša
nevera	nevjera	nevolxa	neđelxa	novina
nxiva	obasjanija	oblačina	obrana	obuća
odbrana	odmena	odmjena	oduka	odvala
odvika	ograda	oka	opaklija	oputa
ora	osovina	osveta	očevina	palata
palača	panada	panahija	papa	para
paša	pedepsa	pedevsija	peta	petlxanija
pećina	pinxata	pita	planina	plata
plaća	pleva	ploča	plxeva	pogača
pogibija	pojata	poka	polaža	polača
polovica	polovina	polutina	pomama	pomrčina
ponta	ponxava	popara	porcija	postelxa
potreba	povala	pošta	praznina	prašina
prda	prdačina	prdnxava	preporuka	pretilina
prevrtača	preša	pređa	prga	prigoda
priguta	profunta	proha	proja	proskura
prosulxa	prtenxara	prtina	prćija	pukotina
putina	pučina	rabota	rakija	rana
raskuća	ravnica	raša	refena	roba
rosa	rtica	ruda	rupa	rusa
rutina	rđa	sekira	sepija	sermija
seta	sila	siromaština	sirotinxa	sisa
sjekira	sjeta	skela	skuža	skvara
slama	slana	slanina	slava	sloboda
sluta	smrekinxa	soba	soha	splačina
sramota	srdyba	sreda	sredina	sreća
srijeda	srma	starina	stena	stijena
stima	stolica	stopa	strana	streha

strela	strijela	strvina	stuba	subota
suvača	suza	suša	svetina	svila
svjetina	tama	tarana	tava	tazbina
tendyera	tepsija	tojaga	tolxaga	trgovina
trka	trpeza	tužica	tugovina	tuta
tuđina	tvrđava	užina	uboština	udaja
ura	usna	usprema	utroba	uvratina
uzda	vajdica	varnica	vasilxena	vasiona
vaslava	vedrina	velxača	vera	veresija
večera	vika	vjera	voda	volxa
voskovarina	vrba	vreća	vuna	zadužbina
zahvala	zakrpa	zaprdica	zaseda	zasjeda
začina	zbilxa	zdela	zdila	zdjela
zgoda	zima	zloba	zolota	zora
zvezda	zvijezda	šala	šapa	šiba
šija	škola	štala	šteta	šuga
šuka	šuma	šuplxika	šuša	ćuprija
ćuskija	čarapa	čaša	čelebija	čengija
čizma	čoha	čorbušta	čuma	čutura

N70.02+

bilxka	kvočka	mačka	ovca	patka
sestra	trešnxa	vaška	višnxa	ćurka

N70.02-

žetva	bedra	bradva	britva	crkva
igla	igra	litra	pesma	pjesma
plećka	puška	voćka	zemlxa	

N70.04+

brzoreka	druga	guska	konoplxika	perunika
snaha	snaša	stoka	vuga	

N70.04-

bisaga	briga	bruka	dika	dlaka
gramatika	guka	hajka	huka	istraga
jabuka	knxiga	motika	muka	muzika
nauka	navika	pliska	poruka	prilika
reka	rijeka	slika	sloga	snaga
struka	trka	tuga	zadruga	zraka
šaka	čorboloka			

N70.05+

devojka	djevojka

N70.05-

daska	drška	igračka	kolevka	kolijevka
kruška	lxuska			

N70.61+

braća	deca	dica	djeca	gospoda

N70.76-

cvancige	demija	dimija	gaće	grablxe
gusle	mekinxe	oje	poklade	talxige
vile	zazubice	zjale	čakšire	

N71.01+

žalosnica	babica	carica	gladnica	gubavica
izjelica	jetrvica	junica	krčmarica	kukavica
kulašica	kučnica	lavica	lisica	lxubica
lxutica	majčica	matica	mučenica	neimalica
nejačica	ovčica	pijanica	pijavica	piskavica
pjanica	prasica	prepelica	punica	ribica
ružica	senica	sjenica	snašica	sudica
tikvica	udovica	varalica	veštica	vilica

N71.01-

žeravica	žičica	aščijnica	banica	berbernica
bezadnxica	bitvica	brašnxevica	brnxica	crkavica
držalica	dušica	dyigerica	fortica	gibanica
glavica	glavnica	gronica	groznica	gubica
guzica	ječmenica	kabanica	kapica	kaplxica
kiselica	kišnica	kobasica	kotarica	krilatica
kućica	lubenica	merica	mešinica	mjerica
mješinica	močionica	mrvica	muzlica	nestašica
nizbrdica	nizdolica	nizgorica	nogavica	ožica
odšalica	ovsenica	palica	patarica	pečenica
podrepnica	pokajnica	polica	pozajmica	pravica
prijeglavica	pšenica	ručica	slaninica	slobodica
slobodičica	suknxica	tamnica	torbica	trica
trupica	ulica	utrobica	uzbrdica	uzglavnica
uzgorica	uzica	varvarica	vedrica	vodenica
vodica	volxica	vrbica	zadnxica	zadušnica
zaušnica	začepica	zenica	zjenica	šenica
štica	šućurica	ćušica	čađavica	česnica

N71.02-

pesnica				

N71.1.01+

mama	popadija	strina		

N72.01+

dikla	gospa	kokša	kurva	kučka
kćerca	leska	lijeska	mazga	mečka
planinka	rotkva	smokva	svekrva	tašta
tikva	zverka	zvjerka	šćerca	ćurka
čaplxa	čavka			

N72.01-

žurba	alva	aspra	avra	bačva
berba	bečka	bočka	crevlxa	družba
duplxa	fajda	flinta	forinta	forteca
furba	gizda	glamnxa	glavnxa	građa
grdnxa	grinta	gužva	halva	halxka
hitnxa	jagma	jufka	kaplxa	karablxa
kavga	klašnxa	kletva	klin-čorba	krivda
krčma	levča	lijevča	lubarda	magla
metla	molba	molitva	mržnxa	natra
naćve	nužda	perda	pećka	plačka
plećka	pogodba	pomnxa	pravda	prdalxka
prošivalxka	prošnxa	pročka	radnxa	revka
sablxa	seka	slamka	služba	sofra
sržba	suknxa	surutka	svadba	svađa
sveća	svijeća	svirala	tajna	taška
tigla	torba	trešnxa	trska	tušta
udadba	unča	uzma	vajda	varka
vatra	zakletva	zatra	zublxa	zvrčka
šajka	šipka	šišarka	šiška	štednxa
čalma	čistomenka	čorba		

N72.02+

kruška	majka			

N72.02-

batalxka	kolevka	kolijevka	kriška	nepravda
vojska				

N72.76-

gajde	jasle	krpele	ostve	

N73.01-

noga	ruka			
N75.01+				
aga	aščija	baća	bača	bena
bostandyija	braja	deda	delibaša	delija
dolibaša	elčija	gazda	gosa	hadyija
handyija	harambaša	haračlija	hatardyija	hodya
hvališa	kadija	kolovođa	kujundyija	kusonxa
mehandyija	papa	paša	pletikrošnxa	popa
prica	provodadyija	radiša	sahibija	saibija
skeledyija	spahija	starešina	starješina	sudija
torbonoša	vjedogonxa	vojvoda	vođa	vratiša
zanatlija	šeširdyija	đačina	đidibaša	đidija
ćaća	ćos-baša	ćosa	čifčija	čipčija
N75.01-				
kiridyija	staniša			
N75.02+				
pletikrošnxa				
N75.03+				
vladika				
N76.01+				
pletikotarica				
N77.01+				
komšija	neznadoša	neznajša	platiša	tata
čiča				
N77.01-				
puniša				
N78.01+				
voko	đido			
N78.1.01+				
sluga				
N80.01+				
junad	klxusad	kopilad	kozlad	kumčad
magarad	nedonoščad	paščad	prasad	ptičad
siročad	telad	tičad	unučad	štenad
čelxad				
N80.01-				
burad	cev	cijev	jesen	narav
pest	pomoć	snet	snijet	stvar
tanad	užad	večer	ćebad	
N80.02-				
ravan				
N82.01+				
kćer	mater			
N82.01-				
žeđ	kap	laž	moć	naruč
peć	raž	reč	riječ	sen
sjen				
N82.02+				
paščad				
N82.02-				
glad	krv	lxubav	mladost	pamet
paprat	plesan	smrt	snet	snijet
N82.03-				
so	čast			
N82.04-				
žalost	bolest	dužnost	ispovest	ispovijest
korist	kost	last	ludost	mast
milost	mrzost	mudrost	nadmenost	napast

obilnost	oblast	oholost	plahost	pripovest
pripovijest	radost	slast	starost	vlast
zapovest	zapovijest	zavist	čovječnost	

N83.01-

noć

N84.01+

uš

N84.01-

kost

N85.01-

uši

N86.01-

oči

N87.01+

kći

N90.00-

doba

N99.01-

mozak

N99.03+

knez	vitez

D.3.2 Pridevi iz rečnika DELA po morfografemskim klasama

A02.01

Bježanov	Bogov	Lazarev	Lizin	Markov
Milićev	Milošev	Petrov	Stojanov	Đurin
Đurov	Đurđev	ženin	babin	biserov
bobov	borov	brajin	carev	drenov
gosin	gospodarev	gospodarov	hatarov	hlebarov
hlxebarov	hrastov	jeftin	junakov	kasalov
knežev	kralxev	kumin	kumov	kurvin
lipov	lozov	lxeskov	majčin	materin
očev	očin	popin	popov	prohin
projin	rastov	sirov	sunčev	svekrvin
tatin	vilin	vladičin	zemlxin	zetov
čarapin				

A03.01

Aleksandrijski	Carigradski	Erdelxski	Jeđupački	Lesandrijski
ženski	adamski	arbanaški	blagosloven	bliži
bratski	carski	ciganski	desni	devojački
djevojački	futoški	gospodski	gradski	hajdučki
hajđidi	horjatski	hrišćanski	hrsuski	hrvatski
iskonski	isti	istinski	jedini	jesenski
kaluđerski	kanarski	kasapski	komšijski	konxski
kosovski	kosturski	kotorski	kranxski	krsni
kukuruzni	kupovni	kućni	lanxski	logovski
lovački	lovni	lubenični	lxudski	marčani
mnogi	morski	mostarski	muški	nebeski
nejedini	nemački	neprijatelxski	nxemački	onomlanxski
orjatski	ostali	očni	peraški	petni
popovski	poskurni	prijatelxski	prosjački	puliješki
ražani	ražni	racki	sarajevski	siromaški
sirotinxski	slxepački	srpski	starinski	svetski
svečarski	svijetski	torni	turski	umrli
utorni	velxi	vodenični	vrdnički	vreočki

zemalxski	zimski	zubni	šokački	đavolski
đački	čifutski	čobanski		
A03.02				
Božiji	Božji	današnxi	domaći	donxi
gladinxi	goveđi	guščiji	jagnxeći	jesenxi
jutarnxi	jutrenxi	kutnxi	letnxi	lxetnxi
magareći	mišiji	mišji	mrzeći	ovčiji
očinxi	pasji	pačji	pletaći	pletići
poslednxi	poslxednxi	potonxi	prednxi	ptičji
riblxi	sinxi	srednxi	stražnxi	suvišnxi
tičji	večernxi	vražiji	vražji	vranxi
vučiji	vučji	zadnxi	đavolxi	
A04.01				
levi	lijevi			
A06.01				
žežen	beličast	besposlen	bezrep	bijen
bos	bređ	cijel	cklen	crnook
drven	dugorep	dur	frug	glavat
golem	gvozden	hud	izjeden	jednak
klenit	kršten	kus	lanen	mehovit
mjehovit	nag	nedobijen	nedrag	nejak
nem	nenačet	neoprošten	neplaćen	nesit
nevaren	neviđen	nijem	opleten	podjednak
poizderat	pozlaćen	prepošten	prepun	prešaren
prodrt	prten	rogat	sed	sijed
slep	slijep	smušen	srebrn	staklen
subjel	svet	trudan	ubrađen	upisan
varen	vet	voden	vrlxook	vruć
vunen	zubat	šut		
A06.03				
dugačak				
A06.51				
lijen				
A07.01				
žut	bijel	blag	brz	crn
drag	dug	gluv	gust	jak
kriv	len	lud	lxut	mlad
prek	prijek	pust	skup	suv
tih	tup	tust	tvrd	čest
A07.02				
rađ	tuđ			
A07.03				
tesan	tijesan			
A08.01				
ždrmnxan	živ	bagav	balav	bambat
blagočastiv	bodlxiv	bogat	boleć	bon
brašnav	brašnxav	bremenit	budalast	crven
crvlxiv	darovan	divalx	dosetlxiv	dosjetlxiv
gadlxiv	garav	gizdav	gluh	gologlav
gotov	govorlxiv	grbav	grk	grozničav
gubav	hrom	istrgan	izdrpan	jalov
kalxav	kilav	klet	klijenit	koštan
kradlxiv	krnx	krnxav	krvav	lajav
lakom	loš	lukav	lxutit	mahnit
manit	mator	meden	mekan	mjeden
mlogogovorlxiv	mnogogovorlxiv	mrk	mršav	mutav
nadut	naoparan	napet	nenadan	nenamazan
neopojan	nepostojan	nepotkan	nepošten	neprav

nesedlan	neslan	nevest	nevješt	nezdrav
neznan	nezvan	nov	novčan	obil
obilat	okat	omrazit	opak	paklen
pijan	pitom	pjan	plav	plašiv
plašlxiv	plemenit	pogan	pološ	pometan
poprdlxiv	poredan	potulxen	pouzdan	pošten
prav	pravcit	preslan	proklet	prost
prten	prčevit	pun	radlxiv	ran
račvast	rovit	rutav	sakat	samohran
samoran	silovit	sit	sjerčan	slab
slan	smolav	smrdlxiv	spor	srdit
srčan	stidlxiv	strašiv	strašlxiv	studen
suh	svrablxiv	tralxav	trom	ubog
ukočen	umilxat	upisan	uroklxiv	valxan
venčan	vešt	viđen	vjenčan	vješt
voštan	zamuzen	zdrav	zelen	zemlxan
zenđil	zločest	znan	šalxiv	šaren
škaklxiv	šogav	šugav	šušnxat	ćelav
ćorav	čađav	čestit	čipav	čist
čitav	čuven			

