Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Master rad

Stohastičke kontekst slobodne gramatike i primene

Mina Aleksandra Konaković, 1143/2013

Mentor: Prof. dr Gordana Pavlović-Lažetić

Beograd, avgust 2014.

Sadržaj

1 Uvod

Opšte je poznato da su računarske metode dale veliki doprinos razvoju raznih nauka, pa tako i bioinformatika ima značajan uticaj na otkrića iz oblasti biologije i medicine. Pogodnosti računara da veoma brzo skladište i obrade veliku količinu podataka, od krucijalne su važnosti za neverovatan pomak u ovim oblastima poslednjih decenija.

Stohastičke kontekst slobodne gramatike (skraćeno *SKSG*) su vrlo moćan računarski alat koji se koristi na raznim poljima, kao što su programski jezici, lingvistika, obrada slika i biologija. Prvenstveno su razvijene za potrebe modelovanja jezika, ali su kasnije našle svoju značajnu primenu i u drugim oblastima.

Jednostavan verovatnosni model koji se javlja kod stohastičkih konteskt slobodnih gramatika, povezuje ih sa skrivenim Markovljevim modelima, koji su se pokazali kao veoma važan koncept u računarstvu. Upotrebom raznih metoda zasnovanih na idejama skrivenih Markovljevih modela (metoda *Unutra-Spolja* je ekvivalent *Backward-Forward* metodi skrivenih Markovljevih modela, *CYK* algoritam je pandan *Viterbi* algoritmu), *SKSG* su dale veliki doprinos u sferama u kojima su do sada našle svoju primenu.

Ova teza prikazuje spoj matematičkih i računarskih tehnika u realizaciji koncepta SKSG sa primenama na biološkom polju. Poglavlja 2 i 3 odnose se na teoriju formalnih jezika i definicije gramatika. Četvrto poglavlje razmatra osnovne algoritme za rad sa *SKSG*. Analogno skrivenim Markovljevim modelima, imamo algoritme za određivanje parametara pomoću skupova za obučavanje i određivanje najverovatnijih putanja kroz stanja. U poglavlju 5 predstavljen je najosnovniji deo teorije o ribonukleinskoj kiselini (skraćeno *RNK*) za potrebe ove teze. Ideja je da se stekne uvid u osnovne pojmove molekularne biologije vezane za *RNK*, ali sa akcentom na informacije potrebne za primenu računarskih metoda.

Poglavlja 6 i 7 bave se vezom između *SKSG* i *RNK*, kao i upotrebom bioinformatičkih sistema za predikciju sekundarne strukture *RNK* sekvenci. Konačno, konkretan način primene svih navedenih tehnika, predstavljen je u poglavlju 8. Naveden je postupak formiranja *SKSG* za modelovanje *RNK* sekvenci pomoću skupova za obučavanje i njihova upotreba u određivanju sekundarne strukture *RNK*, na način koji je strogo matematički i računarski.

2 Kontekstno slobodne gramatike

Teorija formalnih jezika kao disciplina je počela ubrzano da se razvija radom lingviste Noama Čomskog pedesetih godina dvadesetog veka, kada je pokušao da formira preciznu karakterizaciju strukture prirodnih jezika. Njegov cilj je bio da definiše sintaksu jezika koristeći jednostavne i precizne matematičke formule. Tako je nastala poznata hijerarhija Čomskog, koja će biti prikazana u poglavlju 2.1.

Kasnije je otkriveno da se i sintaksa programskih jezika može u potpunosti opisati koristeći jedan od gramatičkih modela Čomskog, poznat kao kontekstno slobodne gramatike. Štaviše, naučnici su došli do zaključka da se sve informacije u računarima - od brojeva, imena, slika, do zvučnih talasa - mogu predstaviti u obliku niski karaktera. Tada su kolekcije niski, poznate kao jezici, postale centralna oblast računarskih nauka, a rukovanje niskama moćan alat. Samim tim, sasvim je logično da su tehnike vezane za gramatike pronašle svoju veliku primenu i u bioinformatici. Na primer, RNK sekvenca se može smatrati nizom karaktera A, C, G i U, pa se mogu formirati gramatike za taj jezik i primeniti neki od već poznatih algoritama za rad sa gramatikama. Slično važi i za druge značajne biološke molekule, kao što su DNK ili proteini. Literatura korišćena u poglavlju 2 je [1][2][3][4].

2.1 Formalni jezici i hijerarhija Čomskog

Prikazaćemo sada preciznije definicije formalnih jezika i gramatika, kao i hijerarhiju Čomskog.

Neka je Σ konačan skup simbola. Tada Σ nazivamo *azbukom*. *Niska* je konačan niz simbola iz azbuke, a Σ^* skup svih konačnih niski koje se mogu formirati simbolima iz Σ, uključujući i praznu nisku. Kako prazna niska ne sadrži ni jedan vidljivi simbol, označavaćemo je sa ε. *Dužina* niske x, u oznaci $|x|$, je broj simbola od kojih je ona sastavljena, a dužina od ε je 0.

Definicija 2.1.1: *(Formalni) jezik L* nad azbukom Σ je bilo koji podskup skupa Σ^*: $L \subseteq \Sigma^*$.

Elementi skupa L, odnosno jezika, nazivaju se *reči.* U bioinformatici, za elemente jezika takođe se koristi izraz *sekvence.*

Definicija 2.1.2: *Gramatika G* je uređena četvorka (Σ, *N, R, S*) gde je:

- Σ *završna azbuka,* čiji se elementi nazivaju *završni* ili *terminalni simboli*
- N *nezavršna azbuka,* čiji se elementi nazivaju *nezavršni* ili *neterminalni simboli,* takva da je $\Sigma \cap N = \varnothing$,
- $S \in N$ koji se naziva *početni simbol*
- $R \subseteq (\Sigma \cup N)^* N (\Sigma \cup N)^* \times (\Sigma \cup N)^*$ - *konačni skup pravila.*

Napomena 2.1.1: Oznaka $(A)^*$ znači pojavljivanje elemenata iz skupa A nula ili više puta.

Definicija 2.1.3: Kažemo da se niska $s_2 = \alpha_1 \beta \alpha_2$ *izvodi* iz niske $s_1 = \alpha_1 A \alpha_2$ (u oznaci $s_1 \Rightarrow s_2$), ukoliko u gramatici postoji pravilo $A \rightarrow \beta$. Ovo izvođenje je izvršeno u jednom koraku. Kada je za izvođenje primenjeno nula ili više pravila gramatike, pišemo $s_1 \Rightarrow^* s_2$.

Sada možemo predstaviti jezik i preko gramatike.

Definicija 2.1.4: *Jezik* opisan gramatikom $G = (\Sigma, N, R, S)$, u oznaci $L(G)$, je

$$L(G) = \{ x \mid x \in \Sigma^* \wedge S \Rightarrow^* x \}.$$

Kažemo da gramatika G *generiše* jezik $L(G)$. Za nisku $x \in L(G)$ kažemo da je *generisana* gramatikom G.

Dakle, jezik opisan zadatom gramatikom čine sve niske završnih simbola zadate gramatike, koje se mogu izvesti iz početnog simbola gramatike. Niska x *pripada* jeziku $L(G)$ ukoliko se iz početnog simbola gramatike G, pravilima izvođenja može dobiti niska x.

Noam Čomski je uspeo da uspostavi hijerarhiju među jezicima, klasifikujući gramatike prema strukturi njihovih pravila. Prema hijerarhiji Čomskog, gramatike se mogu klasifikovati u četiri osnovne kategorije (*Slika 2.1.1*).

Slika 2.1.1 Hijerarhija Čomskog. Slika je konstruisana na osnovu [4]

Sve inkluzije sa *Slike 2.1.1* su prave, odnosno nema jednakosti između nikoja dva podskupa. Dakle, za gramatike važi sledeće:

regularne $\subset$ kontekstno slobodne $\subset$ kontekstno osetljive $\subset$ rekurzivno prebrojive.

Dokaz ovog tvrđenja može se naći u [4, sekcija 11.5].

Regularni jezici su oni koji se mogu generisati regularnim gramatikama, kontekstno slobodni jezici moraju biti generisani kontekstno slobodnim gramatikama, kontekstno osetljivi su proizvod kontekstno osetljivih gramatika, a najopštiji jezici su rekurzivno prebrojivi jezici. To su jezici kod kojih niska može biti prepoznata Tjuringovom mašinom u konačnom vremenu. Za više detalja, pogledati [4].

Nas u ovom radu najviše zanimaju kontekstno slobodni jezici i kontekstno slobodne gramatike, jer su oni osnova stohastičkih kontekst slobodnih gramatika. Takođe, kao što postoje verovatnosni modeli nad kontekstno slobodnim gramatika, oni se javljaju i nad regularnim gramatikama, pod nazivom *stohastičke regularne gramatike* (skraćeno SRG). Između SRG i skrivenih Markovljevih modela postoji jedan-jedan korespodencija.

2.2 Kontekstno slobodne gramatike – osnovne definicije

Definicija 2.2.1: *Kontekstno slobodna gramatika KSG* je uređena četvorka $G = (\Sigma, N, R, S)$, gde je Σ azbuka (skup terminala), N skup neterminala (simbola koji ne mogu biti završni), S početni simbol (iz kojeg počinju sva izvođenja), a R konačan skup pravila izvođenja. Pravila izvođenja su oblika $A \rightarrow \alpha$, gde je $A \in N$, a $\alpha \in (\Sigma \cup N)^*$.

Napomena 2.2.1: U daljem tekstu ćemo koristiti velika slova za neterminale, a mala slova za terminale.

Napomena 2.2.2: Simboli iz Σ se nazivaju terminali, jer jednom kad se pojave u izvođenju, ostaju tu do kraja.

Uobičajeno je da se sva pravila koja se izvode iz istog neterminala A saberu u jedno, tako što se sa leve strane pravila nalazi samo taj neterminal A, a sa desne ostala izvođenja vezana za A, razdvojena znakom ' | '. Dakle, ukoliko imamo skup pravila $\{A \rightarrow \alpha_1, A \rightarrow \alpha_2, A \rightarrow \alpha_3\}$, predstavićemo ga kao $\{A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3\}$.

Jedan od načina da se opišu izvođenja niske $x \in L(G)$ iz pravila gramatike G je *stablo izvođenja.* Svakom izvođenju $S \Rightarrow^* x$ se dodeljuje jedno *stablo izvođenja.*

Definicija 2.2.2: Za datu kontekstno slobodnu gramatiku $G = (\Sigma, N, R, S)$ i nisku $x \in L(G)$, *stablo izvođenja u gramatici G* je uređeno stablo D sa sledećim svojstvima:

- koren stabla je obeležen početnim simbolom S gramatike G

- reč dobijena dopisivanjem listova stabla D sleva udesno, je reč x
- unutrašnji čvorovi su obeleženi simbolima iz N (neterminalima)
- ako je $A \in N$ unutrašnji čvor stabla, a njegovi neposredni potomci sleva udesno $M_1, M_2, ..., M_n \in \Sigma \cup N \cup \{\varepsilon\}$, tada je $A \rightarrow M_1M_2...M_n \in R$. Posebno, ako $A \rightarrow \varepsilon \in R$, onda čvor obeležen sa A može imati samo jednog potomka označenog sa ε.

Dva izvođenja u gramatici G su *ekvivalentna* ako imaju isto stablo izvođenja. Takođe, za jednu nisku $x \in L(G)$ može postojati više različitih stabala izvođenja.

Definicija 2.2.3: KSG $G = (\Sigma, N, R, S)$ je *jednoznačna* ako svaka reč jezika $L(G)$ ima tačno jedno stablo izvođenja. U suprotnom, gramatika je *višeznačna.*

Primer 2.2.1: Neka je data azbuka $\Sigma = \{c, g\}$ i jezik $L = \{s \mid s = c(c)^*g\}$ nad tom azbukom, gde $(c)^*$ označava nula ili više pojavljivanja simbola c. Reči koje pripadaju ovom jeziku su na primer: *cg, ccg, ccccccg, cccg.* Pravila izvođenja gramatike koja generiše ovaj jezik su:

$$\begin{aligned} S &\rightarrow S_1 g \\ S_1 &\rightarrow S_1 S_1 \\ &\mid c \end{aligned}$$

Početni simbol ove gramatike je S, terminali su c i g, a neterminali S i S_1. Očigledno je da reč *ccccg* pripada jeziku L. Jedno od izvođenja ove reči bilo bi sledeće:

$$S \Rightarrow S_1 g \Rightarrow S_1S_1g \Rightarrow cS_1g \Rightarrow cS_1S_1g \Rightarrow ccS_1g \Rightarrow ccS_1S_1g \Rightarrow cccS_1g \Rightarrow ccccg$$

Ovakvo izvođenje zove se *levo izvođenje* (ako se može primeniti više pravila u nekom koraku izvođenja, primenjuje se ono koje se odnosi na prvi neterminal sleva). Primetimo da ovo nije jedino levo izvođenje. Drugo levo izvođenje bilo bi:

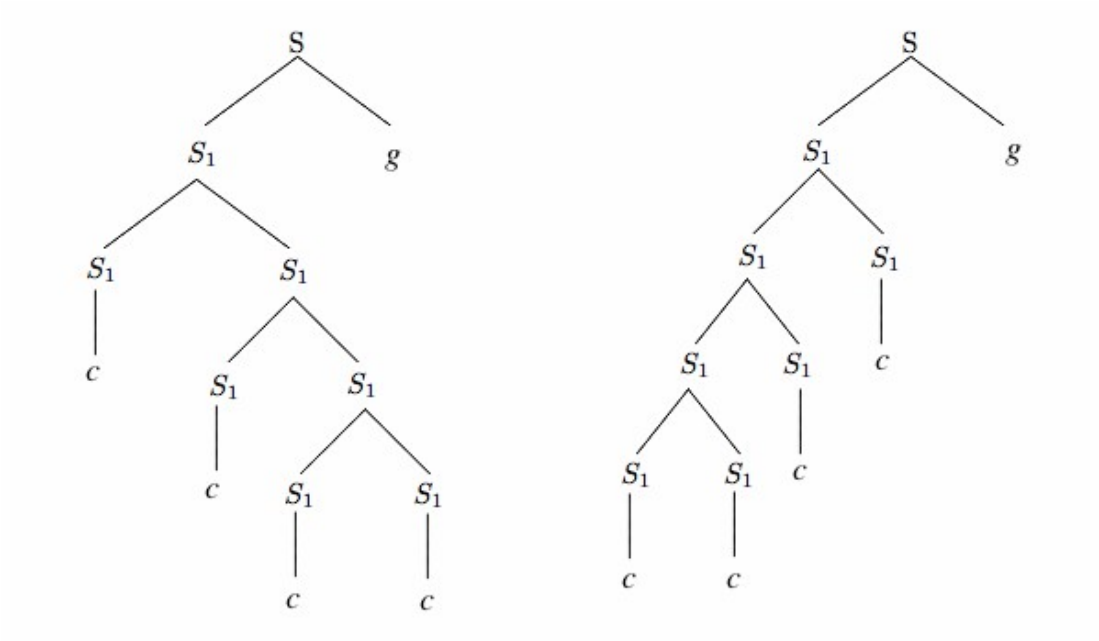
$$S \Rightarrow S_1 g \Rightarrow S_1S_1g \Rightarrow S_1S_1S_1g \Rightarrow S_1S_1S_1S_1g \Rightarrow cS_1S_1S_1g \Rightarrow ccS_1S_1g \Rightarrow cccS_1g \Rightarrow ccccg$$

Na *Slici 2.2.1* su prikazana stabla izvođenja za dva navedena primera izvođenja niske *ccccg.*

Gramatiku za jezik L možemo predstaviti i na sledeći način:

$$\begin{aligned} S &\rightarrow S_1 g \\ S_1 &\rightarrow cS_1 \\ &\mid c \end{aligned}$$

Ovim smo dobili da u svakom koraku izvođenja postoji tačno jedno pravilo koje se može primeniti. Dakle, nova gramatika više nije višeznačna. U radu sa KSG nekad je poželjno imati jednoznačnu gramatiku, a nekad ne. Na primer, gramatike koje opisuju programske jezike moraju biti jednoznačne, dok je u bioinformatici potpuno drugi slučaj. U bioinformatici su gramatike uglavnom višeznačne. Za svaku reč postoji nekoliko stabala izvođenja, pa se bira jedno od tih stabala sa najvećim skorom izvođenja ili najvećom verovatnoćom. Više reči o tome biće u poglavlju stohastičkih kontekst slobodnih gramatika.



