VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉMY PŘEVODNÍKŮ A JEJICH APLIKACE

TRANSDUCER SYSTEMS AND THEIR APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADRIÁN BOROS**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Boros Adrián**
Program: Informační technologie
Název: **Systémy převodníků a jejich aplikace**
Transducer Systems and Their Applications
Kategorie: Teoretická informatika

Zadání:

1. Dle instrukcí vedoucího se seznamte s gramatickými systémy.
2. Zaveďte převodníkové systémy analogicky jako gramatické systémy.
3. Studujte vlastnosti převodníkových systémů. Porovnejte jejich sílu s jinými systémy formálních modelů.
4. Dle pokynů vedoucího uvažujte různé části překladačů. Formalizujte je prostřednictvím převodníkových systémů.
5. Implementujte formalizace navržené v bodě 4 v rámci stávajících nástrojů dle pokynů vedoucího.
6. Zhodnoťte dosažené výsledky. Diskutujte další vývoj projektu.

Literatura:

- Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Volume 1-3, Springer, 1997, ISBN 3-540-60649-1 Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools (2nd Edition), Pearson Education, 2006, ISBN 0-321-48681-1

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

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 2019
Datum odevzdání: 14. května 2020
Datum schválení: 31. října 2019

Abstrakt

Táto práca zavádza novú variantu prekladateľských zariadení. Navrhujeme nový formálny model založený na prepojení niekoľkých spolupracujúcich zásobníkových prevodníkoch. Tento nový model sa nazýva systém prevodníkov. Princíp činnosti týchto systémov je podobný činnosti kooperujúcich distribuovaných (CD) gramatických systémov, umožňujúcich spoluprácu viacerých gramatík nad spoločnou vetnou formou. Ďalej sa práca zaoberá vyjadrovacou silou tohto formálneho modelu. Hlavným výsledkom je aplikácia, ktorá prekladá aritmetické výrazy z infixovej do postfixovej notácie.

Abstract

This thesis introduces a new variant of translation devices. We propose a new formal model based on the interconnection of several cooperating pushdown transducers. This new model is called transducer system. The principle of operation of these systems is similar to the cooperating distributed (CD) grammar systems, enabling a cooperation of several grammar over a common sentential form. Furthermore, the thesis deals with the computational power of this formal model. The main result is an application that converts arithmetic expressions from infix to postfix notation.

Kľúčové slová

Zásobníkové prevodníky, prevodníkový systém, gramatické systémy, CD GS, aritmetické výrazy, postfixová notácia

Keywords

Pushdown transducers, transducer system, grammar system, CD GS, arithmetic expression, postfix notation

Citácia

BOROS, Adrián. *Systémy převodníků a jejich aplikace*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexander Meduna, CSc.

Systémy převodníků a jejich aplikace

Prehlásenie

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

.....
Adrián Boros
25. mája 2020

Poďakovanie

Moje poďakovanie patrí vedúcemu práce, profesorovi Alexandrovi Medunovi za jeho odborné rady, doporučenie materiálov a za pomoc pri tvorbe tejto práce.

Obsah

1 Úvod **2**

2 Konečné prevodníky **4**

3 Zásobníkové prevodníky **9**

4 Prekladové gramatiky **14**

5 Gramatické systémy **18**
5.1 CD gramatické systémy 18
5.2 PC gramatické systémy 20

6 Systémy prevodníkov **24**
6.1 Základné definície 24
6.2 Demonštračné príklady 28
6.3 Systémy prevodníkov – zhrnutie 31

7 Implementácia **32**
7.1 Návrh aplikácie 32
7.1.1 Prevodník $\mathbf{M_1}$ – lexikálny analyzátor 33
7.1.2 Prevodník $\mathbf{M_2}$ – syntaktický analyzátor 33
7.1.3 Prevodník $\mathbf{M_3}$ – kontrola zátvoriek 36
7.1.4 Prevodník $\mathbf{M_4}$ – prevod výrazu 36
7.2 Programová implementácia 38
7.2.1 Trieda LexicalAnalyzer 38
7.2.2 Trieda SyntaxAnalyzer 38
7.2.3 Trieda BracketController 38
7.2.4 Trieda ExpressionConverter 39
7.2.5 Grafické užívateľské rozhranie 39

8 Záver **41**

Literatúra **43**

A Obsah priloženého CD **45**

Kapitola 1

Úvod

Základom formálnych jazykov, ako už aj názov naznačuje, je jazyk. Pomocou jazykov sa vyjadrujú informácie v oblasti informačných technológií. Akúkoľvek úlohu alebo problém je možné popísať pomocou jazyka. V klasickej teórii formálneho jazyka boli úlohy riešené jedným, „centralizovaným“ zariadením. Znamená to, že jazyk bol rozpoznávaný jedným automatom. V modernej informatike hrá rozdelenie úloh a vzájomná spolupráca dôležitú rolu. Rastúci záujem o rýchlejšie a efektívnejšie spracovanie údajov vedie k zovšeobecneniu klasických konceptov teórie formálneho jazyka, vrátane gramatík a prevodníkov. Týmto sa otvára priestor pre vznik nových prekladových modelov, ktoré obsahujú niekoľko komponent.

Jedným z takýchto novovzniknutých modelov je systém prevodníkov, ktorého komponentami sú zásobníkové prevodníky. Samotný zásobníkový prevodník, definovaný v [1], vychádza zo zásobníkového automatu, prípadne z konečného prevodníka. Od zásobníkového automatu sa však odlišuje tým, že okrem spracovávania vstupných symbolov, dokáže taktiež generovať výstup. Vďaka svojej flexibilite pri generovaní výstupných údajov, sa tieto prevodníky uplatňujú najmä v oblasti rozpoznávaní reči [14] alebo lexikálnych analyzátorov [8]. V kapitole 3 sa detailnejšie pozrieme na zásobníkové prevodníky. Na obrázku si ukážeme ich grafickú podobu, t.j. akým spôsobom sa značia stavy a prechody medzi nimi. Ešte predtým, v kapitole 2, si však zavedieme konečné prevodníky, aby sme boli schopní porovnať tieto dva typy. Konečné prevodníky boli zavedené prvýkrát v [6]. Odvtedy sa čoraz viac rozvíjajú vďaka svojej flexibilite pri reprezentácii a generovaní veľkého množstva štrukturálnych údajov. Tieto prevodníky boli rozšírené na iné špeciálne typy, ako napr. algebraický prevodník alebo vážený konečný prevodník.

Transformácia vstupného reťazca na výstupný sa nazýva preklad. Preklad teda tvoria usporiadané dvojice reťazcov. Kompilátor definuje preklad ako pár (zdrojový program, objektový program). Ak si zoberieme do úvahy fázy prekladu (lexikálna analýza, syntaktická analýza a generovanie cieľového kódu), každá z týchto fáz sama definuje preklad. Lexikálnu analýzu možno považovať za preklad, v ktorom sa reťazce predstavujúce zdrojové programy mapujú na reťazce tokenov. Syntaktický analyzátor vytvára z reťazcov tokenov príslušné derivačné stromy. Generátor kódu potom vezme tieto reťazce a preloží ich do jazyka stroja. Kapitola 4 predstavuje ďalší z prístupov na špecifikáciu prekladov, ktorým sú prekladové gramatiky. Zadefinujeme si tento typ prekladového modelu a pre prehľadnejší zápis si zavedieme schému prekladu. Na konci kapitoly si ukážeme postup, ktorým táto gramatika prevedie výraz z infixovej do postfixovej notácie.