A08.02

žedan	žitan	bezobrazan	bezuman	bistar
bogounosan	bolan	cvetan	cvjetan	dostižan
dužan	frišak	gladan	gostovan	grdan
grešan	groban	gulozan	hitar	hladan
hrabar	jadan	jedar	južan	kadar
krasan	krotak	krupan	kužan	lužan
mekoobrazan	miran	modar	mokar	mrsan
mrtav	mrzan	mudar	mutan	nameran
namjeran	naprasan	nazadan	nedobar	nemiran
nemoćan	neobičan	nepravedan	nesložan	nesrećan
neveran	nevjeran	nevolxan	oblačan	okretan
oštar	pametan	pokoran	pomaman	ponizan
potajan	potreban	povolxan	pozan	prazan
predobar	prekoran	pretežan	prijatan	prijekoran
probitačan	ravan	različan	ružan	samovolxan
sitan	slavan	sličan	složan	smočan
smrtan	sraman	sramotan	srećan	star
stidan	strašan	suvišan	tajan	taman
tavan	trezan	trijezan	tužan	udesan
ugodan	umoran	vedar	večan	volxan
vredan	vrijedan	ziman	zlatan	zlosretan
štetan	šupalx	čudan	čudotvoran	

A08.03

žalostan	bolestan	jakostan	koristan	mastan
premastan	častan			

A08.04

gladak	ijedak

A08.05

blizak	mrzak

A08.06

gnxio	kiseo	mio	mukao	nemio
nevalxao	podmukao	svetao	svijetao	truhao
truo	veseo	vreo	zreo	

A08.07

prerastao	prirastao

A08.52

besan	bijesan

A09.01
lak lep lijep mek
A10.01
težak
A10.02
uzak
A10.03
kratak plitak redak rijedak
A10.04
sladak
A10.05
tanak
A12.01
širok
A12.02
žestok dubok visok
A13.01
beo ceo debeo go truo
A14.01
dobar
A14.02
mali
A14.03
malen
A14.04
velik zao
A14.06
rđav
NA14.01
mlada
NA14.01+
sirota

D.3.3 Glagoli iz rečnika DELA po transduktorima

V01.00.2

bacati	batrgati	begati	benetati	birati
bivati	bjegati	brblati	buncati	celivati
cepati	cijepati	cjelivati	crkavati	cvetati
cvjetati	davati	dirati	dobijati	dobivati
dogovarati	dospeti	drijemati	drpati	duhati
duvati	fatati	fatigati	fukati	gatati
gađati	gibati	gledati	gubati	gucati
hitati	hodati	hramati	hrkati	hvatati
igrati	ispijati	ispredati	isturati	izbivati
izmetati	izuvati	izvršavati	kapati	karati
kasati	kidati	kihati	kijati	klimati
klxucati	kopati	korakati	koštati	kucati
kukati	kusati	kušati	kvocati	lipsavati
litati	lupati	lutati	mahati	malaksati
mankati	manxkati	meritati	mešati	migati
miješati	miritati	mirucati	mrdati	mutarati
nadati	nadgledati	naklapati	napuštati	navraćati
obarati	obećavati	obijati	obraćati	odbijati
odgovarati	odzivati	opraštati	otimati	otkidati
otpadati	ozivati	padati	papati	parati

pasati	pevati	pipati	pitati	pišati
pjevati	plačati	plivati	plutati	podsecati
podsijecati	podsmevati	podsmijevati	pogađati	pokivati
pokrivati	ponuđati	postupati	povraćati	poštapati
počivati	praskati	prebirati	pretresati	pretrčavati
pribirati	prigovarati	primati	probati	probijati
probirati	prskati	pružati	prškati	pucati
puhati	puštati	rasipati	raskivati	rastovarati
rastresati	razbijati	razgađati	razgovarati	razlevati
razlijevati	razumeti	razumevati	razumijevati	razvijati
raďati	računati	ridati	rugati	ručati
sanxkati	savijati	sačuvati	secati	sedlati
sevati	sećati	siguravati	sijati	sijevati
sipati	sjati	sjecati	sjećati	skitati
slušati	smatrati	smetati	smeti	snivati
spasavati	spavati	sprdati	spremati	spuštati
starati	stiskati	sveštati	svirati	sviďati
svještati	terati	tiskati	titrati	tišati
tjerati	trebati	tumarati	tumbati	turati
uživati	ubijati	udarati	ugađati	ugnxivati
ugrizati	ujedati	umeti	umilxavati	umivati
upuštati	urlati	uvijedati	uzaimati	uzdati
uzimati	ušati	varati	vatati	večerati
vijati	vladati	vračati	vračati	vrcati
zabadati	zagledati	zaigravati	zapovedati	zapovijedati
zapredati	zatvarati	zavijati	zbijati	zgaďati
zidati	zijati	zjati	šarati	šetati
šikati	šišati	šklocati	šklxocati	dīpati
čekati	čitati	čupati	čuvati	

V01.25.4

dospjeti	razumjeti	umjeti		

V01.50.2

dodijati	dokopati	dolijati	doterati	dotjerati
dočekati	dočuvati	forati	isisati	iskamkati
isklimati	iskopati	isterati	izagnati	izbijati
izdati	izgledati	izlandati	izmešati	izmiješati
izmolxakati	iznxihati	iščupati	krepati	nabalati
nabusati	nadmontati	naduvati	naigrati	nakasati
nakusati	napadati	napipati	napuhati	naspavati
nastradati	naterati	natjerati	načuvati	obećati
obmotati	obrivati	obrugati	obrukati	odrpati
odspavati	ofukati	ogledati	oguglati	okupati
opuhati	osedlati	oterati	otjerati	ošogati
ošugati	pasati	pocrkati	podati	pogledati
poharati	pohvatati	pokarati	pokopati	politati
pomešati	pomiješati	popevati	popijevati	poplxeskati
poslušati	posvaďati	posvirati	poterati	potjerati
pouzdati	pošetati	počupati	predati	pridati
priterati	pritjerati	pričekati	prodati	propevati
propjevati	proterati	protjerati	provešati	provješati
rasklasati	rasparčati	rasterati	rastjerati	razgubati
sazdati	skapati	skuvati	smučkati	udati
ugnati	ukopati	ustumarati	uzvrdati	uščuvati
učuvati	zadati	zaigrati	zakopati	zapevati
zapitati	zapjevati	zapuhati	zarďati	zbabati

V02.00.2

znati

V02.50.2

obaznati poznati saznati
V03.00.2
morati
V04.00.2
imati
V05.00.2
nemati
V06.50.2
dati dodati nadati
V07.00.3
tkati
V07.50.3
natkati potkati
V08.00.2
pasti tresti
V08.50.2
ispasti napasti otresti spasti
V09.00.2
rasti
V09.50.2
narasti obrasti zarasti
V10.50.2
doneti izneti naneti odneti podneti
poneti proneti sneti uneti
V10.75.5
donijeti iznijeti nanijeti odnijeti podnijeti
ponijeti pronijeti snijeti unijeti
V11.00.3
gristi musti vesti vrsti
V11.50.3
dovesti izlxesti izvesti nagrepsti navesti
odgristi odvesti odvrsti ogrepsti ozepsti
povesti prevesti razvrsti ulesti ulxesti
uvesti uzepsti zagrepsti zavesti zavrsti
V12.00.4
grepsti
V13.00.3
krasti presti
V13.01.3
mesti plesti
V13.50.3
dovesti izjesti izvesti najesti navesti
odvesti pojesti povesti prevesti ujesti
ukrasti uvesti zavesti
V13.51.3
obresti omesti pomesti smesti zamesti
zaplesti
V14.50.3
podsesti podsjesti presesti presjesti prosesti
prosjesti sesti sjesti zasesti zasjesti
V15.00.4
gresti
V16.00.3
jesti
V17.50.3
sresti susresti
V18.50.3

popasti	propasti	raspasti	spasti	spopasti
upasti				
V19.00.3				
bosti				
V19.50.3				
obosti	probosti	ubosti	zabosti	zbosti
V20.00.3				
grejati	grijati	hajati	izostajati	kajati
lajati	nedostajati	nestajati	ostajati	pojati
prestajati	smejati	smijati	trajati	ustajati
V20.01.4				
davati	plxuvati	poznavati	prodavati	siguravati
udavati	zadavati			
V20.02.5				
carovati	darovati	gladovati	gospodovati	hladovati
izrizikovati	kmetovati	korotovati	kovati	koštovati
kumovati	kupovati	letovati	lxetovati	milovati
mirovati	mudrovati	naživovati	namudrovati	napredovati
napsovati	obidovati	obradovati	okovati	opsovati
osnovati	otrovati	pirovati	plandovati	popovati
popsovati	potkovati	poštovati	predikovati	psovati
putovati	radovati	ratovati	samotovati	samoćovati
savetovati	savjetovati	skovati	snovati	spakovati
sramotovati	svetkovati	svetovati	svjetovati	trebovati
trgovati	trovati	velxovati	verovati	vjerovati
zakovati	šenkovati	štetovati		
V20.03.5				
darivati	deverivati	djeverivati	dosađivati	isterivati
istraživati	izbacivati	izmenxivati	izvršivati	kazivati
lipsivati	naplaćivati	natrkivati	oblizivati	odrađivati
odrešivati	odrješivati	opakivati	opasivati	osvećivati
pokazivati	posvećivati	potprdivati	presađivati	pripitivati
prokaplxivati	raženxivati	razdrešivati	razgrađivati	zahvalxivati
zakusivati	zasluživati	zastranxivati	zasukivati	zavalxivati
V20.04.5				
vojevati				
V20.50.3				
izlajati	nadlajati	nagrejati	nagrijati	nasmejati
nasmijati	obrijati	ogrejati	ogrijati	očajati
podgrejati	podgrijati	posejati	posijati	sejati
sijati	ugrejati	ugrijati	zagrejati	zagrijati
zalajati				
V20.51.4				
izblxuvati	poplxuvati	rasplxuvati		
V21.00.3				
žderati	derati	izdirati	izvirati	nazirati
odupirati	orati	otirati	ponirati	posati
prebirati	pribirati	proždirati	probirati	revati
rvati	umirati	utirati	češati	
V21.01.3				
drijemati	gibati	hramati	kapati	odzivati
otimati	ozivati	poštapati	rasipati	sipati
uzaimati	uzimati	zobati		
V21.02.4				
ambisati	begenisati	bitisati	bojadisati	disati
eglenisati	jeglenisati	malaksati	metanisati	mirisati
pisati	uzdisati	ćerdisati		
V21.03.4				

duhati jahati kihati mahati puhati
V21.04.4
lagati legati lijegati nalagati odmagati
pomagati raspolagati slagati zalagati
V21.05.4
dizati kazati lizati mazati natezati
podizati potezati prestrizati rzati stezati
stizati sustizati vezati zatezati
V21.06.4
kvocati micati nicati omicati poricati
ticati
V21.07.4
žvakati hrkati hukati jaukati krakati
lokati maukati mukati plakati preskakati
rikati skakati srkati sukati umakati
urlikati vikati šikati
V21.08.4
benetati izmetati izvrtati kakotati kleptati
kretati metati nametati nasrtati obrtati
odletati odlijetati okretati osvitati podmetati
povrtati premetati pretati prevrtati prometati
razmetati skitati svitati treptati upletati
vrtati zametati zapletati šetati
V21.09.4
glodati zidati
V21.10.5
iskati
V21.11.5
iskati stiskati
V21.12.5
drhtati
V21.13.4
klati
V21.50.3
doderati izderati oderati poderati razderati
razorati zaderati
V21.51.3
nazobati skapati
V21.52.4
isisati istesati lipsati nagrajisati napisati
ograjisati omirisati opasati prekasati prokopsati
upisati zapisati
V21.53.4
iznxihati napuhati objahati odjahati opuhati
sjahati uzjahati zapuhati
V21.54.4
otkakotati pošetati slagati
V21.57.4
isplakati nalokati polokati povikati proskakati
zahukati zaplakati zasukati
V21.59.4
oglodati
V21.60.5
pobiskati poiskati zaiskati
V21.61.5
poplxeskati zaiskati
V21.63.4
otklati poklati zaklati

V22.00.4
brati prati srati
V22.01.4
zvati
V22.02.5
slati
V22.50.4
obrati oprati pobrati posrati pribrati
V22.51.4
dozvati nazvati odazvati
V22.52.5
poslati
V22.53.5
izagnati ugnati
V23.00.4
brinuti ginuti grnuti mrknuti tonuti
zrenuti
V23.01.4
kisnuti
V23.02.4
mrznuti
V23.50.3
ciknuti dahnuti darnuti denuti djenuti
dunuti gonenuti graknuti granuti izvrnuti
jeknuti kinuti kucnuti mahnuti metnuti
minuti nadenuti nadjenuti nagnuti nategnuti
obiknuti oblaznuti obliznuti obrnuti odenuti
odsrknuti odviknuti ohronuti okrenuti omrknuti
omrznuti oronuti ostanuti osvanuti otegnuti
otisnuti otkinuti otpočinuti otrgnuti ošinuti
oděnuti panuti planuti plxunuti podmetnuti
poginuti pokliznuti poprdnuti posahnuti posrnuti
potegnuti potisnuti potonuti potrgnuti prdnuti
prekinuti prekrenuti premetnuti pretegnuti prevrnuti
prhnuti prionuti pripnuti pritisnuti prnuti
prometnuti protegnuti raskinuti rasprdnuti razminuti
raščenuti reknuti sažegnuti seknuti sinuti
sjeknuti skinuti spanuti spomenuti spotaknuti
srinuti stisnuti švanuti usahnuti utonuti
uzdahnuti vaskrsnuti vinuti vrgnuti vrnuti
zabrinuti zadenuti zadjenuti zakinuti zamahnuti
zametnuti zapanuti zavrnuti zinuti zovnuti
zveknuti šinuti šušnuti
V24.00.2
biti kriti liti piti sniti
viti šiti čuti
V24.50.2
dobiti izbiti izmiti izriti nabiti
naduti napiti obuti odbiti opiti
otkriti otpiti pobiti podaviti pokriti
politi popiti prebiti preliti prešiti
pribiti priviti probiti proliti razbiti
razviti sakriti saviti skriti sviti
ubiti zaliti zaviti
V25.00.4
mleti
V25.25.5
mlxeti