Slika 2.2.1 Dva različita stabla izvođenja niske *ccccg*.

2.3 Normalne forme KS gramatika

Osnovna definicija KSG dozvoljava da se u gramatici pojave *nepotrebni* neterminali, oni neterminali do kojih se nikada neće stići ako se krene iz zadatog početnog simbola. Ukoliko za neterminal A ne postoji drugo pravilo osim $A \rightarrow AB$, on bi izazvao beskonačno izvođenje, jer nikada ne bismo mogli da izbacimo A iz izvođenja. Takođe, postoje i *neproduktivni ciklusi pravila* $A \rightarrow B \rightarrow C \rightarrow A$, koji ne izvode ni jedan terminal. Gramatiku koja nema ni jedno od navedenih svojstava nazivamo *čistom* (engl. *clean*) [3]. U daljem tekstu ćemo podrazumevati da je gramatika sa kojom radimo čista.

Postoji mnogo različitih gramatika koje generišu jedan isti formalni jezik, čak i ako su sve te gramatike čiste. Međutim, neke od tih gramatika su pogodnije za primenu u algoritmima, nego druge. Tako smo došli do *normalne forme,* koja ima posebno jednostavna pravila izvođenja:

Definicija 2.3.1: *KS* gramatika je u *normalnoj formi Čomskog* ako su sva njena pravila ili oblika $X \rightarrow YZ$ ili oblika $X \rightarrow a$, gde je $X, Y, Z \in N$, $a \in \Sigma$. Ako $\varepsilon \in L(G)$, tada je $S \rightarrow \varepsilon$ jedino pravilo koje sadrži ε, a S se ne pojavljuje na desnoj strani ni jednog pravila gramatike.

Uvek možemo naći *KS* gramatiku u normalnoj formi Čomskog za zadati *KS* jezik. Na primer, pravilo

$$X \rightarrow aYbXZc$$

može biti zamenjeno skupom pravila

$$X \to UA, \quad U \to a, \quad A \to YB, \quad B \to VC, \quad V \to b, \quad C \to XD, \quad D \to ZW, \quad W \to c$$

gde su *A, B, C* i *D* novi neterminali. Ovakva zamena pravila nije uticala na jezik $L(G)$, a pogodnija je za parsiranje, o čemu je reč u narednom poglavlju.

Primer 2.2.1: (*nastavak*) Gramatika koju smo videli u ovom primeru nije u normalnoj formi Čomskog (NFČ), ali je lako možemo preformulisati da bude:

$$\begin{aligned} S &\to S_1 S_2 \\ S_2 &\to g \\ S_1 &\to S_3 S_1 \\ &\quad | \, c \\ S_3 &\to c \end{aligned}$$

Navedena pravila zadovoljavaju uslove NFČ.

3 Stohastičke kontekst slobodne gramatike

Stohastičke kontekst slobodne gramatike (skraćeno SKSG) su u istoriji računarskih nauka počele da se pojavljuju uglavnom kao alat za obradu prirodnih jezika. Međutim, nešto kasnije su našle i svoju primenu u bioinformatici. Najjednostavnije objašnjenje šta zapravo predstavljaju SKSG jeste da su one kontekst slobodne gramatike kojima je priključen verovatnosni model. Na taj način dobijamo mogućnost izračunavanja verovatnoće generisanja neke niske zadatom gramatikom, ili određivanja najverovatnijeg izvođenja kod višeznačnih gramatika. Literatura koja je korišćena u ovom poglavlju je [2][3][5].

3.1 SKSG – osnovni pojmovi i definicije

Pogledajmo preciznu definiciju SKSG:

Definicija 3.1: Stohastička kontekst slobodna gramatika $G = (\Sigma, N, R, S, P)$ je kontekstno slobodna gramatika (Σ, N, R, S), kojoj je priključena $P : N \to (0,1]$ funkcija verovatnoće izvođenja. Odnosno, svakom pravilu izvođenja r iz R pripisana je određena verovatnoća p_r, takva da za svaki neterminal $A \in N$ važi:

$$\sum_{\alpha} P(A \to \alpha) = 1$$

To znači da je zbir verovatnoća pravila izvođenja iz istog neterminala uvek 1. Oznaka $P(A \to \alpha)$ predstavlja verovatnoću izvođenja pravilom $A \to \alpha$.

Primetimo da ni jedna verovatnoća iz P nije jednaka 0, jer pravilo čija je verovatnoća upotrebe 0, jednostavno izostavljamo iz gramatike.

Kako je SKSG u osnovi kontekstno slobodna gramatika, sve osobine i definicije koje smo naveli u poglavlju 2, važe i ovde. Potrebno je samo dopuniti ih verovatnosnim modelom.

Definicija 3.2: Neka je $G = (\Sigma, N, R, S, P)$ SKSG i $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ izvođenje d, takvo da se $x = \alpha_n$ sastoji samo od terminala azbuke Σ. Tada se *verovatnoća izvođenja d* računa kao

$$P(S \Rightarrow_d x) = P(S \Rightarrow \alpha_1) \cdot P(\alpha_1 \Rightarrow \alpha_2) \cdot \ldots \cdot P(\alpha_{n-1} \Rightarrow \alpha_n),$$

gde je $P(\alpha_i \Rightarrow \alpha_{i+1})$ jednaka verovatnoći primene pravila kojim ćemo iz α_i dobiti α_{i+1}.

Već smo pomenuli da se u bioinformatici neretko koriste višeznačne gramatike, tako može postojati više različitih izvođenja niske *x*. Sada možemo odrediti i koje izvođenje je najverovatnije (ima najveću verovatnoću izvođenja), ali i verovatnoću generisanja niske *x* gde se pod verovatnoćom generisanja niske *x* smatra suma verovatnoća svih mogućih izvođenja d_i niske *x*.

Definicija 3.3: *Verovatnoća generisanja niske x* stohastičkom kontekst slobodnom gramatikom *G* je

$$P(x|G)=\sum_{d_i} P(S \Rightarrow_{d_i} x)$$

gde su d_i sva izvođenja niske *x* pravilima gramatike *G*.

Ukoliko je gramatika jednoznačna, postojaće samo jedno izvođenje niske *x*, pa je izračunavanje verovatnoće izvođenja niske *x* trivijalno.

Iz svega navedenog, prirodno sledi da je jezik generisan SKSG *G* skup svih niski koje se mogu generisati gramatikom *G* sa verovatnoćom većom od nule.

Definicija 3.4: Neka je $G = (\Sigma, N, R, S, P)$ SKSG. *Jezik generisan gramatikom G* je

$$L = \{x \mid P(x \mid G) > 0\}.$$

Možemo zaključiti da za svaku SKSG G_1 postoji KSG G_2, takva da važi: $L(G_1) = L(G_2)$, odnosno, SKS gramatike i KS gramatike mogu generisati isti jezik. Ako imamo SKSG G_1 jednostavno možemo izostaviti verovatnosni deo iz svakog pravila izvođenja i dobićemo KSG, i obrnuto, ako imamo KSG G_2 dodavanjem verovatnosne vrednosti svakom pravilu izvođenja, dobićemo SKSG koja odgovara istom jeziku. Prikazaćemo to na primeru iz prethodnog poglavlja.

Primer 2.2.1: (*nastavak*) Jednostavnim dodavanjem verovatnoća svakom pravilu ovog primera KSG gramatike, dobićemo SKSG. Vratićemo se na primer koji nije u normalnoj formi Čomskog, radi jednostavnijeg zapisa:

$$\begin{aligned} S &\rightarrow S_1 g \quad (1) \\ S_1 &\rightarrow cS_1 \quad (0{,}8) \\ &\mid c \quad (0{,}2) \end{aligned}$$

Vrednosti u zagradama predstavljaju verovatnoću izvođenja pravilom pored kojeg su dopisane. Gore predstavljena gramatika jeste primer SKSG. Pogledajmo kako se računa verovatnoća izvođenja niske *ccccg*. Izvođenje je $S \Rightarrow S_1 g \Rightarrow cS_1 g \Rightarrow ccS_1 g \Rightarrow cccS_1 g \Rightarrow ccccg$. Dakle, prema formuli iz definicije 3.2, verovatnoća ovog izvođenja će biti

$$P(S \Rightarrow_d x) = 1 \cdot 0{,}8 \cdot 0{,}8 \cdot 0{,}8 \cdot 0{,}2 = 0{,}1024 .$$

Kako je ova gramatika jednoznačna, ovo je jedino pravilo izvođenja niske *x,* pa je i verovatnoća izvođenja niske *x* jednaka 0,1024.

Kada bismo računali verovatnoću generisanja niske *ccg,* gde je izvođenje $S \Rightarrow S_1 g \Rightarrow cS_1 g \Rightarrow ccg$, $P(S \Rightarrow_d x) = 1 \cdot 0{,}8 \cdot 0{,}2 = 0{,}16$, primećujemo da je verovatnoća izvođenja *ccccg* manja nego verovatnoća izvođenja *ccg.* Očigledno je da je verovatnoća manja što je niska duža.

3.2 Normalne forme SKS gramatika

Kao što smo već napomenuli, SKSG je nadograđena KSG, pa tako sve osobine koje smo u poglavlju 2 naveli za KSG, važiće i za SKSG. Ovo se odnosi i na normalnu formu Čomskog (NFČ) iz tačke 2.3, stoga nećemo navoditi posebne definicije normalne forme za SKSG. Navešćemo samo jedan jednostavan primer određivanja verovatnoća pri formiranju novih pravila izvođenja, koja zadovoljavaju uslove NFČ. Svaka SKSG G se može prevesti u NFČ, a da se ne izgube verovatnoće definisane za $L(G)$.

Primer 3.2.1: Neka je dato pravilo r sa verovatnoćom p_r

$$r: \quad A \rightarrow BCD \qquad p_r$$

Tada se od pravila r mogu formirati dva nova pravila r_1 i r_2 koja su u NFČ, bez narušavanja verovatnoće i jezika koji je gramatika sa pravilom r generisala:

$$r_1: \quad A \rightarrow BX \qquad p_{r1} = p_r$$
$$r_2: \quad X \rightarrow CD \qquad p_{r2} = 1$$

Simbol X je novi neterminal, koji je sa leve strane samo jednog pravila, pa je uslov da je zbir svih verovatnoća izvođenja iz istog neterminala jednak 1 zadovoljen.

4 Algoritmi za rad sa SKSG

Definisanje SKSG je samo prvi korak u određivanju verovatnosnih modela za problem analiziranja sekvenci. Potrebni su nam i algoritmi koji će, pomoću SKSG, moći da:

1) izračunaju najverovatnijeg stabla izvođenja niske sa parametrizacijom stohastičke gramatike
2) izračunaju verovatnoću izvođenja sekvence zadatom SKSG
3) ako je dat skup sekvenci (skup za obučavanje), odrede optimalne verovatnosne parametre za neparametrizovanu SKSG (problem obučavanja).

Predstavićemo tri osnovna algoritma za rad sa SKSG, koji pomoću tehnika dinamičkog programiranja daju rešenja za tri gore navedena problema.

4.1 Algoritam "*Unutra - Spolja*"

Algoritam "*Unutra - Spolja*" (engl. *Inside-Outside*) [2][3] koristi se za SKSG u normalnoj formi Čomskog. Služi za izračunavanje verovatnoće izvođenja neke niske datom SKSG i popravljanje parametara gramatike skupom za obučavanje. Sastoji se iz dva dela – *Unutra* i *Spolja.* Predstavićemo ih pojedinačno.

4.1.1 Algoritam "*Unutra*"

Algoritam "*Unutra*" se koristi za određivanje verovatnoće generisanja neke niske datom SKSG u normalnoj formi Čomskog.

Pretpostavimo da je data SKSG $G = (\Sigma, N, R, S, P)$ i niska x generisana gramatikom G, takva da je $x = x_1 x_2 \dots x_n$ dužine n, gde su x_i simboli iz Σ. Neka je broj neterminala skupa N jednak m, a početni simbol $S = N_1$. Da bismo primenili *Unutra* algoritam nad G, smatraćemo da je G u normalnoj formi Čomskog. Stoga su sva pravila ove gramatike oblika $N_u \rightarrow N_v N_w$ ili $N_u \rightarrow a$, gde su N_u, N_v i N_w neterminali iz N, a a terminal iz Σ. Ideja *Unutra* algoritma je da za svaki neterminal N_u iz N i svaku podsekvencu $x_i \dots x_j$ niske x izračuna verovatnoću $\alpha[i, j, u]$ izvođenja

podsekvence $x_i, \ldots, x_j$ iz neterminala N_u. Time se dinamičkim programiranjem kreira matrica $n \times n \times m$, čiji su elementi $\alpha[i, j, u]$, za svako $u \in \{1, \ldots, m\}$ i svako $i, j \in \{1, \ldots, n\}$, gde je $i \leq j$. Formalno gledano,

$$\alpha[i, j, u] = \sum_{d_i} P(N_u \Rightarrow_{d_i} x_i x_{i+1} \ldots x_j)$$

gde su d_i sva izvođenja niske $x_i \ldots x_j$ iz neterminala N_u.

Izračunavanje elemenata matrice počinje "unutra", od podsekvenci dužine 1, tj. kada je $i = j$. U tom slučaju nas interesuju samo neterminali koji direktno izvode terminale, tj. verovatnoće pravila oblika $N_u \rightarrow x_i$ i postavljamo da je $\alpha[i, i, u] = P(N_u \rightarrow x_i)$ za svako i. Nastavlja se "ka spolja", podsekvencama dužine 2, i tako dalje, dok se ne stigne do sekvence dužine n. U svakom rekurzivnom koraku, računa se verovatnoća da je podsekvenca $x_i \ldots x_j$ izvedena iz neterminala N_u. Kako je ulazna gramatika u normalnoj formi Čomskog, ako je $N_u \rightarrow N_v N_w$, to znači da se iz N_v i N_w redom izvode podsekvence $x_i \ldots x_k$ i $x_{k+1} \ldots x_j$. Lako se vidi da je rezultat koji tražimo element $\alpha[1, n, 1]$. U daljem tekstu prikazan je pseudo-algoritam:

Algoritam 1: (*Unutra*)

Ulaz: SKSG G i niska x
Izlaz: verovatnoća generisanja niske x gramatikom G

begin
// inicijalizacija
n = dužina niske x
m = broj neterminala gramatike G
$\alpha[i, j, u] = 0$ za sve $i, j = 1..n$ i $u = 1..m$

for i =1 **to** n **do**
    for u =1 **to** m **do**
        if postoji pravilo $N_u \rightarrow x_i$ u gramatici G
            $\alpha[i, i, u] = P(N_u \rightarrow x_i)$
        end
    end
end

// iteracija
for i =n-1 **to** 1 **do**
    for j =i+1 **to** n **do**
        for u =1 **to** m **do**

$$\alpha[i, j, u] = \sum_{y=1}^{m} \sum_{z=1}^{m} \sum_{k=i}^{j-1} \alpha[i, k, y] \alpha[k+1, j, z] P(N_u \rightarrow N_y N_z)$$

        end
    end
end

//vraćamo rezultat algoritma
return α[1, *n*, 1]

end

Prostorna složenost ovog algoritma određena je veličinom matrice α. Za nisku dužine *n* i gramatiku sa *m* neterminala, prostorna složenost je $O(n^2m)$. Vremenska složenost najviše zavisi od parametara iterativnih petlji. Ako je broj pravila oblika $N_u \rightarrow N_v N_w$ jednak *k*, na osnovu ugnježdenih iterativnih petlji, vidimo da je vremenska složenost $O(n^3k)$. Sa *m* neterminala najviše može biti $m \cdot m \cdot m = m^3$ pravila pomenutog oblika, jer se N_u može izabrati na *m* načina, kao i N_v i N_w . Dakle, u najgorem slučaju, vremenska složenost algoritma *Unutra* je $O(n^3m^3)$.

