VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA A TRANSFORMACE KÓDŮ ZALOŽENÁ NA GRAMATIKÁCH

CODE ANALYSIS AND TRANSFORMATION BASED ON GRAMMARS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATÚŠ ARBET**
AUTHOR

VEDOUCÍ PRÁCE **prof. RNDr. ALEXANDER MEDUNA, CSc.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Arbet Matúš**
Program: Informační technologie
Název: **Analýza a transformace kódů založená na regulovaných gramatikách**
Code Analysis and Transformation Based on Regulated Grammars
Kategorie: Překladače
Zadání:

1. Seznamte se s regulovanými gramatikami dle instrukcí vedoucího.
2. Seznamte se s metodami analýzy a transformace kódů dle instrukcí vedoucího. Zaměřte se na metody používané v bioinformatice.
3. Upravte metody z bodu 2 tak, aby byly založeny na regulovaných gramatikách.
4. Studujte užití metod navržených v předchozím bodě dle instrukcí vedoucího. Zaměřte se na bioinformatiku.
5. Dle instrukcí vedoucího implementujte vybrané metody z bodu 4.
6. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Meduna, A.: Automata and Languages, Springer, London, 2000

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Meduna Alexander, prof. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 27. října 2020

Abstrakt

Táto práca sa zaoberá analýzou a transformáciou kódu založenej na gramatikách. Práca obsahuje matematický základ operácií použitých v gramatikách a automatoch. Ich definície, sú sprevádzané príkladmi. V závere je popísaný návrh a implementácia aplikácie, so zameraním na oblasť bioinformatiky, založenej na regulovaných gramatikách.

Abstract

This thesis is concerning with Code Analysis and Transformation Based on Grammars. The work contains mathematical basics of operations used in grammars and automata. Their definitions are accompanied by examples. The design and implementation of the application with focus on the field of bioinformatics, based on regulated grammars is discussed at the end of the theses.

Kľúčové slová

konečný automat, zásobníkový automat, rozšírený zásobníkový automat, konfigurácia, gramatika, regulovaná gramatika, DNA, RNA, kodón, aminokyselina, transkripcia, translácia

Keywords

finate autmata, pushdown automata, extended pushdown automata, configuration, grammar, regulated grammar, DNA, RNA, codon, amino acid, transcription, translation

Citácia

ARBET, Matúš. *Analýza a transformace kódů založená na gramatikách.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexander Meduna, CSc.

Analýza a transformace kódů založená na gramatikách

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána prof. RNDr. Alexandra Medunu CSc.. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Matúš Arbet
3. mája 2021

Poďakovanie

Rád by som sa poďakoval pánovi prof. RNDr. Alexandrovi Medunovi CSc. za jeho odbornú pomoc pri tvorbe tejto práce a jeho cenné rady. Taktiež by som rád poďakoval pani Ing. Ivane Burgetovej Ph.D., za poskytnutie materiálov z oblasti molekulárnej biológie, ktoré poslúžili ako základ práce.

Obsah

Kapitola 1

Úvod

Bioinformatika je relatívne novým, ale rýchlo sa rozvíjajúcim odvetvím. Využíva sa nielen v biologickom výskume, ale aj pri vývoji liekov, lekárskej diagnostike, poľnohospodárstve a predovšetkým genetike. Hlavnými témami sú vyhľadávanie génov, porovnávanie sekvencií, pravdepodobnosti proteínových štruktúr, vzájomnými vzťahmi medzi génmi. Gény sú základnou zložkou pri tvorbe proteínov, ktoré sú nositeľmi rôznych funkcií buniek organizmov. Gény určujú napríklad farbu vlasov, očí, ale aj hrajú významnú rolu v ľudskom zdraví. Bioinformatika predstavuje tvorbu a vývoj algoritmov, metód a teórií, ktoré riešia problémy pri správe a analýze dát.

Táto práca sa zaoberá aplikovaním regulovaných gramatík v procese analýzy a prekladu DNA, mRNA sekvencií s cieľom automatizácie a efektívnejšieho riešenia doposiaľ využívaných metód. Jedna zo stávajúcich metód je využitie stochastickej bezkontextovej gramatiky. Tá využíva množinu pravdepodobnosti na produkčných pravidlách. Výpočet pravdepodobnosti derivácie môže byť náročné a môže viesť k nepresnosti výsledkov. Regulované gramatiky sa snažia tento problém redukovať. V práci sa počíta s minimálnou znalosťou danej problematiky, definícií a pojmov, ktoré sú v danej oblasti bežne zaužívané. V jednotlivých kapitolách práce sú definované pojmy, ktoré na seba postupne nadväzujú a sú nevyhnutné pre pochopenie výrazov a pojmov v nasledujúcej kapitole.

Teoretická časť práce pozostáva zo štyroch kapitol. V prvej kapitole sú definované základné matematické operácie, s ktorými sa budete stretávať v nasledujúcich kapitolách. Prvá kapitola, kde sú tieto pravidlá využívané, je kapitola o bezkontextových gramatikách. Tu sa nachádza definícia bezkontextovej gramatiky, derivácie, generovaného jazyka a neskôr nadväzujúcej regulovanej gramatiky. Pre lepšie pochopenie sú ku každej gramatike priložené príklady, ktoré ukazujú, ako každá gramatika pracuje. V ďalšej kapitole sú definované modely automatov. Tie slúžia, ako základná štruktúra pre popis bezkontextovej gramatiky. Na začiatok je definovaný konečný automat, na ktorý nadväzuje zásobníkový a rozšírený zásobníkový automat. Tie sú vo svojej podstate konečné automaty rozšírené o zásobník, s malými odlišnosťami v procese spracovávania vstupov. Rovnako ako v predošlej kapitole, definície sú sprevádzané príkladmi znázorňujúce princíp fungovania. V závere kapitoly je definovaný regulovaný zásobníkový automat a jeho vzťah so zásobníkovým automatom. Regulovaná gramatika síce nezvyšuje silu pri preklade, v porovnaní s obyčajným zásobníkovým automatom, ale poskytuje jednoduchší a prehľadnejší zápis pravidiel, jeho definovanie a využitie pre účel transformácie DNA, mRNA, ktorými sa táto práca zaoberá.

V poslednej kapitole teoretickej časti sú prezentované základné pojmy z oblasti molekulárnej biológie. Tie sú nevyhnutné pre pochopenie procesov translácie a transkripcie, ktoré výsledná aplikácia simuluje. Kapitola je sprevádzaná obrázkami, na ktorých sú znázornené

prvky DNA, mRNA na úrovni Biochémie. Aj keď sa v práci nezaoberáme jednotlivými atómami molekúl, niektoré pojmy si je zložité predstaviť a môžu byť mätúce. Čo je nukleotid, nukleová kyselina, polynukleotidový reťazec, ako to vyzerá, čo sa kde presne na čo viaže a podobne. Taktiež rozdiely medzi DNA a mRNA, proces spracovávania, ich štruktúry sú obsiahnuté v tejto kapitole. Tieto rozdiely sú kľúčovým prvkom v pochopení procesov replikácie, transkripcie a translácie. V akom poradí sa odohrávajú, aký je ich jednotlivý vstup a výstup. Podľa toho je následne možné navrhnúť a definovať automaty v každom procese prekladu DNA, mRNA. Tieto automaty predstavujú základnú štruktúru pre regulované gramatiky.

Druhá polovica práce popisuje návrh aplikácie a detailnejšie popisuje implementáciu jednotlivých častí. Implementácia využíva jazyk Python verzia 3.9. Tento jazyk je použitý v implementácií, kvôli jeho vlastnostiam a jednoduchému použitiu. Hlavným pozitívom jazyka v rámci implementácie je napríklad trieda list, ktorá ponúka vlastnosti zásobníku. V tejto kapitole sú definované jednotlivé triedy, automaty a popísaný proces generovania vstupnej sekvencie.

V závere je spomenuté testovanie aplikácie. To je nevyhnutné v overení správneho fungovania a splnenia požiadaviek, ktoré sú od aplikácie vyžadované. Priebeh testovania prebieha v etapách. V prvej etape testovania je zisťované, či aplikácia zvládne detekovať vstupné chybové zápisy. V ďalšej etape je testované, či aplikácia vykonáva správne operácie pri vstupných parametroch. Tie určujú či aplikácia má vstupnú sekvenciu vygenerovať, prečítať zo súboru, alebo je manuálne zadaná. V závere testovania je nutné jednotlivé časti spojiť v celok a zistiť, či aplikácia dokáže bezchybne vykonávať preklad, ktorý je ešte následne porovnávaný s referenčnou aplikáciou Expasy Traslate Tool[1] od Swiss Institute of Bioinformatics.

[1]Dostupná na https://web.expasy.org/translate/

Kapitola 2

Základné matematické pojmy

V tejto kapitole sú definované matematické pojmy, ktoré sú využité pri definovaní gramatík, pravidiel týchto gramatík, alebo znázornené v demonštračných príkladoch. Definície sú prevzaté z [8] a [5].

2.1 Množina

Množina je súhrn ľubovoľných navzájom rozlíšiteľných objektov. Objekty, ktoré patria do množiny, nazývame prvkami tejto množiny. Množiny obvykle značíme veľkými písmenami a ich prvky malými písmenami latinskej abecedy. Zápis $a \in A$ značí, že a je prvkom množiny A. $a \notin A$ značí, že a nie je prvkom množiny A. Obvykle ich zapisujeme tak, že do zložených zátvoriek zapíšeme jej jednotlivé prvky alebo pomocou tzv. charakteristických vlastností ako napr.: $\{ x \in \mathbb{N}, 12 < x < 17\}$, značí množinu tvorenú prvkami 13, 14, 15, 16.

2.1.1 Rozdelenie množín

- Konečné – konečný počet prvkov.
- Nekonečné – tvorené nekonečne mnoho prvkami.
- Prázdne – neobsahujú žiadny prvok. Značíme ju symbolom $\emptyset$.
- Neprázdne – obsahujú aspoň jeden prvok.

2.1.2 Základné množinové operácie

- Zjednotenie $A \cup B$ množín A, B je množina tých prvkov, ktoré patria aspoň do jednej z týchto množín.
- Prienik $A \cap B$ je množina tých prvkov, ktoré patria do oboch množín súčasne. Ak $A \cap B = \emptyset$, hovoríme, že množiny A, B sú disjunktné.
- Rozdiel $A - B$ je množina tých prvkov z A, ktoré nepatria do množiny B.
- Doplnok (komplement) A' značí doplnok množiny A tak, že vznikne rozdielom v nejakej pevne zvolenej množine U tzn. $A' = U - A$.
- Karteziánsky súčin $A \times B$ množín A, B je množina všetkých usporiadaných dvojíc $[x, y]$, takých kde $x \in A$, $y \in B$.

Príklad 2.1.1 Majme množiny $A = \{0, 1, 2\}$, $B = \{-1, 0\}$ a $U = \{a, b\}$. Aplikovaním základných množinových operácii dostávame:

$$A \cup B = \{-1, 0, 2\}$$
$$A \cap B = \{0\}$$
$$A - B = \{1, 2\}$$
$$A' = U - A = \{a, b\}$$
$$B \times U = \{-1a, -1b, 0a, 0b\}$$

2.2 Abeceda a reťazce

Abeceda a symboly

Abeceda je konečná, neprázdna množina elementov, ktoré nazývame symboly. Ak označíme abecedu Σ, potom $\Sigma = \{a, b, 0, 1\}$ je abeceda obsahujúca symboly a, b, 0, a 1.

Reťazec

Nech Σ je abeceda.

1. ε[1] je reťazec na abecedou Σ.
2. Ak x je reťazec nad Σ a $a \in \Sigma$, potom xa je reťazec nad abecedou Σ.

Dĺžka reťazca

Nech x je reťazec nad abecedou Σ. Dĺžka reťazca x, $\mid x \mid$, vyjadruje celkový počet symbolov v reťazci x a je definovaná:

1. Ak $x = \varepsilon$, potom $\mid x \mid = 0$.
2. Ak $x = a_1..a_n$, potom $\mid x \mid = n$, pre $n \geq 1$ a $a_i \in \Sigma$ pre všetky $i = 1, .., n$.

Konkatenácia reťazcov

Nech x a y sú dva reťazce nad abecedou Σ. Konkatenácia x a y je reťazec xy[2].

Mocnina reťazca

Nech x je reťazec nad abecedou Σ. Pre $i \geq 0$, i-ta mocnina reťazca x, x^i, je definovaná:

1. $x^0 = \varepsilon$
2. Pre $x \geq 1 : x^i = xx^{i-1}$

Príklad 2.2.1 Majme abecedu $\Sigma = \{0, 1\}$. A reťazce $x = 010$ a $y = 1$ nad abecedou Σ. Nezabúdajme pritom, že aj ε je reťazcom nad abecedou Σ. Potom dĺžky zmienených reťazcov sú nasledovné:

[1] Pozn.: ε označuje prázdny reťazec tzn. neobsahuje žiadny symbol.
[2] Pozn.: $\varepsilon x = x\varepsilon = x$

$$| x | = 3, | y | = 1, | \varepsilon | = 0,$$

Konkatenáciou reťazcov dostaneme:

$$xy = 0101,\ y\varepsilon = 1,\ \varepsilon x = 010$$

Tretie mocniny reťazcov x a y sú nasledovné:

$$x^3 = 010010010,\ y^3 = 111$$

2.3 Jazyk

Nech Σ^* značí množinu všetkých reťazcov nad Σ. Každá podmnožina $L \subseteq \Sigma^*$ je jazyk nad Σ.