V25.50.4
samleti
V25.75.5
samlxeti
V26.00.3
žnxeti dreti mreti
V26.25.5
drijeti mrijeti
V26.50.3
odadreti požnxeti podupreti pomreti popreti
proždreti prodreti prostreti razdreti umreti
upreti zatreti
V26.75.5
odadrijeti odrijeti proždrijeti prodrijeti umrijeti
uprijeti zatrijeti
V27.00.3
žeti
V27.50.3
oduzeti oteti uzeti
V27.51.3
osuti posuti prosuti zasuti
V27.52.3
naduti
V27.53.3
dočeti napeti načeti popeti početi
sapeti zapeti
V28.00.4
kleti
V28.50.4
prokleti ukleti zakleti
V29.50.2
nastati nestati ostati postati prestati
pristati rastati sastati stati ustati
V30.00.3
žeti
V30.50.3
požeti
V31.00.3
bežati bježati blejati bojati brojati
bučati držati dreždati drhtati ječati
klečati kreštati ležati mučati pištati
prštati režati tajati tištati trčati
večati vrištati zujati zvečati zviždati
zvrčati šištati šutati ćutati čučati
V31.50.3
natrčati načučati oćutati potrčati pridržati
protrčati údržati utrčati
V32.00.3
bojati brojati
V32.01.5
stajati
V32.50.3
prebrojati ubojati
V33.00.3
goreti hiteti strepeti trpeti
V33.01.3
žariti žuriti arčiti besposličiti bečiti
boriti curiti dičiti dojiti drešiti

driješiti	družiti	dvoriti	gojiti	gospodariti
govoriti	grešiti	griješiti	kaližiti	kobačiti
koriti	kormilariti	kočiti	krojiti	kuburiti
kvariti	kvačiti	lenčariti	lečiti	liječiti
ličiti	manxiti	mariti	meriti	mečiti
miriti	mjeriti	moriti	motriti	mrčiti
mučiti	oblačiti	ortačiti	papriti	pazariti
pečiti	pilxiti	piriti	plašiti	povlačiti
pržiti	prašiti	prlxiti	pušiti	ružiti
služiti	sušiti	tajiti	tlačiti	toriti
tovariti	točiti	tražiti	trošiti	tužiti
tvoriti	uzuriti	učiti	variti	vedriti
viriti	vlačiti	zboriti	širiti	štrojiti
đubriti	ćariti			

V33.25.4

gorjeti	hitjeti	strepjeti	trpjeti	

V33.50.3

dogoreti	doleteti	izgoreti	izleteti	naživeti
nagoreti	oživeti	ogladneti	ogoreti	omileti
poželeti	pobledeti	pocrneti	pocrveneti	pokipeti
poleteti	pomrzeti	povideti	povrveti	pretrpeti
prispeti	rasprdeti	sazreti	strpeti	uprdeti
ušuteti	zaželeti	zaboleti	zabudeti	zacrveneti
zaludeti	zaštedeti			

V33.51.3

baciti	blagosloviti	desiti	dobaviti	dogoditi
dohvatiti	dokoračiti	dokročiti	dokučiti	dosaditi
dovršiti	glasiti	hvatiti	isceniti	iscijeniti
iskamčiti	iskesiti	iskeziti	iskriviti	iskrpiti
iskrvaviti	isplatiti	ispostiti	ispraviti	isprositi
istražiti	izbečiti	izbulxiti	izgubiti	izguliti
izjaloviti	izjediniti	izlečiti	izliječiti	izmamiti
izmeriti	izmjeriti	izmučiti	izraditi	izrebriti
izvaditi	izvaliti	kakiti	kazniti	koračiti
kupiti	latiti	lišiti	mašiti	miti
nadimiti	nagaziti	nagoditi	nagojiti	nahraniti
nahvaliti	najmiti	nakaniti	nakititi	nakvasiti
nalepiti	nalijepiti	naložiti	nalxutiti	nameriti
namesiti	namijesiti	namiriti	namjeriti	namrgoditi
namrštiti	namučiti	naoblačiti	napraviti	napuniti
naružiti	nasaditi	naseliti	nasititi	naslutiti
nasporiti	natmuriti	natoprčiti	natražiti	natrćiti
nauditi	naumiti	naučiti	navaditi	navaliti
navratiti	nazdraviti	načiniti	nehoditi	oženiti
obeseliti	obesiti	objesiti	oboriti	obraniti
obratiti	obrlatiti	obršiti	obveseliti	odbraniti
odgovoriti	odmoritĺ	odrediti	odrešiti	odriješiti
odstupiti	odužiti	oducčiti	odvaliti	ogaditi
oglasiti	ogrešiti	ogriješiti	oguliti	ojariti
okajmačiti	okotiti	okoziti	okratiti	okumiti
okupiti	okusiti	omaciti	omeriti	omijeniti
omitariti	omjeriti	omoriti	omraziti	omrciniti
omrčiti	omučiti	opaziti	opepeliti	opržiti
oprasiti	opraviti	opremiti	oprositi	oprostiti
oprćiti	opslužiti	orazumiti	osiromašiti	osloboditi
osmočiti	osmuditi	osokoliti	osoliti	osramotiti
ostariti	ostaviti	osuditi	osušiti	osvetiti
oteliti	otoboliti	ototoliti	otužiti	otvoriti

oštiti	oćoraviti	očepiti	očistiti	platiti
požaliti	poarčiti	pobeliti	pobijeliti	pobratiti
pobudaliti	podeliti	podijeliti	pogladiti	pogoditi
pogrešiti	pogriješiti	pogubiti	poharčiti	pohuliti
pohvaliti	pokloniti	pokondiriti	pokoriti	pokositi
pokučiti	pokvariti	polxubiti	pomamiti	pomiriti
pomisliti	pomoliti	pomoriti	poništiti	ponuditi
popaliti	poplašiti	popraviti	poraditi	porazgovoriti
poraziti	porediti	posaditi	posiliti	poslužiti
postiditi	posuditi	posvedočiti	posvetiti	posviriti
posvjedočiti	potrošiti	poturčiti	potvoriti	povratiti
pozdraviti	pozlediti	pozlijediti	pregaziti	prekoračiti
prekoriti	prekrstiti	preprtiti	preskočiti	presušiti
pretvoriti	prevaliti	prevariti	prevratiti	prigrliti
prihititi	prilepiti	prilijepiti	primeniti	primijeniti
primiti	pripovediti	pripovijediti	pripraviti	pristaviti
privaliti	progovoriti	prolomiti	promeniti	promijeniti
proputiti	prostiti	proučiti	provaliti	prozliti
prtiti	pružiti	pustiti	raženiti	raniti
rascveliti	rascvijeliti	raseliti	raskrečiti	raspoloviti
rasrditi	rastvoriti	rasušiti	razbluditi	razdvojiti
razgraditi	razmisliti	razmrsiti	razoriti	razvaliti
raširiti	raščepiti	sadružiti	sagrešiti	sagriješiti
sastaviti	satužiti	setiti	sjetiti	skalaburiti
skoliti	skočiti	skrojiti	skrstiti	skupiti
skvasiti	složiti	slomiti	smiriti	smisliti
spratiti	staniti	staviti	stvoriti	svaditi
svršiti	turiti	ucveliti	ucvijeliti	udariti
udaviti	udiviti	udrobiti	ugasiti	ugoditi
ugrabiti	uhvatiti	ujediniti	ukloniti	ukočiti
ukrasiti	ukratiti	ukuburiti	uloviti	umaliti
umesiti	umijesiti	umoliti	umoriti	upaliti
uplašiti	upraviti	uprtiti	uputiti	uraniti
urediti	usmrtiti	usprdežiti	utanxiti	utočiti
uvaliti	uvrediti	uvrijediti	uzajmiti	uzdušiti
učiniti	vratiti	zabaviti	zabeliti	zabijeliti
zaboraviti	zabrazditi	zabuniti	zacrlxeniti	zacrniti
zadaviti	zadesiti	zadužiti	zaglaviti	zahvaliti
zakameniti	zaklapiti	zakloniti	zakrlxeštiti	zakrpiti
zakrvaviti	založiti	zalxubiti	zamisliti	zamlatiti
zamrsiti	zapaliti	zapatiti	zapaziti	zapečatiti
zapopiti	zapotiti	zapovediti	zapovijediti	zaprašiti
zarediti	zariti	zatrubiti	zatvoriti	zavaditi
zavaliti	zaviriti	zazvoniti	začuditi	združiti
zgaditi	zgoditi	zgrčiti	ďipiti	ćušiti
V33.75.4				
doletjeti	izletjeti	naživjeti	nagorjeti	oživjeti
poblijedjeti	pokipjeti	poletjeti	pomrzjeti	povidjeti
povrvjeti	pretrpjeti	prispjeti	rasprdjeti	strpjeti
uprdjeti	ušutjeti	zaštedjeti		
V34.00.3				
živeti	grmeti	srbeti	svrbeti	
V34.01.4				
daviti	drobiti	esapiti	globiti	gnxaviti
grabiti	gubiti	hesapiti	klapiti	krpiti
kumiti	kupiti	lomiti	loviti	lxubiti
mamiti	praviti	rubiti	slaviti	slepiti
slijepiti	sramiti	trubiti	vrviti	zajmiti

škopiti	škripiti			
V34.02.4				
želeti	boleti	voleti		
V34.03.4				
žaliti	deliti	dijeliti	faliti	guliti
hvaliti	moliti	paliti	seliti	soliti
tegliti	teliti	tuliti	veseliti	šaliti
V34.04.4				
žedneti	crveneti			
V34.05.4				
ženiti	braniti	ceniti	cijeniti	dogoniti
goniti	hraniti	kameniti	kloniti	kormaniti
nagoniti	pleniti	plijeniti	progoniti	puniti
raniti	roniti	rumeniti	sapuniti	sniti
trabuniti	zvoniti	činiti		
V34.06.4				
mrzeti				
V34.07.4				
dolaziti	izlaziti	nakaziti	nalaziti	obilaziti
oblaziti	odlaziti	paziti	prolaziti	slaziti
ulaziti	voziti	zalaziti		
V34.08.4				
leteti	lećeti	vrteti		
V34.09.4				
inatiti	kotiti	lxutiti	mlatiti	mutiti
pamtiti	patiti	pratiti	pretiti	prihvatiti
prijetiti	pustiti	slutiti	sramotiti	svetiti
tratiti				
V34.10.4				
bledeti	daždeti	gudeti	lebdeti	prdeti
sedeti	smrdeti	stideti	stideti	videti
vredeti	zavideti	štedeti		
V34.11.4				
besediti	besjediti	dohoditi	gladiti	glediti
goditi	graditi	grditi	gvarditi	hladiti
hoditi	ishoditi	izvoditi	kuditi	nahoditi
nuditi	raditi	roditi	saditi	sediti
sijediti	sladiti	srditi	suditi	truditi
tvrditi	uhoditi	uvoditi	vaditi	voditi
škoditi	čuditi			
V34.12.4				
donositi	gasiti	kositi	mesiti	musiti
nositi	odnositi	ponositi	prositi	visiti
V34.13.5				
besneti				
V34.14.5				
gostiti	pakostiti	postiti	častiti	
V34.15.5				
prazniti				
V34.16.5				
misliti				
V34.25.4				
živjeti	grmjeti	srbeti	svrbjeti	
V34.27.4				
želxeti	živlxeti	bolxeti	volxeti	
V34.29.5				
žednxeti	crvenxeti			
V34.31.5				

mrzjeti
V34.33.5
letjeti vrtjeti
V34.35.5
blijedjeti daždjeti gudjeti lebdjeti prdjeti
sjedjeti smrdjeti stidjeti stidjeti vidjeti
vrijedjeti zaludjeti zavidjeti štedjeti
V34.38.6
bjesnxeti
V34.77.4
omilxeti poželxeti zaželxeti zabolxeti
V34.79.5
ogladnxeti pocrnxeti pocrvenxeti zacrvenxeti
V35.00.5
krstiti
V36.00.3
vreti
V36.50.3
dodreti
V37.25.4
viđeti
V50.00.2
vrći
V50.01.2
peći seći teći tući vući
V50.02.2
žeći leći strići
V50.26.4
sjeći
V50.51.2
dovući ispeći izvući naseći natući
navući obući odseći opeći oseći
podvući poseći potući povući preseći
preteći prevući provući svući upeći
zapeći zaseći zavući
V50.52.2
izleći ostrići prožeći užeći
V50.76.4
nasjeći odsjeći osjeći posjeći presjeći
zasjeći
V51.50.2
isteći nateći obreći odreći preteći
proteći reći steći ureći uteći
zareći
V52.00.2
moći
V53.50.2
pomoći
V54.00.2
ići
V55.50.4
otići
V56.50.2
doći izaći izići naići naći
obići odići poći preći prijeći
proći saći sići snaći ući
zaći
V57.50.5

```
     dignuti      legnuti     ožegnuti    pobegnuti   pobjegnuti
   podignuti     stignuti
V57.51.5
     crknuti    izniknuti      maknuti    naviknuti      niknuti
   odmaknuti     omaknuti     pomaknuti   premaknuti   primaknuti
     puknuti     smaknuti     smrknuti      taknuti    zataknuti
V99.00
       jesam
V99.01.3
  pristojati
V99.02
        biti
V99.03
        biti
V99.04
       hteti
V99.05
      htjeti
V99.06
       kćeti
V99.10
     pospati        spati       zaspati
V99.11
       velim
```