4.1.2 Algoritam "*Spolja*"

Algoritam "*Spolja*" (engleski *Outside*) je drugi deo algoritma *Unutra-Spolja,* koji se zajedno koriste za popravljanje parametara SKSG skupovima za obučavanje. On takođe računa verovatnoću generisanja neke niske datom SKSG u normalnoj formi Čomskog, ali u drugom smeru - "spolja ka unutra". Prvo se određuje verovatnoća za celu nisku *x*, a zatim za svaku manju podsekvencu niske *x*. Za svako *i, j* i *v* izračunava se verovatnoća $\beta[i, j, v]$ stabla izvođenja sekvence $x = x_1 x_2 \ldots x_n$, gde je startni simbol *S* u korenu stabla i isključena su sva moguća podstabla izvođenja podsekvence $x_i, \ldots, x_j$ sa korenom N_v (vidi sliku 4.1.2) . Za računanje vrednosti $\beta[i, j, v]$ koriste se vrednosti $\alpha[i, j, u]$ iz *Unutra* algoritma, kao i tehnike dinamičkog programiranja. Potrebno je popuniti matricu dimenzija $n \times n \times m$. Za algoritam *Spolja* pseudo-algoritam možemo predstaviti na sledeći način:

Algoritam 2: (*Spolja*)

Ulaz: SKSG *G* i niska *x*
Izlaz: verovatnoća generisanja niske *x* gramatikom *G*

begin
// inicijalizacija
n = dužina niske *x*
m = broj neterminala gramatike *G*
$\beta[1, n, 1] = 1$
for *v* =2 **to** *m* **do**
    $\beta[1, n, v] = 0$
end

// iteracija
for *s* =*n*-1 **to** 1 **do**
    for *j* =*s* **to** *n* **do**

for $v = 1$ **to** m **do**

$$i = j - s + 1$$

$$\beta[i, j, v] = \sum_{y,z} \sum_{k=1}^{i-1} \alpha[k, i-1, z]\beta[k, j, y]P(N_y \rightarrow N_z N_v)$$

$$+ \sum_{y,z} \sum_{k=j+1}^{n} \alpha[j+1, k, z]\beta[i, k, y]P(N_y \rightarrow N_v N_z)$$

end

end

end

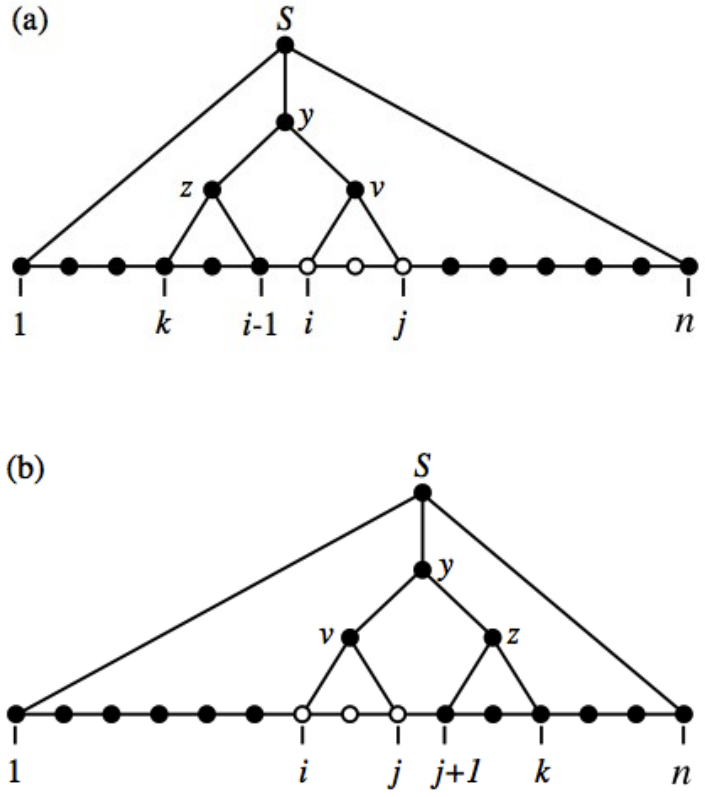
//vraćamo rezultat algoritma

return $P(x|G) = \sum_{v=1}^{m} \beta[i, i, v]P(N_v \rightarrow x_i)$ za svako i

end

Radi lakšeg razumevanja, na slici 4.1.2 (a) i 4.1.2 (b) su predstavljeni, redom, prvi i drugi deo iteracije algoritma *Spolja.*

Spolja algoritam ima istu vremensku i prostornu složenost kao i alogritam *Unutra.*



Slika 4.1.2 Prvi i drugi deo iteracije algoritama *Spolja.* Slike su preuzete iz [2].

4.1.3 Popravljanje parametara SKSG pomoću skupa za obučavanje

Algoritmom *Unutra-Spolja* možemo poboljšati verovatnosne parametre već zadate SKSG, pomoću skupa za obučavanje. Koristi se vrednosti matrica α i β, izračunate u algoritmima *Unutra* i *Spolja* i tehnika maksimizacije očekivanja, kako bi se došlo do bolje procene verovatnoća. Očekivani broj upotrebe neterminala N_v pri izvođenju niske x dužine n gramatikom G je

$$b(N_v)=\frac{1}{P(x|G)}\sum_{i=1}^{n}\sum_{j=i}^{n}\alpha[i,j,v]\beta[i,j,v].$$

Za nalaženje broja pojavljivanja neterminala N_v iz koga se dalje izvodi pravilom $N_v \rightarrow N_u N_w$, koristi se formula

$$b(N_v\rightarrow N_u N_w)=\frac{1}{P(x|G)}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=i}^{j-1}\beta[i,j,v]\alpha[i,k,u]\alpha[k+1,j,w]P(N_v\rightarrow N_u N_w).$$

Tada možemo sračunati nove verovatnoće u SKSG G za pravila izvođenja $N_v \rightarrow N_u N_w$ na sledeći način:

$$\begin{aligned}\hat{P}(N_v\rightarrow N_u N_w) &= \frac{b(N_v\rightarrow N_u N_w)}{b(N_v)}\\ &= \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=i}^{j-1}\beta[i,j,v]\alpha[i,k,u]\alpha[k+1,j,w]P(N_v\rightarrow N_u N_w)}{\sum_{i=1}^{n}\sum_{j=i}^{n}\alpha[i,j,v]\beta[i,j,v]}.\end{aligned}$$

Analogno, za pravila izvođenja oblika $N_v \rightarrow a$, verovatnoću ćemo proceniti formulom:

$$\hat{P}(N_v\rightarrow a) = \frac{b(N_v\rightarrow a)}{b(N_v)} = \frac{\sum_{i|x_i=a}\beta[i,i,v]P(N_v\rightarrow a)}{\sum_{i=1}^{n}\sum_{j=i}^{n}\alpha[i,j,v]\beta[i,j,v]}.$$

Za skupove za obučavanje sa više od jedne niske, podešavanje parametara se vrši jednostavnim sabiranjem broja pojavljivanja neterminala i primena pravila u svakoj od niski, formulama prikazanim u ovom poglavlju.

4.2 CYK algoritam

Naziv CYK dolazi od inicijala tvoraca ovog algoritma, Cocke-Younger-Kasami.

Algoritam CYK [2][3] se koristi za određivanje najverovatnijeg stabla izvođenja niske x.

Sastoji se iz dva dela. Jedan deo izračunava matricu brojeva $\gamma(i, j, v)$, što dovodi do vrednosti $\log P(x, \lambda \mid G)$, gde je G zadata SKSG, a λ najverovatnije stablo izvođenja niske x gramatikom G. Logaritam se koristi da bi se prevazišao problem malih vrednosti koje se javljaju pri računanju verovatnoća. Prvi deo CYK algoritma je, zapravo, vrlo sličan algoritmu *Spolja*, iz prethodnog poglavlja. Razlika je samo u tome što se umesto suma koriste operacije maksimuma, jer nas ne interesuje ukupna verovatnoća, nego samo maksimalna.

Drugi deo pamti vrednosti $\tau(i, j, v)$ koje predstavljaju trojke brojeva (y, z, k) potrebnih za rekonstrukciju najverovatnijeg stabla izvođenja, upotrebom vrednosti iz matrice γ, kreirane dinamičkim programiranjem. Koristićemo iste oznake kao i kod *Unutra–Spolja* algoritma. Pseudo-algoritam za popunjavanje trodimenzionalne matrice γ je:

Algoritam 3.1: (CYK)

Ulaz: SKSG G i niska x
Izlaz: najverovatnije stablo izvođenja λ niske x gramatikom G

begin
// inicijalizacija
n = dužina niske x
m = broj neterminala gramatike G
for i =1 **to** n **do**
    for v =1 **to** m **do**
        $\gamma(i, i, v) = \log P(N_v \rightarrow x_i)$
        $\tau(i, i, v) = (0, 0, 0)$
    end
end

//iteracija
for $i = n - 1$ **to** 1 **do**
    for $j = i + 1$ **to** n **do**
        for v =1 **to** m **do**
            $\gamma(i, j, v) = \max_{y,z} \max_{k=i \ldots j-1}$
                $\{\gamma(i, k, y) + \gamma(k+1, j, z) + \log P(N_v \rightarrow N_y N_z)\}$
            $\tau(i, j, v) = \operatorname{argmax}_{(y, z, k),\, k=i \ldots j-1}$
                $\{\gamma(i, k, y) + \gamma(k+1, j, z) + \log P(N_v \rightarrow N_y N_z)\}$
        end
    end
end

//rezultat algoritma
return $\log P(x, \lambda \mid G) = \gamma(1, n, 1)$

end

Trojke brojeva (y, z, k) za vrednosti $\tau(i, j, v)$ prikazuju da je najverovatnije izvođenje podsekvence $x_i, \ldots, x_j$ iz neterminala N_v izvođenjem $x_i, \ldots, x_k$ iz N_y i $x_{k+1}, \ldots, x_j$ iz N_z, gde pravilo $N_v \rightarrow N_y N_z$ pripada gramatici G.

Drugi deo CYK algoritma, odnosi se na konstruisanje najverovatnijeg stabla izvođenja. Polazeći od vrednosti $\tau(1, n, 1)$, možemo pomoću računarkse strukture "stek", koja radi na principu "prvi unutra, poslednji napolje" (engl. *first in-last out*) i vrednosti trojki τ, odrediti najverovatnije stablo izvođenja niske x. Algoritam za to je:

Algoritam 3.2: (CYK *putanja*)

begin
//inicijalizacija
n = dužina niske x
stavi na stek trojku (1, n, 1)

//iteracija
skini sa vrha steka (i, j, v)
(y, z, k) = $\tau(i, j, v)$
if $\tau(i, j, v) = (0, 0, 0)$
    nadoveži xi kao potomak od v
else
    nadoveži y, z na stablo izvođenja kao potomke od v
    stavi na stek (k +1, j, z)
    stavi na stek (i, k, y)

end

Dakle, u algoritmu 3.2, prvo na stek stavljamo trojku (1, n, 1), a to je trojka koja se odnosi na koren stabla izvođenja koje tražimo, jer predstavlja izvođenje cele niske x dužine n iz početnog neterminala. U sledećem koraku ulazimo u *else* deo algoritma, koji kao potomke početnog simbola dodaje u stablo dva neterminala određena algoritmom 3.1. Dodavanjem dva neterminala u stablo, na stek stavljamo dve nove odgovarajuće trojke. Iteracija se nastavlja dok se sve trojke ne obrade i grane stabla ne završe. Grana stabla se završava kada se u steku naiđe na trojku (0, 0, 0), što znači da se radi o slučaju kada je $i = j$, a to se odnosi na pravilo gramatike koje izvodi terminal.

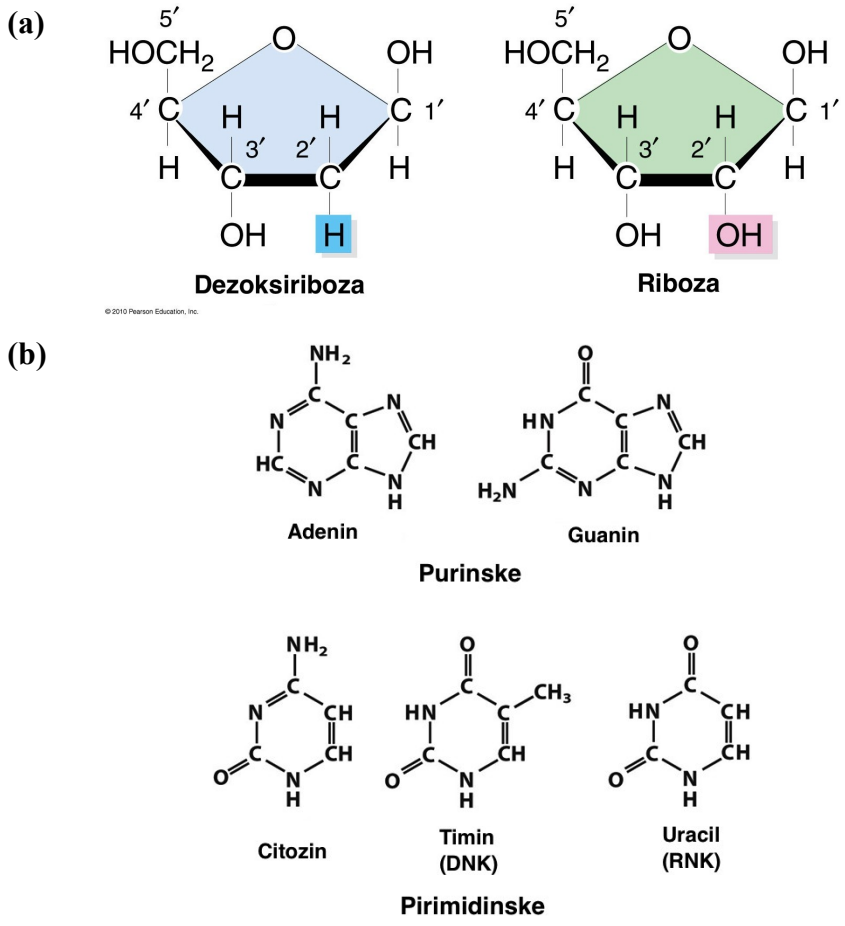
CYK se može koristiti za aproksimaciju *Unutra-Spolja* obučavanja. Računanjem najverovatnijih stabala izvođenja za trening sekvence CYK algoritmom, možemo prebrojati pravila gramatike koja se koriste u svakom stablu i tim frekvencijama odrediti verovatnoće za SKSG.