Konkatenácia jazykov

Nech L_1 a L_2 sú dva jazyky nad Σ. Konkatenácia jazykov L_1 a L_2, L_1L_2 je definovaná:

$$L_1L_2 = \{xy : x \in L_1 \text{ a } y \in L_2\}$$

Mocnina jazyka

Nech L je jazyk nad abecedou Σ. Pre $i \geq 0$, i-ta mocnina jazyka L, L^i, je definovaná:

1. $L^0 = \{\varepsilon\}$
2. Pre $i \geq 1 : L^i = LL^{i-1}$

Operácie nad jazykmi

Nech L_1 a L_2 sú dva jazyky nad Σ. Pre ktoré platí:

1. Zjednotenie $L_1 \cup L_2 = \{x : x \in L_1 \text{ alebo } x \in L_2\}$
2. Prienik $L_1 \cap L_2 = \{x : x \in L_1 \text{ a } x \in L_2\}$
3. Rozdiel $L_1 - L_2 = \{x : x \in L_1 \text{ a } x \notin L_2\}$

Ďalej si v úvahe vezmeme jazyk L nad abecedou Σ. Potom doplnkom jazyka L, označovaný $\overline{L}$, je definovaný:

$$\overline{L} = \Sigma^* - L$$

Iterácie jazyka

Nech L je jazyk nad abecedou Σ. Iterácia jazyka L,L^*, a pozitívna iterácia jazyka L,L^+ sú definované:

$$L^* = \bigcup_{n=0}^{\infty} L^i$$

$$L^+ = \bigcup_{n=1}^{\infty} L^i$$

Príklad 2.3.1 Majme abecedu $\Sigma = \{0, 1\}$ a jazyky $L_1 = \{0, 01\}$ a $L_2 = \{1, 01\}$ nad abecedou Σ. Predošle definovanými operáciami nad týmito jazykmi dostaneme:

$$L_1 \cup L_2 = \{0, 1, 01\}$$
$$L_1 \cap L_2 = \{01\}$$
$$L_1 - L_2 = \{0\}$$
$$L_1 L_2 = U - A = \{01, 001, 011, 0101\}$$

Ďalej si definujme jazyk $L = \{10, 11\}$ potom:

$$\overline{L} = \Sigma^* - \{10, 11\}$$
$$L^2 = \{1010, 1011, 1110, 1111\}$$
$$L^* = \{\varepsilon, 10, 11, 1010, 1011, 1110, 1111, ...\}$$
$$L^+ = \{10, 11, 1010, 1011, 1110, 1111, ...\}$$

Kapitola 3

Gramatiky

Bezkontextová gramatika definuje bezkontextový jazyk, ktorého slová sa tvoria nezávisle na predchádzajúcich krokoch. Je tvorená *neterminálmi*, ktoré reprezentujú premenné, *terminálmi*, ktoré reprezentujú konštanty a pravidlá. Tie každému *neterminálu* definujú, na čo sa dá prepísať. Bezkontextové gramatiky tvoria neodmysliteľný základ pre pochopenie regulovaných gramatík, ktoré sú na nich založené. Nasledujúca kapitola vychádza z poznatkov knihy [8], ktorá je dobrým úvodom do problematiky.

3.1 Bezkontextová gramatika

Bezkontextová gramatika je štvorica $G = (N, T, P, S)$, kde:

- N je abeceda *neterminálov*
- T je abeceda *terminálov*, kde $N \cap T \neq \emptyset$
- P je konečná množina pravidiel v tvare $A \to x$, kde $A \in N$, $x \in (N \cup T)^*$
- S je počiatočný symbol, kde $S \in N$

Pre používanie gramatík boli zavedené konvencie pre lepšie pochopenie. *Neterminály* sa obvykle zapisujú veľkými písmenami a *terminály* symbolmi, číslicou, alebo malými písmenami. Konečná množina P je v podstate relácia z N do $(N \cup T)^*$, ktorú miesto relačného zápisu $(A, x) \in P$ zapisujeme ako $p : A \to x$, kde p značí poradie pravidla v množine. Toto označenie môže byť buď číselné $1, 2, \ldots, n$ alebo písmenami malej abecedy s príslušným indexom napr.: $p_1, p_2, \ldots, p_n$, kde $n \geq 1$.

3.1.1 Priama derivácia

Nech $G = (N, T, P, S)$ je bezkontextová gramatika, kde $p : A \to w \in P$ a $x, y \in (N \cup T)^*$. Potom xAy priamo derivuje xwy, ktorú zapíšeme ako $xAy \Rightarrow xwy[p]$, alebo zjednodušene $xAy \Rightarrow xwy$.

3.1.2 Sekvencia derivačných krokov

Majme bezkontextovú gramatiku $G = (N, T, P, S)$, kde $u \in (N \cup T)^*$. G vykoná nula derivačných krokov z u do u, ktoré zapíšeme:

$$u \Rightarrow^0 u[\varepsilon]$$

zjednodušene:

$$u \Rightarrow^0 u$$

Pre $u_0, \ldots, u_n \in (N \cup T)^*$ a $p_i \in P$, kde $n \geq 1$, $i = 1, \ldots, n$, jednotlivé derivačné kroky vyzerajú nasledovne:

$$u_0 \Rightarrow u_1[p_1] \Rightarrow u_2[p_2] \ \ldots \ \Rightarrow u_n[p_n]$$

Potom n-tú deriváciu podľa gramatiky G z u_0 do u_n zapisujeme:

$$u_0 \Rightarrow^n u_n[p_1, \ldots, p_n]$$

zjednodušene:

$$u_0 \Rightarrow^n u_n$$

Matematicky, $\Rightarrow^n$ označuje n-tú mocninu $\Rightarrow$. Na základe toho $\Rightarrow^+$ reprezentuje tranzitívny uzáver, ak $n \geq 1$ a $\Rightarrow^*$ reflexívny uzáver ak $n \geq 0$.

3.1.3 Generovaný jazyk

Nech $G = (N, T, P, S)$ je bezkontextová gramatika, ak $S \Rightarrow^* w$ v G. Potom w označuje *vetnú formu*. Ak $w \in T^*$, potom w je *veta* generovaná gramatikou G. Generovaný jazyk L(G), je množina všetkých *viet*, ktoré gramatika G generuje a je definovaná:

$$L(G) = \{w : w \in T^*, S \Rightarrow^* w\}$$

Príklad 3.1.1 Majme gramatiku $G = (N, T, P, S)$ a jazyk $L(G) = \{a^n b^n : n \geq 0\}$, kde:

- $N = \{S\}$
- $T = \{a, b\}$
- $P = \{1 : S \to aSb, 2 : S \to \varepsilon\}$

Gramatika G generuje nasledujúce reťazce:

$S \Rightarrow^* \varepsilon$

$\Rightarrow^* ab[12]$

$\Rightarrow^* aabb[112]$

$\Rightarrow^* aaabbb[1112]$

$\vdots$

3.2 Regulovaná gramatika

Regulované gramatiky sú postavené na bezkontextových gramatikách, rozšírené o matematický mechanizmus, ktorým sa kontroluje použitie pravidiel pri generovaní jazyka. Týmto pracujú viac deterministicky než neregulované bezkontextové gramatiky. To zaručuje silnejšiu generatívnu silu. Existujú rôzne typy regulovaných gramatík, ale v tejto práci sa

budeme zaoberať len dvoma typmi. Predtým než, ale tak urobíme, potrebujeme poznať odpoveď na otázku: Prečo sa v tejto práci zaoberáme regulovanými gramatikami? V knihe [6] sú popísané metódy molekulárnej biológie v bioinformatike. Kde sa dočítame o použití takzvaných stochastických bezkontextových gramatík. Stochastické bezkontextové gramatiky boli prvý krát použité v kontexte modelovania prirodzeného jazyka. Až neskôr našli svoje uplatnenie v predikcii sekundárnej štruktúry RNA. Detailnejší popis o tom čo je RNA je venovaná kapitola 5. Stochastická bezkontextová gramatika je takmer rovnako definovaná ako v kapitole 3.1 s tým rozdielom, že množina pravidiel P je asociovaná s pravdepodobnosťou. Takže zápis $A \to x$ v P vyzerá nasledovne: $Pr(A \to x)$. U stochastických gramatík, ale vzniká problém počítania pravdepodobnosti. Toto je riešené zavedením poľa pravidiel a postupnosti. Toto pole pripomína bezkontextovú gramatiku obohatenú o tzv. regulovaný jazyk v regulovaných gramatikách. Nasledujúce definície gramatík vychádzajú z knihy [1].

3.2.1 Stavová gramatika

Stavová gramatika je bezkontextová gramatika rozšírená o mechanizmus stavov. V derivačnom kroku, gramatika prepisuje najľavejší *neterminál* a zároveň mení *stav*, v ktorom sa nachádza a ktorý ovplyvňuje voľbu pravidla použitú v nasledujúcom kroku. Stavová gramatika je pätica $G = (V, Q^1, T, P, S)$, kde:

- V je celková *abeceda*
- Q je konečná množina *stavov*
- $T \subset V$ je abeceda *terminálov*
- $S \in V - T$ je *počiatočný neterminál*
- $P \subseteq (Q \times (V - T)) \times (Q \times V^+)$ je konečná množina *pravidiel*

Používame zápis $(q, A) \to (p, v) \in P$ namiesto zápisu $(q, A, p, v) \in P$. Ak $(q, A) \to (p, v) \in P, x, y \in V^*$ tak potom zápis derivačného kroku (q, xAy) do (q, xvy) vyzerá nasledovne:

$$(q, xAy) \Rightarrow (q, xvy)[(q, A) \to (p, v)]$$

K-tá mocnina *derivačného kroku* pre $k \geq 0$ je taktiež značená $\Rightarrow^k$. Obdobne sa označuje aj reflexívny $\Rightarrow^*$ a tranzitívny $\Rightarrow^+$ uzáver. Potom jazyk generovaný stavovými gramatikami G zapisovaný $L(G)$ je definovaný ako:

$$L(G) = \{w \in T^* \mid (q, S) \Rightarrow^* (p, w), q, p \in Q\}.$$

Príklad 3.2.1 Majme stavovú gramatiku $G = (V, Q, T, P, S)$ kde:

- $V = \{S, A, B, a, b\}$
- $Q = \{q_1, q_2, p_1, p_2\}$
- $T = \{a, b\}$
- $P = \{(1.p_1, S) \to (q_1, AB), \quad (2.q_1, A) \to (q_2, aA), \quad (3.q_2, B) \to (q_1, bBa),$
  $(4.q_1, A) \to (p_2, a), \quad (5.p_2, B) \to (p_2, ba)\}$

[1] V knižnej definícii často označované ako W, ale kvôli jednotnosti použitého zápisu použijeme označenie Q.

Jazyk generovaný gramatikou G je definovaný ako $L(G) = \{a^k b^k a^k, k \geq 1\}$ a jednotlivé derivačné kroky sú nasledovné:

$$\begin{aligned}(p_1, S) &\Rightarrow (q_1, AB)[(1.p_1, S) \rightarrow (q_1, AB)] \\ &\Rightarrow (q_2, aAB)[(2.q_1, A) \rightarrow (q_2, aA)] \\ &\Rightarrow (q_1, aAbBa)[(3.q_2, B) \rightarrow (q_1, bBa)] \\ &\Rightarrow (p_2, aabBa)[(4.q_1, A) \rightarrow (p_2, a)] \\ &\Rightarrow (p_2, aabbaa)[(5.p_2, B) \rightarrow (p_2, ba)]\end{aligned}$$

3.2.2 Gramatika s rozptýleným kontextom

Gramatika s rozptýleným kontextom G je založená na sekvencií pravidiel bezkontextových gramatík. To umožňuje, aby v jednom *derivačnom kroku* bolo prepísaných viac *neterminálov* zároveň. Táto gramatika je definovaná ako štvorica $G = (V, T, P, S)$, kde:

- V je celková *abeceda*
- $T \subset V$ je abeceda *terminálov*
- P je konečná množina *pravidiel* vo forme $(A_1, \ldots, A_n) \Rightarrow (x_1, \ldots, x_n)$, kde $n \geq 1, A_i \in V - T, x_i \in V^*$ pre všetky i, kde $1 \geq i \geq n$
- $S \in V - T$ je *počiatočný neterminál*

K-tá mocnina *derivačného kroku* pre $k \geq 0$ je značená ako $\Rightarrow_G^k$, reflexívny $\Rightarrow_G^*$ a tranzitívny $\Rightarrow_G^+$ uzáver. Potom jazyk generovaný gramatikou s rozptýleným kontextom G zapisovaný $L(G)$ je definovaný ako:

$$L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}.$$

Príklad 3.2.2 Majme gramatiku s rozptýleným kontextom $G = (V, T, P, S)$ kde:

- $V = \{S, A, B, C, a, b, c\}$
- $T = \{a, b, c\}$
- $P = \{(S) \rightarrow (ABC), (A, B, C) \rightarrow (aAa, bBb, cCc), (A, B, C) \rightarrow (a, b, c)\}$

Jazyk generovaný gramatikou G je definovaný ako $L(G) = \{a^k b^k c^k, (2K + 1) \geq 0\}$ a jednotlivé derivačné kroky sú nasledovné:

$$\begin{aligned}(S) &\Rightarrow (ABC)[(S) \rightarrow (ABC)] \\ &\Rightarrow (aAabBbcCc)[(A, B, C) \rightarrow (aAa, bBb, cCc)] \\ &\Rightarrow (aaAaabbBbbccCcc)[(A, B, C) \rightarrow (aAa, bBb, cCc)] \\ &\Rightarrow (aaaaaabbbbbbcccccc)[(A, B, C) \rightarrow (a, b, c)]\end{aligned}$$

Kapitola 4

Automaty

Ďalším modelom pre popis bezkontextového jazyka sú automaty. Automat si je možné predstaviť ako zariadenie, ktoré postupnými krokmi v určitej postupnosti, závislosti na vstupnej množine symbolov a množine stavov, rozhodne či je vstupné slovo prijaté, alebo nie. Vstupné slovo je prijaté len vtedy, ak je načítaný celý vstupný reťazec a automat sa nachádza v koncovom stave. Množinu všetkých vstupných slov prijatú automatom tvorí jazyk. Definície automatov sú prevzaté z kníh [8], [2] a [1].