Hlavnou motiváciou pre tvorbu a skúmanie nových modelov je možnosť rozdelenia problému na menšie, ľahšie riešiteľné časti. Jednotlivé podproblémy tak môžu byť riešené sek-

venčne alebo paralelne. Analýza rôznych výpočtov, napr. v počítačových sieťach alebo distribuovaných databázach, vedie k pojmom ako sú distribúcia, paralelizmus a komunikácia. Gramatické systémy boli vyvinuté ako model pre distribuované výpočty, kde je možné tieto pojmy analyzovať. Podľa [5] je gramatický systém skupina gramatík, ktoré spolupracujú a vytvárajú jeden jazyk. Jeden z typov gramatických systémov sú kooperujúce distribuované (CD) gramatické systémy, ktoré sa stali základom pre vyššie spomínané systémy prevodníkov. V prípade týchto systémov platí, že všetky komponenty spoločne vytvárajú jednu vetnú formu. V jednom momente je aktívna iba jedna gramatika, ktorá prepisuje aktuálnu vetnú formu. Po dokončení svojej práce predá aktuálnu vetnú formu ďalšej komponente, ktorá opäť vykoná zmeny. Druhým typom sú paralelne komunikujúce (PC) gramatické systémy. Tieto systémy taktiež pozostávajú z niekoľkých gramatík, ktoré však pracujú paralelne na svojich vetných formách a vygenerované reťazce si navzájom predávajú na vyžiadanie. Kapitola 5 obsahuje informácie o oboch typoch spomínaných gramatických systémov a zavádza aj ich formálne definície. U CD gramatických systémoch sú následne podrobne vysvetlené ich derivačné režimy. U PC gramatických systémoch je popísaný princíp komunikácie pomocou komunikačných symbolov. Činnosť obidvoch typov je predvedená na jednoduchých príkladoch.

Systémy prevodníkov pozostávajú z niekoľkých zásobníkových prevodníkov, ktoré postupne pracujú na vstupnom reťazci. Každý prevodník je v jednom zo stavov z jeho vlastnej konečnej množiny stavov a pristupuje k najvyššiemu symbolu svojho vlastného zásobníka. Komunikácia prevodníkov v systéme prebieha pomocou protokolov a stratégií podobných tým, ktorými spolupracujú komponenty v kooperujúcich distribuovaných gramatických systémoch. Táto komunikácia je založená na tom, že výstupná páska jednej komponenty je stotožnená so vstupnou páskou nasledujúcej komponenty. Ľubovoľný prevodník teda po vykonaní svojho prekladu odošle výsledok na vstup ďalšieho prevodníka, ktorý opäť vykoná svoj preklad. Výsledkom prekladu systému je výstup posledného prevodníka. Kapitola 6 zavádza novú, formálnu definíciu týchto systémov a následne skúma jej ďalšie vlastnosti. Na príkladoch na konci kapitoly je ukázané, prečo je systém prevodníkov efektívnejší a silnejší ako samotný prevodník.

Kapitola 7 popisuje implementáciu systému prevodníkov, ktorý vykonáva prevod výrazov z infixovej do postfixovej notácie. Obsahuje návrh a popis jednotlivých prevodníkov ako aj návrh celého systému a taktiež aj jeho programovú implementáciu – vytvorené triedy, ich funkcie a taktiež popis jednotlivých častí výslednej aplikácie.

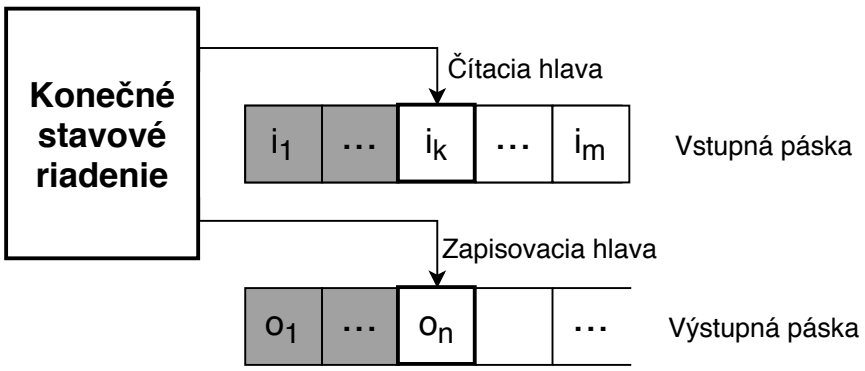
Táto práca predpokladá znalosť pojmov a symbolov preberaných v predmete IFJ – Formální jazyky a překladače. Ak čitateľ tento predmet neabsolvoval, môže si potrebné pojmy naštudovať z [11].

Kapitola 2

Konečné prevodníky

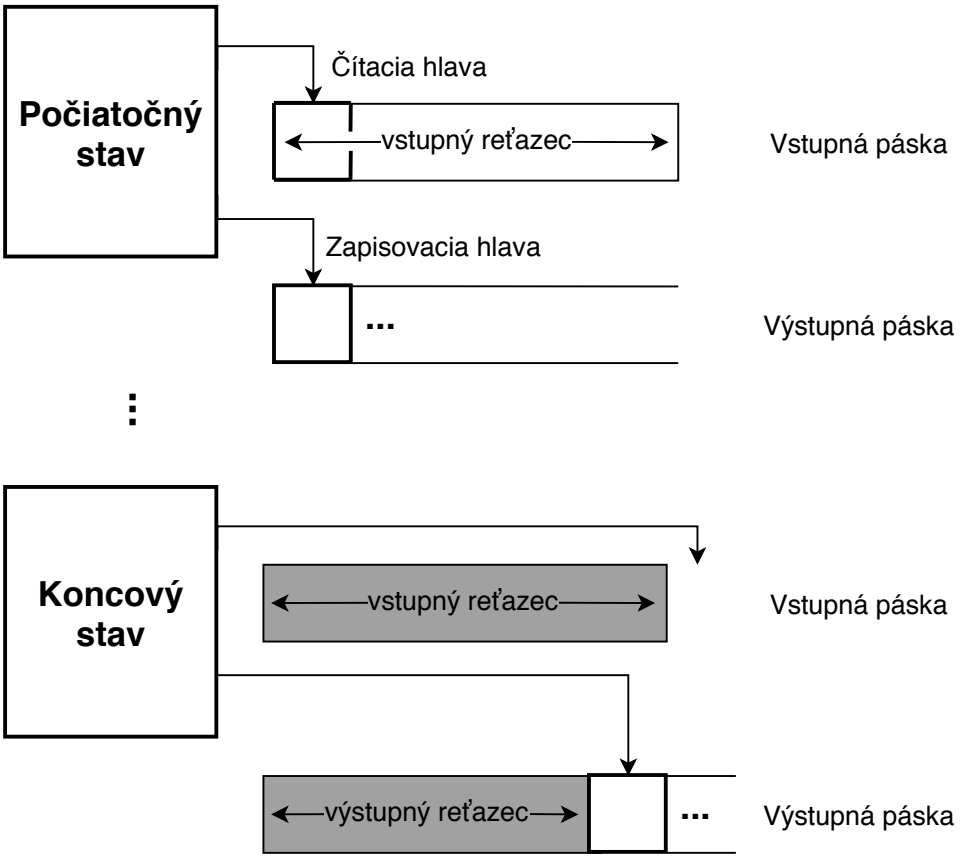
Konečný prevodník, ktorého definíciu je možné nájsť v [11], prípadne v [1], vychádza z konečného automatu, od ktorého sa však odlišuje tým, že okrem vstupnej pásky, z ktorej číta vstupné symboly, obsahuje aj výstupnú pásku na ktorú generuje výstup. Prevodníky sú teda schopné pri každom prechode zapisovať výstupné symboly. Môžeme ich chápať, že sú to svojim spôsobom „prekladacie stroje“. Môžu byť však použité aj v iných režimoch ako v prekladovom. V generovacom režime prevodníky zapisujú symboly na obe pásky a v prekladovom režime dokážu čítať z obidvoch pások. Taktiež je možné obrátiť smer ich prekladu [17]. Konečný automat definuje formálny jazyk definovaním sady akceptovaných reťazcov symbolov, zatiaľ čo konečný prevodník definuje vzťahy medzi množinami reťazcov [19].