Dodatak E

Primeri čitanja elektronskog teksta

E.1 Elektronski tekst podvučen elektronskim rečnikom

Primer prikazuje povezivanje teksta Vukovih narodnih poslovica sa elektronskim rečnikom. Sa rečnikom je povezan samo tekst poslovica (označen SGML etiketama <pv> i </pv>) a ne i objašnjenje koje ga, eventualno, prati. U primeru je dat početak poslovica koje počinju slovom G.

```
<divletter letter=G type='odeljak'>
<head>G.</head>
<divp id=P658 n=658>
<pv>   <w a=.N07.01+*:msn+>Gad</w>
   <w a=ovaj.ProA01:msg*:nsg*:msa+>ovoga</w>
   <w a=svijet.N08.01-J:msg->svijeta</w>
</pv>
<pexp>
    Reče se za ružno čeljade.
</pexp>
</divp>
<divp id=P659 n=659>
<pv>   <w a=.N70.01-*:fsn-:fpg->Galija</w>
   <w a=jedan.Num01*:msg*:msa+:nsg*>jednoga</w>
   <w a=.Par*>ne</w>
   <w a=čekati.V01.00.2*:P3s:A2s:A3s>čeka</w>
</pv>
</divp>
<divp id=P660 n=660>
<pv>   <w a=.N05.01-*:msn-:msa->Garbin</w>
   <w a=lxutiti.V34.09.4*:P3s:Y2s:A2s:A3s,lxut.A07.01*:p@msn*:p@msa-:p*msv*:p*mpn*:p*mpv*>ljuti</w>
   <w a=.ProA07:msn*:msa-:mpn*>koji</w>
   <w a=mora.N70.01-*:fsg-:fpn-:fpa-:fpv-,.N60.01-*:nsn-:nsa-:nsv-,mor.N13.01-*:mpa-,moriti.V33.01.3*:P3p
   <w a=.Pre*,.Adv*,.N15.08-*:msn-:msa->do</w>
   <w a=dno.N51.01-*:nsg-:npn-:npg-:npa-:npv->dna</w>
   <w a=mutiti.V34.09.4*:P3s:Y2s:A2s:A3s>muti</w>
</pv>
<pexp>
    U <placeName type='mesto'>Dubrovniku</placeName>
    <term rend='italic'>Garbin</term> se zove
    nekakav vjetar, čini mi se zapadni.
</pexp>
```

```
</divp>
<divp id=P661 n=661>
<pv>   <w a=gatati.V01.00.2*:PPsf:PPpn>Gatala</w>
   <w a=.N70.01+*:fsn+:fpg+,babo.N40.01+*:msg+:msa+:mpg+>baba</w>
   <w a=.Con*,.Par*,dati.V06.50.2*:P3s:A2s:A3s>da</w>
   <w a=jesam.V99.00*:P3s>nije</w>
   <w a=mraz.N23.01-*:msg->mraza</w>
   <w a=.Con*,.Par*>pa</w>
   <w a=osvanuti.V23.50.3*:PPms>osvanuo</w>
   <w a=.N04.02-J:msn-:msa-,.N08.02-J:msn-:msa->snijeg</w>
   <w a=.Pre*,.Adv*,.N15.08-*:msn-:msa->do</w>
<obs.ph value='guzice' resp='CV'>g...e
</obs.ph></pv>
</divp>
<divp id=P662 n=662>
<pv>   <w a=.N60.01-*:nsn-:nsa-:nsv->Gvožđe</w>
   <w a=valxati.V01.00.2*:VP3s:VA2s:VA3s>valja</w>
   <w a=.V20.02.5*:W>kovati</w>
   <w a=.Con*>dok</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=vruć,A06.01*:p*mpa*:p*fsg*:p*fpn*:p*fpa*:p*fpv*:p*nsn*:p*nsa*:p*nsv*>vruće</w>
</pv>
<pexp>
    Ili: <s.a target=P663 resp='Vuk'></s.a>
</pexp>
</divp>
<divp id=P663 n=663>
<pv>   <w a=.N60.01-*:nsn-:nsa-:nsv->Gvožđe</w>
   <w a=sebe.ProN11:**g*:**a*>se</w>
   <w a=kuja.N70.01+*:fsg+:fpn+:fpa+:fpv+,kovati.V20.02.5*:P3s>kuje</w>
   <w a=.Con*>dok</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=vruć.A06.01*:p*mpa*:p*fsg*:p*fpn*:p*fpa*:p*fpv*:p*nsn*:p*nsa*:p*nsv*>vruće</w>
</pv>
<ptr target=P662 resp='CV'>
</divp>
<divp id=P664 n=664>
<pv>   <w a=.AdvE,.ConE>Gde</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=.N08.01-E:msn-:msa->cvet</w>
   <w a=.Adv*,taj.ProA01:fsa*>tu</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=.N08.01-*:msn-:msa->med</w>
</pv>
<pexp>
    Čele bez cvijeta ne mogu meda skupiti.
</pexp>
</divp>
<divp id=P665 n=665>
<pv>   <w a=gizdav.A08.01*:p*fsd*:p*fsl*>Gizdavoj</w>
   <w a=neva.N70.01+*:fsd+:fsl+>nevi</w>
   <w a=.Par*>ne</w>
   <w a=?>moš</w>
   <w a=.N22.)1-*:msn-:msa->ovrlj</w>
   <w a=.Pre*,oka.N70.01-*:fsv-,.N54.01-*:nsn-:nsa-:nsv->oko</w>
   <w a=glava.N70.01-*:fsg-:fpn-:fpa-:fpv->glave</w>
   <w a=.V01.50.2*:W>obmotati</w>
</pv>
```

```
<pexp>
   Kad je kome teško ugoditi.
</pexp>
</divp>
<divp id=P666 n=666>
<pv>   <w a=.N72.01-*:fsn->Gizda</w>
   <w a=lomiti.V34.01.4*:P3s:Y2s:A2s:A3s>lomi</w>
   <w a=.Con*,.Adv*,.Int*>a</w>
   <w a=.N82.02-*:fsn-:fsa->glad</w>
   <w a=mora.N70.01-*:fsd-:fsl-,mor.N13.01-*:mpn-:mpv-,moriti.V33.01.3*:P3s:Y2s:A2s:A3s>mori</w>
</pv>
<pexp>
    Gledaj: <s.a target=P2963 resp='Vuk'>Malo jede, al' se lijepo
    nosi.</s.a>
</pexp>
</divp>
<divp id=P667 n=667>
<pv>   <w a=.N70.01-*:fsn-:fpg->Glava</w>
   <w a=.Con*,.Adv*,.Int*>a</w>
<n.i.prov type='pokret'>   <w a=?>podignuvši</w>
   <w a=ruka.N73.01-*:fsg-:fpn-:fpa-:fpv->ruke</w>
   <w a=.Con*,.Par*>i</w>
   <w a=raširiti.V33.51.3*:AdvPs>rašrivši</w>
   <w a=šaka.N70.04-*:fsg-:fpn-:fpa-:fpv->šake</w>
   <w a=.Pre*>prema</w>
   <w a=glava.N70.01-*:fsd-:fsl->glavi</w>
</n.i.prov>   <w a=.Con*,.Adv*,.Int*>a</w>
   <w a=.Pre*,.Int*>u</w>
   <w a=glava.N70.01-*:fsd-:fsl->glavi</w>
   <w a=.Pre*,.Int*>na</w>
</pv>
<pexp>
    Kao uhvativši se palcem i kažiprstom za zube, i znači: glava velika,
    ali prazna.
</pexp>
</divp>
<divp id=P668 n=668>
<pv>   <w a=.N70.01-*:fsn-:fpg->Glava</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=skup.A07.01*:k0fsn*:k0fsv*:k0npn*:k0npa*:k0npv*,skupljati.V01.00.2*:VP3s:VA2s:VA3s>skuplja</w>
   <w a=.Pre*>s</w>
   <w a=jezik.N04.04-*:msi->jezikom</w>
   <w a=.Con*>nego</w>
   <w a=.Pre*>bez</w>
   <w a=jezik.N04.04-*:msg-:mpg->jezika</w>
</pv>
<pexp>
    Vredniji je čoek koji umije govoriti nego onaj koji ne umije.
</pexp>
</divp>
<divp id=P669 n=669>
<pv>   <w a=.N70.01-*:fsn-:fpg->Glava</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=star.A08.02*:k0fsn*:k0fsv*:k0npn*:k0npa*:k0npv*>starija</w>
   <w a=.Pre*>od</w>
   <w a=knxiga.N70.04-*:fsg-:fpn-:fpa-:fpv->knjige</w>
</pv>
</divp>
```

```
<divp id=P670 n=670>
<pv>   <w a=.N70.01-*:fsn-:fpg->Glava</w>
   <w a=.Par*,liti.V24.00.2*:A2s:A3s>li</w>
   <w a=ono.ProN06:nsg*:nsa*,on.ProN05:msg*:msa*>ga</w>
   <w a=bosti.V19.00.3*:PPpm,bolxeti.V34.27.4J:P3s:Y2s,boleti.V34.02.4E:P3s:Y2s>boli</w>
</pv>
<pexp>
    Kad se kazuje da ko nema uzroka s čime biti nezadovoljan.
</pexp>
</divp>
<divp id=P671 n=671>
<pv>   <w a=.N70.01-*:fsn-:fpg->Glava</w>
   <w a=.Pre*>pred</w>
   <w a=glava.N70.01-*:fsa->glavu</w>
</pv>
<pexp>
    Gledaj: <s.a target=P504 resp='Vuk'>Varka pred <persName>Marka</persName>
    </s.a>
</pexp>
</divp>
<divp id=P672 n=672>
<pv><lg.prov><l.prov>   <w a=.N71.01-*:fsn-:fpg->Glavica</w>
   <w a=ono.ProN06:nsd*,on.ProN05:msd*>mu</w>
   <w a=?>cv'jeća</w>
   <w a=iskati.V21.10.5*:P3s>ište</w>
</l.prov><l.prov>   <w a=.Con*,.Adv*,.Int*>A</w>
<obs.ph value='guzica' resp='CV'>g...a
</obs.ph>   <w a=gaće.N70.76-*:fpg->gaća</w>
   <w a=nemati.V05.00.2*:P3s:A2s:A3s,nem.A06.01E:p#msg*:p#msa+:p#nsg*:p*fsn*:p*fsv*:p*npn*:p*npa*:p*npv*>
</l.prov></lg.prov></pv>
</divp>
<divp id=P673 n=673>
<pv>   <w a=.A08.04*:p#msn*:p#msa->Gladak</w>
   <w a=.Con*,.Adv*>kao</w>
   <w a=rast.N08.01-*:mpg-,rastov.A02.01*:p#msg*:p#msa+:p#fsn*:p#fsv*:p#nsg*:p#npn*:p#npa*:p#npv*>rastova
   <w a=.N70.01-*:fsn-:fpg->kora</w>
</pv>
</divp>
<divp id=P674 n=674>
<pv>   <w a=.A08.02*:p#msn*:p#msa->Gladan</w>
   <w a=.Con*,.Par*>i</w>
   <w a=.N12.01+*:msn+>patrijar</w>
   <w a=hlxeb.N08.01-J:msg->hljeba</w>
   <w a=hteti.V99.04E:P3s:P3p,htjeti.V99.05J:P3s:P3p>će</w>
   <w a=.V13.50.3*:W>ukrasti</w>
</pv>
</divp>
<divp id=P675 n=675>
<pv>   <w a=.A08.02*:p#msn*:p#msa->Gladan</w>
   <w a=.Con*,.Adv*>kao</w>
   <w a=.N01.04+*:msn+>kurjak</w>
</pv>
</divp>
<divp id=P676 n=676>
<pv>   <w a=.A08.02*:p#msn*:p#msa->Gladan</w>
   <w a=.Con*,.Adv*>kao</w>
   <w a=.N51.01+*:nsn+:nsa+:nsv+>pseto</w>
</pv>
```

```
</divp>
<divp id=P677 n=677>
<pv>   <w a=.A08.02*:p#msn*:p#msa->Gladan</w>
   <w a=.N01.04+*:msn+>kurjak</w>
   <w a=.Pre*>usred</w>
   <w a=selo.N51.01-*:nsg-:npn-:npg-:npa-:npv-,sesti.V14.50.3*:PPsf:PPpn>sela</w>
   <w a=ići.V54.00.2*:P3s:A2s:A3s>ide</w>
</pv>
</divp>
<divp id=P678 n=678>
<pv>   <w a=.A08.02*:p#msn*:p#msa->Gladan</w>
   <w a=.N08.01-*:msn-:msa-,.N12.51+*:msn+>pas</w>
   <w a=.Pre*,.Int*>o</w>
   <w a=?>komadu</w>
   <w a=sanxati.V01.00.2*:VP3s:VA2s:VA3s>sanja</w>
</pv>
</divp>
<divp id=P679 n=679>
<pv>   <w a=.N82.02-*:fsn-:fsa->Glad</w>
   <w a=.Con*,.Par*>i</w>
   <w a=kurjak.N01.04+*:msg+:msa+:mpg+>kurjaka</w>
   <w a=.Pre*>iz</w>
   <w a=šuma.N70.01-*:fsg-:fpn-:fpa-:fpv->šume</w>
   <w a=?>išćera</w>
</pv>
</divp>
<divp id=P680 n=680>
<pv>   <w a=gladiti.V34.11.4*:PPsf:PPpn>Gladila</w>
   <w a=sebe.ProN11:**g*:**a*>se</w>
   <w a=.N71.1.01+*:fsn+:fsv+:fpg+>popadija</w>
   <w a=.Pre*>porad</w>
   <w a=đak.N01.04+*:msg+:msa+:mpg+>đaka</w>
</pv>
<pexp>
    Nije to za tebe pripravljeno.
</pexp>
</divp>
<divp id=P681 n=681>
<pv>   <w a=gladan.A08.02*:p@msn*:p@msa-:p*msv*:p*mpn*:p*mpv*>Gladni</w>
   <w a=?>hrti</w>
   <w a=.Adv*,bolxeti.V34.27.4J:A2s:A3s,dobar.A14.01*:k@mpa*:k@fsg*:k@fpn*:k@fpa*:k@fpv*:k@nsn*:k@nsa*:k@
   <w a=lov.N04.01-*:msv-:mpa-,loviti.V34.01.4*:P3p>love</w>
</pv>
<pexp>
    Kad se hoće da kaže da je siromah čoek bolji za kakav posao nego bogat.
</pexp>
</divp>
<divp id=P682 n=682>
<pv>   <w a=gladan.A08.02*:p*nsn*:p*nsa*:p*nsv*>Gladno</w>
   <w a=?>gospostvo</w>
<obs.ph value='govnu' resp='CV'>g...u
</obs.ph>   <w a=.N02.01+*:msn+>brat</w>
</pv>
<pexp>
    U <placeName type='mesto'>Dubrovniku</placeName>
</pexp>
</divp>
<divp id=P683 n=683>
```

```
<pv>   <w a=gladan.A08.02*:p*nsn*:p*nsa*:p*nsv*>Gladno</w>
   <w a=.Pre*,oka.N70.01-*:fsv-,.N54.01-*:nsn-:nsa-:nsv->oko</w>
   <w a=.Par*>ne</w>
   <w a=spavati.V01.00.2*:P3s:A2s:A3s>spava</w>
</pv>
</divp>
<divp id=P684 n=684>
<pv>   <w a=gladan.A08.02*:p#msd*:p#msl*:p#nsd*:p#nsl*:p*fsa*>Gladnu</w>
   <w a=svat.N07.01+*:msd+:msl+>svatu</w>
   <w a=.Con*,.Par*>i</w>
   <w a=divlxak.N01.04+*:mpa+>divljake</w>
   <w a=.Pre*,.Int*>u</w>
   <w a=.N82.04-*:fsn-:fsa->slast</w>
   <w a=ići.V54.00.2*:P3p>idu</w>
</pv>
</divp>
<divp id=P685 n=685>
<pv>   <w a=gladan.A08.02*:p#msd*:p#msl*:p#nsd*:p#nsl*:p*fsa*>Gladnu</w>
   <w a=.N84.01+*:fsn+:fsa+>uš</w>
   <w a=.Adv*>mučno</w>
   <w a=ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=.V33.51.3*:VW>nasititi</w>
</pv>
</divp>
<divp id=P686 n=686>
<pv>   <w a=gladan.A08.02*:p#msd*:p#msl*:p#nsd*:p#nsl*:p*fsa*>Gladnu</w>
   <w a=?>čoeku</w>
   <w a=sladak.A10.04*:p*mpa*:p*fsg*:p*fpn*:p*fpa*:p*fpv*>slatke</w>
   <w a=jesam.V99.00*:P3p>su</w>
   <w a=.Con*,.Par*>i</w>
   <w a=divlxak.N01.04+*:mpa+>divljake</w>
</pv>
</divp>
<divp id=P687 n=687>
<pv>   <w a=?>Gladom</w>
   <w a=patiti.V34.09.4*:P3p>pate</w>
   <w a=.Con*,.Adv*,.Int*>a</w>
   <w a=čist.A08.01*:p*nsn*:p*nsa*:p*nsv*>čisto</w>
   <w a=nos.N08.01-*:msv-,nositi.V34.12.4*:VP3p>nose</w>
</pv>
</divp>
<divp id=P688 n=688>
<pv>   <w a=.N82.02-*:fsn-:fsa->Glad</w>
   <w a=oči.N86.01-*:fpg->očiju</w>
   <w a=nemati.V05.00.2*:P3s:A2s:A3s,nem.A06.01E:p#msg*:p#msa+:p#nsg*:p*fsn*:p*fsv*:p*npn*:p*npa*:p*npv*>
</pv>
<pexp>
    Glad može čoeka načerati na svako zlo.
</pexp>
</divp>
```