Kako ovaj algoritam funkcioniše sa gramatikama u normalnoj formi Čomskog, njegova složenost je $O(n^3m^3)$, gde je n dužina ulazne niske, a m broj različitih neterminala. Za neke ograničene *SKSG* postoje druge verzije algoritama *CYK* i *Unutra-Spolja* za koje ulazna gramatika ne mora biti u normalnoj formi Čomskog i čija je vremenska složenost $O(n^3m)$. Ta složenost je opet velika, ali u slučajevima gramatika sa puno neterminala, daje osetnu razliku.

5 Ribonukleinska kiselina (RNK)

Mnogima je pojam DNK poznatija od pojma RNK. Iz tog razloga, pomenućemo najpre šta je DNK i uporediti ga sa RNK.

Dezoksiribonukleinska kiselina (skraćeno DNK) ne samo da je nosač genetskog materijala kod svih živih bića, već i sadrži instrukcije potrebne za razvoj svakog organizma. Pored RNK i proteina, DNK je jedan od glavnih makromolekula koji učestvuju u funkcionisanju svake ćelije. *Gen* je deo DNK sekvence koji sadrži informacije potrebne za sintezu jednog molekula RNK ili jednog proteina. On je "zadužen" za prenos naslednog materijala na potomke.



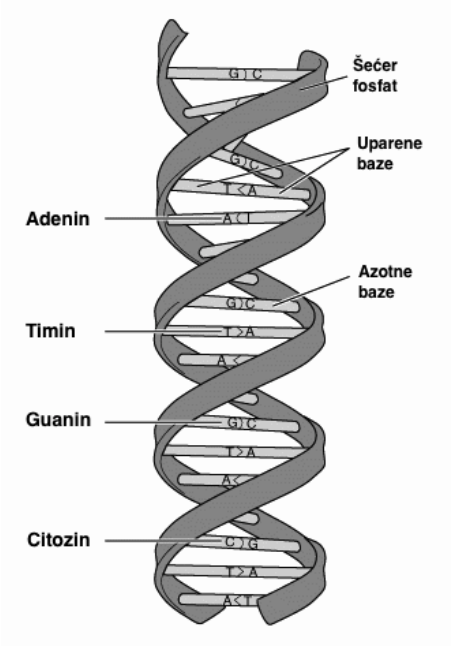
Slika 5.1 Osnovne komponente nukleotida i građa nukleinskih kiselina. (a) Šećeri koji ulaze u sastav DNK i RNK. (b) Azotne baze *A,G, C, T* i *U*. Prva slika je preuzeta iz [7], a druga iz [8] i prevedene su na srpski.

Nukleinske kiseline su biološki makromolekuli, polimerne (gr. πολύς (polus)

– mnogo, više + μέρος (meros) – deo) prirode. *Nukleotidi*, osnovne jedinice koje se ponavljaju, se sastoje od (*i*) šećera *pentoze* (*riboze* ili *dezoksiriboze*), (*ii*) azotnih baza (*pirimidina* - *timina* (***T***), *citozina* (***C***) i *uracila* (***U***) i *purina* - *adenina* (***A***) i *guanina* (***G***)) i (*iii*) *fosforne kiseline* (slika 5.1).

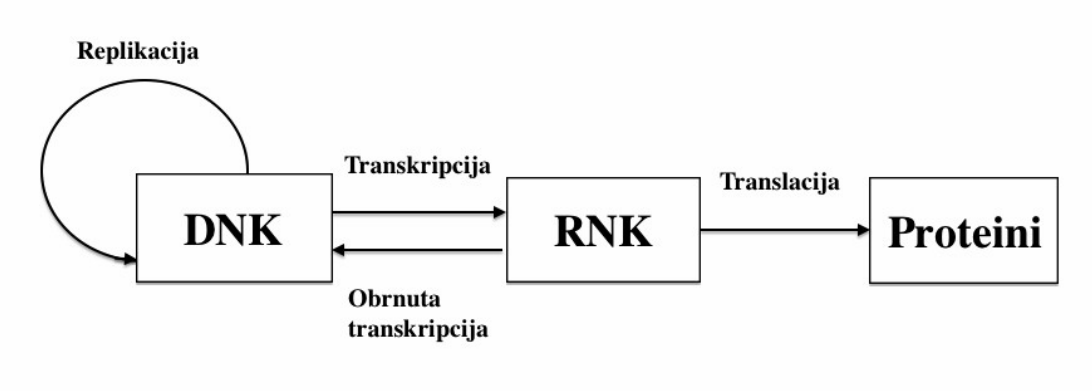
Dezoksiribonukleinska kiselina (DNK) je izgrađena od nuklotida u čiji sastav ulaze sećer *dezoksiriboza* i azotne baze *A*, *T*, *C* i *G*. Ribonukleinska kiselina (RNA) je izgrađena od nukleotida u čiji sastav ulaze šećer *riboza* i nuklotidi *A*, *U*, *C* i *G*. Nukleotidi su međusono povezani (*polimerizovani*) preko fosforne kiseline. Azotne baze se mogu međusobno povezivati nekovalentnim, vodoničnim vezama, tako da se purin *A* vezuje sa pirimidinima *T* ili *U*, a purin *G* sa pirimidinom *C*. Ovo uparivanje poznato je kao *Watson-Crick uparivanje baza.*

Molekul DNK se najčešće sastoji od dva međusobno, preko azotnih baza, uparena polinukleotidna lanca u obliku *dvostruke helikoide* (zavojnice) (slika 5.2). Broj nukleotida po lancu je više miliona (npr. pojedinačni DNK lanac ljudskog hromozoma 1 se sastoji od 200.000.000 nukleotida).



Slika 5.2 Dvostruka zavojnica DNK. Slika je preuzeta sa [9] i prevedena na srpski.

Biološka informacija proističe iz redoseleda nukleotida (tj. primarne strukture) polinukleotidnog lanca (slika 5.3). Tako, npr. pojedini segmenti DNK, označeni kao geni, kodiraju, odnosno, određuju redosled amino kiselina u polipeptidnom lancu proteina, koji su glavni izvršni biomolekuli (npr. enzimi). Svaka tri uzastopna nukleotida iz *informaciona RNK* (iRNK) kodiraju jednu od dvadeset *amino-kiselina* koje grade proteine i određuju njihovu ulogu (funkciju). Grupa od tri nukelotida naziva se *kodon.* Na osnovu centralne dogme molekularne biologije (više o tome može se pročitati u [10]), mnogi misle da je uloga RNK samo u prenošenju informacija sa DNK do proteina. Međutim, noviji eksperimenti potvrđuju da RNK ima i druge uloge. Takođe, RNK hipoteza govori o tome da se u procesu evolucije prvo pojavila RNK, a zatim DNK i proteini. Više o tome može se pronaći u [11].



Slika 5.3 Centralna dogma molekularne biologije. DNK se umnožava u procesu **replikacije**. Informacija, tj. redosled nukleotida, u segmentima DNK (genima) koji kodiraju proteine, se prepisuje na RNK molekul u procesu **transkripcije**. Dalje se informacija sadržana u RNK prevodi u redosled aminokiselina u proteinskom lancu u procesu **translacije**. Slika je preuzeta iz [12] i prevedena na srpski.

Pored već navedene, literatura koja je korišćena za ovo poglavlje je i [3][6].

5.1 Strukture RNK molekula

U naredne tri tačke predstavićemo tri vrste struktura u kojima se javlja RNK. Nas u ovom radu prvenstveno interesuje sekundarna struktura RNK.

5.1.1 Primarna struktura RNK

Primarna struktura RNK je linearni niz sastavljen od nukleotida *A, U, C* i *G*. To je niska slova nad azbukom $\Sigma=\{A, U, C, G\}$. Komparativna analiza sekvenci nukleotida je osnovni preduslov za klasifikaciju RNK sekvenci i predikciju njihovih funkcija i struktura. Na slici 5.1.1 prikazan je primer primarne strukture RNK.

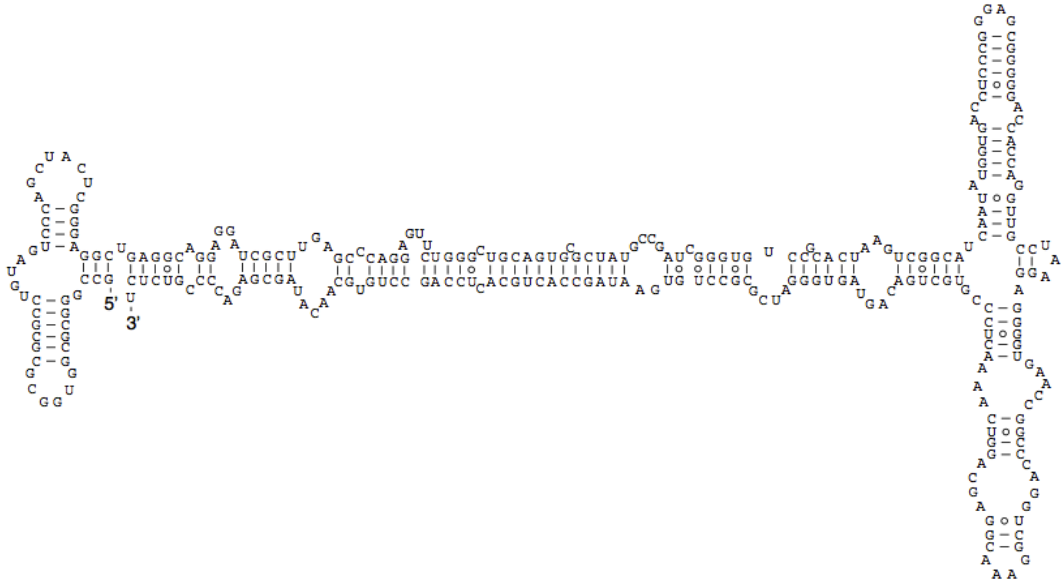
5.1.2 Sekundarna struktura RNK

Vezivanje azotnih baza rezultuje formiranjem karakterističnih motiva i oblika RNK molekula, što se može prikazati dvodimenzionalnom skicom. Ona predstavlja

sekundarnu strukturu RNK. Motivi koji se javljaju uparivanjem baza su razne vrste petlji i grananja. Obično se stvaraju vodonične veze između komplementarnih baza *A-U* i *G-C* (“*Watson Crick”* uparivanje baza), ali to ne važi u svim slučajevima. Veza između *G* i *U* je takođe energetski povoljna i često se sreće u molekulima RNK. Primer sekundarne strukture prikazan je na slici 5.1.2.1.

CCTCGTTCTCTTGCAGAACTTTGATTTTAA
CGAACTTAAATAAAAGCCCTGTTGTTTAGC
GTATTGTTGCACTTGTCTGGTGGGATTGT
GCATTAATTTGCCTGCTCATCTAGGCAGTG
GACATATGCTCAACACTGGGTATAATTCTA
ATTGAATACTATTTTTCAGTTAGAGCGTCG
TGTCTCTTGTACGTCTCGGTCACAATACAC
GGTTTCGTCCGGTGCGTGGCAATTCGGGGC

Slika 5.1.1 Deo primarne strukture RNK sekvence izolata MERS CoV virusa. Sekvenca je preuzeta sa [13].



Slika 5.1.2.1 *S*ekundarne strukture RNK *SRP-RNA* psa*, Canis Familiaris* [2]

Uparivanjem baza u jednolančanim podsekvencama RNK nastaju takozvane *petlje.* Na slici 5.1.2.2 su predstavljene vrste petlji koje se javljaju kod sekundarne strukture RNK. Petlja na kraju *uparivanja* naziva se *petlja ukosnica. Heliks* je petlja jednostavnog uparivanja, u čijem nizu nema neuparenih baza. Deo uparivanja gde se javlja niz neuparenih baza sa jedne strane uparene jednolančane sekvence naziva se *ispupčena petlja,* a ukoliko se sa dve strane uparene sekvence nađu nizovi neuparenih baza, onda je to *unutrašnja petlja. Višestruko-razgranate petlje* su one iz kojih proizilazi tri ili više grana niza uparivanja. Petlje u RNK sekvencama su uglavnom ugnježđene (mogu se nalaziti jedna unutar druge, ali ne i da se delimično preklapaju).

Slika 5.1.2.2 Vrste petlji koje se javljaju kod uparivanja u sekundarnim strukturama RNK. Tačke predstavljaju nukleotide, a poprečne crte veze između njih (slika je preuzeta iz [6] i prevedena na srpski).

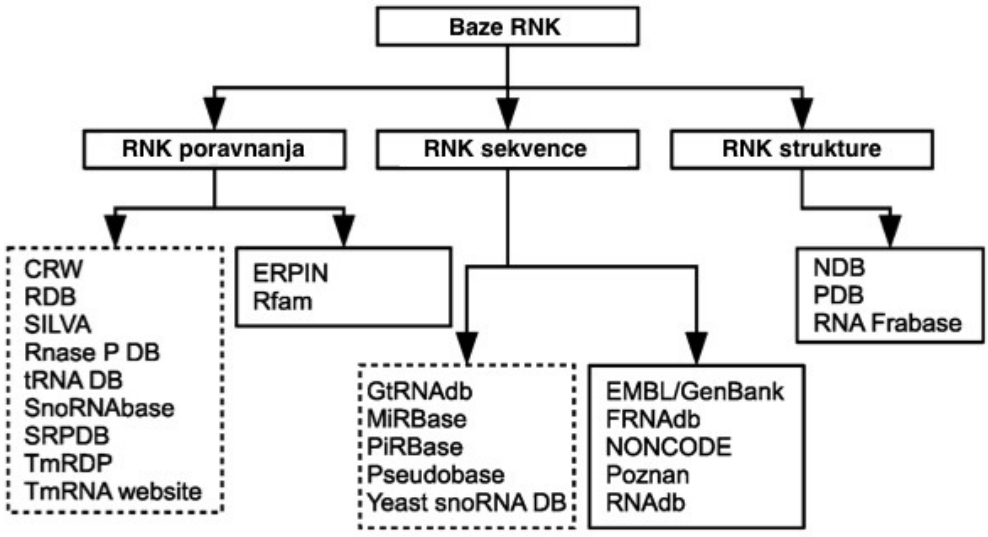
5.1.3 Tercijarna struktura RNK

Tercijarna struktura RNK je prostorna (trodimenzionalna) struktura. Ona prikazuje vodonične veze između baza, koje se javljaju pored onih koje su prikazane u sekundarnoj strukturi. U realnom svetu RNK molekuli se uvek javljaju u trodimenzionalnoj (tercijarnoj) strukturi. Međutim, određivanje ove strukture eksperimentalnim metodama, kao što su kristalografija i difrakcija *X-zracima*, su skupi i dugotrajni procesi. Iz tih razloga javlja se potreba za računarskim metodama, koje mogu makar približno da odrede kako će izgledati prava struktura RNK sekvence. Kod računarskih metoda se korisnim pokazuju sekundarne strukture. One su, zapravo, apstrakcija pravog modela, daju manje informacija o realnoj RNK strukturi, ali ipak prikazuju više nego primarne strukture. Mnoge podsekvence sekundarnih sekvenci su termodinamički vrlo stabilne, pa će se javljati i u tercijarnoj strukturi. Zbog toga je vrlo efikasno prvo odrediti sekundarnu strukturu i to iskoristiti kao osnovu za predviđanje tercijarne strukture RNK.