Vstupná aj výstupná páska prevodníka sú rozdelené na štvorce, kde každý štvorec predstavuje jeden symbol vstupného alebo výstupného reťazca. Na začiatku prekladu obsahuje vstupná páska reťazec symbolov vstupnej abecedy a výstupná páska je prázdna. Čítacia hlava sa nachádza vždy nad aktuálnym symbolom na vstupe. Táto hlava sa pohybuje iba jedným smerom, a to zľava doprava. Zapisovacia hlava, pohybujúca sa tiež zľava doprava, umožňuje zapisovať symboly výstupnej abecedy na výstupnú pásku. Tá je nekonečná z pravej strany. V priebehu výpočtu sa zapisovacia hlava nachádza vždy na prvom voľnom štvorci výstupnej pásky. Všetky tieto skutočnosti vidíme na obrázku 2.1. Symbol i_k predstavuje aktuálny symbol na vstupe (*input symbol*) a o_n je posledným výstupným symbolom (*output symbol*).



Obr. 2.1: Konečný prevodník s popísanými jednotlivými časťami (obrázok bol prevzatý z [11] a následne bol editovaný)

Čítacia aj zapisovacia hlava sú riadené konečným počtom stavov spolu s prechodovou reláciou, ktorá je špecifikovaná ako súbor výpočtových pravidiel. V jednom kroku, prevodník aplikáciou týchto pravidiel prečíta žiaden alebo jeden vstupný symbol a na základe aktuálneho stavu zapíše symbol (môže ich byť aj viac) na výstupnú pásku. Po prečítaní symbolu sa čítacia hlava posunie na nasledujúci vstupný symbol a zapisovacia hlava sa posunie na prvé voľné miesto za výstupný reťazec. Jeden zo stavov sa označuje ako počiatočný a prevodník môže mať jeden alebo viac stavov koncových. Ak prevodník v konečnom počte krokov preloží vstupný reťazec na výstupný a skončí v niektorom z koncových stavov, tak tomuto procesu hovoríme preklad. Spracovanie vstupného reťazca na výstupný vidíme na obrázku 2.2.



Obr. 2.2: Preklad vstupného reťazca na výstupný (obrázok prevzatý z [11])

Definícia 2.1 *Konečný prevodník M je pätica: $M = (Q, \Sigma, R, s, F)$ kde*

- *Q je konečná množina stavov;*
- *Σ je abeceda symbolov, pre ktorú platí že, $\Sigma \cap Q = \Delta$ a $\Sigma = \Sigma_I \cup \Sigma_O$, kde Σ_I je abeceda vstupných symbolov a Σ_O je abeceda výstupných symbolov;*
- *$R \subseteq Q(\Sigma_I \cup \{\varepsilon\}) \times Q\Sigma_O^*$ je relácia predstavujúca množinu pravidiel;*
- *$s \in Q$ je počiatočný stav;*
- *$F \subseteq Q$ je množina koncových stavov.*

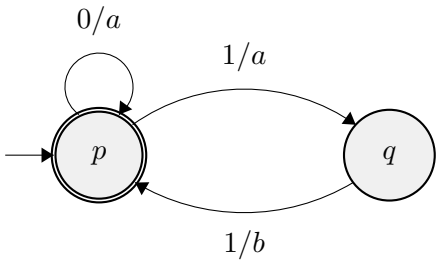
Ak máme pravidlo $(pa, qz) \in \Sigma$ kde $p, q \in Q, a \in \Sigma_I \cup \{\varepsilon\}$ a $z \in \Sigma_O^*$, tak okrem zápisu (pa, qz) môžeme použiť ekvivalentný zápis v tvare $pa \Vdash qz$. Tento zápis značí, že prevodník pri prečítaní symbolu a urobí prechod zo stavu p do q a na výstupnú pásku zapíše symbol z. Ak sa $a = \varepsilon$, zo vstupnej pásky nie je prečítaný žiadny symbol. Aby sa na jednotlivé pravidlá dalo odkazovať, sú označené. Ak chceme vyjadriť že sa jedná o pravidlo r, zapíšeme pravidlo ako $r : pa \Vdash qz$, kde pa označuje *left-hand side – lhs(r)* a qz *right-hand side – rhs(r)* pravidla r.

Prevodníky, rovnako ako konečné automaty, môžeme znázorniť ako graf, kde uzly označujú jednotlivé stavy. Počiatočný stav je označený šípkou a koncový stav dvojitým krúžkom. Prechody sú orientované hrany z jedného do druhého (alebo aj do toho istého) uzlu a sú pomenované ako *vstupný / výstupný* symbol.

Príklad 2.1 *Majme konečný prevodník* $M = (Q, \Sigma, R, s, F)$ *kde*

- $Q = \{p, q\}$;
- $\Sigma = \{0, 1, a, b\}$;
- $R = \{p0 \Vdash pa, p1 \Vdash qa, q1 \Vdash pb\}$;
- $s = p$;
- $F = \{p\}$.

Grafickú reprezentáciu tohto prevodníka vidíme na obrázku 2.3.



Obr. 2.3: Konečný prevodník M z príkladu 2.1

Definícia 2.2 *Nech* $M = (Q, \Sigma, R, s, F)$ *je konečný prevodník. Konfigurácia prevodníka* M *je trojica* (q, x, y)*, kde*

(1) $q \in Q$ *je aktuálny stav;*

(2) $x \in \Sigma_I^*$ *je vstupný reťazec na vstupnej páske, s najľavejším symbolom* x *pod čítacou hlavou;*

(3) $y \in \Sigma_O^*$ *je doposiaľ vygenerovaná časť výstupného reťazca.*

Konfigurácia prevodníka teda určuje jeho aktuálny stav. Pomocou týchto informácií (aktuálny stav, vstupný reťazec a vygenerovaný reťazec) sme schopní určiť budúce správanie prevodníka. Použitím pravidiel prevodník prechádza medzi jednotlivými stavmi, čiže v každom kroku zmení svoju konfiguráciu.