4.1 Konečný automat

Konečný automat je pätica $M = (Q, \Sigma, R, s, F)$, kde:

- Q je konečná množina *stavov*
- Σ je vstupná *abeceda*
- $R \subseteq Q \times \Sigma^*$[1] $\times Q$ je konečná množina *pravidiel*
- $s \in Q$ je *počiatočný stav*
- $F \subseteq Q$ je množina *koncových stavov*

Vždy predpokladáme, že množiny Q a Σ sú disjunktné tzn. $Q \cap \Sigma = \emptyset$. Používame zápis $pa \to q \in R$ namiesto zápisu $(p, a, q) \in R$. Ak konečný automat M má byť deterministický, znamená to, že $a \neq \varepsilon$. *Konfigurácia* automatu M je každý reťazec χ z $Q\Sigma^*$. Relácia *priameho prechodu* je symbolicky značená $\vdash$ a definovaná ako $pax, qx \in Q\Sigma^*$, ak $pa \to q \in R$, tak potom:

$$pax \vdash qx.$$

K-tá mocnina *prechodu* pre $k \geq 0$ je značená $\vdash^k$. Obdobne sa označuje aj reflexívny $\vdash^*$ a tranzitívny $\vdash^+$ uzáver. Jazyk prijatý automatom M zapisovaný $L(M)$ je definovaný ako:

$$L(M) = \{w \in \Sigma^* \mid sw \vdash^* f, f \in F\}.$$

[1] V knižnej definícií taktiež zaznačené ako $(\Sigma \cup \{\varepsilon\})$

4.2 Zásobníkový automat

Zásobníkový automat je sedmica $M = (Q, \Sigma, \Gamma, R, s, S, F)$, kde:

- Q, Σ, s a F sú rovnako definované, ako v prípade konečného automatu viď. 4.1
- Γ je *zásobníková abeceda*
- $R \subseteq \Gamma \times Q \times \Sigma^* \times \Gamma^* \times Q$ je konečná množina *pravidiel*
- $S \in \Gamma$ je *počiatočný symbol* v zásobníku

Taktiež rovnako predpokladáme, že množina Q je disjunktná s množinami Σ a Γ tzn. $Q \cap \Sigma = \emptyset$, $Q \cap \Gamma = \emptyset$. Používame zápis $Apa \to wq \in R$ namiesto zápisu $(A, w, p, a, q) \in R$. *Konfigurácia* automatu M je každý reťazec χ z $Q\Sigma^*\Gamma^*$. Relácia *priameho prechodu* je rovnako symbolicky značená $\vdash$ a definovaná ako $xApay, xwqy \in Q\Sigma^*\Gamma^*$, ak $Apa \to wq \in R$, tak potom:

$$xApay \vdash xwqy.$$

Obr. 4.1: Znázornenie zásobníkového automatu v stave p

K-tá mocnina *prechodu* pre $k \geq 0$ je taktiež značená $\vdash^k$. Obdobne sa označuje aj reflexívny $\vdash^*$ a tranzitívny $\vdash^+$ uzáver. Existujú tri možnosti, v ktorých automat prijme jazyk:

1. Automat prejde do koncového stavu
2. Vyprázdnením zásobníka
3. Prejdením do koncového stavu a vyprázdnením zásobníka

Jazyk prijatý automatom M pri prechode do koncového stavu, zapisovaný $L(M)_f$ je definovaný ako:

$$L(M)_f = \{w \in \Sigma^* \mid Ssw \vdash^* \gamma f, f \in F, \gamma \in \Gamma^*\}.$$

Jazyk prijatý automatom M vyprázdnením zásobníka, zapisovaný $L(M)_e$ je definovaný ako:

$$L(M)_e = \{w \in \Sigma^* \mid Ssw \vdash^* q, q \in Q\}.$$

Jazyk prijatý automatom M vyprázdnením zásobníka a prechodom do koncového stavu, zapisovaný $L(M)_{ef}$ je definovaný ako:

$$L(M)_{ef} = \{w \in \Sigma^* \mid Ssw \vdash^* f, f \in Fs\}.$$

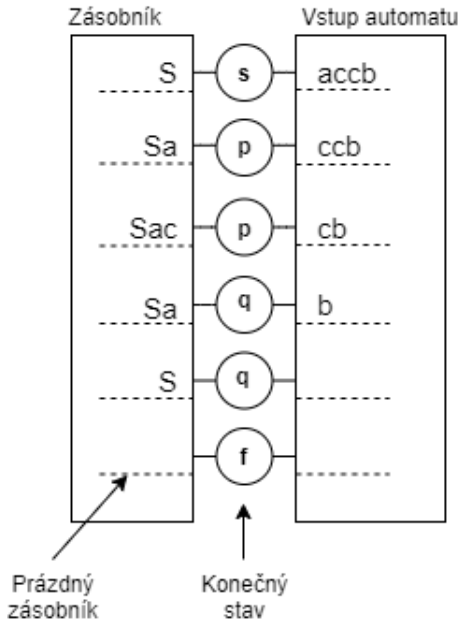
Príklad 4.2.1 Majme zásobníkový automat $M = (Q, \Sigma, \Gamma, R, s, S, F)$. Príjme automat reťazec *accb*? Ak je automat definovaný nasledovne:

- $Q = \{s, p, q, f\}$
- $\Sigma = \{a, b, c\}$
- $\Gamma = \{S\}$
- $R = \{Ssa \to Sap, apc \to acp, cpc \to q, aqb \to q, Sq \to f\}$
- $F = \{f\}$

Použitím pravidiel z množiny R dostaneme sekvenciu prechodov nasledovnú:

$$Ssaccb \vdash Sapccb \vdash Sacpcb \vdash Saqb \vdash Sq \vdash f.$$

Reťazec $accb$ je automatom M prijatý, keďže zásobník automatu je prázdny a zároveň automat prešiel do koncového stavu. Takže platí $accb \in L(M)_{ef}$.



Obr. 4.2: Grafické znázornenie prechodov

4.3 Rozšírený zásobníkový automat

Rozšírený zásobníkový automat je sedmica $M = (Q, \Sigma, \Gamma, R, s, S, F)$, kde $Q, \Sigma, \Gamma, s, S, F$ sú rovnako definované ako u definícii zásobníkového automatu v predošlej kapitole 4.2. Rozdiel je v zápise konečnej množiny pravidiel R, ktorá umožňuje čítanie reťazca z vrcholu zásobníka, kde v prípade zásobníkového automatu bolo možné čítanie len jedného symbolu. Toto pravidlo sa zapisuje v tvare: $vpa \to wq$, kde $v, w \in \Gamma^*, p, q \in Q, a \in \Sigma^*$. Ak máme dve konfigurácie rozšíreného automatu $xvpay$ a $xwqy$, kde $x, v, w, \in \Gamma^*, p, q \in Q, a, y \in \Sigma^*$ pri $vpq \to wq \in R$, potom priamy prechod z $xvpay$ a $xwqy$ pri použití pravidla z množiny R je značený obdobne ako u zásobníkového automatu a vyzerá nasledovne:

$$xvpay \vdash xwqy.$$

Zápis a značenie K-tej mocniny prechodu, tranzitívneho, reflexívneho uzáveru a jazyka prijatého rozšíreným zásobníkovým automatom, zostávajú rovnaké ako u zásobníkového automatu. Pre každý rozšírený zásobníkový automat M existuje zásobníkový automat M', pre ktorý platí $L(M)_f = L(M')_f$. Dôkaz sa nachádza v knihe [8].

4.4 Regulovaný zásobníkový automat

Regulované zásobníkové automaty sú predstavené v knihe [1]. Vychádzajú z konceptu zásobníkových automatov. Nasledujúca kapitola sa zaoberá zápisom a definovaním regulovaného zásobníkového automatu a prevodu regulovaného zásobníkového automatu regulovaným ľubovoľnou regulovanou gramatikou na zásobníkový automat. Regulovaný zásobníkový automat je dvojica $M_R = (M, \Xi)$, kde:

- $M = (Q, \Sigma, \Gamma, R, s, S, F)$je zásobníkový automat definovaný ako v kapitole 4.2,
- $\Xi \subseteq \Psi^*$ označuje riadiaci jazyk

Znak Ψ je množina symbolov označenia pravidiel, kde platí, že $card(\Psi) = card(R)$. Potom Ψ^* označuje množinu sekvencií všetkých označení pravidiel z Ψ. Konfigurácia, prechody a sekvencie zostávajú rovnako definované ako u zásobníkových automatov s rozdielom, že musí platiť $r \in \Xi$, kde r reprezentuje sekvenciu použitých pravidiel. Obdobne ako u zásobníkových automatov existujú tri možnosti kedy zásobníkový automat príjme jazyk, tak je tomu rovnako aj v prípade regulovaných zásobníkových automatoch, kedy:

- $L(M, \Xi, 1)$ jazyk je prijatý ak automat prejde do koncového stavu
- $L(M, \Xi, 2)$ jazyk je prijatý s vyprázdnením zásobníku
- $L(M, \Xi, 3)$ jazyk je prijatý ak automat prejde do koncového stavu s vyprázdnením zásobníku

Nech χ je konfigurácia regulovaného zásobníkového automatu M_R, kde $\chi \in \Gamma^* Q \Sigma^*$. Potom koncové konfigurácie pre $i = 1, 2, 3$, sú definované ako:

1. $\chi \in F$
2. $\chi \in Q$
3. $\chi \in \Gamma^* F$

Po zavedení koncových konfigurácii automatov, jazyky $L(M, \Xi, i)$ nimi prijaté sú definované nasledovne:

$$L(M, \Xi, i) = \{w \in \Sigma^* \mid Ssw \vdash^* \chi_i[r], r \in \Xi\}.$$

Ak je ale riadiaci jazyk regulárny, tak potom regulovanie automatu riadiacim jazykom nemá žiadny vplyv na jeho generatívnej sile. Dôkaz o tom nájdeme v knihe [1]. Jazyk sekvencie DNA a RNA je regulárny, keďže existuje regulárny výraz, ktorý tento jazyk označuje. Transformáciou na obyčajný zásobníkový automat sa zväčší prehľadnosť zápisu a zjednoduší samotnú implementáciu aplikácie. Transformácia prevedie ľubovoľnú regulárnu gramatiku G a ľubovoľný automat A na obyčajný zásobníkový automat M, kde potom platí $L(M) = L(A, L(G), i)$ pri $i \in 1, 2, 3$ koncovej konfigurácii automatu.

Príklad 4.4.1 Majme stavovú gramatiku $G = (V_G, Q_G, T_G, P_G, S_G)$. Zavedením dolného indexu G tak identifikujeme, že V, Q, T, P a S sú komponentami gramatiky G. Obdobne zapíšeme aj automat $A = (Q_A.\Sigma_A, \Gamma_A, R_A, s_A, S_A, F_A)$. Stavová gramatika G je definovaná nasledovne:

- $V_G = \{S, A, T, U, a, t, u\}$
- $Q_G = \{s, q_1, q_2, f\}$
- $T_G = \{a, t, u\}$
- $P_G = \{1.(s, A) \to (q_1, Sa), 2.(q_1, T) \to (q_2, t), 3.(q_2, U) \to (f, u)\}$
- $S_G = \{S\}$

Potom automat A je definovaný ako:

- $Q_A = \{s, q_1, q_2, f\}$
- $\Sigma_A = \{S, A, T, U\}$
- $\Gamma_A = \{S, a, t, u\}$
- $R_A = \{sA \to Saq_1, aq_1T \to tq_2, tq_2U \to uf\}$
- $s_A = \{s\}$
- $S_A = \{S\}$
- $F_A = \{f\}$

Transformáciou na zásobníkový automat $M = (Q_M, \Sigma_M, \Gamma_M, s_M, S_M, F_M, R_M)$ bude definovanie vyzerať nasledovne:

- $Q_M = \{q \in Q_A\}$
- $\Sigma_M = \Sigma_A$
- $\Gamma_M = \Gamma_A$
- $R_M = \{1.sA \to Saq_1 \in R_A, (s, A) \to (q_1, Sa) \in P_G\}$
  $\cup\{2.aq_1T \to tq_2 \in R_A, (q_1, T) \to (q_2, t) \in P_G\}$
  $\cup\{3.tq_2U \to uf \in R_A, (q_2, U) \to (f, u) \in P_G\}$
- $s_M = \{s_A S_G\}$
- $S_M = \{S_A\}$
- $F_M = \{q \in F_A\}$

Uskutočnenie prvého prechodu v zásobníku M je dané pravidlom $1.(s, A) \to (q_1, Sa) \in P_G$ v gramatike G. Pri pohľade na pravidlá v zásobníku je vidieť, že automat M príjme vstupný reťazec w, len ak sa automat A dostane do koncového stavu po prečítaní celého slova $L(G)$, teda platí $L(M) = L(A, L(G), 1)$

Kapitola 5

Molekulárna biológia

Molekulárna biológia je oblasť biológie, ktorá študuje štruktúru, zloženie a vzťahy molekúl buniek, ako napríklad nukleové kyseliny, proteíny, DNA a RNA. Nasledujúca kapitola vychádza z [9], [7] a [4].