5.2 Baze poznatih RNK sekvenci

Razvoj modernih baza RNK sekvenci je u mnogome uticao na poznavanje RNK sa biološke tačke gledišta, kao i na unapređivanje infomacionih tehnologija za rad sa njima. Prve baze RNK sekvenci fokusirale su se na dugo poznate klase RNK, kao što su tRNK i rRNK. Jedna od tih baza, "*Sprinzl tRNA",* objavljena je u naučnom časopisu [14] i sadržala je oko 700 sekvenci. Današnje baze, kao što je, na primer, poznata *Rfam* baza podataka [15], sadrže i više od milion RNK sekvenci. Problem skladištenja, pretrage i izmene baza sa tom količinom podataka, rešen je pomoću relacionih baza dostupnih na vebu, odnosno internet tehnologiji. Relacione baze podataka omogućavaju da se iz jednog skladišta dinamički generiše više različitih izlaznih skupova podataka. Oraganizacija baze dopušta primenu kompleksnih bulovskih upita za dobijanje željenih informacija, što znatno olakšava manipulisanje sekvencama.

Potreba za velikim brojem baza RNK sekvenci javlja se iz više razloga. Jedan od njih jeste da različite klase RNK imaju različite osobine koje ih definišu i na koje treba staviti akcenat u bazi. Sa druge strane, potrebu da se formira više baza stvara i obim skupa neophodnih sekvenci kojim baratamo u nekom istraživanju. Nekad su nam od značaja baze sa više sličnih (*specijalizovane baze*) ili različitih (*generalizovane baze*) sekvenci, dok su u drugom slučaju potrebne baze koje sadrže samo jedan element. Jedna od osnovnih razlika među bazama RNK jeste da li su podaci iz baze otkriveni i potvrđeni eksperimentima, ili su pak dobijeni automatski, računskim putem. Prvi način se uglavnom koristi kod *specijalizovanih* baza. RNK baze mogu se klasifikovati kao na slici 5.2.



Slika 5.2 Klasifikacija RNK baza, sa navedenim nekim primerima baza iz svake klase. Isprekidanom linijom su označene *klasno-specijalizovane* baze, a punom linijom su uokvirene *generalizovane* baze [3].

Spisak svih trenutno poznatih baza RNK sekvenci može se naći na internet

stranici [16]. Sada ćemo pomenuti samo neke od najznačajnijih baza RNK iz svake gore pomenute klase.

Prvenstvena namena baza iz klase RNK sekvenci (slika 5.2) jeste skladištenje pojedinačnih sekvenci. Baze sekvenci uglavnom ne sadrže detaljnije informacije svake pojedinačno nekodirajuće RNK (*ncRNA*), njihove međusobne odnose ili nomenklaturu. Nedostatak takvih informacija čini te baze nepoželjnim kod određenih korisnika. Ipak, ovakav način čuvanja RNK pojednostavljuje dodavanje novih otkrivenih sekvenci, rad osoba koje unapređuju bazu, kao i onih koji su samo krajnji korisnici.

Sa druge strane, baze RNK poravnanja bolje opisuju raznovrsnost nekodirajućih RNK, familije sekvenci na osnovu njihovih očekivanih zajedničkih predaka koristeći sličnosti u strukturama i sekvencama. Ove sekvence se mogu poravnati, iz čega proizilazi procena raznolikosti svakog nukleotida. Mana ovakvog pristupa jeste relativno spora obrada podataka.

Trenutno najiscrpnija generalizovana baza sekvenci RNK je baza ***Noncode***, formirana 2005. godine [17]. Ona se fokusira na predstavljanje pojedinačnih sekvenci sa relevantnim informacijama. Za razliku od mnogih drugih baza, u bazi *Noncode*, većina sekvenci dobijena je eksperimentalnim putem, čak više od 80% podataka, što je jako veliki broj, uzimajući u obzir da je 2007. godine ova baza sadržala preko 200.000 sekvenci. *Noncode* bazi je priključen bulovski mehanizam za pretraživanje, koji omogućava korisniku da postavlja kompleksne upite i efikasno dođe do željenih podataka.

"***miRBase***" je specijalizovana baza sekvenci miRNK (*microRNA*) [18]. Ona je trenutno najkompletnija baza miRNK, raznovrsnih eukariotskih RNK koji učestvuju u ekspresiji gena [19]. Osnovni cilj baze *miRBase* je prikupljanje svih poznatih, eksperimentalno proverenih miRNK sekvenci. Sastoji se iz tri dela. Prvi je *registar*, u kome se čuvaju nove informacije, koje je potrebno proveriti pre ubacivanja u bazu, kako bi tačnost podataka ostala na najvišem nivou. Drugi deo je skoncentrisan na pojedinačne miRNK sekvence i detaljnije informacije o svakoj od njih. Treći deo povezuje individualne miRNK sa drugim bazama koje se bave predikcijom mogućih miRNK. *miRBase* ne podržava bulovske upite, ali omogućava pretraživanje skupa predefinisanih kriterijuma kao što su određene vrste, lokacije genoma, ili slobodna pretraga.

Baza familija RNK, ***Rfam*** je najveća do sada poznata generalizovana baza poravnanja [15][20]. *Rfam* definiše višestruka poravnanja sekvenci i kovarijansne modele (engl. *covariance models*) [2]. Osnovni cilj *Rfam* baze je da prikupi nove članove poznatih RNK familija, posebno cele genome. Mana koja se javlja kod ovakvog pristupa konstruisanju baze je ta što će neke sekvence koje se u određenoj meri razlikuju od ostalih iz familije, biti izostavljene. Kao rešenje ovog problema, *Rfam* uvodi pojam *klanova,* koji su viša kategorija grupisanja od familija. Informacije i alati koje sadrži *Rfam* su raznovrsni. Potpuna poravnanja svih elemenata baze, kao i evolutivnih stabala obezbeđuju uvid u raznolikost i filogenetsku distribuciju svake familije.

Među prvim RNK sekvencama iz kojih su sačinjene RNK baze, bile su rRNK

(*ribosomalRNA*), shodno činjenici da se rRNK često koriste kod filogenetskih izračunavanja. Do sada najkompletnija specijalizovana baza poravnanja jeste baza ***SILVA*** [21]. Ova baza služi za proveru kvaliteta i poravnanje svih poznatih rRNK sekvenci. Pretraga podataka baze *SILVA* može se vršiti po raznim kriterijumima, kao što su, na primer, nazivi organizama ili publikacije vezane za tu sekvencu.

Iz treće klase baza RNK, nas će u ovom radu posebno zanimati baza ***RNA STRAND***, koja sadrži sekundarne strukture RNK sekvenci i njihove motive. Više o tome u poglavlju 8.1.

6 Stohastičke kontekst slobodne gramatike i sekundarne strukture RNK

Stohastičke kontekst slobodne gramatike su prvenstveno našle svoju primenu u računarskoj obradi prirodnih jezika. Međutim, kako RNK strukture i prirodni jezici imaju mnogo zajedničkih osobina, ideja da se SKSG upotrebe za modelovanje RNK sekvenci, bila je sasvim logična. Azbuka RNK sekvenci sastoji se iz četiri simbola – *A, U, C,* i *G*, koji predstavljaju nukleotide. Kod sekundarne strukture niz nukleotida zadovoljava određenu strukturnu konfiguraciju, što se može opisati pravilima gramatike. Verovatnosni model SKSG se može formirati na osnovu skupa za obučavanje sastavljenog od poznatih sekvenci RNK. Verovatnosni parametri nam pomažu da odredimo koliko je verovatno da se neka sekvenca izvede u jeziku zadate gramatike i koje je njeno najverovatnije izvođenje.

Jedan od prvih članaka u kome se povezuju SKSG i RNK sekvence jeste [22], objavljen 1994. godine, čiji je jedan od glavnih autora bio J. Sakakibara. Predstavićemo sada metodu koja je prikazana u tom radu.

6.1 Azbuka i pravila izvođenja gramatika za modelovanje RNK sekvenci

Ono što je zajedničko za sve gramatike za rad sa RNK jeste azbuka $\Sigma=\{A, U, C, G\}$ i četiri vrste pravila izvođenja. Neka je S oznaka za neterminal gramatike G i a oznaka za bilo koji terminal iz G. Za uparivanje baza koriste se pravila oblika $S \rightarrow aSa$. Na primer, za uparivanje *A*-*U,* koristiće se pravilo $S \rightarrow ASU$. Pravila $S \rightarrow aS$ i $S \rightarrow a$ opisuju neuparene baze, kao na primer $S \rightarrow CS$ i $S \rightarrow C$, za neuparene baze *C*. Za grananje koriste se pravila oblika $S \rightarrow SS$, dok $S \rightarrow S$ predstavlja deleciju kod poravnanja RNK sekvenci [2]. Nije neophodno da sa desne i leve strane pravila budu isti neterminali. Ako su S_i , S_j i S_k proizvoljni neterminali, tada npr. pravilo grananja može biti $S_i \rightarrow S_j S_k$. Pogledajmo jedan jednostavan primer kontekstno slobodne gramatike za modelovanje RNK niski:

Primer 6.1: Neka su $S, S_1, S_2 , \ldots , S_{10}$ neterminali, gde je S startni simbol, a *A, U, C* i *G* terminali. Neka je gramatika za modelovanje jednog tipa RNK sekvenci:

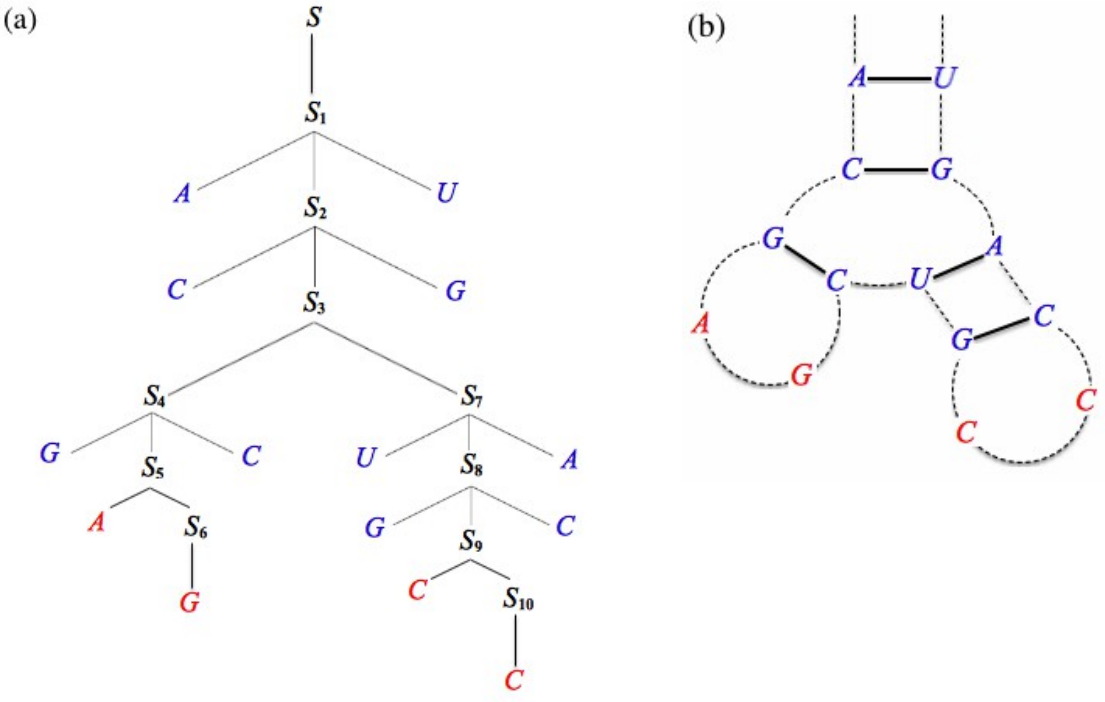
$$S \rightarrow S_1 \qquad\qquad S_6 \rightarrow G$$

$S_1 \rightarrow AS_2U$
$S_2 \rightarrow CS_3G$
$S_3 \rightarrow S_4S_7$
$S_4 \rightarrow GS_5C$
$S_5 \rightarrow AS_6$
$S_7 \rightarrow US_8A$
$S_8 \rightarrow GS_9C$
$S_9 \rightarrow CS_{10}$
$S_{10} \rightarrow C$

Očigledno je da će neterminali S_1, S_2, S_4, S_7 i S_8 proizvesti uparivanje baza, dok će se pravilima S_5, S_6, S_9 i S_{10} dobiti neuparene baze, a pravilom S_3 grananje. Izvođenje sekvence *ACGAGCUGCCCAGU* ovom gramatikom je:

$$S \Rightarrow S_1 \Rightarrow AS_2U \Rightarrow ACS_3GU \Rightarrow ACS_4S_7GU \Rightarrow ACGS_5CS_7GU \Rightarrow ACGAS_6CS_7GU \Rightarrow$$
$$ACGAGCS_7GU \Rightarrow ACGAGCUS_8AGU \Rightarrow ACGAGCUGS_9CAGU \Rightarrow$$
$$ACGAGCUGCS_{10}CAGU \Rightarrow ACGAGCUGCCCAGU$$

Na slici 6.1 prikazano je stablo izvođenja za sekvencu *ACGAGCUGCCCAGU*, kao i njena sekundarna struktura dobijena datom gramatikom.



Slika 6.1 Na slici (a) je prikazano stablo izvođenja sekvence *ACGAGCUGCCCAGU,* gde su uparene baze obeležene plavom bojom, a neuparene crvenom. Na slici (b) se nalazi sekundarna struktura sekvence *ACGAGCUGCCCAGU,* koja je očigledno refleksija stabla izvođenja. Punim linijama su označena uparivanja baza. U ovoj strukturi se javljaju dva heliksa dužine dva (*A-U, C-G* i *U-A, G-C*), jedan heliks dužine jedan (*G-C*), dve petlje ukosnice i jedna višesturko-razgranata petlja. Način prikaza je preuzet iz [22].

Napomena 6.1: Da bi gramatika *G* iz primera 6.1 bila SKSG nedostaje jos samo verovatnosni model, odnosno verovatnoće za svako pravilo izvođenja. U daljem tekstu će biti reči o tome kako se te verovatnoće određuju.

Primer 6.2: Jedna od najjednostavnijih gramatika za rad sa RNK sekvencama sadrži samo jedan neterminal *S* i četiri terminala *A, U, C* i *G*.

$S \rightarrow ASU \mid USA \mid CSG \mid GSC$ // *pravila za uparivanje baza*
$S \rightarrow AS \mid US \mid CS \mid GS$ // *pravila za neuparene baze*
$\quad \mid A \mid U \mid C \mid G$
$S \rightarrow SS$ // *pravilo grananja*

Lako se primećuje da je ova gramatika višeznačna (videti poglavlje 2.2). Problem izbora stabla izvođenja ovde rešava verovatnosni model SKSG. Bira se ono stablo sa najvećom verovatnoćom izvođenja. Iako broj mogućih stabala izvođenja eksponencijalno raste sa povećanjem dužine sekvence, izbor se efikasno može izvesti primenom CYK algoritma iz poglavlja 4.2.

Videli smo kako u osnovi izgledaju pravila i azbuka gramatika za modelovanje RNK sekvenci, kao i da su sekundarne strukture zapravo forma koja se reflektuje iz stabala izvođenja. Potrebno je još da spomenemo kako se određuju verovatnosni parametri, da bi SKSG bila kompletna. Postoji više načina da se odrede vrednosti verovatnoća pravila izvođenja, ali mi ćemo navesti samo neke, koji se najčešće javljaju u praksi.

6.2 Obučavanje SKSG gramatika za modelovanje RNK sekvenci

Gramatike za rad sa RNK su uglavnom oformljene tako da modeluju sekvence koje su pronađene u prirodi. Postavljanje parametara gramatika na optimalne vrednosti često se naziva "*obučavanje*" (kao i kod skrivenih Markovljevih modela). Ukoliko imamo skup poznatih sekvenci i njihovih sekundarnih struktura, taj skup nazivamo "*skupom za obučavanje*" i pomoću njega možemo lako odrediti potrebne verovatnoće izvođenja. Jednostavnim postavljanjem verovatnosnih parametara za svako pravilo izvođenja prema broju koliko je puta to pravilo upotrebljeno u stablima izvođenja ulaznih sekvenci, dobićemo kompletnu SKSG koja sa velikom preciznošću modeluje sekvence RNK slične ulaznim.