Definícia 2.3 *Nech* $M = (Q, \Sigma, R, s, F)$ *je konečný prevodník a nech* $C_1 = (q, ax, y)$, $C_2 = (r, x, yz)$ *sú dve konfigurácie, kde* $q, r \in Q, a \in (\Sigma \cup \{\varepsilon\}), x \in \Sigma_I^*$ *a* $y, z \in \Sigma_O^*$. *Ak máme pravidlo* $p : qa \vdash rz$, *potom* M *môže vykonať krok z* C_1 *do* C_2 *použitím pravidla* p. *Zapisujeme*

$$(q, ax, y) \vdash (r, x, yz) \qquad [p]$$

alebo zjednodušene

$$(q, ax, y) \vdash (r, x, yz).$$

Na obrázku 2.3 je možné vidieť, že sa tam nachádzajú iba pravidlá, ktorých aplikovaním sa zo vstupu prečíta a aj na výstup zapíše symbol, avšak existujú aj pravidlá, pri ktorých je vstupný alebo výstupný symbol rovný ε. V prípade, že vstupný symbol je ε, tak prevodník urobí krok z jedného stavu do druhého, na výstup zapíše výstupný symbol, avšak čítacia hlava sa neposunie ďalej, takže v ďalšom kroku prevodník prečíta rovnaký vstupný symbol. V druhom prípade, ak je výstupný symbol ε, opäť prevodník prejde z jedného stavu do druhého, prečíta vstupný symbol, ale na výstup nič nezapíše. Okrem týchto dvoch prípadov, môže ešte nastať situácia, keď vstupný aj výstupný symbol sú rovné ε. V tejto situácii prevodník iba prejde z jedného stavu do druhého, ale zo vstupu neprečíta žiadny symbol a takisto na výstup nič nevypíše.

Definícia 2.4 *Nech* $M = (Q, \Sigma, R, s, F)$ *je konečný prevodník a nech* C *a* C' *sú dve konfigurácie prevodníka* M.

1. M *urobí 0 prechodov z* C *do* C' *iba keď* $C = C'$. *Zapisujeme*

   $$C \vdash^0 C' \quad [\varepsilon]$$

   alebo zjednodušene

   $$C \vdash^0 C'.$$

2. *Majme sekvenciu konfigurácií* $C_0, \ldots, C_k$ *pre* $k \geq 1$ *také, že platí* $C_{i-1} \vdash C_i$ *pre všetky* i, $i = 1, \ldots, k$. *Môžeme povedať, že* M *urobí* k *prechodov z* C_0 *do* C_k *na základe pravidiel* $r_1, \ldots, r_k$. *Zapisujeme*

   $$C_0 \vdash^k C_k \quad [r_1, \ldots, r_k]$$

   alebo jednoduchšie

   $$C_0 \vdash^k C_k.$$

Pokiaľ $C \vdash^k C'$ *pre nejaké* $k \geq 1$, *potom píšeme* $C \vdash^+ C'$.
Pokiaľ $C \vdash^k C'$ *pre nejaké* $k \geq 0$, *potom píšeme* $C \vdash^* C'$.

Definícia 2.5 *Uvažujme prevodník* $M = (Q, \Sigma, R, s, F)$ *a nech* $x \in \Sigma_I^*$ *a* $y \in \Sigma_O^*$. *Môžeme povedať, že* y *je výstupom pre vstup* x, *práve keď* $(q_0, x, \varepsilon) \vdash^* (q, \varepsilon, y)$ *pre nejaké* $q \in F$. *Preklad definovaný prevodníkom* M, $T(M)$, *je nasledovný:*

$$T(M) = \{(x, y) : x \in \Sigma_I^*, y \in \Sigma_O^*, (q_0, x, \varepsilon) \vdash^* (q, \varepsilon, y) \text{ pre nejaké } q \in F\}$$

Vstupný jazyk, odpovedajúci prekladu $T(M)$, je definovaný ako

$$L_I(M) = \{x : (x, y) \in T(M) \text{ pre nejaké } y \in \Sigma_O^*\}$$

Výstupný jazyk, odpovedajúci prekladu $T(M)$, je definovaný ako

$$L_O(M) = \{y : (x, y) \in T(M) \text{ pre nejaké } x \in \Sigma_I^*\}$$

Predtým, ako sa dá výstupný reťazec y považovať za preklad vstupného reťazca x, musí vstupný reťazec x prejsť prevodníkom M z počiatočného do koncového stavu.

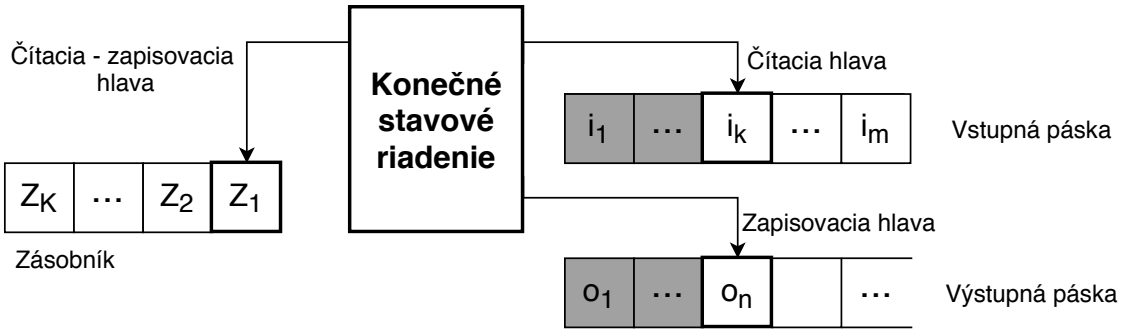
Príklad 2.2 *Uvažujme konečný prevodník z príkladu 2.1. Preskúmaním zadania a prislúchajúceho obrázku 2.3 zistíme, že preklad daný týmto prevodníkom je nasledovný:*

$$T(M) = \{(0^n 1 1^k, a^n a b^k) : n \geq 0, k \ \geq 1\}$$

Kapitola 3

Zásobníkové prevodníky

Zásobníkový prevodník, ktorý je definovaný v [11] alebo v [1], je založený na zásobníkovom automate. Od zásobníkového automatu sa však odlišuje tým, že okrem zásobníkovej a vstupnej pásky obsahuje aj výstupnú pásku, na ktorú vypisuje symboly výstupnej abecedy (na obrázku 3.1 symboly $o_1 \dots o_n$). Od konečného prevodníka sa odlišuje pomocným zásobníkom, na ktorý zapisuje a z ktorého číta symboly zásobníkovej abecedy (na obrázku znázornené ako $Z_1 \dots Z_k$). Všetky spomenuté časti a symboly sú znázornené na obrázku 3.1.



Obr. 3.1: Zásobníkový prevodník s jednotlivými časťami (obrázok bol prevzatý z [11] a následne bol editovaný)

Zásobníkový prevodník pracuje na rovnakom princípe ako konečný prevodník. Na začiatku sa nachádza v počiatočnom stave, vstupná páska obsahuje vstupný reťazec, výstupná páska je prázdna a naviac sa na zásobníku nachádza počiatočný symbol. V počiatočnom stave sa čítacia hlava nachádza na prvom symbole vstupného reťazca. Každý krok prevodníku znamená, že čítacia hlava neprečíta žiaden alebo prečíta jeden symbol zo vstupného reťazca a podľa tohto symbolu, stavu v ktorom sa nachádza a symbolu na vrchole zásobníku vykoná prechod. Podľa pravidla prechodu následne zásobníková hlava prepíše symbol z vrcholu zásobníku reťazcom symbolov zásobníkovej abecedy a zapisovacia hlava vypíše výstupný symbol. Za preklad vstupného reťazca na výstupný môžeme považovať to, ak prevodník prečíta celý vstup, z počiatočného stavu v konečnom počte krokov skončí v niektorom z koncových stavov a vypíše výstupný reťazec.