5.1 Bunka

Bunka patrí medzi základné štruktúry rastlín a živočíchov. Má vlastný metabolizmus, a preto je tento organizmus schopný samostatného života či prenosu genetickej informácie rozmnožovaním. Každá bunka bez ohľadu na funkciu, tvar alebo druh sa skladá z jadra a obalu. Jadro je tvorené cytoplazmou, kde prebiehajú metabolické procesy a oblasť ktorá obsahuje genetickú informáciu. Obal je tvorený plazmatickou membránou. Existujú dva typy buniek:

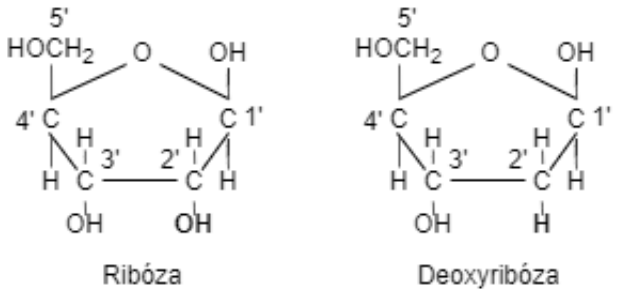
- Prokaryotická bunka - sú najpočetnejšie na zemi s jednoduchou štruktúrou. Obsahuje prokaryotické jadro (nukleoid) v cytoplazme spolu s deoxyribonukleovou kyselinou. Neobsahuje mitochondrie, plastidy a golgiho aparát. Ich veľkosť sa pohybuje v rozmedzí od 1 do 10μm.
- Eukaryotická bunka - má eukaryotické jadro oddelene od cytoplazmy. Toto jadro obsahuje deoxyribonukleovú kyselinu (DNA), ktorá je nositeľom genetickej informácie. Na rozdiel od prokaryotickej bunky, obsahuje mitochondrie, plastidy a golgiho aparát. Sú niekoľko násobne väčšie od prokaryotickej bunky o veľkosti od 10 do 100μm.

5.2 Nukleové kyseliny

Nukleové kyseliny tvoria nevyhnutnú časť každej bunky. Ich hlavnou funkciou je uchovávanie genetickej informácie a jej prenos do dcérskej bunky. Rozlišujeme dva druhy nukleových kyselín:

- kyselina deoxyribonukleová (DNA)
- kyselina ribonukleová (RNA)

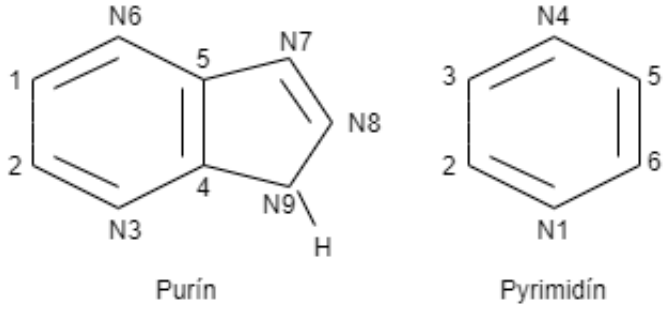
Obe sa skladajú z troch častí, a to z dusíkatej bázy (zásaditá časť), kyseliny fosforečnej (kyslá časť) a pentózy (cukornatá časť). O tom, o akú kyselinu sa jedná, rozhoduje sacharid v cukornatej časti. Pre DNA je to deoxyribóza a u RNA ribóza.



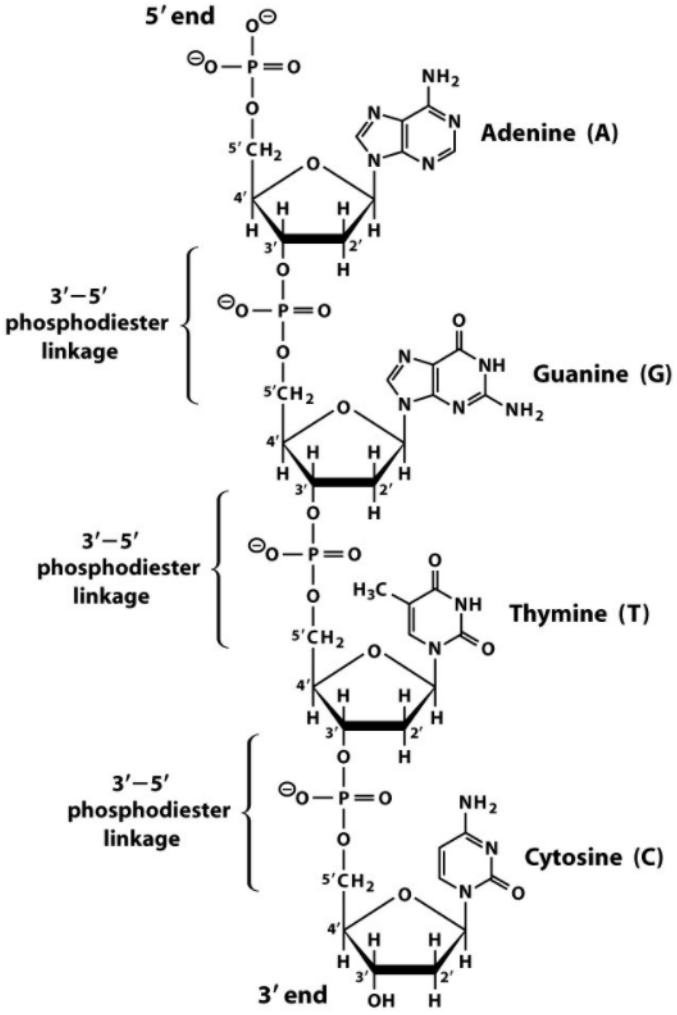
Obr. 5.1: Štruktúry pentóz

5.3 Nukleotidy a polynukleotidový reťazec

Molekuly nukleotidov sú štruktúrnymi derivátmi buď purínu alebo pyrimidínu. Najbežnejšími purínmi sú adenín (A) a guanín (G). U pyrimidínu je to cytozín (C), uracil (U) a tymín (T). Puríny vytvárajú väzby s pentózou (5−uhlíkový cukor) cez svoje $N9$ atómy dusíka, kde u pyrimidínov je to cez $N1$ atómy dusíka. Spojením dusíkatej bázy a pentózy vzniká nukleosid. Ich názvy sú odvodené od dusíkatej bázy, ktorú obsahujú, ako napr.: adenosin, cytidin, uridin a podobne. Ak sa na $5'$ atóm uhlíku pentózy naviaže kyselina fosforečná vzniká nukleotid (adenosinfosfat, uridinfosfat, atď.). Pomocou kyseliny fosforečnej sú jednotlivé nukleotidy pospájané v polynukleotidový reťazec medzi atómami uhlíka $5'$ a $3'$ ribózy, alebo deoxyribozy. Tento reťazec je základom štruktúry nukleových kyselín.



Obr. 5.2: Štruktúry purínu a pyrimidínu



Obr. 5.3: Príklad polynukleotidového reťazca[1]

5.4 Deoxyribonukleová kyselina (DNA)

DNA pozostáva z dvoch komplementárnych vlákien, ktoré formujú tvar tzn. dvojšrúbovice. Tento tvar vzniká pomocou vodíkových väzieb medzi dusíkovými bázami. Tieto bázy môžu obsahovať adenín (A), guanín (G), cytozín (C) a tymín (T). Adenín (A) tvorí pár s tymínom (T) pomocou dvoch vodíkových väzieb (A = T) . Cytozín (C) a guanín (G) tvoria pár pomocou troch vodíkových väzieb (C ≡ G). Vlákna DNA sú si navzájom antiparalelné, čo znamená, že jedno vlákno je orientované od 5′ konca po 3′ koniec. U druhého je to od konca 3′ po koniec 5′. Tieto konce označujú začiatok, koniec a smer polynukleoidového reťazca DNA, v ktorom je uchovaná genetická informácia.

5.4.1 Komplementarita

Princíp komplementárneho párovania báz je základným prvkom DNA štruktúr a má veľký význam v molekulárnej biológii. Párovanie báz je mechanizmus, ktorým je sekvencia mole-

[1]Prevzané z https://sandwalk.blogspot.com/2007/07/dna-is-polynucleotide.html

kuly DNA uchovaná počas replikácie, čo je rozhodujúce, ak sa informácia obsiahnutá v géne nesmie zmeniť alebo stratiť počas jej delenia. Ako už bolo spomenuté v predchádzajúcej kapitole 5.4, adenín (A) sa viaže s tymínom (T) a cytozín (C) s guanínom (G). Nasledujúci príklad popisuje tvorbu komplementárneho polynukleotidového reťazca DNA.

Príklad 5.4.1 Majme dva reťazce A a reťazec B. Reťazec A = CCATATGGCC, ktorý je orientovaný od $5'$ konca po koniec $3'$. Z tohto reťazca budeme vychádzať a podľa princípov komplementarity nám vznikne príslušný reťazec B. Na obrázku 5.4 je vidieť, že reťazec B je voči A antiparalelný a reťazec začína koncom $3'$ a končí s $5'$.

A: 5' - CCATATGGCC - 3'

B: 3' - GGTATACCGG - 5'

Obr. 5.4: Komplementárny reťazec B voči reťazcu A

5.4.2 Replikácia

Replikácia DNA je semikonzervatívna, tzn. z každej materskej molekuly DNA vzniknú dve dcérske molekuly DNA. Počas replikácie iba určitá oblasť materskej DNA je rozpletená. Replikácia nastáva po oddelení dvoch vlákien. To zahrňuje rozbitie slabých vodíkových väzieb, ktoré držia bázy protiľahlých vlákien pohromade. Táto oblasť, v ktorej nastane toto oddelenie vlákien sa označuje pod pojmom *replikačná vidlica*. Pre rozdelenie týchto dvoch vlákien sa používa enzým *helikáza*, ktorý zabraňuje ich okamžitému opätovnému spojeniu. Nové väzby s komplementárnym vláknom sú spojené pomocou enzýmu *polymerázy*, ktorý má za úlohu nielen vytvoriť väzby s komplementárnym vláknom, ale aj zabezpečiť to, že replikácia prebehne bez chýb a nová dcérska molekula DNA bude identická k materskej molekule DNA. Syntéza polymerázy prebieha iba v smere od $5'$ konca k $3'$ konca.

5.5 Ribonukleová kyselina(RNA)

RNA podobne ako DNA obsahuje nukleotidové väzby pomocou fosforečných väzieb. Hlavným rozdielom však je v cukornatej časti, kde sa u RNA nachádza už spomenutá ribóza. RNA pozostáva z jedného vlákna namiesto dvoch, ako v prípade DNA. Ďalší rozdiel je v obsahu zásaditých zložiek, kde je tymín (T) nahradený uracilom (U). Existuje niekoľko druhov RNA. Tie sa rozdeľujú podľa toho, či sú nositeľmi genetickej informácie na kodinujúce a nekodinujúce. Medzi kodinujúce patrí mediátorova RNA (mRNA), ktorá je ako jediná nositeľom genetickej informácie. Nekodinujúcich RNA je niekoľko, ale medzi základné patrí transferová RNA (tRNA) a ribozómová RNA (rRNA).

5.5.1 Transkripcia

Každé molekuly RNA sú kopírované z tzn. DNA šablóny cez proces transkripcie. Proces transkripcie je podobný proces ako replikácia, ale základným rozdielom je dĺžka DNA šablóny použitej pri transkripcií. U replikácií, všetky nukleotidy sú kopírované, kde u transkripcií je použitá iba menšia časť DNA. Je to preto, lebo nie všetky gény sú pri transkripcií potrebné. Šablónou použitou pri tvorbe RNA je jedno vlákno DNA, kde u replikácií DNA boli použité obe vlákna DNA. Počas transkripcie, vzniknutý reťazec RNA je komplementárny a antiparalelný k vláknu DNA šablóny. Tento nový reťazec RNA je taktiež označovaný

pod pojmom RNA transkript. Ako enzýmy rozpoznajú, ktorá transkripčná jednotka sa použije, kde začína a kde končí, je zakódované v samotnej DNA sekvencie.

Transkripcia je katalyzovaná enzýmom RNA polymerázy, ktorý sa viaže na oblasť promótora a rozpletá časť DNA, označovanú pojmom transkripčná bublina. Promótor je sekvencia DNA indikujúca, ktoré vlákno bude použité ako šablóna a smer, v ktorom bude prebiehať transkripcia. Promótor taktiež určuje začiatok transkripcie. Po vytvorení transkripčnej bubliny, nastáva proces inicializácie, v ktorom je syntetizovaných prvých 9 až 12 nukleotidov RNA. Po skončení inicializácie nastáva proces elongácie. Počas procesu elongácie enzým RNA polymerázy sa už viac neviaže na oblasť promótora, ale pohybuje sa v smere transkripcie po vlákne šablóny DNA a predlžuje RNA reťazec. Koniec transkripcie určuje tzn. terminátor. Je to sekvencia nukleoidov DNA podobná promótoru. V tomto bode sa viac nepokračuje v transkripcii a enzým RNA polymerázy je uvoľnený ako aj výsledný RNA reťazec. Tento reťazec RNA je označovaný ako mediátorova RNA (mRNA).