Ukoliko nam nisu poznati parovi sekvenci i njihovih sekundarnih struktura za skup za obučavanje, verovatnoće možemo odrediti pomoću algoritma *Unutra-Spolja* (poglavlje 4.1), metodom maksimizacije očekivanja. Verovatnosni paramteri se mogu odrediti pomoću očekivanog broja upotrebe svakog pravila u izvođenju svake podsekvence ulazne sekvence. Ukratko, postupak za ovakav način određivanja parametara je sledeći: postavimo neke inicijalne vrednosti za svako pravilo izvođenja gramatike, pomoću algoritma *Unutra-Spolja* odrede se potrebna očekivanja prema inicijalnim verovatnoćama gramatike, ažuriraju se verovatnoće prema vrednostima dobijenim za očekivanja, svaki od ovih koraka se ponovi dva-tri puta, dok se ne dobiju konvergentne verovatnoće (ili do neke proizvoljne, unapred zadate tačnosti). Za detalje pogledati poglavlje 4.1.3.

Za određivanje parametara SKSG mogu se uzeti i neke druge informacije,

kao što su na primer evolutivne informacije. One mogu poboljšati procenu parametara, a samim tim i predikciju sekundarne strukture RNK. Više o tome će biti reči u narednom poglavlju.

7 Predikcija sekundarne strukture RNK

Predikcija struktura, funkcija ili uvijanja makromolekula je izazovan problem, kako u teoriji tako i u praksi. Neki od najvažnijih zadataka analize i poređenja sekvenci makromolekula jesu detektovanje zajedničkih obrazaca sekvenci, izvođenje višestrukog poravnanja sekvenci, razdvajanje članova familije od onih koji to nisu, kao i otkrivanje novih članova familije. Sekundarnu strukturu RNK i interakcije kod uparivanja baza daju mnogo više informacija nego same sekvence. Ovo znatno otežava analizu RNK, ali poznavanje sekundarne strukture čini veoma važnim. Način na koji su baze uparene u nekoj sekvenci određuje funkciju molekula, kao i interakciju sa drugim molekulima. Uparivanje baza je na najrealnijem nivou prikazano tercijarnom stukturom RNK sekvenci, ali kako je znatno teže doći do tercijarne strukture, u praksi se češće koriste sekundarne strukture za proučavanje uparivanja baza. Stoga se naučnici već više od dve decenije bave ozbiljnim istraživanjem na temu predikcije RNK sekundarne strukture. Postoje razne metode određivanja sekundarne strukture, ali mi ćemo navesti samo nekoliko osnovnih, sa akcentom na one koje uključuju SKSG u modelovanju.

7.1 Osnovne metode predikcije sekundarne strukture RNK

Među prvim metodama za predikciju sekundarne strukture RNK javile su se one koje koriste tehnike *dinamičkog programiranja.* Jedna od njih jeste primena *Nusinovog* algoritma [2], koji na efikasan način konstruiše strukturu sa najviše uparenih baza. Ovo je najjednostavniji način da se odredi sekundarna struktura i nije dovoljno precizan, ali je njegov princip kasnije primenjen u naprednoj metodi predikcije *minimizacijom energije*.

Osnovna ideja metoda zasnovanih na pravilima *termodinamike* jeste da je najverovatnija sekundarna struktura ona koja koristi najmanje energije. Ta ideja potiče iz činjenice da molekuli uglavnom formiraju strukture koje minimizuju energiju potrebnu za održavanje stabilnih veza unutar strukture, koja se neće lako rasformirati. *Zukerov* algoritam [2] je jedna od metoda minimizacije energije. Primenom dinamičkog programiranja i vrednosti kao što su količina energije potrebna za uparivanje baza u petlji ukosnice ili heliksu, energija koju emituju neuparene baze u strukturi, itd., mogu se vrlo precizno predvideti sekundarne strukture RNK sekvenci. Glavni izazov kod ove tehnike predikcije predstavlja dolaženje do tačnih energetskih parametara. Ukoliko parametri nisu dovoljno precizni, rezultujuće sekundarne strukture neće biti zadovoljavajuće.

Drugu grupu metoda za predikciju čine *metode sravnjivanja* (engl. *Matching methods*). Jedna od njih jeste metoda *sravnjivanja maksimalne težine.* Problem pronalaženja sravnjivanja uparenih baza ovde je formulisan kao pronalaženje maksimalnog broja težinskog sravnjivanja između dva skupa baza. Za više informacija pogledati [23].

Nešto drugačiji pristup predikciji sekundarne strukture RNK jeste pomoću e*volutivnih algoritama* (EA). Ova metoda čuva skupove mogućih rešenja, pod nazivom *populacije*. Iz tih skupova, raznim izračunavanjima, izmenama i izborima najboljih kandidata, dolazi se do konačnog rezultata. Nedostatak EA jeste vrlo sporo dobijanje konačnog rešenja [24].

U osnovne metode predikcije sekundarne strukture RNK mogu se svrstati i one koje su zasnovane na primeni *gramatika.* Postoji više vrsta gramatika koje se mogu iskoristiti (videti hijerarhiju Čomskog u poglavlju 2.1), ali u ovom radu ćemo se posvetiti upotrebi stohastičkih kontekst slobodnih gramatika.

Među bitnijim problemima koji su od biološkog interesa pri analiziranju RNK su sledeća dva: prvi problem je predikcija sekundarne strukture RNK za pojedinačne sekvence; drugi problem vezan je za analizu višestrukih poravnjanja familija srodnih RNK sekvenci. Metode predikcije pomoću SKSG bave se sa oba navedena problema.

7.2 Predikcija sekundarne strukture RNK pomoću stohastičkih kontekst slobodnih gramatika: prediktor Pfold

Značajan doprinos predikciji sekundarne strukture RNK dali su *Knudsen* i *Hein* (Bjarne Knudsen, Jotun Hein). U svom radu [25] oni pored SKSG koriste i filogeniju u predikciji. Server *Pfold* [26] koji su konstruisali za potrebe predikcije sekundarne RNK strukture dostupan je na internetu [27]. Predstavićemo ukratko osnovne principe na kojima je *Pfold* zasnovan.

Pfold je prediktor sekundarne strukture RNK, koji su razvili Knudsen i Hein iz svog algoritma KH-99. Kao ulaz prima niz RNK sekvenci za koje se pretpostavlja da imaju istu sekundarnu strukturu. Izračunava se njihovo poravnanje, kao i filogenetsko stablo. Zatim se pomoću jednostavne SKSG i CYK algoritma izračunava najverovatnija sekundarna struktura.

Ideja za algoritam KH-99 potiče od Goldmanove metode [28] za predikciju sekundarne strukture proteina pomoću skrivenih Markovljevih modela i filogenetskih informacija. KH-99 se sastoji iz dva zasebna dela. Prvi je SKSG, a drugi evolutivni model.

Gramatika koju su koristili Knudsen i Hein je vrlo jednostavna. Početni

neterminal je, kao i obično, označen znakom *S.* Ostali neterminali su *L* i *F*, dok su terminali zabeleženi malim slovima.

$$S \rightarrow LS \mid L$$
$$F \rightarrow dFd \mid LS$$
$$L \rightarrow s \mid dFd$$

Znak *s* označava baze koje ostaju neuparene, dok *d* predstavlja baze koje se uparuju. Neterminal *S* izvodi *petlje*, *F* izvodi *helikse*, dok *L* odlučuje da li će biti *grananja* ili ne (vidi 5.1.2). *Heliks* može biti proizvolje dužine, dok petlja mora imati najmanje dve pozicije, što je određeno pravilima ove gramatike. U ovom kontekstu se podrazumeva da *pozicija* može biti ne samo jedna baza, već i početak heliksa.

Verovatnoće primene pravila izvođenja određuju izvođenje sekundarne strukture. Verovatnosni parametri SKSG su određeni pomoću skupova za obučavanje. U skupovima za obučavanje su date primarne strukture i odgovarajuće sekundarne strukture RNK. Tako se jednostavnim prebrojavanjem različitih neuparenih baza, kao i uparenih, mogu odrediti verovatnoće njihovog pojavljivanja. Procena parametara SKSG u algoritmu KH-99 izvršena je *Unutra-Spolja* algoritmom (4.1) nad skupom za obučavanje koji se sastojao iz 1968 tRNK sekvenci i 305 LSU rRNK sekvenci (LSU - *large subunit*). Konačna SKSG izgleda ovako (u zagradama su naznačene verovatnoće primene pravila izvođenja pored kojih se nalaze):

$$S \rightarrow LS\ (86{,}9\%) \mid L\ (13{,}1\%)$$
$$F \rightarrow dFd\ (78{,}8\%) \mid LS\ (21{,}2\%)$$
$$L \rightarrow s\ (89{,}5\%) \mid dFd\ (10{,}5\%).$$

Ukoliko za ulazne sekvence nije poznato filogenetsko stablo, potrebno ga je odrediti, nekim od poznatih metoda, kao što je, npr. Svofordov (*Swofford*) [29].

Višestruko poravnanje ulaznih sekvenci može se konstruisati nekim od poznatih algoritama, kao što je, npr. *CLUSTAL W alignment*, ako poravnanje nije unapred poznato. Primećena je značajno veća preciznost u predikciji ukoliko je na ulazu zadato više sekvenci. Takođe, utvrđeno je i da je preciznost veća pri korišćenju podataka iz filogenetskog stabla.

Postoji i novija verzija algoritma KH-99, implementirana u *PPfold* prediktoru [30][31]. *PPfold* je paralelizovana verzija *Pfold-a,* koja znatno brže izračunava sekundarnu strukturu i za razliku od *Pfold* prediktora, kao ulaz može da primi veći broj znatno dužih sekvenci. *PPfold* je pogodan za primenu kod velikih sekvenci RNK, kao što su, npr. RNK sekvence raznih virusa.

8 Predikcija sekundarne strukture RNK pomoću SKSG: eksperiment i rezultati

Cilj eksperimentalnog dela ove teze jeste da se na konkretnom primeru pokaže korišćenje SKSG u svrhe predikcije sekundarnih struktura RNK sekvenci. Rezultati su dobijeni jednostavnom primenom metoda predstavljenih u prethodnim poglavljima, uz korišćenje nekih gotovih alata koji će kasnije biti navedeni.

8.1 Baza podataka

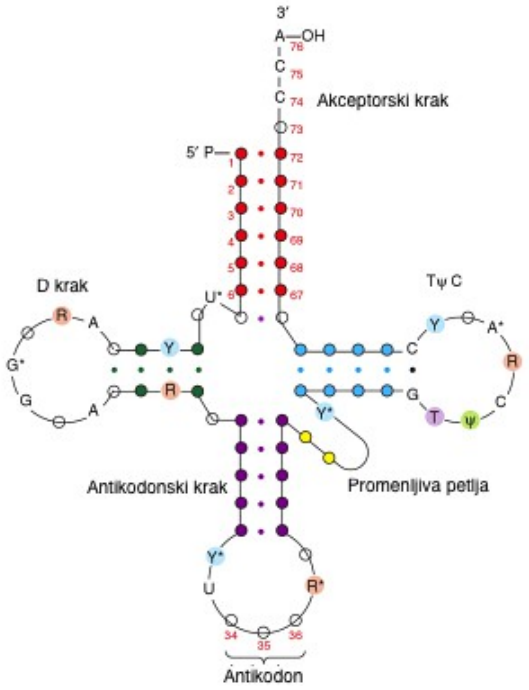
Kako bismo ilustrovali primenu prethodne teorije za predikciju, potrebna nam je baza podataka koja sadrži RNK sekvence i njihove sekundarne strukture. Jedna od takvih baza jeste i *RNA STRAND* (engl. skraćenica: *The RNA secondary STRucture and statistical ANalysis Database*). *RNA STRAND* [32][33-40] se sastoji iz poznatih sekundarnih struktura RNK raznih tipova i organizama. Krajnji cilj ove baze je da sakupi obimnu kolekciju svih poznatih sekundarnih strukutra RNK i da obezbedi jednostavne i moćne metode analiziranja, pretraživanja i ažuriranja navedenih podataka. U trenutku pisanja ovog rada, pomenuta baza sadrži ukupno 4666 različitih sekundarnih struktura RNK. *RNA STRAND* spada u klasu baza RNK struktura (vidi 5.2).

Sekvence se u bazi *RNA STRAND* mogu pretraživati po raznim kriterijumima. Tako, na primer, postoji opcija izlistavanja svih sekundarnih struktura RNK po tipovima. Neki od tipova koji se najčešće javljaju su *tRNA* (engl. *transfer RNA*), *16S rRNA* (engl. *16S ribosomal RNA*), *tmRNA* (engl. *Transfer Messenger RNA*), *23S rRNA*, *SRP RNA* (engl. *Signal Recognition Particle RNA*), *RNase P RNA* (engl. *Ribonuclease P RNA*), itd. Obučavanje i testiranje ćemo izvoditi nad uzorkom sekvenci tipa *tRNA*.

Transportna RNK (*tRNK;* engl. *tRNA*) je molekul koji ima funkciju fizičke veze između sekvenci sastavljenih od nukleotida (DNK i RNK) i sekvenci amino kiselina (proteina). Ona prenosi amino kiseline do organela zvanih *ribozomi.* Redosled nukleotida u *tRNK* određuje amino kiseline. Svaka tri uzastopna nukleotida (ova trojka se naziva *kodon*), kodiraju jednu od 20 amino kiselina. Transportna RNK je uglavnom dužine manje od 100 nukleotida. Prema centralnoj dogmi molekularne biologije (vidi poglavlje 5), ona igra glavnu ulogu u procesu *translacije* (sinteze proteina iz RNK).

Sekundarna struktura *tRNK* ima karakterističan oblik, koji podseća na detelinu sa tri lista. Dvolančani deo te strukture nastaje uparivanjem nukleotida vodoničnim vezama i on predstavlja "dršku" deteline, dok su listovi jednolančane petlje.

U trenutku pisanja ovog rada, u bazi *RNA STRAND* postoji 707 različitih *tRNK* sa poznatim sekundarnim strukturama. Mi ćemo za obučavanje i testiranje iskoristiti ukupno 100 sekvenci različitih dužina i iz raznih organizama.



Slika 8.1 Karakteristična sekundarna struktura *tRNK* u obliku deteline sa tri lista. Slika je preuzeta iz [41] i prevedena na srpski.

8.2 Obučavanje SKSG

U prethodnim poglavljima ovog rada smo razmatrali nekoliko kontekstno slobodnih i stohastičkih kontekst slobodnih gramatika kojima se mogu modelovati RNK sekvence. Jedna od jednostavnijih gramatika, koja se pokazala kao vrlo povoljna za predikciju sekundarne strukture RNK, jeste takozvana gramatika KH-99, koju su *Knudsen* i *Hein* koristili u svom radu [25]. Kao program za obučavanje koristićemo implementaciju algoritma *Unutra-Spolja* [42]. Skup za obučavanje biće 70 *tRNK* sekvenci, dužina od 17 do 90 nukleotida.