E.2 Prepoznavanje varijanti u elektronskom tekstu

```
<divletter letter=V type='odeljak'>
<head>V.</head>
<divp id=P490 n=490>
<pv>   <w a=,.Adv*>Vazda</w>
   <w a=,.Con*,.Par*>i</w>
<persName reg='sveta Marija'>   <w a=,.N70.01+*:fsn+:fpg+>Marija</w>
</persName></pv>
<pexp>
    Odgovore kršćani oko
    <placeName type='mesto' reg='Splita'>Spljeta</placeName> kad im se nazove:
    <q>Faljen Isus!</q>
</pexp>
</divp>
<divp id=P491 n=491>
<pv>   <w a=?>Va</w>
<corr sic='ustinu' resp='Nolit'>   <w a=,istina.N70.01-*:fsa->istinu</w>
</corr>   <w a=?>vaskrs</w>
</pv>
<pexp>
    Gledaj: <s.a target=P6045 resp='Vuk'>Hristos vaskrs!</s.a>
</pexp>
</divp>
<divp id=P492 n=492>
<pv>   <w a=?>Va</w>
   <w a=,istina.N70.01-*:fsa->istinu</w>
   <w a=,roditi.V34.11.4*:P3s:Y2s:A2s:A3s>rodi</w>
</pv>
<pexp>
    Gledaj: <s.a target=P6046 resp='Vuk'>Hristos se rodi!</s.a>
</pexp>
</divp>
<divp id=P493 n=493>
<pv>   <w a=,.Par*>Valaj</w>
   <w a=,sebe.ProN11:**d*,jesam.V99.00*:P2s>si</w>
   <w a=,.Con*,.Par*>i</w>
   <w a=?>čoek</w>
   <w a=,.Con*,.Par*>i</w>
   <w a=,.N70.01+*:fsn+:fpg+>žena</w>
</pv>
<pexp>
    U <placeName type='mesto'>Risnu</placeName> reče se valjanoj ženi.
</pexp>
</divp>
<divp id=P494 n=494>
<pv>   <w a=,.Par*>Vala</w>
   <w a=,mi.ProN03:*pd*>nam</w>
   <w a=,ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=,.Adv*,sav.ProA04.01:nsn*:nsa*:fsg*:mpa*:fpn*:fpa*>sve</w>
   <w a=,jedan.Num01*:nsn*:nsa*:nsv*>jedno</w>
   <w a=,nemati.V05.00.2*:P1p>nemamo</w>
   <w a=,nikakav.ProA03.01:msg*:nsg*:fsn*:npn*:npa*>nikakva</w>
   <w a=,posao.N15.58-*:msg-,poslati.V22.52.5*:A2s:A3s>posla</w>
   <w a=,.Con*,.Par*>ni</w>
   <w a=,.Pre*>kod</w>
```

```
</pv>
<pexp>
    Kazali psi bolesnom konju, kad su oni čekali više njega dok lipše,
    a on im govorio da idu kući, jer on ne će lipsati.
</pexp>
</divp>
<divp id=P495 n=495>
<pv>   <w a=,valxati.V01.00.2*:VP3s:VA2s:VA3s>Valja</w>
   <w a=,.Con*,.Par*,dati.V06.50.2*:P3s:A2s:A3s>da</w>
   <w a=,ona,ProN07:fsg*:fsa*,jesam,V99.00*:P3s>je</w>
   <w a=,nestati.V29.50.2*:PPsn>nestalo</w>
   <w a=,voda.N70.01-*:fsg-:fpn-:fpa-:fpv-,voditi.V34.11.4*:P3p>vode</w>
   <w a=,.Con*>ili</w>
   <w a=,drvo.N51.01-*:nsg-:npn-:npg-:npa-:npv->drva</w>
</pv>
<pexp>
    Kazao magarac, kad su ga pozvali na svadbu.
</pexp>
</divp>
<divp id=P496 n=496>
<pv>   <w a=,valxati.V01.00.2*:VP3s:VA2s:VA3s>Valja</w>
   <w a=,.Con*,.Par*,dati.V06.50.2*:P3s:A2s:A3s>da</w>
   <w a=,hteti.V99.04E%#E3.08:P3s:P3p,htjeti.V99.05J%hteti#E3.08:P3s:P3p>će</w>
   <w a=?>jedanput</w>
   <w a=,.V99.02E:W,.V99.03J:W,.V24.00.2*:W>biti</w>
   <w a=,.Con*,.Par*>i</w>
   <w a=,.Pre*,.Int*>u</w>
   <w a=,pakao.N15.58-*:msd-:msv-:msl->paklu</w>
   <w a=,.N04.01-*:msn-:msa->vašar</w>
</pv>
<pexp>
    Kad se nada da će se ćto obrnuti na bolje.
    U <placeName type='region'>vojvodstvu</placeName>.
</pexp>
</divp>
<divp id=P497 n=497>
<pv>   <w a=,valxati.V01.00.2*:VP3s:VA2s:VA3s>Valja</w>
   <w a=,.V27.53.3*:W>zapeti</w>
   <w a=,.Con*,.Par*>pa</w>
   <w a=,.V01.50.2J%zapevati#E4.12:W>zapjevati</w>
</pv>
<pexp>
    Bez truda ne može se ništa učiniti.
</pexp>
</divp>
<divp id=P498 n=498>
<pv>   <w a=,valxati.V01.00.2*:VPPsn>Valjalo</w>
   <w a=,biti.V99.03J:A2s:A3s,biti.V99.02E:A2s:A3s,biti.V24.00.2*:A2s:A3s>bi</w>
   <w a=,ono.ProN06:nsg*:nsa*,on.ProN05:msg*:msa*>ga</w>
   <w a=,.V99.02E:W,.V99.03J:W,.V24.00.2*:W>biti</w>
   <w a=,.Con*,.Par*>pa</w>
   <w a=,ono.ProN06:nsd*,on.ProN05:msd*>mu</w>
   <w a=,.Par*>ne</w>
   <w a=,.V06.50.2*:W>dati</w>
   <w a=,.V21.07.4*:W>plakati</w>
</pv>
</divp>
<divp id=P499 n=499>
```

```
<pv>   <w a=,valxati.V01.00.2*:VPPsn>Valjalo</w>
   <w a=,biti.V99.03J:A2s:A3s,biti.V99.02E:A2s:A3s,biti.V24.00.2*:A2s:A3s>bi</w>
   <w a=,ono.ProN06:nsg*:nsa*,on.ProN05:msg*:msa*>ga</w>
   <w a=,.Pre*,.Adv*>po</w>
   <w a=,trbuh.N13.07-*:msd-:msv-:msl->trbuhu</w>
   <w a=,.V99.02E:W,.V99.03J:W,.V24.00.2*:W>biti</w>
<opt.ph rend='() bold'>   <w a=,.Con*,.Adv*,.Int*>a</w>
   <w a=,.Par*>ne</w>
   <w a=,.Adv*,.Con*>kud</w>
   <w a=,sebe.ProN11:**g*:**a*>se</w>
   <w a=,.N03.11+*:mpn+:mpg+:mpv+,.N26.1.01+*:mpn+:mpv+>ljudi</w>
   <w a=,biti.V24.00.2*:P3p>biju</w>
</opt.ph></pv>
</divp>
<divp id=P500 n=500>
<pv>   <w a=,valxati.V01.00.2*:VPPsn>Valjalo</w>
   <w a=,biti.V99.03J:A2s:A3s,biti.V99.02E:A2s:A3s,biti.V24.00.2*:A2s:A3s>bi</w>
   <w a=,ono.ProN06:nsd*,on.ProN05:msd*>mu</w>
   <w a=,.V01.00.2*:W>čitati</w>
   <w a=,velik.A14.04*:p#msd*:p#msl*:p#nsd*:p#nsl*:p*fsa*>veliku</w>
   <w a=,molitva.N72.01-*:fsa->molitvu</w>
</pv>
<pexp>
    Kad koji što ludo govori, kao da nije pri sebi.
</pexp>
</divp>
<divp id=P501 n=501>
<pv>   <w a=,varati.V01.00.2*:P3s:A2s:A3s>Vara</w>
   <w a=,ono.ProN06:nsg*:nsa*,on.ProN05:msg*:msa*>ga</w>
   <w a=,.Con*,.Adv*>kao</w>
   <w a=,.N65.01+J%dete#E2.03.09:nsn+:nsa+:nsv+>dijete</w>
   <w a=,šaren.A08.01*:p*msi*:p*nsi*:p**pd*:p**pi*:p**pl*>šarenim</w>
   <w a=,jaje.N65.01+*:nsi+>jajetom</w>
</pv>
</divp>
<divp id=P502 n=502>
<pv><lg.prov><l.prov><persName>   <w a=,varvar.N01.01+*:msg+:msa+:mpg+>Varvara</w>
</persName>   <w a=,variti.V33.01.3*:P3s:Y2s:A2s:A3s>vari</w>
</l.prov><l.prov><persName reg='Sava'>   <w a=?>Sav-Sava</w>
</persName>   <w a=,hladiti.V34.11.4*:P3s:Y2s:A2s:A3s>hladi</w>
</l.prov><l.prov><persName>   <w a=,.N77.01+*:msn+:msv+:mpg+>Nikola</w>
</persName>   <w a=,kus.N08.01-*:msg-,kusati.V01.00.2*:P3s:A2s:A3s,kus.A06.01*:p#msg*:p#msa+:p#nsg*:p*fsn
</l.prov></lg.prov></pv>
<pexp>
    Ili: <s.a target=P503 resp='Vuk'></s.a>
</pexp>
</divp>
<divp id=P503 n=503>
<pv><lg.prov><l.prov><persName>   <w a=,.N71.01-*:fsn-:fpg->Varvarica</w>
</persName>   <w a=,variti.V33.01.3*:P3s:Y2s:A2s:A3s>vari</w>
</l.prov><l.prov>   <w a=,.Con*,.Adv*,.Int*>A</w>
<persName>   <w a=,.N71.01-*:msn-:mpg-,.N76.01+*:msn+:mpg+>Savica</w>
</persName>   <w a=,hladiti.V34.11.4*:P3s:Y2s:A2s:A3s>hladi</w>
</l.prov><l.prov><persName>   <w a=,.N76.01+*:msn+:mpg+>Nikolica</w>
</persName>   <w a=,kus.N08.01-*:msg-,kusati.V01.00.2*:P3s:A2s:A3s,kus.A06.01*:p#msg*:p#msa+:p#nsg*:p*fsn
</l.prov></lg.prov></pv>
<pexp>
    Varica se na <dateStruct><occasion type='slava'>Varin dan</occasion></dateStruct> vari.
```

```
    na <dateStruct><occasion type='slava'>Savin dan</occasion></dateStruct> hladi,
    a na <dateStruct><occasion type='slava'>Nikolj dan</occasion></dateStruct> jede; jer su ta tri sveca
    za drugim.
</pexp>
<ptr target=P502 resp='CV'>
</divp>
<divp id=P504 n=504>
<pv>   <w a=,.N72.01-*:fsn->Varka</w>
   <w a=,.Pre*>pred</w>
<persName>   <w a=,Marko.N42.01+*:msg+:msa+>Marka</w>
</persName></pv>
<pexp>
    Sjela četiri kaluđera da ručaju; kad se donose pred njih u jednoj činiji
    s čorbom riba rasječena na troje, onda iguman uzme glavu govoreći:
    <q rend='99l 66u'>Glava pred glavu;</q> drugi uzme srijedu govoreći:
    <q rend='99l 66u'>Rebra sjeverova i grad carja velikago;</q> a treći,
    kome je valja da je bilo ime <persName>Marko</persName>, uzme varku govoreći:
    <q rend='99l 66u'>Varka pred <persName>Marka</persName>;</q> četvrti pak
    videći da oni svu ribu uzeše, uzme činiju s čorbom vrućom, pa po njima
    svoj trojici govoreći: <q rend='99l 66u'>Izlija sja na vas blagodat
    Božija!</q>
</pexp>
<ptr target=P671 resp='CV'>
</divp>
<divp id=P505 n=505>
<pv>   <w a=,.N60.01-*:nsn-:nsa-:nsv->Vaskrsenije</w>
   <w a=,ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
   <w a=,.N15.51-*:msn-:msa->ručak</w>
   <w a=,.Pre*>bez</w>
   <w a=,večera.N70.01-*:fsg-:fpn-:fpa-:fpv->večere</w>
</pv>
<pexp>
    Valja da se misli prema Božiću, a osobito za to što se u početku proljeća
    premaklo svašta: od stare godine se pojelo, a od nove još nije prispjelo.
</pexp>
</divp>
<divp id=P506 n=506>
<pv>   <w a=,vi.ProN04:*pg*:*pa*>Vas</w>
   <w a=,.N08.01-J%svet#E3.03:msn-:msa->svijet</w>
   <w a=,živeti.V34.00.3E%#E4.23:P3s:Y2s,živjeti.V34.25.4J%živeti#E4.23:P3s:Y2s,živ.A08.01*:p@msn*:p@msa-
   <w a=,.Pre*,.Int*>na</w>
   <w a=,vjera.N70.01-J%vera#E2.12:fsa->vjeru</w>
   <w a=,.Con*,.Par*>i</w>
   <w a=,.Pre*,.Int*>na</w>
   <w a=,.N04.01-*:msn-:msa->amanat</w>
</pv>
<pexp>
    U <placeName type='država'>Crnoj Gori</placeName>.
</pexp>
</divp>
<divp id=P507 n=507>
<pv>   <w a=?>Vati</w>
   <w a=,kuja.N70.01+*:fsa+,kovati.V20.02.5*:P3p>kuju</w>
<persName>   <w a=,.N42.01+*:msn+:msv+>Rejo</w>
</persName></pv>
<pexp>
    <persName>Rejo</persName> je nekako ime, ali ja ne znam kako je. Kaže se kad ko
    čera koga. Ja sam ovo slušao u detinjstvu.
```

```
</pexp>
</divp>
<divp id=P508 n=508>
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,vatra.N72.01-*:fsa->vatru</w>
   <w a=,.Par*>ne</w>
   <w a=,žeći.V50.02.2*:P3s:A2s:A3s>žeže</w>
</pv>
<pexp>
    Ili: <s.a target=P509 resp='Vuk'></s.a>
</pexp>
</divp>
<divp id=P509 n=509>
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,vatra.N72.01-*:fsa->vatru</w>
   <w a=,.Par*>ne</w>
   <w a=,pržiti.V33.01.3*:P3s:Y2s:A2s:A3s>prži</w>
</pv>
<pexp>
    (prikučivši ono đe se ko opržio, kao da vatra izvuče vatru i bol).
</pexp>
<ptr target=P508 resp='CV'>
</divp>
<divp id=P510 n=510>
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,.Con*,.Par*>i</w>
   <w a=,.N70.01-*:fsn-:fpg->voda</w>
   <w a=,dobar.A14.01*:p*mpa*:p*fsg*:p*fpn*:p*fpa*:p*fpv*>dobre</w>
   <w a=,jesam.V99.00*:P3p>su</w>
   <w a=,sluga.N78.1.01+*:msg+:mpn+:mpa+:mpv+>sluge</w>
   <w a=,.Con*,ala.N70.01-*:fsd-:fsl-,ala.N70.01+*%hala#H1.07:fsd+:fsl+>ali</w>
   <w a=,zao.A14.04*:p@msn*:p@msa-:p*msv*:p*mpn*:p*mpv*>zli</w>
   <w a=,gospodar.N01.01+*:mpn+:mpv+,gospodariti.V33.01.3*:P3s:Y2s:A2s:A3s>gospodari</w>
</pv>
</divp>
<divp id=P511 n=511>
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,.Con*,.Par*>i</w>
   <w a=,.N70.01-*:fsn-:fpg->slama</w>
   <w a=,.Par*>ne</w>
   <w a=,moći.V52.00.2*:P3p,moći.V52.00.2*:P1s>mogu</w>
   <w a=,.Pre*,.Int*>u</w>
   <w a=,.Pre*,.Adv*>blizu</w>
   <w a=,.V32.01.5*:W>stajati</w>
</pv>
</divp>
<divp id=P512 n=512>
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,sebe.ProN11:**g*:**a*>se</w>
   <w a=?>zeitinom</w>
   <w a=,.Par*>ne</w>
   <w a=,gasiti.V34.12.4*:VP3s:VY2s:VA2s:VA3s>gasi</w>
</pv>
<pexp>
    Ili, koje je u narodu običnije: <s.a target=P513 resp='Vuk'></s.a>
</pexp>
</divp>
<divp id=P513 n=513>
```

```
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,sebe.ProN11:**g*:**a*>se</w>
   <w a=,slama.N70.01-*:fsi->slamom</w>
   <w a=,.Par*>ne</w>
   <w a=,gasiti.V34.12.4*:VP3s:VY2s:VA2s:VA3s>gasi</w>
</pv>
<pexp>
    Ili: <s.a target=P514 resp='Vuk'></s.a>
</pexp>
<ptr target=P512 resp='CV'>
</divp>
<divp id=P514 n=514>
<pv>   <w a=,.N72.01-*:fsn->Vatra</w>
   <w a=,sebe.ProN11:**g*:**a*>se</w>
   <w a=,ulxe.N60.01-*:nsi->uljem</w>
   <w a=,.Par*>ne</w>
   <w a=,tuliti.V34.03.4*:P3s:Y2s:A2s:A3s>tuli</w>
</pv>
<pexp>
    U <placeName type='država'>Crnoj Gori</placeName>.
</pexp>
<ptr target=P513 resp='CV'>
</divp>
<divp id=P515 n=515>
<pv>   <w a=,vaš.ProA04:nsn*:nsa*:fsg*:mpa*:fpn*:fpa*>Vaše</w>
   <w a=,.N60.01-*:nsn-:nsa-:nsv->prišestvije</w>
   <w a=,.ProA04:msn*:msa->naš</w>
   <w a=,.N04.04-*:msn-:msa->praznik</w>
</pv>
<pexp>
    Kažu da govore kaluđeri kad im dođu gosti, jer se i oni onda malo bolje
    počaste.
</pexp>
</divp>
<divp id=P516 n=516>
<pv>   <w a=,vaš.ProA04:mpn*>Vaši</w>
   <w a=,igrati.V01.00.2*:P3p>igraju</w>
   <w a=,.Con*,.Adv*,.Int*>a</w>
   <w a=,naš.ProA04:mpn*>naši</w>
   <w a=,ridati.V01.00.2*:P3p>ridaju</w>
</pv>
</divp>
<divp id=P517 n=517>
<pv>   <w a=,vaš.ProA04:mpn*>Vaši</w>
   <w a=,piti.V24.00.2*:P3p>piju</w>
   <w a=,.Con*,.Adv*,.Int*>a</w>
   <w a=,naš.ProA04:mpn*>naši</w>
   <w a=,pjan.A08.01*:p*mpa*:p*fsg*:p*fpn*:p*fpa*:p*fpv*>pjane</w>
   <w a=,biti.V24.00.2*:P3p>biju</w>
</pv>
</divp>
<divp id=P518 n=518>
<pv>   <w a=,vaška.N70.02+*:fsg+:fpn+:fpa+:fpv+>Vaške</w>
   <w a=,.Adv*>onda</w>
   <w a=,.Adv*,visok.A12.02*:s@mpa*:s@fsg*:s@fpn*:s@fpa*:s@fpv*:s@nsn*:s@nsa*:s@nsv*>najviše</w>
   <w a=,bjesnxeti.V34.38.6J%besneti#E2.20#E5.47:P3p>bjesne</w>
   <w a=,.Adv*,.Con*>kad</w>
   <w a=,ona.ProN07:fsg*:fsa*,jesam.V99.00*:P3s>je</w>
```