5.5.2 Translácia

Translácia sa uskutočňuje na ribozómoch, tie je možné považovať za pohyblivé, proteín syntetizujúce stroje. Ribozóm sa pripája blízko konca 5′ reťazca mRNA a prekladá sa smerom k 3′ koncu. Ide o preklad polynukleotidového reťazca mRNA do polypeptidového reťazca aminokyselín prostredníctvom genetického kódu. Tento proces sa skladá zo štyroch krokov:

1. Prvým krokom translácie je väzba molekúl tRNA na príslušné aminokyseliny. Každá tRNA je špecifická pre konkrétnu aminokyselinu. Kľúčom pri špecifikácii medzi aminokyselinou a jej tRNA je enzým aminoacyl-tRNA syntetáza. Bunka má 20 rôznych aminoacyl-tRNA syntetáz, jedna pre každú z 20 aminokyselín. Rozpoznanie tRNA, pomocou enzýmu syntetáz, závisí od rôznych nukleotidových sekvencií tRNA.

2. Tento krok je nazývaný iniciácia a pozostáva z troch hlavných krokov. Syntéza proteínov sa začína od štart-signálu mRNA AUG kodónu. Najskôr sa mRNA viaže na malú časť ribozómu. Neskôr je iniciačná tRNA naviazaná k mRNA prostredníctvom párovaní báz medzi kodónom a antikodónom. Nakoniec sa ribozóm pripája k iniciačnému komplexu.

3. Ďalším krokom je elongácia, v ktorej sú aminokyseliny spojené, aby vytvorili polypeptidový reťazec. Ribozóm obsahuje tri väzbové miesta, ktoré môže zabrať tRNA:

   - *aminoacyl* nasledovná prichádzajúca aminokyselina vo forme aminoacyl-tRNA
   - *peptidyl* posledná naviazaná aminokyselina vznikajúceho polypeptidového reťazca
   - *exit* alebo *prázdna* predchádzajúca tRNA

   Elongácia prebieha v troch krokoch. V prvom kroku sa tRNA viaže na aminoacylové miesto. Kde sa antikodón tRNA viaže na kodón mRNA. V druhom kroku elongácie sa formuje peptidová väzba medzi aminokyselinami, ktoré sú naviazané na aminoacyl-tRNA miesta. Vytváranie tejto peptidovej väzby uvoľňuje aminokyselinu v peptidylovom mieste od jej tRNA. V treťom kroku sa ribozóm posunie v smere od 5′ konca do 3′ konca po mRNA. Ribozóm v tomto kroku je nad novým kodónom a proces elongácie sa môže opakovať.

4. Posledným krokom je terminácia translácie, kde sa na aminoacylové miesto dostaví terminačný kodón. Pretože neexistuje žiadny antikodón komplementárny k terminačnému kodónu, žiadna tRNA sa nenaviaže na aminoacylové miesto. Tieto kodóny sa viažu na tzv. uvoľňovacie faktory, ktoré blokujú naviazanie ďalšej tRNA do aminoacylového miesta a pomáhajú hydrolyzovať väzbu medzi aminokyselinami a tRNA v peptidylovom mieste. Vytvorený polypeptidový reťazec je z ribozómu uvoľnený.

5.6 Bielkoviny

Bielkoviny sú vo všetkých procesoch organizmov hlavnou zložkou. Mnohé bielkoviny sú enzýmami (biologickými katalyzátormi), ktoré riadia chemické reakcie buniek. Iné sú štrukturálne komponenty, ktoré poskytujú podporu pre membrány, vlákna, kosti atď.. Niektoré bielkoviny pomáhajú transportovať látky. Ďalšie majú regulačné, komunikačné, alebo obranné funkcie. Všetky proteíny sú zložené z aminokyselín, ktoré sú vzájomne prepojené.

Poznáme 22 aminokyselín. Ale len 20 z nich je bielkovinotvorných. Jedna aminokyselina je kódovaná tromi, po sebe nasledujúcimi nukleotidmi v mRNA. Týmito nukleotidmi môže byť už spomenutý adenín (A), guanín (G), cytozín (C) a uracil (U). Celkovo je teda 64 kodónov. Tieto kodóny sa nachádzajú v tabuľke 5.5. Tri z týchto kodónov sú stop kodóny, určujúce koniec prekladu. Každá aminokyselina pozostáva z atómu uhlíka, naviazaného na aminoskupinu, atóm vodíka, karboxylová skupina a skupina R (radikál), ktorá sa líši pre každú aminokyselinu. Aminokyseliny v bielkovinách sú spojené peptidovými väzbami. Rovnako ako nukleové kyseliny majú aj polypeptidy polaritu, pričom jeden koniec má voľnú aminoskupinu a druhý koniec obsahuje voľnú karboxylovú skupinu. Niektoré bielkoviny pozostávajú iba z niekoľkých aminokyselín, zatiaľ čo iné sa môžu skladať z tisícok.

Štruktúra bielkovín rovnako ako v prípade nukleových kyselín, má niekoľko úrovní organizácie. Primárnou štruktúrou bielkovín je jeho aminokyselinová sekvencia kyseliny. Interakciami medzi susednými aminokyselinami, polypeptidový reťazec sa prekladá a krúti do sekundárnej štruktúry. Sekundárne štruktúry sa skladajú, aby vytvorili terciárnu štruktúru. Nakoniec niektoré bielkoviny, obsahujúce dve a viac polypeptidových reťazcov, vytvoria kvartérnu štruktúru.

	U		C		A		G	
U	UUU	Fenylalanín	UCU	Serín	UAU	Tyrozín	UGU	Cysteín
	UUC		UCC		UAC		UGC	
	UUA	Leucín	UCA		UAA	Stop	UGA	Stop
	UUG		UCG		UAG	Stop	UGG	Tryptofán
C	CUU	Leucín	CCU	Prolín	CAU	Histidín	CGU	Arginín
	CUC		CCC		CAC		CGC	
	CUA		CCA		CAA	Glutamín	CGA	
	CUG		CCG		CAG		CGG	
A	AUU	Izoleucín	ACU	Treonín	AAU	Asparagín	AGU	Serín
	AUC		ACC		AAC		AGC	
	AUA		ACA		AAA	Lyzín	AGA	Arginín
	AUG	Metionín	ACG		AAG		AGG	
G	GUU	Valín	GCU	Alanín	GAU	Kyselina asparágová	GGU	Glycín
	GUC		GCC		GAC		GGC	
	GUA		GCA		GAA	Kyselina glutámová	GGA	
	GUG		GCG		GAG		GGG	

Obr. 5.5: Zoznam aminokyselín a kodónov

5.7 Univerzálny štandardizovaný formát

Textový formát používaný v bioinformatike a biochémii je označovaný ako FASTA formát. Tento formát sa využíva pre reprezentovanie nukleových sekvencií, alebo sekvencií aminokyselín. Sekvencia začína jednoriadkovým popisom, za ktorým nasledujú riadky sekvencie dát. Popisný riadok musí vždy začínať symbolom '>' a môže mu predchádzať riadok s komentárom začínajúci symbolom ';'. Riadky sekvencie dát by nemali presiahnuť dĺžku viac ako 80 znakov a musia byť zadané v IBU/IUPAC štandardných kódoch. Tie môžu byť zapísane malými alebo veľkými písmenami latinskej abecedy. Súbory, do ktorých je možné tieto sekvencie uložiť, majú viacero prípon, ktoré značia čo za informáciu je v nich uložená. Medzi základné patria:

- .fasta, .fa: ľubovoľný druh sekvencie
- .fna: sekvencia nukleových kyselín
- .faa: sekvencia aminokyselín

FASTA formát môže taktiež obsahovať špeciálne identifikátory. Tieto identifikátory definuje NCBI (National Center for Biotechnology Information), ktoré slúžia ako označenie referencie k záznamu v databáze. Tieto identifikátory nie sú podporované aplikáciou v práci, ale nájdete ich na [3].

Kapitola 6

Implementácia

6.1 Návrh

Cieľom tejto práce je navrhnúť a implementovať aplikáciu pre prevod DNA a mRNA sekvencii na sekvencie proteínov reprezentované formou reťazcov aminokyselín. Jazyk zvolený pre implementáciu aplikácie je objektovo orientovaný `Python 3.9`[1]. Pre vývoj aplikácie bol použitý program `PyCharm`[2]. Aplikácia je realizovaná pre operačný systém Linux a Windows.

Vzhľadom na to, že aplikácia má realizovať využitie regulovaných gramatík v odvetví Bioinformatiky a pre prípadné rozšírenia použitia a funkcionality aplikácie, je zbytočné implementovať užívateľské rozhranie. Dôležitá je funkčnosť a použiteľnosť aplikácie. Preto bola zvolená pre implementáciu konzolová aplikácia pre jednoduché zobrazenie výsledkov a interakciu s užívateľom. Užívateľ aplikácie je schopný si vstupnú sekvenciu aj vygenerovať. Proces, akým je sekvencia generovaná, je popísaný v sekcií 6.4. Aplikácia je schopná rozoznať o aký typ vstupnej sekvencie sa jedná, ak bola zadaná manuálne, alebo zo súboru a uskutočniť jej preklad. Ten spočíva vo využití troch automatov. Prvý automat $M1$ slúži ako lexikálny analyzátor, vykonávajúci validáciu vstupnej sekvencie ako aj určenie o aký typ vstupnej sekvencie sa jedná (DNA alebo mRNA). Keďže nie je potrebné v tomto kroku využívať zásobník, je využitý konečný automat. Druhý automat $M2$ je využívaný v procese transkripcie. Tento automat je realizovaný ako zásobníkový automat regulovaný stavovou gramatikou. Automat $M3$ vykonáva proces translácie a je realizovaný ako rozšírený zásobníkový automat, regulovaný gramatikou s rozptýleným kontextom. Viac o tom prečo boli zvolené práve tieto gramatiky, sa dozviete v kapitole 6.3.

6.2 Triedy aplikácie

V nasledujúcej sekcii sa nachádza popis jednotlivých tried implementovaných v aplikácii. Každá časť obsahuje stručný výpis kódu, ktorý obsahuje prototyp metód a jednotlivé premenné. Celkovo v aplikácií je definovaných päť tried.

6.2.1 Trieda Stack

Trieda `Stack` reprezentuje abstraktnú implementáciu zásobníka. Metódy triedy umožňujú podporu operácii nad zásobníkom, ktorých význam je nasledovný:

[1] Python 3.9 - dokumentácia https://docs.python.org/release/3.9.0/
[2] PyCharm Python IDE - dostupný na https://www.jetbrains.com/pycharm/

- `init` - konštruktor vytvárajúci prázdny zásobník
- `isEmpty` - zistí, či je zásobník prázdny
- `push` - pridá prvok na vrchol zásobníka
- `pop` - odoberie prvok z vrcholu zásobníka
- `size` - vráti počet prvkov v zásobníku
- `top` - vráti prvok nachádzajúci sa na vrchole zásobníka

```
1 class Stack:
2 def __init__(self):
3 def isEmpty(self):
4 def push(self, item):
5 def pop(self):
6 def size(self):
7 def top(self):
```

Výpis 6.1: Metódy triedy `Stack`

6.2.2 Trieda Sequence

Trieda `Sequence` reprezentuje vstupnú sekvenciu, ktorá je ďalej spracovaná automatmi. Premenná `code` obsahuje reťazec vstupnej sekvencie. Typ vstupnej sekvencie je získaný z výstupu automatu $M1$. Ak vstupná sekvencia je validnou DNA, alebo mRNA, tak sa po spracovaní automatom $M2$ zapíše do premenných `start_kodon`, `stop_kodon` informácia o pozícii štart, a stop kordónov sekvencií. Ak sa jedná o DNA tak to premennej `transcription` je uložená mRNA získaná z transkripcie. Keďže vstupná sekvencia môže obsahovať viac preložiteľných sekvencií, premenná `s_array` slúži pre zápis týchto sekvencií. Podobne aj premenná `amino` slúži pre uchovanie, ale už aminokyselín získaných z translácie sekvencií z `s_array`. Metóda `getSequence` ukladá do týchto dvoch premenných sekvencie získané zo zásobníka a zároveň odstráni nežiadúce symboly ako štart(S) a stop symbol(#).

```
1 class Sequence:
2 code;
3 start_kodon;
4 stop_kodon;
5 type;
6 transcription;
7 s_array;
8 amino;
9 def getSequence(self, stack, destination):
```

Výpis 6.2: Metódy a premenné triedy `Sequence`

6.2.3 Triedy automatov

Každý automat aplikácie má vlastnú triedu. Triedy `AutomataM2` a `AutomataM3` pre svoju funkčnosť pracujú s inštanciou triedy `Stack`(6.2.1). Keďže automat $M1$ je konečný, neobsahuje `stack`. Premenná `input` je rovnaká pre všetky automaty. Slúži ako vstup pre automat. Na tento vstup je privedený `code` objektu triedy `Sequence`. Premenná `state` slúži pre ukladanie stavu automatu, v ktorom sa nachádza. Premenná `top` označuje symbol, ktorý sa momentálne nachádza na vrchole zásobníka.

```
class AutomataM1:
input;
state;
```

Výpis 6.3: Premenné triedy `AutomataM1`

```
class AutomataM2(Stack):
input;
state;
stack;
top;
```

Výpis 6.4: Premenné triedy `AutomataM2`

```
class AutomataM3(Stack):
input;
stack;
top;
```

Výpis 6.5: Premenné triedy `AutomataM3`

6.3 Automaty aplikácie

Aplikácia využíva tri automaty. Ako už bolo spomenuté v návrhu 6.1. Každý z automatov plní odlišnú funkciu vo fázach prekladu vstupnej sekvencie. Týmito fázami sú:

1. lexikálna analýza - validácia symbolov vstupnej sekvencie
2. transkripcia - preklad DNA sekvencie do mRNA
3. translácia - preklad sekvencie mRNA do reťazca aminokyselín