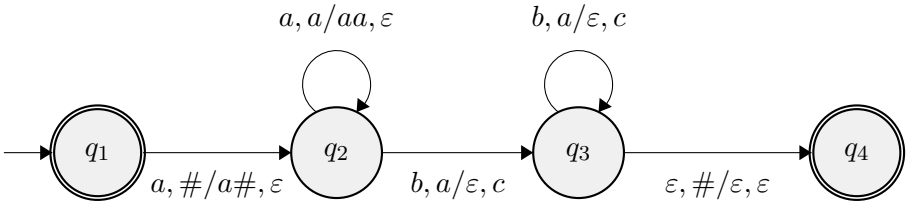
Definícia 3.1 *Zásobníkový prevodník M je osmica [7]: $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$ kde*

- *Q je konečná množina stavov;*

- Σ, Γ, Δ *sú abecedy, kde* Σ *je vstupná abeceda M, jej elementy sa nazývajú vstupné symboly.* Γ *je zásobníková abeceda M, jej elementy sa nazývajú zásobníkové symboly.* Δ *je výstupná abeceda M, jej elementy sa nazývajú výstupné symboly;*
- δ *je prechodová relácia z* $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ *do* $Q \times \Gamma^* \times \Delta^*$. δ *sa nazýva prechodová tabuľka prevodníka M, jej elementy sa nazývajú prechodové pravidlá prevodníka M;*
- $q_0 \in Q$ *je počiatočný stav;*
- $Z_0 \in \Gamma$ *je počiatočný symbol na zásobníku;*
- $F \subseteq Q$ *je množina koncových stavov.*

Každá prechodová relácia je v tvare $(q, \mu, \beta, p, \gamma, \rho)$, *kde* q *a* p *sú stavy v* Q, μ *je vstupný symbol alebo* ε, β *je zásobníkový symbol alebo* ε, γ *je reťazec, ktorý nahradí vrchol zásobníku a* ρ *je výstupný symbol alebo* ε.

Na obrázku 3.2 vidíme grafickú reprezentáciu zásobníkového prevodníka, ktorá sa od konečného prevodníka líši iba v označení prechodu. V tomto prípade sa ešte uvádza aj aktuálny stav zásobníka a reťazec, ktorým sa nahradí symbol na vrchole zásobníka. Označenie prechodu má teda tvar: *vstupný symbol, symbol na vrchole zásobníka / reťazec symbolov, ktorý nahradí vrchol zásobníka, výstupný symbol.* Namiesto zápisu $(q, \mu, \beta, p, \gamma, \rho)$, zvyčajne píšeme: $r = q\mu\beta \Vdash p\gamma\rho$ a platí $(r \in \delta)$.



Obr. 3.2: Grafická reprezentácia zásobníkového prevodníka

Definícia 3.2 *Nech* $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$ *je zásobníkový prevodník. Vstupný zásobníkový automat* M_I *je definovaný ako*

$$M_I = (Q, \Sigma, \Gamma, \delta_I, q_0, Z_0, F)$$

kde $\delta_I = \{qAa \vdash pu : qAa \Vdash puv \in \delta \text{ pre nejaké } v \in \Delta^*\}$.

Definícia 3.3 *Konfigurácia prevodníka* $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$ *je štvorica* (q, x, α, y), *kde*

(1) $q \in Q$ *reprezentuje aktuálny stav;*

(2) $x \in \Sigma^*$ *reprezentuje vstupný reťazec na vstupnej páske, s najľavejším symbolom x pod čítacou hlavou. Ak sa* $x = \varepsilon$ *predpokladá sa, že celý vstupný reťazec je prečítaný;*

(3) $\alpha \in \Gamma^*$ *reprezentuje obsah zásobníka, kde najľavejší symbol z* α *je braný ako symbol na vrchole zásobníka. Ak sa* $\alpha = \varepsilon$, *predpokladá sa, že zásobník je prázdny;*

(4) $y \in \Delta^*$ *reprezentuje doposiaľ vyprodukovaný výstup.*

Pre každý vstupný reťazec má zásobníkový prevodník M množinu možných konfigurácií. Ak si označíme doposiaľ prečítanú časť reťazca písmenom u, tak celý vstupný reťazec má tvar $v = ux$. Konfigurácia sa nazýva *počiatočná*, ak $q = q_0, u = \varepsilon$ a $\alpha = Z_0$. Znamená to, že prevodník sa nachádza v počiatočnom stave q_0, ešte neprečítal žiadny symbol, výstupná páska je prázdna a na vrchole zásobníka sa nachádza počiatočný symbol. Konfiguráciu považujeme za *akceptujúcu* v prípade, že $x = \varepsilon$, čiže vstupná páska je prázdna a q je jedným z koncových stavov.

Definícia 3.4 *Nech $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$ je zásobníkový prevodník. Krok vykonaný prevodníkom M je binárnou reláciou $\vdash$, rovnako ako pri konečnom prevodníku. Zapisujeme*

$$(q, ax, Z\gamma, y) \vdash (r, x, \alpha\gamma, yz)$$

ak δ obsahuje (q, a, Z, r, α, z) kde $q, r \in Q, a \in (\Sigma \cup \{\varepsilon\}), x \in \Sigma^*, Z \in (\Gamma \cup \{\varepsilon\}), \alpha, \gamma \in \Gamma^*$ a $y, z \in \Delta^*$. Platí, že $(q, ax, Z\gamma, y)$ a $(r, x, \alpha\gamma, yz)$ označujú konfigurácie prevodníka M.

Prechodové pravidlá sa používajú na definovanie možných krokov prevodníka. Uvažujme prechod z definície 3.4 Ak $a \neq \varepsilon$ a v prípade, že sa prevodník nachádza v stave q, vstupným symbolom je symbol a a symbol na vrchole zásobníka je Z, potom môže prevodník urobiť krok do stavu r. Čítacia hlava sa posunie o jeden symbol doprava, symbol na vrchole zásobníka sa nahradí reťazcom α. Ak $\alpha = \varepsilon$, zásobník je vyprázdnený a žiadne ďalšie kroky nie sú možné. V prípade, že $a = \varepsilon$, prechod sa nazýva $\varepsilon - prechod$. Vstupný symbol sa v tomto prípade neberie do úvahy, čítacia hlava sa neposúva, avšak stav prevodníka sa môže zmeniť a môže dojsť aj k úprave zásobníka.

Definícia 3.5 *Ak $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$ je zásobníkový prevodník a ak máme sekvenciu konfigurácií $C_0, \ldots, C_k$ pre $k \geq 1$ také, že platí $C_{i-1} \vdash C_i$ pre všetky i, $i = 1, \ldots, k$, tak prevodník M urobí k prechodov z C_0 do C_k podľa pravidiel $r_1, \ldots, r_k$. Zapisujeme*

$$C_0 \vdash^k C_k \quad [r_1, \ldots, r_k]$$

alebo zjednodušene

$$C_0 \vdash^k C_k.$$

Ak $C_0 \vdash^k C_k$ pre nejaké $k \geq 1$, potom píšeme $C_0 \vdash^+ C_k$.
Ak $C_0 \vdash^k C_k$ pre nejaké $k \geq 0$, potom píšeme $C_0 \vdash^* C_k$.