Pre svega, da bismo primenili algoritam *Unutra-Spolja*, gramatiku KH-99 moramo modifikovati da bude u normalnoj formi Čomskog. Prevođenjem pravila gramatike tako da sva budu oblika $X \rightarrow YZ$ ili oblika $X \rightarrow a$, gde su X, Y i Z neterminali, a a terminal, dobijamo sledeću kontekstno slobodnu gramatiku:

$$W_1 \rightarrow W_2W_3 \mid W_4W_5 \mid s$$
$$W_2 \rightarrow W_4W_5 \mid s$$
$$W_3 \rightarrow W_2W_3 \mid W_4W_5 \mid s$$
$$W_4 \rightarrow d$$
$$W_5 \rightarrow W_6W_4$$
$$W_6 \rightarrow W_4W_5 \mid W_2W_3 \ .$$

W_1, W_2, W_3, W_4, W_5 i W_6 predstavljaju neterminale, dok su s terminali neuparenih, a d terminali uparenih baza *A, U, C* i *G*. Ova gramatika je očigledno u normalnoj formi Čomskog. Da bi postala SKSG, potrebno je još dodati verovatnosni model. Kao inicijalne verovatnoće, postavićemo proizvoljne vrednosti u intervalu od 0 do 1, ali tako da zadovoljavaju pravilo SKSG, da suma verovatnoća svih pravila izvođenja koji sa leve strane pravila imaju isti neterminal, bude jednaka 1. Pretpostavili smo da pravila koja izvode terminale imaju manju verovatnoću od pravila grananja, ukoliko se izvode iz istog neterminala. Kasnije se, pomoću skupa za obučavanje, te vrednosti menjaju i bolje modeluju realne sekvence. Evo kako izgleda inicijalna SKSG:

$$\begin{aligned}
W_1 &\rightarrow W_2W_3 \ (0.2) \mid W_4W_5 \ (0.2) \\
&\quad \mid A \ (0.12) \mid U \ (0.12) \mid C \ (0.12) \\
&\quad \mid G \ (0.12) \mid N \ (0.12) \\
W_3 &\rightarrow W_2W_3 \ (0.2) \mid W_4W_5 \ (0.2) \\
&\quad \mid A \ (0.12) \mid U \ (0.12) \mid C \ (0.12) \\
&\quad \mid G \ (0.12) \mid N \ (0.12) \\
W_6 &\rightarrow W_4W_5 \ (0.5) \\
&\quad \mid W_2W_3 \ (0.5) \\
W_5 &\rightarrow W_6W_4 \ (1.0) \\
W_2 &\rightarrow W_4W_5 \ (0.4) \\
&\quad \mid A \ (0.12) \mid U \ (0.12) \mid C \ (0.12) \\
&\quad \mid G \ (0.12) \mid N \ (0.12) \\
W_4 &\rightarrow A \ (0.2) \mid U \ (0.2) \mid C \ (0.2) \\
&\quad \mid G \ (0.2) \mid N \ (0.2) \ .
\end{aligned}$$

Oznaka *N* služi za prepoznavanje elemenata sekvenci različitih od *A, U, C* i *G*. U zagradama su predstavljene vrednosti verovatnoća.

Ovu SKSG zajedno sa 70 sekvenci *tRNK* prosleđujemo kao ulaz programu u kome je implementiran algoritam *Unutra-Spolja*, a kao izlaz dobijamo SKSG sa popravljenim parametrima za predikciju sekundarne strukture RNK. Tu modifikovanu SKSG ćemo kasnije testirati u predikciji, poređenjem sa poznatim sekundarnim strukturama. Vrednosti verovatnoća u programu smo zaokruživali na sedam decimala.

Konačno, obučena SKSG za predikciju sekundarne strukture tRNK izgleda ovako:

$$\begin{aligned}
W_1 &\rightarrow W_2W_3 \ (0.1634020) \mid W_4W_5 \ (0.8365975) \\
&\quad \mid A \ (0.0000001) \mid U \ (0.0000001) \mid C \ (0.0000001) \\
&\quad \mid G \ (0.0000001) \mid N \ (0.0000001) \\
W_3 &\rightarrow W_2W_3 \ (0.2362123) \mid W_4W_5 \ (0.3779777)
\end{aligned}$$

$| A \ \ (0.0002261\ ) \ | U \ \ (0.0390719) \ | C \ \ (0.1798469)$
$| G \ \ (0.166665) \ | N \ \ (0.0000001)$

$W_6 \rightarrow W_4W_5 \quad (0.5361326)$
$| W_2W_3 \quad (0.4638674)$

$W_5 \rightarrow W_6W_4 \quad (1.0)$

$W_2 \rightarrow W_4W_5 \quad (0.0532827)$
$| A \ \ (0.1777583) \ | U \ \ (0.1646913) \ | C \ \ (0.0219265)$
$| G \ \ (0.5823411) \ | N \ \ (0.0000001)$

$W_4 \rightarrow A \ \ (0.0949165) \ | U \ \ (0.1729965) \ | C \ \ (0.2823022)$
$| G \ \ (0.4497847) \ | N \ \ (0.0000001)$.

Vremenska i prostorna složenost *Unutra-Spolja* algoritma zavise od dužine ulaznih sekvenci, kao i broja neterminala koji se javljaju u gramatici. Poređenja radi, istim programom i istim skupom sekvenci, obučili smo i još jednu SKSG gramatiku koja je vrlo česta u modelovanju RNK sekvenci. Ova gramatika jasnije određuje moguća uparivanja baza:

$S \rightarrow LS \mid L$	*-pravilo za formiranje petlji*
$F \rightarrow LS$	
$F \rightarrow aFu \mid uFa \mid cFg \mid gFc$	*-Watson-Crick uparivanje baza*
$F \rightarrow gFu \mid uFg$	*-nestabilno uparivanje baza*
$L \rightarrow a \mid c \mid g \mid u \mid n$	*-neuparene baze*
$L \rightarrow aFu \mid uFa \mid cFg \mid gFc$	*-Watson-Crick uparivanje baza u petljama*
$L \rightarrow gFu \mid uFg$	*-nestabilno uparivanje baza u petljama*

Očigledno, kako ima više pravila od KH-99, prevođenjem u normalnu formu Čomskog, dobili smo i dva puta više neterminala, što znatno usporava obradu podataka. Vremenska složenost *Unutra-Spolja* algoritma je $O(n^3m^3)$, gde je n dužina ulazne sekvence, a m broj neterminala, pa je očekivano da je programu za istih 20 ulaznih sekvenci bilo potrebno približno $2^3 = 8$ puta više vremena da završi sa radom. Program za obučavanje je kao izlaz dao:

$W_1 \rightarrow W_2W_3 \ \ (0.0011483) \mid W_4W_5 \ \ (0.0000001) \mid W_6W_7 \ \ (0.0000076)$
$\mid W_8W_9 \ \ (0.0000001) \mid W_{10}W_{11} \ \ (0.9988434) \mid W_{10}W_5 \ \ (0.0000001)$
$\mid W_6W_9 \ \ (0.0000001) \mid A \ \ (0.0000001) \mid U \ \ (0.0000001)$
$\mid C \ \ (0.0000001) \mid G \ \ (0.0000001) \mid N \ \ (0.0000001)$

$W_3 \rightarrow W_2W_3 \ \ (0.4997714) \mid W_4W_5 \ \ (0.0000568) \mid W_6W_7 \ \ (0.2084028)$
$\mid W_8W_9 \ \ (0.0000001) \mid W_{10}W_{11} \ \ (0.0000001) \mid W_{10}W_5 \ \ (0.0000003)$
$\mid W_6W_9 \ \ (0.0000001) \mid A \ \ (0.2490264) \mid U \ \ (0.0405814)$
$\mid C \ \ (0.0021582) \mid G \ \ (0.0000016) \mid N \ \ (0.000001)$

$W_{12} \rightarrow W_2W_3 \ \ (0.2514318) \mid W_4W_5 \ \ (0.1252876) \mid W_6W_7 \ \ (0.0001827)$
$\mid W_8W_9 \ \ (0.1239123) \mid W_{10}W_{11} \ \ (0.4991854) \mid W_{10}W_5 \ \ (0.0000001)$
$\mid W_6W_9 \ \ (0.0000001)$

$W_5 \rightarrow W_{12}W_6 \quad (1.0)$

$W_7 \rightarrow W_{12}W_4 \quad (1.0)$

$W_9 \rightarrow W_{12}W_{10} \quad (1.0)$

$W_{11} \rightarrow W_{12}W_8 \quad (1.0)$

$W_2 \rightarrow A \ \ (0.2486284) \mid U \ \ (0.2088189) \mid C \ \ (0.497422)$
$\mid G \ \ (0.0022754) \mid N \ \ (0.0000001)$
$\mid W_4W_5 \ \ (0.0000068) \mid W_6W_7 \ \ (0.0404234) \mid W_8W_9 \ \ (0.0023654)$

$| W_{10}W_{11}$ (0.0000024) | $W_{10}W_5$ (0.0000005) | W_6W_9 (0.0000567)
$W_4 \rightarrow A$ (1.0)
$W_6 \rightarrow U$ (1.0)
$W_8 \rightarrow C$ (1.0)
$W_{10} \rightarrow G$ (1.0) .

8.3 Testiranje

Sada kada imamo obučenu SKSG, pomoću *CYK* algoritma (4.2), za svaku sekvencu tRNK možemo odrediti njeno najverovatnije stablo izvođenja, a time i sekundarnu strukturu. Za obučavanje smo iskoristili 70 sekvenci tRNK iz uzorka, pa ćemo preostalih 30 upotrebiti za testiranje. Kako u bazi *RNA STRAND* postoje tačne sekundarne strukture svih sekvenci iz uzorka, možemo ih uporediti sa sekundarnim strukturama predviđenim pomoću SKSG i *CYK* algoritma. Opisaćemo sada tačan postupak testiranja.

Prvo smo svaku od 30 sekvenci tRNK za testiranje propustili kroz *CYK* algoritam i za svaku pojedinačno sačuvali njeno najverovatnije stablo izvođenja. Kao što smo videli u tački 6.1 ovog rada, to stablo izvođenja se direktno prevodi u sekundarnu strukturu te RNK sekvence. Tako smo izvršili predikciju sekundarnih struktura sekvenci skupa za testiranje.

Za određivanje preciznosti predviđenih sekundarnih struktura RNK koristili smo metod koji se navodi i u radu Knudsena i Heina [25]. Pre svega se odredi poravnanje predviđene strukture i tačne strukture u tačka-zagrada notaciji. Tačke predstavljaju neuparene baze, a odgovarajući parovi otvorenih i zatvorenih zagrada predstavljaju uparene baze. Zatim se prebroje sve pozicije u poravnanju koje se podudaraju, a različite su od belina. Taj broj se podeli ukupnim brojem pozicija u poravnanju, različitih od belina. Tako dobijamo meru tačnosti predikcije jedne sekvence. Ukupnu meru tačnosti skupa od 30 sekvenci za testiranje, dobili smo nalaženjem aritmetičke sredine mera tačnosti svake pojedinačne predviđene sekundarne strukture.

Poravnanja smo računali pomoću internet verzije alata *MiGal* za poređenje RNK sekundarnih struktura [44]. Ovaj alat poredi strukture na više nivoa, pomoću posebne *MiGal* strukture [45], ali nama je dovoljan bio rezultat poravnanja struktura u tačka-zagrada notaciji. Primera radi, navodimo jednu sekvencu tRNK iz našeg uzorka preuzetog iz baze *RNA STRAND*.

Primer 8.3.1: Transportna RNK sekvenca sa identifikacionom šifrom PDB_00376 u bazi *RNA STRAND* je sekvenca iz organizma *Escherichia coli*. Njena primarna struktura ima 73 baze (slika 8.3.1).

GGGGUAUCGCCAAGCGGUAAGGCACCGGAUUCUGAUUCCGGAGGUCGAGGUUCGAAUC
CUCGUACCCCAGCCA

Slika 8.3.1 Primarna struktura tRNK sa šifrom PDB_00376 u bazi *RNA STRAND*[32]

Predikcijom pomoću SKSG dobijamo da je sekundarna struktura ove sekvence kao na slici 8.3.2 (a), dok je na slici 8.3.2 (b) prikazana tačna sekundarna struktura iz baze.

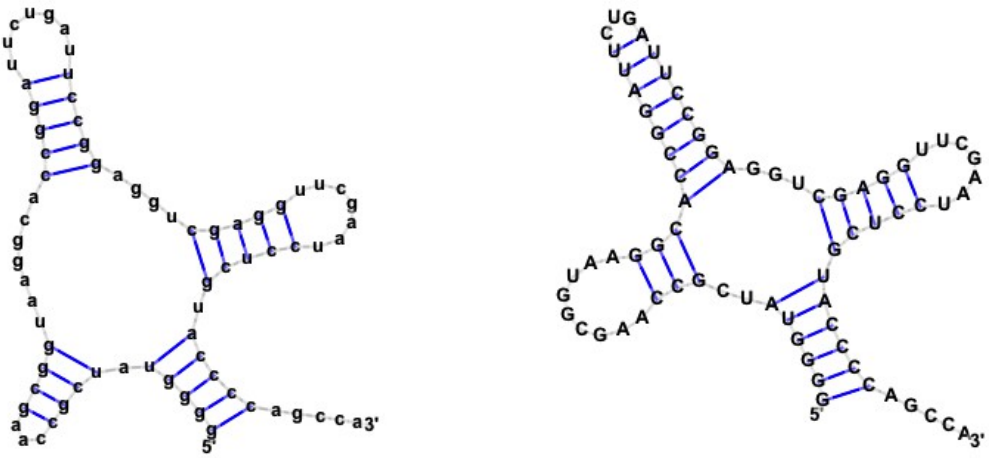
Poravnanje ove dve sekundarne strukture dobijeno alatom *MiGal* i prikazano u tačka-zagrada notaciji, gde je belina označena znakom "-", izgleda ovako:

(((((.--((---((--...)))).....-(((((.....))))).-...(((((.....))))).)))))....
((((((..(((.....)-))-------(((((((...)))))))-)...(((((.....)))))))))).....

U ovom primeru, preciznost procene je 87,5%, što je ispod proseka ukupne preciznosti predikcije pomoću SKSG na ovom uzorku. Iz tih razloga je ovaj primer i izabran.

Tačnost predikcije izmerena na celom skupu sekvenci za testiranje iznosi 89,76%. Od ukupnog broja testiranih sekvenci, čak 30% predviđenih struktura se u potpunosti podudaralo sa tačnim strukturama. Izmeren je najlošiji procenat tačnosti od 75%, kod sekvence od 148 baza, koja odstupa od karakterističnih tRNK.

(a) **(b)**



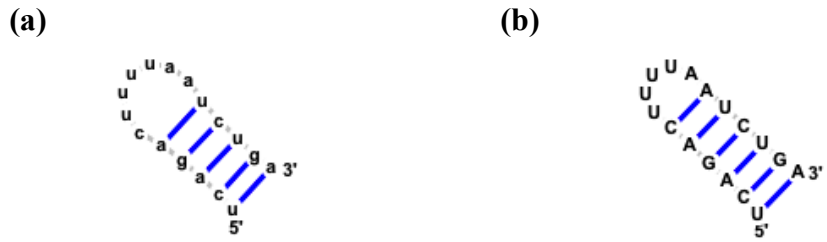
Slika 8.3.2 Na slici su prikazane (a) predikcija i (b) tačna sekundarna struktura tRNK sekvence PDB_00376 iz baze *RNA STRAND*. Za grafički prikaz korišćen je program *jViz* [46][47].