1811 Ja umio a ne imao, ja imao a ne umio.
1811 Ja umeo a ne imao, ja imao a ne umeo.

1812 Jača rđa od bijesa.
1812 Jača rđa od besa.

1813 Jača su dvojica nego sam Radojica.

1814 Jačega kapom i nejačega šakom.

1815 Jače je đelo nego besjeda.
1815 Jače je delo nego beseda.

1816 Jače je selo od svatova.

1817 Jače selo od međeda.

1818 Ja što mogoh pomogoh.

1819 Jedan drobi a drugi kusa.

1820 Jedan dušom, drugi ćesom.

1821 Jedan put mi je umrijeti.
1821 Jedan put mi je umreti.

1822 Jedan put se mre.

1823 Jedan put se počinje.

1824 Jedan rekao binjašče a drugi birašče, pa su se pogodili.

1825 Jedan ti Bog a jedna Božja vjera!
1825 Jedan ti Bog a jedna Božja vera!

1826 Jedan u klinac, drugi u ploču.

1827 Jedan čoek ne može sve znati.

1828 Jedva kraj s krajem sastavljam.

1829 Jedva mu koža kosti drži.

1830 Jedva sam pasjim zubima otklao.

1831 Jede kao mećava.

1832 Jedem, jedem, pa ne mogu.

1833 Jedina ruko Božja!

1834 Jedna glava hiljada jezika.

1835 Jedna kći kao nijedna, dvije ka' i jedna, a tri misli ti.

1836 Jedna koža ne može dva mesa dati.

1837 Jedna kruška na deset međeda.

1838 Jedna lasta ne čini proljeća.
1838 Jedna lasta ne čini proleća.

1839 Jedna mu noga drugoj dobra ne misli.

1840 Jedna palica ni pred carem ne gori.

1841 Jedna smrt tisuća uzroka.

1842 Jedna smrt trista uzroka.

1843 Jednaci kao linjaci.

1844 Jedna šteta sto grijeha.
1844 Jedna šteta sto greha.

1845 Jedna šugava ovca cijelo stado ošuga.

1846 Jednim do zuba a drugim do suza.

1847 Jedno ali vrijedno. 1847 Jedno ali vredno.

1848 Jedno ali jedro.

1849 Jednoga nazivalo a drugoga izdiralo.

1850 Jedno drugomu a Bog će svakomu.

1851 Jedno drugom do uha.

1852 Jedno zlo ne može čoeku dosaditi.

1853 Jedno kolo iz blata a drugo u blato.

1854 Jedno misli, drugo govori, a treće tvori.

1855 Jedno misli pijanica, a drugo mehandžija.

1856 Jednom mjeri, drugom kroj.
1856 Jednom meri, drugom kroj.

1857 Jednom nogom stao koračio u grob.

1858 Jednom rukom daje a dvjema uzima.

1859 Jednom se rađa a jednom umire.

1860 Jedno mu je na srcu a drugo na jeziku.

1861 Jedno radi a drugo odrađuje.

1862 Jednu manje!

1863 Jednu crkvu kvariti a drugu graditi slaba je zadužbina.

1864 Jeđi zelje iz svoga vrta.

1865 Jeza parala, g..a plaćala.

1866 Jezik gore može posjeći nego mač.
1866 Jezik gore može poseći nego mač.

1867 Jezik za zube!

1868 Jeko ječi a zdrav zveči.

1869 Jelesije proso sije, Ide Vide da obide.
1869 Jelesije proso seje, Ide Vide da obide.

1870 Je li zrela lubenica?

1871 Jemac platac.

1872 Jeremije u polje, A sve zmije u more.

1873 Jesen je bogata a zima rogata.

1874 Jesi li digao rosu sa srca?

1875 Jesi li pri sebi?

1876 Jesi li ti padao s tavana?

1877 Jesi li čitav?

1878 Jest puška, da je praha.

1879 Jeftin espap kesu prazni.

1880 Jecalu iezik smeta. a mene ne.

1882 Još kad je car kaplar bio.

1883 Još mu nijesu oči padale.

1884 Još nije ni do vode došao, a gaće zasukuje.

1885 Još se nije zima izjalovila.

1886 Još nije lula duvana poginula.

1887 Još se nije koza okozila, a kozle igra po polju.

1888 Jug babi za furunom.

1889 Južnu Božiću i prijateljskom kolaču ne valja se radovati.

1890 Junakova majka najprije zaplače.

1891 Junački je umro.

1892 Juče vezir danas rezil.

E.4.2 Konkordancije sa promenjenim pojmom ključne reči

```
  davati.V
<pv>Jednom rukom daje a dvjema uzima.</pv>
  delo#E2.39.N
<pv>Jače je delo nego besjeda.</pv>
  deo#E3.04.10.N
<pv>Jadan je onaj koji nema dijela od Boga.</pv>
  derati.V
<pv>Ja derem jarca a on kozu.</pv>
<pv>Jao moj <name type='?'>Vlaho</name>! ni ti rad tu ležati ni se ja više tebe derati, ali ne dadu <name
  deset.Num
<pv>Jedna kruška na deset međeda.</pv>
  desni.A
<pv>Ja sam to držao kao u desnoj ruci.</pv>
  dignuti.V
<pv>Jarmove na tavan dignuti.</pv>
<pv>Jesi li digao rosu sa srca?</pv>
  dlaka.N
<pv>Ja psu, a pas repu, a rep dlaci: idi dlaka ti si laka.</pv>
  do.Pre
<pv>Jao moj do groba, a od groba ko bude <opt.ph rend='bold'>moj</opt.ph>.</pv>
<pv>Jednim do zuba a drugim do suza.</pv>
<pv>Jedno drugom do uha.</pv>
<pv>Još nije ni do vode došao, a gaće zasukuje.</pv>
  dockan.Adv
<pv>Jabuka koja dockan sazri, dugo stoji.</pv>
  doneti#E4.02.V
<pv>Ja poslah sina u <name type='?'>Rim</name> da primijeni turin, a on kad dođe iz <name type='?'>Rima</
  dosaditi.V
<pv>Jedno zlo ne može čoeku dosaditi.</pv>
  doći.V
<pv>Ja vuk ne doći, ja šuke ne naći.</pv>
<pv>Ja poslah sina u <name type='?'>Rim</name> da primijeni turin, a on kad dođe iz <name type='?'>Rima</
<pv>Još nije ni do vode došao, a gaće zasukuje.</pv>
  držalica.N
<pv>Ja pravim držalicu, a ono se načini kijak: ja kijak i volim.</pv>
  držati.V
<pv>Jami mrtve prijatelje, koć drži za moć.</pv>
```

```
<pv>Ja sam to držao kao u desnoj ruci.</pv>
<pv>Jedva mu koža kosti drži.</pv>
  drago.Adv
<pv>Ja tebe, a ti kome ti drago.</pv>
  drem#E3.03.N
<pv>Ja kad vidjeh zelen drijen, predadoh mu vas moj drijem i lijen.</pv>
  dren#E3.07.01.N
<pv>Ja kad vidjeh zelen drijen, predadoh mu vas moj drijem i lijen.</pv>
  drobiti.V
<pv>Jedan drobi a drugi kusa.</pv>
  drugi.Ord
<pv>Jednu crkvu kvariti a drugu graditi <opt.ph rend='bold'>slaba je zadužbina</opt.ph>.</pv>
<pv>Jedno radi a drugo odrađuje.</pv>
<pv>Jedno mu je na srcu a drugo na jeziku.</pv>
<pv>Jednom mjeri, drugom kroj.</pv>
<pv>Jedno misli pijanica, a drugo mehandžija.</pv>
<pv>Jedno misli, drugo govori, a treće tvori.</pv>
<pv>Jedno kolo iz blata a drugo u blato.</pv>
<pv>Jedno drugomu a Bog <opt.ph rend='bold'>će</opt.ph> svakomu.</pv>
<pv>Jednoga nazivalo a drugoga izdiralo.</pv>
<pv>Jedna mu noga drugoj dobra ne misli.</pv>
<pv>Jedan u klinac, drugi u ploču.</pv>
<pv>Jedan rekao binjašče a drugi birašče, pa su se pogodili.</pv>
<pv>Jedan dušom, drugi ćesom.</pv>
<pv>Jedan drobi a drugi kusa.</pv>
<pv>Jedno drugom do uha.</pv>
<pv>Jednim do zuba a drugim do suza.</pv>
  dugo.Adv
<pv>Jabuka koja dockan sazri, dugo stoji.</pv>
  duvan.N
<pv>Još nije lula duvana poginula.</pv>
  duša.N
<pv>Ja ću umrijeti, a ne ću lako đavolu dušu dati.</pv>
<pv>Jedan dušom, drugi ćesom.</pv>
  dva.Num
<pv>Jedna koža ne može dva mesa dati.</pv>
<pv>Jedna <opt.ph rend='bold'>kći</opt.ph> kao nijedna, dvije ka' i jedna, a tri misli ti.</pv>
<pv><lg.prov><l.prov>Jako je magare,</l.prov><l.prov>Ali dva tovare.</l.prov></lg.prov></pv>
<pv>Ja poslah sina u <name type='?'>Rim</name> da primijeni turin, a on kad dođe iz <name type='?'>Rima</
<pv>Jednom rukom daje a dvjema uzima.</pv>
  dvojica.Num
<pv>Jača su dvojica nego sam <name type='?'>Radojica</name>.</pv>
  espap.N
<pv>Jeftin espap kesu prazni.</pv>
  furuna.N
<pv>Jug babi za furunom.</pv>
  gaće.N
<pv>Ja kuj, ja ne mrči gaća.</pv>
<pv>Još nije ni do vode došao, a gaće zasukuje.</pv>
  glas.N
<pv>Jao vrane, zla i toga glasa!</pv>
<pv>Ja tebe glas, ti mene muštuluk.</pv>
  glava.N
<pv>Jedna glava hiljada jezika.</pv>
  gore.Adv
<pv>Jezik gore može posjeći nego mač.</pv>
  goreti#E4.13.V
<pv>Jedna palica ni pred carem ne gori.</pv>
```

```
  govoriti.V
<pv>Ja govorim, a on ni u uho.</pv>
<pv>Ja govorim, a on ni u baltu.</pv>
<pv>Jedno misli, drugo govori, a treće tvori.</pv>
  graditi.V
<pv>Jednu crkvu kvariti a drugu graditi <opt.ph rend='bold'>slaba je zadužbina</opt.ph>.</pv>
  greh#E3.03.04.N
<pv>Jedna šteta sto grijeha.</pv>
  grob.N
<pv>Jednom nogom stao ((koračio)) u grob.</pv>
<pv>Jao moj do groba, a od groba ko bude <opt.ph rend='bold'>moj</opt.ph>.</pv>