Definícia 3.6 *Ak máme zásobníkový prevodník $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$, tak môžeme povedať, že $y \in \Delta^*$ je výstupom pre $x \in \Sigma^*$, ak $(q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \alpha, y)$ pre nejaké $q \in F$ a $\alpha \in \Gamma^*$.*
Preklad definovaný prevodníkom M, označovaný ako $T(M)$, je:

$$T(M) = \{(x, y) : x \in \Sigma^*, y \in \Delta^*, (q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \alpha, y) \text{ pre nejaké } q \in F \text{ a } \alpha \in \Gamma^*\}.$$

Aj v prípade prázdneho zásobníka môžeme tvrdiť, že $y \in \Delta^*$ je výstupom pre $x \in \Sigma^*$ ak $(q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon, y)$ pre nejaké $q \in F$.
Preklad definovaný prevodníkom M, v prípade prázdneho zásobníka, označovaný ako $T_e(M)$ je:

$$T_e(M) = \{(x, y) : x \in \Sigma^*, y \in \Delta^*, (q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon, y) \text{ pre nejaké } q \in Q\}.$$

Vstupný jazyk L_I, korešpondujúci prekladu $T(M)$, je definovaný ako:

$$L_I(M) = \{x : (x, y) \in T(M) \text{ pre nejaké } y \in \Delta^*\}$$

Výstupný jazyk L_O, korešpondujúci prekladu $T(M)$, je definovaný ako:

$$L_O(M) = \{y : (x, y) \in T(M) \text{ pre nejaké } x \in \Sigma^*\}$$

Preskúmaním definície 3.1 je možné zistiť, že z jednej konfigurácie môže prevodník vykonať viacero krokov, čo môže predstavovať problém. Hovoríme, že prevodník pracuje nedeterministicky. Existuje však aj deterministický prevodník, ktorý tento problém odstraňuje.

Definícia 3.7 *Nech $M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F)$ je zásobníkový prevodník. Prevodník M je deterministický, ak pre každé pravidlo $r \in \delta$ s ľavou stranou v tvare pAa kde $a \in \Sigma \cup \{\varepsilon\}$, spĺňa nasledujúcu vlastnosť:*

$$\{r\} = \{r' : r' \in \delta, pAa = lhs(r') \text{ alebo } pA\varepsilon = lhs(r')\}$$

Pre každý stav $q \in Q, A \in \Gamma, a \in \Sigma \cup \{\varepsilon\}$, ak má ľavá strana pravidla r tvar pAa, tak neexistuje žiadne ďalšie pravidlo s ľavou stranou v tvare pAa alebo pA. Znamená to, že ak je na vstupe hocijaký vstupný reťazec, tak prevodník M vykoná jedinečnú sekvenciu prechodov.

Príklad 3.1 *Zoberme si do úvahy zásobníkový prevodník z obrázka 3.2. Z tohto obrázka môžeme zostaviť formálnu definíciu tohto prevodníka, ktorá je nasledovná:*

$$M = (Q, \Sigma, \Gamma, \Delta, \delta, q_o, Z_o, F) \text{ kde platí:}$$

- $Q = \{q_1, q_2, q_3\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, \#\}$;
- $\Delta = \{c\}$;
- $\delta = \{(q_1, a, \#, q_2, a\#, \varepsilon), (q_2, a, a, q_2, aa, \varepsilon), (q_2, b, a, q_3, \varepsilon, c), (q_3, b, a, q_3, \varepsilon, c),$ $(q_3, \varepsilon, \#, q_4, \varepsilon, \varepsilon)\}$
- $q_0 = q_1$;
- $Z_0 = \#$;
- $F = \{q_1, q_4\}$.

Z relácie δ a z obrázku 3.2 vidíme, že prvým spracovávaným symbolom vstupného reťazca musí byť symbol a, ktorý sa hneď uloží na zásobník. Po spracovaní tohto symbolu prejde prevodník do stavu q_2. V tomto stave zostáva, až kým je vstupný symbol rovný a. Spracovaním prvého symbolu b, prevodník prejde do stavu q_3, zo zásobníka odoberie jeden symbol a a na výstup vypíše c. V tomto stave zostáva, až kým je vstupným symbolom b. Do koncového stavu q_4 prejde prevodník vtedy, ak zo vstupu nič neprečíta a ak je na vrchole zásobníku počiatočný symbol #. Z tohto vyplýva, že prevodník prijíma reťazce v tvare $a^i b^i$, kde $i \geq 0$ a produkuje výstup v tvare c^i. V prípade ak $i = 0$, prevodník ihneď skončí v stave q_1.

Preklad prevodníka si ilustrujeme na vstupnom reťazci v tvare *aaabbb*. Prevodník vykoná nasledujúcu sekvenciu krokov:

$$
\begin{aligned}
(q_1, aaabbb, \#, \varepsilon) &\vdash (q_1, aabbb, a\#, \varepsilon) && [1]\\
&\quad (q_1, abbb, aa\#, \varepsilon) && [1]\\
&\quad (q_1, bbb, aaa\#, \varepsilon) && [1]\\
&\quad (q_2, bb, aa\#, c) && [2]\\
&\quad (q_2, b, a\#, cc) && [3]\\
&\quad (q_2, \varepsilon, \#, ccc) && [3]\\
&\quad (q_3, \varepsilon, \varepsilon, ccc) && [4]
\end{aligned}
$$

Túto sekvenciu krokov môžeme zapísať aj v tvare:

$$(q_1, aaabbb, \#, \varepsilon) \vdash^7 (q_3, \varepsilon, \varepsilon, ccc) \qquad [1112334]$$

alebo zjednodušene:

$$(q_1, aaabbb, \#, \varepsilon) \vdash^7 (q_3, \varepsilon, \varepsilon, ccc)$$

V tomto príklade má i hodnotu 3, čiže výstupom je reťazec *ccc*.

Kapitola 4

Prekladové gramatiky

Preklad je funkcia alebo všeobecnejšie, mapovanie reťazcov medzi dvomi textami, ktoré majú rovnaký význam, ale sú napísané v rôznych jazykoch. Existujú dva možné prístupy k špecifikácii prekladov [3]. Prvým sú prevodníky, ktoré boli bližšie skúmané v kapitolách 2 a 3. V tejto kapitole sa budeme zaoberať druhým prístupom, a to prekladovými gramatikami, ktoré sú jedným z formálnych systémov na opis syntaxou riadených prekladov [13]. Prekladová gramatika G je veľmi podobná bezkontextovej gramatike, s tým rozdielom, že každé pravidlo v G má na pravej strane dva reťazce. Použitím pravidla, teda G generuje dvojice reťazcov, ktoré patria do prekladu definovaného touto gramatikou. Pomocou prekladových gramatík je možné pri každom preklade dokázať rovnocennosť medzi vstupnými a výstupnými reťazcami [11].

Definícia 4.1 *Prekladová gramatika je štvorica [11]* $G = (N, T, P, S)$ *kde:*

- N *je abeceda neterminálov;*
- T *je abeceda terminálov, pre ktorú platí, že* $T \cap N = \varnothing$ *a* $T = T_I \cup T_O$*, kde* T_I *je vstupná abeceda a* T_O *je výstupná abeceda;*
- P *je konečná množina prepisovacích pravidiel v tvare*

$$A \twoheadrightarrow u_0 B_1 u_1 \dots B_n u_n | v_0 B_1 v_1 \dots B_n v_n$$

  kde $|$ *je špeciálny symbol, pre ktorý platí* $| \notin T \cap N$*,* $B_j \in N$ *pre* $j = 1, \dots, n$*,* $u_i \in T_I^*$ *pre* $i = 0, \dots, n$ *a* $v_i \in T_O^* (n = 0$ *implikuje* $x = u_0$ *a* $y = v_0)$*;*

- $S \in N$ *je počiatočný symbol*

Ak si zoberieme do úvahy pravidlo $p : A \twoheadrightarrow u_0 B_1 u_1 \dots B_n u_n | v_0 B_1 v_1 \dots B_n v_n$, tak A reprezentuje ľavú stranu p, $lhs(p)$. Okrem toho, $u_0 B_1 u_1 \dots B_n u_n$ označuje vstup pravej strany p, $irhs(p)$[1] a $v_0 B_1 v_1 \dots B_n v_n$ je výstup pravej strany pravidla p, $orhs(p)$[2].