6.3.1 Automat $M1$

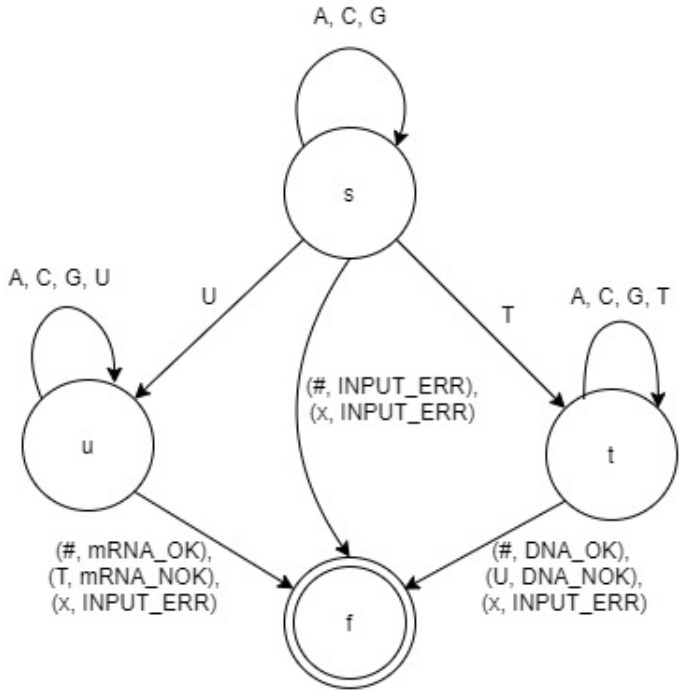
V procese lexikálnej analýzy prebieha ku klasifikácii sekvencie vstupných symbolov. K tomu nám dobre poslúži konečný automat. Ten vstupnú sekvenciu prejdením do koncového stavu označí za validnú, ak zodpovedá pravidlám definovaným v množine pravidiel automatu. V opačnom prípade sekvenciu označí ako chybnú a preklad skončí s príslušnou chybovou hláškou. Konečný automat $M1 = (Q_1, \Sigma_1, R_1, s, F_1)$ je definovaný nasledovne:

- $Q_1 = \{s, u, t, (f, (\text{INPUT\_ERR} \mid \text{DNA\_OK} \mid \text{DNA\_NOK} \mid \text{mRNA\_NOK} \mid \text{mRNA\_OK}))\}$,
- $\Sigma_1 = \Sigma \cup x$, kde $x \in \Sigma'$ pričom platí $\Sigma \cap x = \emptyset$,
- $\Sigma = \{A, C, G, U, T, \#\}$,
- $R_1 = \{1.sA \to s, 2.sC \to s, 3.sG \to s, 4.sT \to t, 5.sU \to u, 6.uA \to u, 7.uC \to u, 8.s\# \to (f, \text{INPUT\_ERR}), 9.uG \to u, 10.uU \to u, 11.tA \to t, 12.tC \to t, 13.tG \to t, 14.tT \to t, 15.u\# \to (f, \text{mRNA\_OK}), 16.uT \to (f, \text{mRNA\_NOK}), 17.t\# \to (f, \text{DNA\_OK}), 18.tU \to (f, \text{DNA\_NOK}), 19.sx \to (f, \text{INPUT\_ERR}), 20.ux \to (f, \text{INPUT\_ERR}), 21.tx \to (f, \text{INPUT\_ERR})\}$,
- $s \in Q_1$,

- $F_1 = \{(f, (\text{INPUT\_ERR} \mid \text{DNA\_OK} \mid \text{DNA\_NOK} \mid \text{mRNA\_NOK} \mid \text{mRNA\_OK}))\}$

Algoritmus 1 popisuje kroky realizované automatom $M1$. Ak sa automat dostane do koncového stavu f, na tento stav sú viazané signály indikujúce výsledný typ vstupnej sekvencie, ktorých význam je nasledovný:

- INPUT_ERR - vo vstupnej sekvencii sa nachádza nevalidný symbol
- DNA_OK - sekvencia na vstupe je validnou DNA
- DNA_NOK - vo vstupnej sekvencii sa nachádza chybný výskyt uracilu
- mRNA_OK - sekvencia na vstupe je validnou mRNA
- mRNA_NOK - vo vstupnej sekvencii sa nachádza chybný výskyt tymínu



Obr. 6.1: Diagram automatu $M1$

Algoritmus 1: Automat $M1$

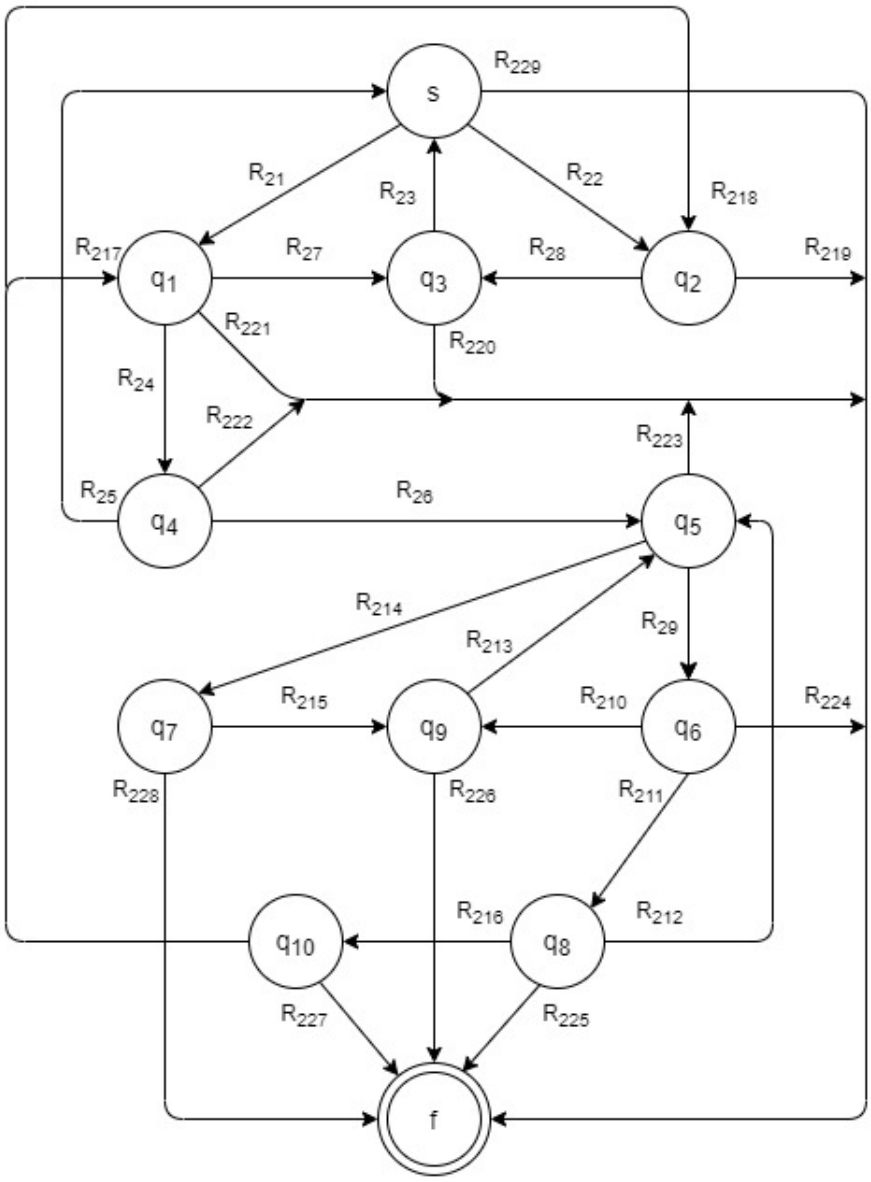
```
Result: Sequence
Class AutomataM1 contains
    input;
    state;
end
Class Sequence contains
    code;
    type;
     ⋮
end
AutmataM1.input = Sequence.code;
AutmataM1.input = += '#'; // zarážka
for symbol in AutmataM1.input do
    if (symbol == x) or (symbol == '#' and stav == 's') then
        prechod do stavu 'f';
        Sequence.type = INPUT_ERR;
        return Sequence;
    else if (stav == 't') and (symbol == 'U') then
        prechod do stavu 'f';
        Sequence.type = mRNA_NOK;
        return Sequence;
    else if (stav == 't') and (symbol == '#') then
        prechod do stavu 'f';
        Sequence.type = mRNA_OK;
        return Sequence;
    else if (stav == 'u') and (symbol == 'T') then
        prechod do stavu 'f';
        Sequence.type = DNA_NOK;
        return Sequence;
    else if (stav == 'u') and (symbol == '#') then
        prechod do stavu 'f';
        Sequence.type = DNA_OK;
        return Sequence;
    else if (stav == 's') and (symbol == 'T') then
        prechod do stavu 't';
    else if (stav == 's') and (symbol == 'U') then
        prechod do stavu 'u';
    else
        // načítaj ďalší symbol a ak sa nachádzame v stave 's' tak musí
           platiť symbol ∈ {A, C, G}, ak sme v stave 'u' tak
           symbol ∈ {A, C, G, U} a ak sme v stave 't' tak
           symbol ∈ {A, C, G, T}.
    end
end
```

6.3.2 Automat $M2$

Vo fáze transkripcie prebieha preklad validnej DNA, ktorú automat $M2$ obdrží z výstupu automatu $M1$. Automat $M2$ preloží túto sekvenciu na sekvenciu mRNA. Okrem prekladu sekvencie dochádza k získaniu pozícii štart a stop kodónov, ktoré sú potrebné vo fáze translácie automatom $M3$. Okrem sekvencie DNA, automat $M2$ môže prijať aj sekvenciu mRNA. Tá sa už síce neprekladá, ale sú z nej získané pozície kodónov ak sú prítomné v sekvencii. Pozície kodónov sú dôležitým prvkom, ktoré uľahčia proces translácie, keďže sa nie všetky kodóny v polynukleotidovom reťazci prekladajú. Regulovaný zásobníkový automat $M2 = (Q_2, \Sigma_2, \Gamma_2, R_2, s, S_2, F_2)$ je regulovaný stavovou gramatikou G, ktorý je definovaný nasledovne:

- $Q_2 = \{s, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, f\}$,
- $\Sigma_2 = \{A, C, G, U, T, \#\}$,
- $\Gamma_2 = \{A, C, G, U, S, \#\}$,
- $R_2 = R_{21} \cup R_{22} \cup R_{23} \cup R_{24} \cup R_{25} \cup R_{26} \cup R_{27} \cup R_{28} \cup R_{29} \cup R_{210} \cup R_{211} \cup R_{212} \cup R_{213} \cup R_{214} \cup R_{215} \cup R_{216} \cup R_{217} \cup R_{218} \cup R_{219} \cup R_{220} \cup R_{221} \cup R_{222} \cup R_{223} \cup R_{224} \cup R_{225} \cup R_{226} \cup R_{227} \cup R_{228} \cup R_{229}$, kde $R_{2i}, i =< 1, 29 >, (R_3 - R_{2i}) \cup R_{2i} = \emptyset$
- $s \in Q_2$,
- $F_2 = \{f\}$

Množiny pravidiel R_{2i}, pre $i =< 1, 18 >$, sú obsiahnuté v tabuľke 6.1. Množiny R_{2i}, pre $i =< 19, 29 >$, obsahujú koncové pravidlá, kedy automat narazí na zarážku. Tie sú v tvare $n.(p, p\#) \to (f, f)$, kde $n = 41 + i$, $p \in (Q_2 - \{f\}), f \in F_2$.



Obr. 6.2: Diagram automatu $M2$

Množina	**Pravidlá**
R_{21}	$1.(s; sA) \rightarrow (q_1; Aq_1)$,
R_{22}	$2.(s; sC) \rightarrow (q_2; Cq_2), 3.(s; sT) \rightarrow (q_2; Uq_2), 4.(s; sG) \rightarrow (q_2; Gq_2)$, $5.(s; sU) \rightarrow (q_2; Uq_2)$,
R_{23}	$6.(q_3; q_3A) \rightarrow (s; As), 7.(q_3; q_3T) \rightarrow (s; Us), 8.(q_3; q_3G) \rightarrow (s; Gs)$, $9.(q_3; q_3U) \rightarrow (s; Us), 10.(q_3; q_3C) \rightarrow (s; Cs)$,
R_{24}	$11.(q_1; Aq_1T) \rightarrow (q_4; AUq_4), 12.(q_1; Aq_1U) \rightarrow (q_4; AUq_4)$,
R_{25}	$13.(q_4; q_4A) \rightarrow (s; As), 14.(q_4; q_4C) \rightarrow (s; Cs), 15.(q_4; q_4T) \rightarrow (s; Us), 16.(q_4; q_4U) \rightarrow (s; Us)$,
R_{26}	$17.(q_4; Uq_4G) \rightarrow (q_5; UGq_5)$,