  hadumac#H1.07.N
<pv>Ja <opt.ph rend='bold'>mu</opt.ph> kažem adumac sam, a on pita koliko dece imam!</pv>
  hiljada.Num
<pv>Jedna glava hiljada jezika.</pv>
  hteti#E3.08.V
<pv><lg.prov><l.prov>Ja te zovem i molim,</l.prov><l.prov>A kad ne ćeš, još volim.</l.prcv></lg.prov></pv
<pv>Ja ću umrijeti, a ne ću lako đavolu dušu dati.</pv>
<pv>Jedno drugomu a Bog <opt.ph rend='bold'>će</opt.ph> svakomu.</pv>
  i.Con
<pv>Južnu Božiću i prijateljskom kolaču ne valja se radovati.</pv>
<pv>Jačega kapom i nejačega šakom.</pv>
<pv>Ja sam lanjski i onomlanjski.</pv>
<pv>Jaoh zlotvoru i caru <name type='?'>Turskome</name>!</pv>
<pv>Ja ga krstim, a on <obs.ph value='?'>p..i</obs.ph> <opt.ph rend='bold'>i ne bilo živo!</opt.ph>.</pv>
<pv><lg.prov><l.prov>Ja te zovem i molim,</l.prov><l.prov>A kad ne ćeš, još volim.</l.prov></lg.prov></pv
<pv>Jao vrane, zla i toga glasa!</pv>
<pv>Jao moj <name type='?'>Vlaho</name>! ni ti rad tu ležati ni se ja više tebe derati, ali ne dadu <name
<pv>Ja kad vidjeh zelen drijen, predadoh mu vas moj drijem i lijen.</pv>
  i.Par
<pv>Jaku brastvu jaka i pravdu.</pv>
<pv>Jaje za jaje, ako i nije šareno.</pv>
<pv><lg.prov><l.prov>Ja te zovem i molim,</l.prov><l.prov>A kad ne ćeš, još volim.</l.prov></lg.prov></pv
<pv>Jedna <opt.ph rend='bold'>kći</opt.ph> kao nijedna, dvije ka' i jedna, a tri misli ti.</pv>
<pv>Ja pravim držalicu, a ono se načini kijak: ja kijak i volim.</pv>
  igrati.V
<pv>Još <opt.ph rend='bold'>se</opt.ph> nije koza okozila, a kozle igra po polju.</pv>
  imati.V
<pv>Ja umio a ne imao, ja imao a ne umio.</pv>
<pv>Ja <opt.ph rend='bold'>mu</opt.ph> kažem adumac sam, a on pita koliko dece imam!</pv>
  iz.Pre
<pv>Jeđi zelje iz svoga vrta.</pv>
<pv>Ja tikvu u vodu, a tikva iz vode.</pv>
<pv>Ja pseto iz bunara vadim, a ono me ujeda.</pv>
<pv>Ja poslah sina u <name type='?'>Rim</name> da primijeni turin, a on kad dođe iz <name type='?'>Rima</
<pv>Jedno kolo iz blata a drugo u blato.</pv>
  izdiralo.N
<pv>Jednoga nazivalo a drugoga izdiralo.</pv>
  izjaloviti.V
<pv>Još se nije zima izjalovila.</pv>
  ići.V
<pv>Ja ga mećem na policu, a on ide pod policu.</pv>
<pv>Ja psu, a pas repu, a rep dlaci: idi dlaka ti si laka.</pv>
<pv><lg.prov><l.prov>Jelesije proso sije,</l.prov><l.prov>Ide <name type='?'>Vide</name> da obide.</l.pro
  ja.Con
<pv>Ja sad, ja nikad.</pv>
<pv>Ja pravo, ja nikako.</pv>
<pv>Ja je čela, ja je brus, ja od motike štene.</pv>
```

```
<pv>Ja ja, ja on.</pv>
<pv>Ja umio a ne imao, ja imao a ne umio.</pv>
<pv>Ja kuj, ja ne mrći gaća.</pv>
<pv>Ja vuk ne doći, ja šuke ne naći.</pv>
  ja.ProN
<pv>Jecalu jezik smeta, a mene ne.</pv>
<pv>Jedan put mi je umrijeti.</pv>
<pv>Ja što mogoh pomogoh.</pv>
<pv>Ja ugnjivat' a neko uživat'.</pv>
<pv>Ja ti tamo kolač ne spremam.</pv>
<pv>Ja tebe rogom, a ti mene rodom!</pv>
<pv>Ja silom ne ulovih zeca.</pv>
<pv>Ja njega ni u kvak.</pv>
<pv>Ja mu ugađam kao čiru na prstu.</pv>
<pv>Ja mu tamo kolača ne spremam.</pv>
<pv>Ja zapovjedih đaku a đak crkvenjaku.</pv>
<pv>Ja ga ljubim, a on se utire.</pv>
<pv>Ja ja, ja on.</pv>
<pv>Ja ga mećem na policu, a on ide pod policu.</pv>
<pv>Ja tikvu u vodu, a tikva iz vode.</pv>
<pv>Ja pseto iz bunara vadim, a ono me ujeda.</pv>
<pv>Ja sam lanjski i onomlanjski.</pv>
<pv>Ja ga krstim, a on <obs.ph value='?'>p..i</obs.ph> <opt.ph rend='bold'>i ne bilo živo!</opt.ph>.</pv>
<pv><lg.prov><l.prov>Ja te zovem i molim,</l.prov><l.prov>A kad ne ćeš, još volim.</l.prov></lg.prov></pv
<pv>Ja <opt.ph rend='bold'>mu</opt.ph> kažem adumac sam, a on pita koliko đece imam!</pv>
<pv>Ja govorim, a on ni u uho.</pv>
<pv>Ja govorim, a on ni u baltu.</pv>
<pv>Ja tebe glas, ti mene muštuluk.</pv>
<pv>Ja ću umrijeti, a ne ću lako đavolu dušu dati.</pv>
<pv>Ja kad viđeh zelen drijen, predadoh mu vas moj drijem i lijen.</pv>
<pv>Ja tebe, a ti kome ti drago.</pv>
<pv>Ja pravim držalicu, a ono se načini kijak: ja kijak i volim.</pv>
<pv>Ja poslah sina u <name type='?'>Rim</name> da primijeni turin, a on kad dođe iz <name type='?'>Rima</
<pv>Ja psu, a pas repu, a rep dlaci: idi dlaka ti si laka.</pv>
<pv>Ja sam to držao kao u desnoj ruci.</pv>
<pv>Ja derem jarca a on kozu.</pv>
<pv>Jao moj <name type='?'>Vlaho</name>! ni ti rad tu ležati ni se ja više tebe derati, ali ne dadu <name
  jabuka.N
<pv>Jabuka koja dockan sazri, dugo stoji.</pv>
  jad.N
<pv>Jade nesiti!</pv>
  jadan.A
<pv>Jadan <name type='?'>Duka</name>, zaludu ti muka!</pv>
<pv>Jadan je onaj koji nema dijela od Boga.</pv>
  jaje.N
<pv>Jaje za jaje, ako i nije šareno.</pv>
  jak.A
<pv>Jače selo od međeda.</pv>
<pv>Jače je selo od svatova.</pv>
<pv>Jača rđa od bijesa.</pv>
<pv>Jako brastvo brzo zapleće.</pv>
<pv>Jak kao međed.</pv>
<pv>Jak kao zemlja.</pv>
<pv>Jak kao vjedogonja.</pv>
<pv>Jaki na braniku.</pv>
<pv>Jaka je kobila narod!</pv>
<pv>Jaku brastvu jaka i pravdu.</pv>
<pv>Jačega kapom i nejačega šakom.</pv>
```

```
<pv>Jača su dvojica nego sam <name type='?'>Radojica</name>.</pv>
<pv><lg.prov><l.prov>Jako je magare,</l.prov><l.prov>Ali dva tovare.</l.prov></lg.prov></pv>
<pv>Jače je delo nego besjeda.</pv>
  jama.N
<pv>Jami mrtve prćije, već drži za moć.</pv>
  jao.Int
<pv>Jao ti je još kako je <name type='?'>Lazo</name> na <name type='?'>Kosovu</name> poginuo.</pv>
<pv>Jao onome koga žale!</pv>
<pv>Jao moj tupi tupane! S tobom tupo, a bez tebe zarubljeno.</pv>
<pv>Jao majci, svi ti smo jednaci!</pv>
<pv>Jao vrane, zla i toga glasa!</pv>
<pv>Jao moj do groba, a od groba ko bude <opt.ph rend='bold'>moj</opt.ph>.</pv>
<pv>Jao moj <name type='?'>Vlaho</name>! ni ti rad tu ležati ni se ja više tebe derati, ali ne dadu <name
  jaoh.Int
<pv>Jaoh zlotvoru i caru <name type='?'>Turskome</name>!</pv>
  jarac.N
<pv>Ja derem jarca a on kozu.</pv>
  jaram.N
<pv>Jarmove na tavan dignuti.</pv>
  jecalo.N
<pv>Jecalu jezik smeta, a mene ne.</pv>
  jedan.Num
<pv>Jednu manje!</pv>
<pv>Jedno ali jedro.</pv>
<pv>Jedno ali vrijedno.</pv>
<pv>Jedna šugava ovca cijelo stado ošuga.</pv>
<pv>Jedna šteta sto grijeha.</pv>
<pv>Jedna smrt trista uzroka.</pv>
<pv>Jedna smrt tisuća uzroka.</pv>
<pv>Jedna lasta ne čini proljeća.</pv>
<pv>Jedan čoek ne može sve znati.</pv>
<pv>Jedan ti Bog a jedna Božja vjera!</pv>
<pv>Jedan put se počinje.</pv>
<pv>Jedan put se mre.</pv>
<pv>Jedan put mi je umrijeti.</pv>
<pv>Jednom nogom stao ((koračio)) u grob.</pv>
<pv>Jedna palica ni pred carem ne gori.</pv>
<pv>Jedna glava hiljada jezika.</pv>
<pv>Jedna koža ne može dva mesa dati.</pv>
<pv>Jedna <opt.ph rend='bold'>kći</opt.ph> kao nijedna, dvije ka' i jedna, a tri misli ti.</pv>
<pv>Jednu crkvu kvariti a drugu graditi <opt.ph rend='bold'>slaba je zadužbina</opt.ph>.</pv>
<pv>Jedno radi a drugo odraduje.</pv>
<pv>Jedno mu je na srcu a drugo na jeziku.</pv>
<pv>Jednom mjeri, drugom kroj.</pv>
<pv>Jedno misli pijanica, a drugo mehandžija.</pv>
<pv>Jedno misli, drugo govori, a treće tvori.</pv>
<pv>Jedno kolo iz blata a drugo u blato.</pv>
<pv>Jedno drugomu a Bog <opt.ph rend='bold'>će</opt.ph> svakomu.</pv>
<pv>Jednoga nazivalo a drugoga izdiralo.</pv>
<pv>Jedna mu noga drugoj dobra ne misli.</pv>
<pv>Jedan u klinac, drugi u ploču.</pv>
<pv>Jedan rekao binjašče a drugi birašče, pa su se pogodili.</pv>
<pv>Jedan dušom, drugi česom.</pv>
<pv>Jedan drobi a drugi kusa.</pv>
<pv>Jedno zlo ne može čoeku dosaditi.</pv>
<pv>Jedno drugom do uha.</pv>
<pv>Jednim do zuba a drugim do suza.</pv>
<pv>Jednom rukom daje a dvjema uzima.</pv>
```