Schéma syntaktického prekladu, spojená s prekladovou gramatikou, je množina párov získaná z pravidiel prekladovej gramatiky. Táto sada pravidiel obsahuje zdrojovú gramatiku G_1 a cieľovú gramatiku G_2 prekladovej schémy. Prekladová gramatika a schéma syntaktického prekladu sú iba variácie toho istého modelu.

[1] input right-hand side pravidla p
[2] output right-hand side pravidla p

Príklad 4.1 *Prekladová gramatika G pre obrátenie reťazca. Vezmime si napríklad reťazec* 001*. Jeho prekladom je zrkadlový reťazec v tvare* 100*. Táto prekladová gramatika G je tvorená nasledujúcimi produkciami:*

1. $S \twoheadrightarrow 0S|S0$
2. $S \twoheadrightarrow 1S|S1$
3. $S \twoheadrightarrow \varepsilon|\varepsilon$

Prekladovú gramatiku G je možné nahradiť nasledujúcou schémou prekladu (G_1, G_2) :

	Zdrojová gramatika G_1	Cieľová gramatika G_2
(1)	$S \to 0S$	$S \to S0$
(2)	$S \to 1S$	$S \to S1$
(3)	$S \to \varepsilon$	$S \to \varepsilon$

Tabuľka 4.1: Schéma prekladu (G_1, G_2)

Vstupno-výstupný pár pravidla sa získa vygenerovaním sekvencie párov reťazcov $\alpha|\beta$ nazývaných prekladová forma, kde α je vstupná forma a β je výstupná forma. Počiatočným symbolom v tejto gramatike je symbol S, teda prekladová forma má na začiatku tvar $S|S$. Na túto formu môžeme aplikovať prvé pravidlo, čiže prvé S expandujeme použitím produkcie $S \to 0S$. Následne nahradíme výstupnú formu z S na $S0$. Tomuto expandovaniu hovoríme, že prekladová gramatika G derivuje dvojicu slov na inú dvojicu slov. Výsledkom aplikácie prvého pravidla je prekladová forma $0S|S0$. Opätovným použitím pravidla (1), opäť rozšírime každé S a získame tak $00S|S00$. Ak použijeme pravidlo (2), dostávame $001S|S100$, a následným použitím pravidla (3), dostaneme 001|100. Na túto prekladovú formu už nie je možné aplikovať žiadne pravidlo a preto je 001|100 v preklade definovanom touto prekladovou gramatikou.

Definícia 4.2 *Nech* $G = (N, T, P, S)$ *je prekladová gramatika, kde* $x, y, u, v \in (N \cup T)^*$, $p \in P$*. Nech* x, u *obsahujú rovnaký počet neterminálnych symbolov. Môžeme povedať, že* $x\ lhs(p)\ y|u\ lhs(p)\ v$ *priamo derivuje* $x\ irhs(p)\ y|u\ orhs(p)\ v$ *podľa pravidla p v gramatike G. Zapisujeme:*

$$x\ lhs(p)\ y|u\ lhs(p)\ v \Rrightarrow x\ irhs(p)\ y|u\ orhs(p)\ v \qquad [p]$$

alebo zjednodušene:

$$x\ lhs(p)\ y|u\ lhs(p)\ v \Rrightarrow x\ irhs(p)\ y|u\ orhs(p)\ v$$

Príklad 4.2 *Nech* $G = (N, T, P, S)$ *je prekladová gramatika, ktorá prevádza aritmetické výrazy do z infixovej do prefixovej notácie. Vstupná a výstupná abeceda sú nasledovné:*

$$T_I = \{+, *, (,), i\} \qquad T_O = \{+, *, i'\}$$

Pravidlá prekladovej gramatiky G sú tieto:

1. $E \twoheadrightarrow E + T| + ET$
2. $E \twoheadrightarrow T|T$

3. $T \twoheadrightarrow T * F| * TF$

4. $T \twoheadrightarrow F|F$

5. $F \twoheadrightarrow (E)|E$

6. $F \twoheadrightarrow i|i'$

Gramatika G obsahuje tri neterminály – E, T a F, kde E je počiatočný symbol. Môžeme tvrdiť, že gramatika G spraví priamy derivačný krok podľa definície 4.2. Zapisujeme:

$$F * T| * FT \ggg (E) * T| * ET \qquad [5]$$

kde

$$5.\ F \twoheadrightarrow (E)|E$$

Aplikovaním uvedeného pravidla sa výraz na ľavej strane prepíše na odpovedajúci výraz vpravo. Tento príklad však popisuje možnosť, keď je pravidlo použité iba raz. V praxi je však nutné pravidlá použiť aj viackrát, preto je nutné v definícii 4.2 rozšíriť $\ggg$ na $\ggg^n$ pre $n \geq 0$.

Definícia 4.3 *Nech $G = (N, T, P, S)$ je prekladová gramatika.*

1. *Pre akékoľvek $u \in (N \cup T_I)^*$ a akékoľvek $v \in (N \cup T_O)^*$, G urobí nula derivačných krokov z $u|v$ do $u|v$ podľa ε, zapisujeme:*

$$u|v \ggg^0 u|v \qquad [\varepsilon]$$

2. *Nech $u_0, \ldots, u_n \in (N \cup T_I)^*$ a $v_0, \ldots, v_n \in (N \cup T_O)^*$, pre nejaké $n \geq 1$ a nech platí $i = 1, \ldots, n$,*

$$u_{i-1}|v_{i-1} \ggg u_i|v_i \qquad [p_i]$$

kde $p_i \in P$. Potom G urobí n derivačných krokov z $u_0|v_0$ do $u_n|v_n$ na základe pravidiel $p_1, \ldots, p_n$, zapisujeme ako:

$$u_0|v_0 \ggg^n u_n|v_n \qquad [p_1, \ldots, p_n]$$

Vychádzajúc z príkladu 4.1, ak máme daný vstupný reťazec x, je možné vytvoriť deriváciu x zo S pomocou produkcií v prekladovej schéme. Predpokladajme, že $S = \alpha_0 \twoheadrightarrow \alpha_1 \twoheadrightarrow \alpha_2 \twoheadrightarrow \ldots \twoheadrightarrow \alpha_n = x$ je derivácia. Na základe toho je možné vytvoriť deriváciu prekladovej formy, zapísané ako $\alpha_0|\beta_0 \twoheadrightarrow \alpha_1|\beta_1 \twoheadrightarrow \ldots \twoheadrightarrow \alpha_n|\beta_n$ alebo jednoduchšie $\alpha_0|\beta_0 \ggg^n \alpha_n|\beta_n$. Platí, že $\alpha_0|\beta_0 = S|S$ kde S je počiatočný symbol, $\alpha_n|\beta_n = x|y$, kde každé β_i sa získa z β_{i-1} aplikáciou pravidla cieľovej gramatiky. Toto pravidlo odpovedá pravidlu zdrojovej gramatiky použitej pri prechode z α_{i-1} do α_i. Reťazec y je výstupom pre reťazec x.