8.4 Diskusija

Uzorak sekvenci za obučavanje i testiranje u ovom eksperimentu sadrži raznovrsne sekvence. Dužina sekvenci varira od 17 do 148 baza koje čine jednu tRNK. Ove tRNK takođe nemaju sve sekundarnu strukturu u obliku deteline sa tri lista, što dodatno otežava predikciju. Da smo kao uzorak uzeli skup sekvenci koje imaju približno istu sekundarnu strukturu, SKSG bi bila obučena tako da sa znatno većom

verovatnoćom dobija preciznu strukturu. Takođe, preciznost zavisi i od broja sekvenci koje čine uzorak. Što je broj elemenata skupa za obučavanje veći, parametri preciznije opisuju željene podatke. To dokazuje i rad Knudsena i Heina [25], gde je pokazano da se sa većom tačnošću vrši predikcija sekundarne strukture RNK sekvenci ako se metodi za predikciju prosledi više sekvenci za koje se pretpostavlja da imaju istu sekundarnu strukturu.

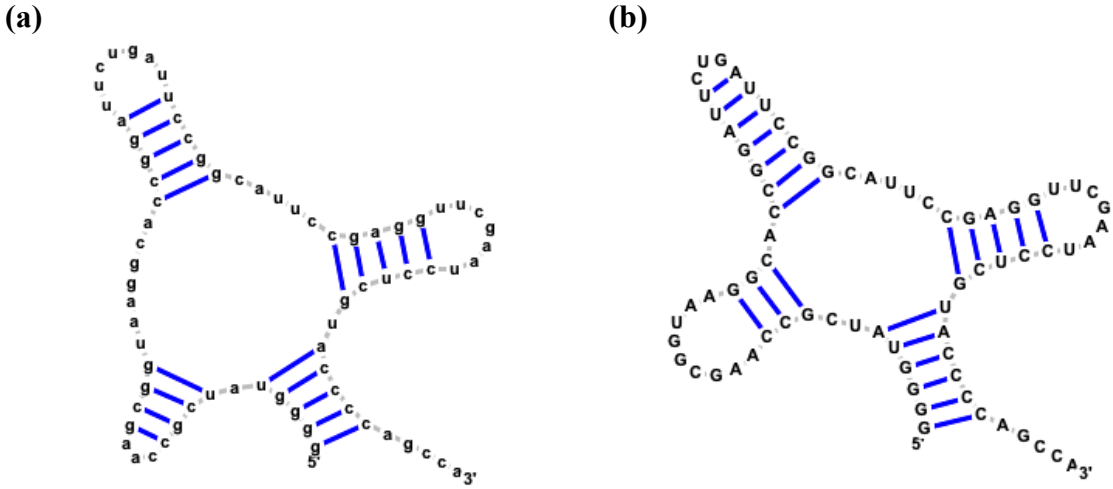
Lako se zaključuje da kod kraćih sekvenci, male greške u predikciji rezultuju manjim procentom mere preciznosti, pa tako na primeru sekvence PDB_00024 iz baze *RNA STRAND,* koja je sastavljena iz svega 17 baza, dobijamo rezultate kao na slici 8.4.1.



Slika 8.4.1 (a) Sekundarna struktura tRNK sekvence PDB_00024 iz baze *RNA STRAND* određena pomoću SKSG. (b) Tačna sekundarna struktura tRNK sekvence PDB_00024. Za grafički prikaz korišćen je program *jViz* [46][47].

Razlika sekundarne strukture izračunate pomoću SKSG i tačne strukture je samo u jednom uparenom paru baza, ali mera preciznosti iznosi svega 88,2%.

Znatno duža sekvenca PDB_426 ima približno istu preciznost, od 88%, ali je razlika u uparenim bazama dosta veća (slika 8.4.2).



Slika 8.4.2 (a) Sekundarna struktura tRNK sekvence PDB_00426 iz baze *RNA STRAND* određena pomoću SKSG. (b) Tačna sekundarna struktura tRNK sekvence PDB_00426. Za grafički prikaz korišćen je program *jViz* [46][47].

Primere u kojima je preciznost veća ne navodimo, jer nisu interesantni za poređenje. Ali važno je imati na umu da je samo u ovom malom uzorku bilo čak 30% potpuno pogođenih struktura, a ukupna mera prezinosti je bila blizu 90%. Ni u jednom primeru opšta forma strukture nije bila promašena predikcijom.

9 Zaključak

Videli smo da se stohastičke kontekst slobodne gramatike, koje predstavljaju uopštenje kontekstno slobodnih gramatika, mogu koristiti kao kompletan verovatnosni sistem za modelovanje, baš kao i skriveni Markovljevi modeli (engl. skrać. *HMM*). Naveli smo tri osnovna algoritma za rešavanje neophodnih problema u ovakvim sistemima. *Unutra-Spolja* algoritam za obučavanje gramatika je isto što i *Forward-Backward* algoritam kod *HMM*. *Unutra* algoritam nam daje verovatnoću izvođenja neke niske zadatom gramatikom, baš kao što *Forward* algoritam to čini kod *HMM*. *CYK* algoritmom se određuje najverovatnije stablo izvođenja. On je pandan *Viterbi* algoritmu kod *HMM*.

RNK sekvence se sa računarske tačke gledišta mogu posmatrati kao niske karaktera sa nekim pravilima formiranja. Stoga je vrlo pogodno iskoristiti veoma razvijenu teoriju formalnih jezika za njihovo modelovanje i predikciju sekundarne strukture pomoću verovatnosnih parametara.

Pokazali smo kako se SKSG koriste zajedno sa biološkim modelima. Videli smo da su matematičke i računarske tehnike vrlo povoljne za baratanje genetskim podacima. RNK sekvence igraju važnu ulogu u svim organizmima, pa je tako i njihovo izučavanje vrlo bitno.

Kako su u sekundarnim strukturama RNK sekvenci sabrane najvažnije informacije o uparivanju nukleotida, a samim tim i funkciji molekula RNK, jasno je da je poznavanje sekundarne struktre vrlo važan problem. Takođe, od načina na koji su baze uparene unutar nekog molekula, zavisi kako će on dalje interagovati sa ostalim molekulima.

Postoji mnogo načina da se dođe do sekundarne strukture RNK, ali ti načini su vrlo često neizvodljivi u praksi ili su previše zahtevni. Zbog toga je predikcija sekundarne strukture dobila na značaju. Razvijeno je desetine metoda za predikciju sekundarne strukture RNK, koji se koriste različitim heuristikama.

Metoda predikcije sekundarne strukture pomoću SKSG koristi isključivo računarske alate. Nije potrebno dodatno poznavanje fizičkih i hemijskih pojmova i podataka. Pokazali smo da se na osnovu matematičkih metoda može doći do vrlo efikasnog modelovanja RNK sekvenci.

Mana SKSG jeste što je potreban veliki broj podataka za obučavanje parametara gramatike, kako bi modelovanje bilo što preciznije, kao i složenost algoritama koja rad sa velikim skupovima čini skupim. Nije uvek jednostavno doći do dovoljnog broja sekvenci za obučavanje, pa predikcija može biti lošijeg kvaliteta. Već smo pomenuli da su istraživanja u pravcu kombinovanja metoda SKSG i

filogenetskih stabala, kao i poravnanja više sličnih sekvenci za koje se pretpostavlja da imaju istu sekundarnu strukturu, dala dobre rezultate i poboljšala tačnost predikcije. Ali za to je takođe potrebno dosta ulaznih podataka. Ukoliko nam je na raspolaganju obiman skup sekvenci za obučavanje, za koje se pretpostavlja da imaju sličnu sekundarnu strukturu kao i sekvence za koje je potrebna predikcija, metode SKSG će dati rezultate visoke tačnosti.

Reference

[1] Vitas, D. (2006) *Prevodioci i interpretatori,* Matematički fakultet, Beograd.

[2] Durbin, R., Eddy, S., Krogh, A., Mitchison, G. (1998) *Biological sequence analysis*, Cambridge University Press.

[3] Gorodkin, I., Ruzzo, W. (2014) *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods,* Humana Press.

[4] Martin, J. C. (1997) *Introduction to languages and the theory of computation,* McGraw-Hill Inc.

[5] Böckenhauer, H. J., Bongartz, D. (2007) *Algorithmic Aspects of Bioinformatics*, Springer Science & Business Media.

[6] Polanski, A., Kimmel, M. (2007) *Bioinformatics,* Springer Science & Business Media.

[7] Russell, P. J. (2009) *iGenetics*, Benjamin Cummings, 3rd edition.

[8] Nelson, D. L., Cox, M. M. (2008) *Lehninger Principles of Biochemistry*, W. H. Freeman, 5th edition.

[9] National Institutes of Health, *Talking Glossary of Genetic Terms,* National Human Genome Research Institute. 10 August 2014, http://www.genome.gov/glossary/

[10] Watson, J., Baker, T., Bell, S., Gann, A., Levine, M., Losick, R. (2013) *Molecular biology of the Gen*, Benjamin-Cummings Publishing Company.

[11] Gesteland R. F., Cech T. R., Atkins J. F., (1993), *The RNA World : the Nature of Modern RNA Suggests a Prebiotic RNA*, Cold Spring Harbor Laboratory Press.

[12] Setubal, J., Meidanis, J. (1997) *Introduction to Computational Molecular Biology,* International Thomson Publishing, 1st edition.

[13] The National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov

[14] Sprinzl M, Vorderwülbecke T, Hartmann T (1985) *Compilation of sequences of tRNA genes,* Nucleic Acids Res 13: 51–104.

[15] Gardner PP, Daub J, Tate J, Moore BL, Osuch IH, Griffiths-Jones S et al (2011) *Rfam: Wikipedia, clans and the "decimal" release*, Nucleic Acids Res 39 (Database issue): D141– D145.

[16] Oxford Journals, Life Sciences, Nucleic Acids Research http://www.oxfordjournals.org/our_journals/nar/database/a/

[17] Liu, C., Bai, B., Skogerbø, G., Cai, L., Deng, W., Zhang Y et al (2005) *NONCODE: an integrated knowledge database of non- coding RNAs,* Nucleic Acids Res 33 (Database issue): D112–D115.

[18] Griffiths-Jones, S., Grocock, R. J., van Dongen, S., Bateman, A., Enright, A. J. (2006) *miR- Base: microRNA sequences, targets and gene nomenclature,* Nucleic Acids Res 34 (Database issue): D140–D144.

[19] Bartel, D. P. (2009) *MicroRNAs: target recognition and regulatory functions,* Cell 136 (2): 215–233.

[20] Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., Eddy, S. R. (2003) *Rfam: an RNA family database,* Nucleic Acids Res 31 (1): 439–441.

[21] Pruesse, E., Quast, C., Knittel, K., Fuchs, B. M., Ludwig, W., Peplies, J. et al (2007) *SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB*, Nucleic Acids Res 35 (21): 7188–7196.

[22] Sakakibara, Y., Brown, M., Underwood, R., Mian, I. S., Haussler, D. (1994) *Stochastic context-free grammars for modeling RNA,* 27th Hawaii International Conference on System Sciences, 284–283, Honolulu, IEEE Computer Society Press.

[23] Ruan, J., Stormo, G., Zhang, W. (2004) *An iterated loop matching approach to the prediction of rna secondary structures with pseudoknots*, Bioinformatics, 20 (1): 58–66.

[24] Wiese, K. C., Deschenes, A., Glen, E. (2003) *Permutation based rna secondary structure prediction via a genetic algorithm,* Sarker, R., Reynolds, R., Abbass, H., Tan, K. C., McKay, B., Essam, D., Gedeon, T. editors, Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, IEEE Press, 335–342.

[25] Knudsen, B., Hein, J. (1999) *Rna secondary structure prediction using stochastic context-free grammars and evolutionary history,* Bioinformatics, 15: 446–454.

[26] Knudsen, B., Hein, J. (2003) *Pfold: RNA secondary structure prediction using stochastic context-free grammars*, Nucleic Acids Research, 31 (13),

3423-3428.

[27] Knudsen, B., Hein, J., Sukosd, Z., *Pfold*, http://daimi.au.dk/~compbio/pfold/

[28] Goldman, N., Thorne, J. L., Jones, D. T. (1996) *Using evolutionary trees in protein secondary structure prediction and other comparative sequence analyses*, Journal of Molecular Biology, 263, 196-208.

[29] Swofford, D. L., Olsen, G. J., Waddell, P. J., Hillis, D. M. (1996) Phylogenetic inference, In Hillis, D. M., Moritz, C. (eds), Molecular Systematics, 2nd edition, Sinauer Associates, 407-514.

[30] Sukosd, Z., Knudsen, B., Kjems, J., Pedersen, C. N. S. (2012) *PPfold 3.0: PPfold 3.0: Fast RNA secondary structure prediction using phylogeny and auxiliary data*, Bioinformatics 28 (16).

[31] Sukosd, Z., Knudsen, B., Kjems, J., Pedersen, C. N. S. (2014) *PPfold*, http://daimi.au.dk/~compbio/pfold/downloads.html

[32] *RNA STRAND v2.0 - The RNA secondary STRucture and statistical ANalysis Database*, http://www.rnasoft.ca/strand/

[33] Andronescu, M., Bereg, V., Hoos, H. H., Condon, A. (2008) *RNA STRAND: The RNA Secondary Structure and Statistical Analysis Database*, BMC Bioinformatics, 9 (1): 340.

[34] Westbrook, J., Feng, Z., Chen, L., Yang, H., Berman, H. (2003) *The Protein Data Bank and structural genomics*, Nucleic Acids Res, 31: 489-491.

[35] Cannone, J., Subramanian, S., Schnare, M., Collett, J., D'Souza, L., Du, Y., Feng, B., Lin, N., Madabusi, L., Muller, K., Pande, N., Shang, Z., Yu, N., Gutell, R. (2002) *The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs*, BMC Bioinformatics, 3:15.

[36] Andersen, E. S., Rosenblad, M. A., Larsen, N., Westergaard, J. C., Burks, J., Wower, I. K., Wower, J., Gorodkin, J., Samuelsson, T., Zwieb, C. (2006) *The tmRDB and SRPDB resources*, Nucleic Acids Res, 34 (Database issue): 163-168.

[37] Sprinzl, M., Vassilenko, K. (2005) *Compilation of tRNA sequences and sequences of tRNA genes*, Nucleic Acids Res, 33 (Database issue): 139-140.

[38] Brown, J. (1999) *The Ribonuclease P Database*, Nucleic Acids Res, 27: 314-314.

[39] Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S., Bateman, A. (2005) *Rfam: annotating non-coding RNAs in complete genomes*, Nucleic Acids Res, 33 (Database issue): 121-124.

[40] Berman, H. M., Olson, W. K., Beveridge, D. L., Westbrook, J., Gelbin, A., Demeny, T., Hsieh, S. H., Srinivasan, A. R., Schneider, B. (1992) *The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids*, Biophysical Journal, 63 (3): 751-759.

[41] Tropp, B. E. (2010) *Molecular Biology: Genes To Proteins*, Jones & Bartlett Learning, 3rd edition.

[42] Johnson, M. (last update 2013) *Inside-Outside algorithm for estimating PCFGs from terminal strings*, http://web.science.mq.edu.au/~mjohnson/Software.htm

[43] Johnson, M. (last update 2006) *CKY PCFG parser*, http://web.science.mq.edu.au/~mjohnson/Software.htm

[44] Allali, J., *MiGaL*, http://www-igm.univ-mlv.fr/~allali/migal/index.php

[45] Allali, J., Sagot, M. F. (2008) *A Multiple Graph Layers Model with Application to RNA Secondary Structures Comparison*, Software-Practice & Experience Volume 38 Issue 8, 775-792.

[46] *jViz.Rna 2.0 Visual Comparison of RNA Secondary Structure*, http://jviz.cs.sfu.ca/index.html

[47] Wiese, K. C., Glen, E., Vasudevan, A. (2005) *jViz.Rna - A Java Tool for RNA Secondary Structure Visualization*, IEEE Transactions on NanoBioscience, Vol. 4 (3), 212-218.