R_{27}	$18.(q_4; Aq_4A) \rightarrow (s; AAs), 19.(q_4; Aq_4C) \rightarrow (s; ACs), 20.(q_4; Aq_4G) \rightarrow (s; AGs),$
R_{28}	$21.(q_2; q_2A) \rightarrow (q_3; Aq_3), 22.(q_2; q_2T) \rightarrow (q_3; Uq_3), 23.(q_2; q_2G) \rightarrow (q_3; Gq_3),$ $24.(q_2; q_2U) \rightarrow (q_3; Uq_3), 25.(q_2; q_2C) \rightarrow (q_3; Cq_3),$
R_{29}	$26.(q_5; q_5T) \rightarrow (q_6; Uq_6), 27.(q_5; q_5U) \rightarrow (q_6; Uq_6),$
R_{210}	$28.(q_6; Uq_6T) \rightarrow (q_9; UUq_9), 29.(q_6; Uq_6C) \rightarrow (q_9; UCq_9), 30.(q_6; Uq_6U) \rightarrow (q_9; UUq_9),$
R_{211}	$31.(q_6; Uq_6A) \rightarrow (q_8; UAq_8), 32.(q_6; Uq_6G) \rightarrow (q_8; UGq_8),$
R_{212}	$33.(q_8; Aq_8T) \rightarrow (q_5; AUq_5), 34.(q_8; Aq_8C) \rightarrow (q_5; ACq_5), 35.(q_8; Aq_8U) \rightarrow (q_5; AUq_5), 36.(q_8; Gq_8C) \rightarrow (q_5; GCq_5), 37.(q_8; Gq_8U) \rightarrow (q_5; GUq_5),$ $38.(q_8; Gq_8T) \rightarrow (q_5; GUq_5),$
R_{213}	$39.(q_9; q_9A) \rightarrow (q_5; Aq_5), 40.(q_9; q_9T) \rightarrow (q_5; Uq_5), 41.(q_9; q_9G) \rightarrow (q_5; Gq_5),$ $42.(q_9; q_9U) \rightarrow (q_5; Uq_5), 43.(q_9; q_9C) \rightarrow (q_5; Cq_5),$
R_{214}	$44.(q_5; q_5A) \rightarrow (q_7; Aq_7), 45.(q_5; q_5C) \rightarrow (q_7; Cq_7), 46.(q_5; q_5G) \rightarrow (q_7; Gq_7),$
R_{215}	$47.(q_7; q_7A) \rightarrow (q_9; Aq_9), 48.(q_7; q_7T) \rightarrow (q_9; Uq_9), 49.(q_7; q_7G) \rightarrow (q_9; Gq_9),$ $50.(q_7; q_7U) \rightarrow (q_9; Uq_9), 51.(q_7; q_7C) \rightarrow (q_9; Cq_9),$
R_{216}	$52.(q_8; Aq_8A) \rightarrow (q_{10}; AAq_{10}), 53.(q_8; Aq_8G) \rightarrow (q_{10}; AGq_{10}), 54.(q_8; Gq_8A) \rightarrow (q_{10}; GAq_{10}),$
R_{217}	$55.(q_{10}; q_{10}A) \rightarrow (q_1; Aq_1),$
R_{218}	$56.(q_{10}; q_{10}C) \rightarrow (q_2; Cq_2), 57.(q_{10}; q_{10}T) \rightarrow (q_2; Uq_2), 58.(q_{10}; q_{10}G) \rightarrow (q_2; Gq_2), 59.(q_{10}; q_{10}U) \rightarrow (q_2; Uq_2),$

Tabuľka 6.1: Tabuľka množín pravidiel automatu $M2$

6.3.3 Automat $M3$

Rozšírený zásobníkový automat $M3$ regulovaný gramatikou s rozptýleným kontextom 3.2.2 nám umožňuje v procese translácie, zo vstupnej sekvencie prečítať viac symbolov (nukleotidov) naraz. V procese translácie sú to vždy tri nukleotidy pre každú aminokyselinu. Aby sme boli schopní tieto symboly naraz preložiť, k tomu nám slúži už spomenutá regulovaná gramatika. Proces translácie prebieha nasledovne. Automat $M3$ obdrží z výstupu automatu $M2$ objekt triedy `Sequence`. Ten obsahuje pozície štart a stop kodónov, ktoré vymedzujú preložiteľné časti vstupnej sekvencie. Automat obsahuje 2 stavy s, f. Ten sa pri načítavaní vždy troch symbolov nachádza v stave s pokiaľ nenarazí na zarážku. Použitím pravidla z množiny R_3 preloží na príslušnú skratku aminokyseliny, ktorú zapíše na vrchol zásobníka. Skratka príslušnej aminokyseliny pozostáva z prvých troch písmen názvu. Jednotlivé názvy aminokyselín boli spomenuté v tabuľke 5.5. Zarážka # signalizuje koniec procesu translácie a automat prechádza do koncového stavu f. Automat $M3 = (Q_3, \Sigma_3, \Gamma_3, R_3, s, S, F_3)$ je definovaný nasledovne:

- $Q_3 = \{s, f\}$,
- $\Sigma_3 = \{A, C, G, U, T, S, \#\}$,
- $\Gamma_3 = \{Phe, Leu, Ile, Met, Val, Ser, Pro, Thr, Ala, Tyr, His, Gln, Ans, Lys, Cys, Trp, Arg, Glu, Asp, Gly\}$,
- R_3 viz tabuľka 6.2,

- $s \in Q_3$,
- $F_3 = \{f\}$

R_3	$1.sUUU \to Phes, 2.sUUC \to Phes, 3.sUUA \to Leus,$ $4.sUUG \to Leus, 5.sCUU \to Leus, 6.sCUC \to Leus,$ $7.sCUA \to Leus, 8.sCUG \to Leus, 9.sAUU \to Iles,$ $10.sAUC \to Iles, 11.sAUA \to Iles, 12.sAUG \to Mets,$ $13.sGUU \to Vals, 14.sGUC \to Vals, 15.sGUA \to Vals,$ $16.sGUG \to Vals, 17.sUCU \to Sers, 18.sUCC \to Sers,$ $19.sUCA \to Sers, 20.sUCG \to Sers, 21.sCCU \to Pros,$ $22.sCCC \to Pros, 23.sCCA \to Pros, 24.sCCG \to Pros,$ $25.sACU \to Thrs, 26.sACC \to Thrs, 27.sACA \to Thrs,$ $28.sACG \to Thrs, 26.sGCU \to Alas, 27.sGCC \to Alas,$ $28.sGCA \to Alas, 29.sGCG \to Alas, 30.sUAU \to Trys,$ $31.sUAC \to Trys, 32.sCAU \to Hiss, 33.sCAC \to Hiss,$ $34.sCAA \to Glns, 35.sCAG \to Glns, 36.sAAU \to Anss,$ $37.sAAC \to Anss, 38.sAAA \to Lyss, 39.sAAG \to Lyss,$ $40.sUGU \to Cyss, 41.sUGC \to Cyss, 42.sUGG \to Trps,$ $43.sCGU \to Args, 44.sCGC \to Args, 45.sCGA \to Args,$ $46.sCGG \to Args, 47.sAGA \to Args, 48.sAGG \to Args,$ $49.sAGU \to Sers, 50.sAGC \to Sers, 51.sGAU \to Asps,$ $52.sGAC \to Asps, 53.sGGU \to Glys, 54.sGGC \to Glys,$ $55.sGGA \to Glys, 56.sGGG \to Glys, 57.s\# \to f,$

Tabuľka 6.2: Tabuľka pravidiel množiny R_3 automatu $M3$

Príklad 6.3.1 Majme vstupnú sekvenciu $x = CTGATGCTTTAA$, ktorá bude vstupom pre automat $M1$. Použitím pravidiel z množiny R_1 dostaneme sekvenciu prechodov nasledovnú:

$sCTGATGCTTTAA \vdash sTGATGCTTTAA \vdash tGATGCTTTAA \vdash tATGCTTTAA \vdash$
$tTGCTTTAA \vdash tGCTTTAA \vdash tCTTTAA \vdash tTTTAA \vdash tTTAA \vdash tTAA \vdash tAA \vdash$
$tA \vdash t\# \vdash f(DNA\_OK)[2, 4, 13, 11, 14, 13, 12, 14, 14, 14, 11, 11, 17]$

Automat prešiel do koncového stavu f so signálom DNA_OK. Indikujúc, že sa jedná o validnú DNA a môže byť ďalej použitá pre vstup automatu $M2$. Ten použitím pravidiel z množiny R_2, sekvencia prechodov automatu je nasledovná:

$SsC \vdash Cq_2T \vdash Uq_3G \vdash GsA \vdash Aq_1T \vdash AUq_4G \vdash UGq_5C \vdash GCq_7T \vdash CUq_9T \vdash UUq_5T \vdash$
$UUq_6A \vdash UAq_8A \vdash AAq_{10}\# \vdash f[2, 22, 8, 1, 11, 17, 45, 48, 40, 26, 31, 52, 68]$

Výstupná sekvencia CUGAUGCUUUAA automatu $M2$ je uložená na zásobníku. Ak pri aplikovaní pravidiel, automat využije postupne za sebou pravidlá z množín $R_{21}, R_{24}aR_{26}$, automat narazil na štart kodón sekvencie. V tomto príklade sa jedná o pravidlá $1, 11a17$. Pozícia štart kodónu sa uloží do príslušnej premennej objektu triedy `Sequence`. Obdobne to platí pre pozíciu stop kodónu. K tomu je potrebné aplikovať pravidlá postupne z množiny $R_{29}, R_{211}aR_{216}$. Týmito pravidlami sú $26, 31, 52$. Proces transkripcie končí narazením automatu na zarážku a použitím pravidla 68 z množiny R_{227}. V poslednej fáze automat $M3$

obdrží výstup automatu $M2$. Tým sa začne proces translácie, ktorého sekvencia prechodov automatu vyzerá nasledovne:

$SsAUG \vdash MetsCUU \vdash Leus\# \vdash f[12, 5, 57]$

Výsledná sekvencia aminokyselín je uložená na zásobníku. V tomto príklade sa jedná o *MetioninLeucin* skrátene *MetLeu*. Automat $M3$ využil pozície štart a stop kodónu získané automatom $M2$. Preklad prebehol len za použitia troch pravidiel, keďže kodón CUG a stop kodón UAA sa neprekladá na žiadnu aminokyselinu. Tie boli automatom $M3$ odignorované, a teda vôbec nedošlo k ich prečítaniu. Týmto sme ušetrili niekoľko krokov použitím vhodných regulovaných gramatík a automatov.

6.4 Generovanie

Jednou z možností vstupnej sekvencie aplikácie je jej vygenerovanie. To musí zabezpečiť, že sa v sekvencii bude nachádzať štart a stop kodón. Vygenerovaná sekvencia môže nadobúdať dĺžky v rozmedzí od 2 do 40 kodónov. Princíp generovania je nasledovný.

Najprv sa vygeneruje dĺžka sekvencie. Ďalej je potrebné vygenerovať časti sekvencie, ktoré budú preložiteľné, tzn. budú obsahovať štart a stop kodón. Ich počet závisí od celkovej vygenerovanej dĺžky. Napríklad, ak vstupná sekvencia má mať dĺžku 3 kodóny, tak musí obsahovať štart, stop kodón a jeden zvyšný kodón, ktorý sa môže nachádzať buď medzi, nimi alebo mimo nich. Ak máme vygenerovanú celkovú dĺžku sekvencie, aj počet sekvencií, ktoré bude obsahovať, je nutné ešte vygenerovať ich jednotlivé dĺžky. Po každom vygenerovaní časti sekvencie sa odráta vygenerovaný podiel od celkovej dĺžky sekvencie. To sa cyklicky opakuje pre každú sekvenciu. Na záver, ak nám ostanú zvyšné kodóny, ktoré nie sú použité v sekvenciách, tie sa rovnomerne rozložia na začiatok a koniec každej sekvencie. Na nasledujúcom príklade 6.4.1 je znázornený priebeh generovania.

Príklad 6.4.1 Majme generátor $g()$, ktorý nám vygeneruje DNA sekvenciu dlhú 10 kodónov. Generátor $g()$ vygeneroval, že vstupná sekvencia bude obsahovať 2 sekvencie alebo iným slovom podsekvencie. Prvej podsekvencii bola vygenerovaná dĺžka 3 kodóny. Druhej podsekvencii bola vygenerovaná dĺžka 5 kodónov. Zvyšné 2 kodóny sú rozdelené, buď na koniec, alebo na začiatok podsekvencie. Výsledná sekvencia môže mať podobu ako na nasledujúcom obrázku.

Obr. 6.3: Príklad vygenerovanej sekvencie

Kapitola 7

Testovanie

V nasledujúcej kapitole sa nachádza testovanie a porovnanie výsledkov s webovou aplikáciou Expasy Traslate Tool[1] od Swiss Institute of Bioinformatics (SIB). Testovanie je rozdelené do troch častí. V prvej časti je testovaná aplikácia na spracovanie nevalidných vstupných symbolov na vstupe aplikácie ako aj limity vstupu. V druhej časti sú testované parametre aplikácie a ich kombinácie. V závere, a teda poslednej časti testovania, sú výsledky porovnané s výstupom aplikácie Expasy Traslate Tool.

7.1 Testovanie vstupu aplikácie

V tejto časti je aplikácia testovaná na nevalidné znaky vo vstupnej sekvencii. Týmito znakmi sú všetky znaky z ASCII[2] tabuľky, okrem znakov nukleotidov a to A, C, T, U ,G. Tie môžu byť zadané malými aj veľkými písmenami latinskej abecedy. Okrem toho prebieha testovanie limít vstupu. Týmito limitami sú maximálna dĺžka generovaného reťazca, ako už bolo spomenuté v sekcii o generovaní 6.4 a maximálna dĺžka riadku sekvencie v súbore formátu FASTA.