<pv>Jedna kruška na deset medveda.</pv>

E.4.3 Analiza pojavljivanja oblika

```
Božić.N24.01-* - :msd-
Božji.A03.02* - :p@fsn*:p@fsv*
Bog.N01.02+* - :msg+:msn+
Bog.N07.02+* - :msg+:msn+
Duka.N70.01+* - :msn+
Jelesije.N46.01+* - :msn+
Jeremije.N60.01+* - :msn+
Kosovo.N51.01-* - :nsd-
Lazo.N42.01+* - :msn+
Radojica.N71.01+* - :msn+
Rim.N08.01-* - :msa-:msg-
Vlah.01.07+* - :mpn+
Vlah.Nxx - :msv+
žaliti.V34.03.4* - :P3p
živ.A08.01* - :p*nsn*
a.Adv* -
a.Con* -
a.Int* -
adumac.N17.61+*%hadumac#H1.07 - :msn+
ako.Con* -
ali.Con* -
baba.N70.01+* - :fsd+
badava.Adv* -
besjeda.N70.01-J%beseda#E4.12 - :fsn-
bez.Pre* -
bijes.N08.01-J%bes#E2.03.02 - :msg-
bilo.Con* -
binjašče.N66.01-* - :nsn-
birašče.N66.01-* - :nsn-
biti.V99.02E - :PPsn:P3s:PPsm
biti.V99.03J - :PPsn:P3s:PPsm
blato.N51.01-* - :nsg-:nsa-
bogat.A08.01* - :p*fsn*
branik.N13.04-* - :msd-
brastvo.N51.02-* - :nsn-
brus.N08.01-* - :msn-
brzo.Adv* -
bunar.N04.01-* - :msg-
car.N19.01+* - :msd+:msi+:msn+
cijel.A06.01J - :p*nsa*
crkva.N70.02-* - :fsa-
crkvenxak.N01.04+* - :msd+
da.Con* -
da.Par* -
danas.Adv* -
dati.V06.50.2* - :P3p:W
davati.V20.01.4* - :P3s
derati.V21.00.3* - :P1s:W
deset.Num04* -
desni.A03.01* - :p@fsd*
dignuti.V57.50.5* - :W:PPms
```

```
lako.Adv* -
lanjski.A03.01* - :p@msn*
lasta.N70.01+* - :fsn+
ležati.V31.00.3* - :W
li.Par -
li.Par* -
lijen.N04.01-J%len#E2.03 - :msa-
linxak.N01.04+* - :mpn+
lubenica.N71.01-* - :fsn-
lula.N70.01-* - :fsn-:fpg-
lxubiti.V34.01.4* - :P1s
magare.N65.01+* - :nsn+
majka.N72.02+* - :fsd+:fsn+
manje.Adv* -
mač.N23.01-* - :msn-
mehandyija.N75.01+*%#H3.08 - :msn+
meso.N51.01-* - :nsg-
metati.V21.08.4* - :P1s
među.N01.01+J - :msn+:msg+:mpg+
mećava.N70.01-* - :fsn-
misliti.V34.16.5* - :Y2s:P3s
mjeriti.V33.01.3J%meriti#E2.25 - :Y2s
moj.ProA06 - :msa-:msv*:msn*
moliti.V34.03.4* - :P1s
more.N60.01-* - :nsa-
motika.N70.04-* - :fsg-
moć.N82.01-* - :fsa-
moći.V52.00.2* - :A1s:P3s:P1s
mreti.V26.00.3E%#E3.03 - :P3s
mrijeti.V26.25.5J%mreti#E3.03 - :P3s
mrtav.A08.02* - :p*fpa*
mrčiti.V33.01.3* - :Y2s
muka.N70.04-* - :fsn-
muštuluk.N13.04-* - :msa-
na.Pre* -
najprije.Adv* -
narod.N01.01+* - :msn+
nazivalo.N51.01-* - :nsn-
naći.V56.50.2* - :W
načiniti.V33.51.3* - :P3s
ne.Par* -
nego.Con* -
nejak.A07.01* - :k@msg*
neko.ProN12 - :**n+
nemati.V05.00.2* - :P3s
nesit.A06.01* - :p*msv*
ni.Con* -
ni.Par* -
nijedan.Num01* - :fsn*
nikad.Adv* -
nikako.Adv* -
noga.N73.01-* - :fsn-:fsi-
obići.Vxx - :P3s
od.Pre* -
odrađivati.V20.03.5* - :P3s
okoziti.V33.51.3* - :PPsf
on.ProN05 - :msa*:msn*:msd*
ona.ProN07 - :fsg*:fsa*
```

```
onaj.Par* -
onaj.ProA01 - :msn*:msd*
ono.Par -
ono.ProN06 - :nsa*:nsd*:nsn*
onomlanjski.A03.01* - :p@msn*
oslati.V22.52.5* - :A1s
otklati.V21.63.4* - :PPsm
ovca.N70.02+* - :fsn+
ošugati.V01.50.2* - :P3s:A3s
oči.N86.01-* - :fpn-
pa.Con* -
padati.V01.00.2* - :PPsm:PPpf
palica.N71.01-* - :fsn-
parati.V01.00.2* - :PPsf
pas.N08.01-* - :msn-
pas.N12.51+* - :msd+:msn+
pasji.A03.02* - :p@*pi*
pijanica.N71.01+* - :fsn+
pitati.V01.00.2* - :P3s
piti.V24.00.2* - :W
platac.N17.16+* - :msn+
plaćati.V01.00.2* - :PPsf
ploča.N70.01-* - :fsa-
po.Pre* -
pod.Pre* -
poginuti.V23.50.3* - :PPms:PPfs:PPnp
pogoditi.V33.51.3* - :PPpm
polica.N71.01-* - :fsa-
polje.N60.01-* - :nsa-
polxe.N60.01-* - :nsd-
pomoći.V53.50.2* - :A1s
posjeći.V50.76.4J%poseći#E4.12 - :W
počinxati.V20.00.3* - :P3s
prah.N15.01-* - :msg-
praviti.V34.01.4* - :P1s
pravo.Adv* -
prazniti.V34.15.5* - :P3s
pred.Pre* -
predati.V06.50.2 - :A1s
pri.Pre* -
prijatelxski.A03.01* - :p@msd*
primijeniti.V33.51.3J%primeniti#E5.02 - :A3s
prolxeće.N60.01-J%proleće#E5.28 - :nsg-
proso.N51.01-* - :nsa-
prst.N04.01-* - :msd-
prst.N26.01-* - :msd-
prćija.N70.01-* - :fpa-
pseto.N51.01+* - :nsa+
put.Adv* -
puška.N70.02-* - :fsn-
pčela.N70.01+* - :fsn+
rad.Axx - :p#msn*
radi.Pre* -
raditi.V34.11.4* - :P3s
radovati.V20.02.5* - :W
rakija.N70.01-* - :fsn-
raďati.V01.00.2* - :P3s
rep.N08.01-* - :msd-:msn-
```

ProN:sn* ProN:sa V:P1s Con V:P1s, Con Con V:P2s, Adv V:P1s. - 1805
ProN:sn* ProN:sa V:P1s Pre N:sa, Con ProN:sn V:P3s Pre N:sa. - 1758
ProN:sn* ProN:sa V:P1s, Con ProN:sn ProN:*a V:P3s. - 1757
ProN:sn* ProN:sa V:P1s, Con ProN:sn V:P3s Con Par Con A:p#sn! - 1756
ProN:sn* ProN:sd Adv N:sa Par V:P1s. - 1807
ProN:sn* ProN:sd Adv N:sg Par V:P1s. - 1781
ProN:sn* ProN:sd V:P1s Con N:sd Pre N:sd. - 1782
ProN:sn* ProN:sd V:P1s N:sn V:P1s, Con ProN:sn V:P3s Adv N:pg V:P1s. - 1780
ProN:sn* V:A1s N:sa Pre N:sa Con V:A3s N:sa, Con ProN:sn Con V:A3s Pre N:sg, V:A3s Num:pa N:sg. - 1792
ProN:sn* V:A1s N:sd Con N:sn N:sd. - 1765
ProN:sn* V:P1s A:p@sn Con A:p@sn. - 1799
ProN:sn* V:P1s N:sa Con ProN:sn N:sa. - 1764
ProN:sn* V:P1s N:sa, Con Par Par V:P3s N:sn: ProN:sn* N:sn Par V:P1s. - 1793
ProN:sn* V:P1s ProA:sa V:PPs Adv Pre A:p@sd N:sd. - 1800
ProN:sn* V:P1s V:W, Con V:P1s Adv N:sd N:sa V:W. - 1808
ProN:sn* V:P1s, Con ProN:sn Par Pre N:sa. - 1760
ProN:sn* V:W Con ProN:*n V:W. - 1809
V:P1s, V:P1s, Con Par V:P1s. - 1831
V:P2s Par A:p#sn? - 1876
V:P2s Par Pre ProN:*d? - 1874
V:P2s Par ProN:sn V:PPs Pre N:sg? - 1875
V:P2s Par V:PPs N:sa Pre N:sg? - 1873
V:P3s Con N:sn. - 1830
V:P3s Par A:p*sn N:sn? - 1869
V:Y2s N:sa Pre ProA:sg N:sg. - 1863