7.2 Testovanie parametrov aplikácie

V aplikácii sú definované dva druhy parametrov. A to základné a voliteľné. Medzi základné patria:

- -s: manuálne zadanie vstupnej sekvencie
- -f: súbor vo formáte FASTA ako vstup aplikácie
- -h: výpis nápovedy ako používať parametre aplikácie
- -g -d: vygenerovanie vstupnej DNA sekvencie
- -g -r: vygenerovanie vstupnej mRNA sekvencie

Voliteľnými parametrami sú:

- -t: výpis tabuľky genetického kódu

[1] Dostupná na https://web.expasy.org/translate/
[2] Dostupná na http://www.asciitable.com

- -c: zadaná vstupná sekvencia je komplementárna
- -sub: výpis preložiteľných častí sekvencie

Voliteľné parametre môžu byť použité v kombinácií so základnými parametrami až na pár výnimiek. Parameter pre nápovedu (-h) nie je použiteľný s ani jedným voliteľným parametrom a parameter generovania nepodporuje voliteľný parameter pre komplementárnu sekvenciu. Na obrázku 7.1 je zobrazený príklad manuálne zadanej sekvencie a parametra -t. Ako vstupnú sekvenciu využijeme sekvenciu z príkladu 6.3.1 . V prípade generovania, je aplikácia spustená s dodatočným parametrom -sub, ktorý je znázornený na obrázku 7.2. Parameter -sub, vo výpise prekladu, vypíše časti vstupnej sekvencie spolu s indexom štart a stop kodónu, ktoré sú aplikáciou preložené na reťazce proteinotvorných aminokyselín.

```
xarbet00@merlin: ~/IBT/IBTproject$ ./app.py -s CTGATGCTTTAA -t
+-------------------------------------------------------------------------------+
|                           Table of genetic code                               |
+-----------------------+----------------+------------------+------------------+
| TTT - Phenylalanine   | TCT - Serine   | TAT - Tyrosine   | TGT - Cysteine   |
| TTC   (Phe)           | TCC   (Ser)    | TAC   (Tyr)      | TGC   (Cys)      |
+-----------------------+ TCA            +------------------+------------------+
| TTA - Leucine         | TCG            | TAA - STOP CODON | TGA - STOP CODON |
| TTG   (Leu)           +----------------+ TAG              |------------------+
| CTT                   | CCT - Proline  |------------------| TGG - Tryptophan |
| CTC                   | CCC   (Pro)    | CAT - Histidine  |       (Try)      |
| CTA                   | CCA            | CAC   (His)      |------------------|
| CTG                   | CCG            |------------------| CGT - Arginine   |
+-----------------------+----------------+ CAA - Glutamine  | CGC   (Arg)      |
| ATT - Isoleucine      | ACT - Threonine| CAG   (Gln)      | CGA              |
| ATC   (Ile)           | ACC   (Thr)    +------------------+ CGG              |
| ATA                   | ACA            | AAT - Asparagine |------------------+
+-----------------------+ ACG            | AAC   (Asn)      | AGT - Serine     |
| ATG - Methionine (Met)|----------------+------------------+ AGC   (Ser)      |
|-----------------------+ GCT - Alanine  | AAA - Lysine     |------------------+
| GTT - Valine          | GCC   (Ala)    | AAG   (Lys)      | AGA - Arginine   |
| GTC   (Val)           | GCA            +------------------+ AGG   (Arg)      |
| GTA                   | GCG            | GAT - Aspartic   |------------------+
| GTG                   |----------------| GAC   acid (Asp) | GGT - Glycine    |
+-----------------------+                |------------------| GGC   (Gly)      |
+ The bases: adenine (A), cytosine (C),  | GAA - Glutamic   | GGA              |
+            guanine (G), thymine (T)    | GAG   acid (Glu) | GGG              |
+-----------------------+----------------+------------------+------------------+
--------------------------------------------------------------------------------
|   Input DNA sequence:     CTGATGCTTTAA                                       |
--------------------------------------------------------------------------------
|   Transcription result:   CUGAUGCUUUAA                                      |
--------------------------------------------------------------------------------
|   Translation result:                                                        |
|   Protein  1 : MetLeu                                                        |
--------------------------------------------------------------------------------
```

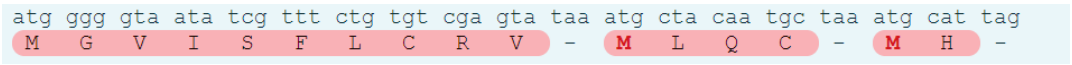
Obr. 7.1: Príklad manuálneho zadania vstupnej sekvencie

```
xarbet00@merlin: ~/IBT/IBTproject$ ./app.py -g -r -sub
-------------------------------------------------------------------------------
|   Input mRNA sequence:    AUGGGGGUAAUAUCGUUUCUGUGUCGAGUAUAAAUGCUACAAUGCUA    |
|                           AAUGCAUUAG                                          |
-------------------------------------------------------------------------------
|   Subsequence [1, 31]:    AUGGGGGUAAUAUCGUUUCUGUGUCGAGUA                      |
|   Subsequence [34, 46]:   AUGCUACAAUGC                                        |
|   Subsequence [49, 55]:   AUGCAU                                              |
-------------------------------------------------------------------------------
|   Translation result:                                                         |
|   Protein  1 : MetGlyValIleSerPheLeuCysArgVal                                 |
|   Protein  2 : MetLeuGlnCys                                                   |
|   Protein  3 : MetHis                                                         |
-------------------------------------------------------------------------------
```

Obr. 7.2: Príklad vygenerovania sekvencie mRNA

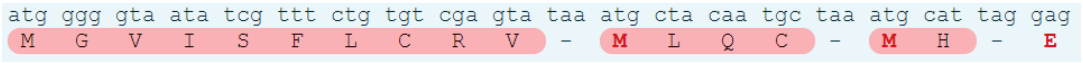
7.3 Validácia výsledkov aplikácie

Výsledky aplikácie v procese validácie budú porovnávané už spomenutou aplikáciou Expasy Traslate Tool od Swiss Institute of Bioinformatics (SIB). Ako príklad si vezmeme vygenerovanú sekvenciu 7.2 z predošlej kapitoly. Výstup referenčnej aplikácie je zobrazený na nasledujúcom obrázku.



Obr. 7.3: Výstup aplikácie Expasy Traslate Tool

Na prvý pohľad je vidno, že výstupy aplikácii sa líšia. To je spôsobené tým, že aplikácia Expasy Traslate Tool využíva NCBI značenie, ktoré je znázornené v tabuľke 7.5. Oblasť vyznačená červenou farbou má význam tzv. Open reading frames. Čo v preklade zamená, časť sekvencie, ktorá je schopná byť preložená. Tá začína, štart kodónom označením ako *M* a končí stop kodónom označené znakom −. Ak do vstupnej aplikácie vložíme kodón, ktorý sa nachádza mimo časť prekladu, aplikácia Expasy Traslate Tool tento kodón preloží. Naša aplikácia tieto kodóny ignoruje. Ak napríklad za koniec sekvencie pridáme kodón *GAG*, tak aplikácia to vyhodnotí ako kyselinu glutámovú. Tento výsledok je zobrazený na obrázku 7.4



Obr. 7.4: Výstup aplikácie Expasy Traslate Tool s kodónom navyše

Znak	Význam	Znak	Význam	Znak	Význam	Znak	Význam
F	Fenylalanín	S	Serín	H	Histidín	E	Kyselina glutámová
L	Leucín	P	Prolín	G	Glutamín	C	Cysteín
I	Izoleucín	T	Treonín	Q	Asparagín	W	Tryptofán
M	Metionín	A	Alanín	N	Lyzín	R	Arginín
V	Valín	Y	Tyrozín	K	Kyselina asparágová	G	Glycín

Obr. 7.5: Tabuľka NCBI kódov

Kapitola 8

Záver

Cieľom práce bolo navrhnúť a implementovať aplikáciu, ktorá využíva regulované gramatiky pre spracovanie sekvencií DNA, alebo mRNA a taktiež získanie nových vedomostí z oblasti bioinformatiky a formálnych jazykov, bez ktorých by sa práca nezaobišla.

V práci boli formálne definované gramatiky a automaty využité pri preklade sekvencií a ich porovnanie s jednou z využívaných metód. Bolo potrebné dôkladné štúdium molekulárnej biológie, na ktorej sa práca zakladá. Dôležité bolo pochopiť jednotlivé postupy a procesy. Bez nich by nebolo možné správne definovať gramatiky. Gramatika je pojem, ktorý označuje štruktúru, ktorá definuje formálny jazyk. Pri preklade je neoddeliteľnou súčasťou a bez nej by preklad nemohol začať. Tá udáva každý krok, určuje, aké bude nasledujúce použité pravidlo danej gramatiky. Gramatika nám tvorí najdôležitejšiu súčasť celého prekladu. Avšak, samotná gramatika k prekladu nestačí. K tomu je potrebný automat. Tento automat sa riadi pravidlami gramatiky, bez ktorých by nevykonal ani krok. V práci je dôkladne rozobraná bezkontextová gramatika z formálneho hľadiska, na ktorej sa regulovaná gramatika zakladá. Podarilo sa ukázať, že gramatika, ktorá je regulovaná určitým spôsobom, je prínosom pri preklade sekvencií za použitia zásobníkových automatov. V prípade stavovej gramatiky, kedy sa pri každom kroku mení stav, prípadne neterminál, nám umožní spoľahlivo určiť začiatok a koniec sekvencie v procese transkripcie. Toto zistenie môže mať v niektorých prípadoch obrovskú výhodu, kedy sekvencia môže byť dlhá niekoľko desiatok kodónov, ale len malá časť z nej bude preložená. To nám ušetrilo niekoľko zbytočných krokov v nasledujúcom procese translácie. Tu sa nám osvedčila regulovaná gramatika s rozptýleným kontextom. Keďže kodóny sú tvorené tromi nukleotidmi, bolo by neefektívne ich spracovávať po jednom, keď máme gramatiku, kde tieto tri nukleotidy je možné naraz preložiť v jednom kroku. To sa nám podarilo využiť v kombinácii s rozšírením zásobníkovým automatom, kedy môžeme načítať viac nukleotidov naraz zo vstupnej sekvencie.

Pri implementácií bolo potrebné overiť, či takto definované automaty regulované gramatikou, je možné využiť v preklade sekvencie DNA alebo RNA. K tomu bol využitý jazyk Python 3.9, ktorá sa ukázala ako dobrá voľba, vďaka vstavaným funkciám a typom. Gramatiky, ktoré boli formálne definované v práci, boli použité v implementácií a následne otestované na konkrétnych príkladoch, ktoré ukázali, či takto definovanú gramatiku je automat schopný správne využiť pri preklade. Na konkrétnych príkladoch bolo dokázané, že aplikácia funguje správne podľa požiadaviek.

Práca poskytuje dobrý základ pre pokračovanie na diplomovú prácu. Zaujímavé by bolo pozorovať využitie regulovaných gramatík a automatov v procese génovej mutácie, ktorá prebieha na úrovni vlákien DNA. Jedná sa o zmeny v poradí nukleotidov, oproti normál-

nemu poradiu v sekvencií. Tu by bolo zaujímavé zistenie, či vo vlákne pobehlo k nejakému druhu mutácie alebo nie. Ak prebehlo k určitému druhu mutácie, bolo by zaujímavé zistiť jej dopad. U živočíchov, väčšinou mutácie pôsobia negatívne, či už v podobe chorôb, nádorových ochorení, alebo k tomu môžu prispievať. U rastlín mutácie môžu mať pozitívny vplyv. To sa môže prejaviť tak, že rastlina produkuje viac ovocia, plody sú väčšie, alebo odolnejšie voči chorobám. Využitie takýchto mutácií je zreteľne viditeľné v poľnohospodárstve pri šľachtení plodín. Tu by bolo zaujímavé pozorovať jednotlivé plodiny a ich predchádzajúce odrody, z ktorých boli vyšľachtené, aké charakteristické vlastnosti si ponechali, ktoré stratili alebo aké získali. Pre prípadné prepojenie aplikácie s databázou bude potrebná úprava pravidiel, keďže tie sú pevne stanovené. Okrem samotnej úpravy pravidiel, by bolo prínosom aplikáciu rozšíriť o identifikátory NCBI. Napríklad v databáze NCBI, súbory FASTA obsahujúce sekvencie nukleotidov obsahujú znaky N, alebo X, ktoré majú význam neznámeho nukleotidu. Prepojenie s databázou by taktiež umožňovalo získanie konkrétneho názvu proteínu z výstupu aplikácie.

Literatúra

[1] Alexander Meduna, P. Z. *Regulated Grammars and Their Transformations*. 1. vyd. Brno University of Technology, 2010. ISBN 978-80-214-4203-0.

[2] Alexander Meduna, P. Z. *Regulated Grammars and Automata*. Springer, 2014. ISBN 978-1-4939-0369-6.

[3] Biotechnology Information, N. C. for. *FASTA Sequence ID Format* [online]. [cit. 2021-03-28]. Dostupné z: https://ncbi.github.io/cxx-toolkit/pages/ch_demo#ch_demo.id1_fetch.html_ref_fasta.

[4] Donald Voet, C. W. P. *Fundamentals of Biochemistry life at the molecular level*. 3. vyd. Kaye Pace, 2013. ISBN 978-0470-12930-2.

[5] Eduard, F. *Matematická logika a teorie množin*. 1. vyd. Plzeň : Fraus, 2013. ISBN 978-80-7489-138-0.

[6] Gorodkin J., W. L. R. *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*. 1. vyd. Humana Press, 2014. ISBN 978-1-62703-708-2.

[7] Hoy, M. A. *Insect Molecular Genetics An Introduction to Principles and Applications*. 3. vyd. Academic Press, 2013. ISBN 0-12-415874-9.

[8] Meduna, A. *Automata and Languages: Theory and Applications*. 1. vyd. Springer London, 2000. ISBN 1-85233-074-0.

[9] Pierce, B. A. *Genetics: A Conceptual Approach*. 4. vyd. Kate Ahr Parker, 2010. ISBN 1-4292-3250-1.

Príloha A

Obsah priloženého CD

- **src** - adresár obsahujúci zdrojové súbory aplikácie a textu bakalárskej práce
- **doc** - adresár obsahujúci manuál a písomnú správu vo formáte PDF