Definícia 4.4 *Nech $G = (N, T, P, S)$ je prekladová gramatika. Preklad definovaný gramatikou G, $T(G)$, je nasledovný:*

$$T(G) = \{u|v : S|S \ggg^* u|v \text{ kde } u \in T_I^* \text{ a } v \in T_O^*\}$$

Príklad 4.3 *Zoberme si do úvahy pravidlá prekladovej gramatiky z príkladu 4.2. Nech máme vstup v tvare $i+i*i$, pre ktorý nájdeme výstup. Prekladová gramatika vykoná nasledujúcu deriváciu:*

$$\begin{aligned} E|E &\Rrightarrow E+T|+ET \\ &\Rrightarrow T+T|+TT \\ &\Rrightarrow F+T|+FT \\ &\Rrightarrow i+T|+i'T \\ &\Rrightarrow i+T*F|+i'*TF \\ &\Rrightarrow i+F*F|+i'*FF \\ &\Rrightarrow i+i*F|+i'*i'F \\ &\Rrightarrow i+i*i|+i'*i'i' \end{aligned}$$

Vidíme, že prekladová gramatika preložila vstupný reťazec $i+i*i$ na $+i'*i'i'$, pričom vykonala osem derivačných krokov. Celú deriváciu je možné zapísať aj v skrátenom tvare:

$$E|E \Rrightarrow^{8} i+i*i|+i'*i'i'$$

Definícia 4.5 *Nech $G=(N,T,P,S)$ je prekladová gramatika. Existujú dve bezkontextové gramatiky, ktoré sú základom každej prekladovej gramatiky. Prvá je vstupná gramatika G_I, ktorá sa získa odstránením druhého reťazca na pravej strane každého pravidla v G. Je definovaná nasledovne:*

$$G_I=(N,T_I,P_I,S) \text{ kde } P_I=\{A\to x : A\twoheadrightarrow x|y\in P\}.$$

Druhá je výstupná gramatika G_O, ktorú získame odstránením prvého reťazca na pravej strane každého pravidla v G. Jej definícia je nasledovná:

$$G_O=(N,T_O,P_O,S) \text{ kde } P_O=\{A\to y : A\twoheadrightarrow x|y\in P\}.$$

Vstupná gramatika G_I z príkladu 4.2 obsahuje tieto pravidlá:
$E\to E+T$
$E\to T$
$T\to T*F$
$T\to F$
$F\to (E)$
$F\to i$

Výstupná gramatika G_O je zložená z nasledujúcich pravidiel:
$E\to +ET$
$E\to T$
$T\to *TF$
$T\to F$
$F\to E$
$F\to i'$

Kapitola 5

Gramatické systémy

V klasickej teórii formálneho jazyka sa gramatiky používajú samostatne, jedna gramatika vytvára jeden jazyk. Gramatický systém, prvýkrát skúmaný v [12], je súbor gramatík, ktoré spolupracujú na základe špecifikovaného protokolu na vytváraní jedného jazyka. Modelovanie distribúcie, vyššia popisná sila a nižšia zložitosť sú iba niektoré z dôvodov zavedenia gramatických systémov. Tieto systémy sú ďalšou z možností ako spracovávať bezkontextové, ale aj iné typy jazykov. Je možné rozlíšiť dva základné typy gramatických systémov: *sekvenčné* a *paralelné* [5]. V nasledujúcich podkapitolách sú uvedené základné definície oboch typov gramatických systémov. Väčšina definícií je prebratá z [5].

5.1 CD gramatické systémy

Kooperujúci distribuovaný gramatický systém, skrátene CD gramatický systém, je zástancom sekvenčných gramatických systémov. Pozostáva z niekoľkých gramatík, ktoré generujú jednu spoločnú vetnú formu striedaním procesu prepisovania. Štruktúra CD gramatického systému je podobná štruktúre tzv. „blackboard“ modelu, ktorý sa používa v oblasti riešenia problémov [15]. Práca tohto systému je založená na tom, že v jednom momente je aktívna iba jedna gramatika, takže pravidlá jednotlivých gramatík sú spracovávané postupne jedno za druhým. Kooperačný protokol určuje, ktorá gramatika bude v danom okamihu aktívna a aj to, kedy sa táto gramatika stane neaktívnou a prenechá tak doposiaľ vygenerovanú vetnú formu ďalšej gramatike.

Definícia 5.1 *CD gramatický systém stupňa n, $n \geq 1$, je (n+3)-tica:*

$$\Gamma = (N, T, S, P_1, \ldots, P_n)$$

kde

- *N je abeceda neterminálov;*
- *T je abeceda terminálov, pričom platí $N \cap T = \varnothing$;*
- *$S \in N$ je počiatočný symbol;*
- *P_i, pre $1 \leq i \leq n$, je konečná množina prepisovacích pravidiel nad $N \cup T$, nazývané komponenty gramatického systému*

Ak chceme explicitne špecifikovať gramatiky ako komponenty CD gramatického systému, môžeme zapísať Γ v tvare $\Gamma = (N, T, S, G_1, \ldots, G_n)$ kde $G_I = (N, T, S, P_i)$, pre $1 \leq i \leq n$.

Definícia 5.2 *Nech* $\Gamma = (N, T, S, P_1, \ldots, P_n)$ *je gramatický systém.*

1. *Pre každé i,* $1 \leq i \leq n$*, je ukončovacia derivácia i-tou komponentou (označená* $\Rightarrow^t_{P_i}$*) definovaná ako:*

$$x \Rightarrow^t_{P_i} y \iff x \Rightarrow^*_{P_i}, \text{ pričom neexistuje žiaden reťazec } z \text{ taký, že } y \Rightarrow_{P_i} z.$$

2. *Pre každé i,* $1 \leq i \leq n$*, je k-kroková derivácia i-tou komponentou (označená* $\Rightarrow^{=k}_{P_i}$*) definovaná ako:*

$$x \Rightarrow^k_{P_i} y \iff \text{existuje } x_1, \ldots x_{k+1} \in (N \cup T)^* \text{ také, že } x = x_1, y = x_{k+1} \text{ a pre každé } j = 1, \ldots, k \text{ platí } x_j \Rightarrow_{P_i} x_{j+1}.$$

   Ďalej platí, že $x \Rightarrow^0_{P_i} y$*, práve vtedy ak* $x = y$*.*

3. *Pre každé i,* $1 \leq i \leq n$*, je najviac k-kroková derivácia i-tou komponentou (označená* $\Rightarrow^{\leq k}_{P_i}$*) definovaná ako:*

$$x \Rightarrow^{\leq k}_{P_i} y \iff x \Rightarrow^{=k'}_{P_i} \text{ pre nejaké } k' \leq k.$$

4. *Pre každé i,* $1 \leq i \leq n$*, je najmenej k-kroková derivácia i-tou komponentou (označená* $\Rightarrow^{\geq k}_{P_i}$*) definovaná ako:*

$$x \Rightarrow^{\geq k}_{P_i} y \iff x \Rightarrow^{=k'}_{P_i} \text{ pre nejaké } k' \geq k.$$

Normálny $*$-režim derivácie, $\Rightarrow^*_{P_i}$, znázorňuje prípad, keď komponenta pracuje tak dlho ako chce a nie je pri tejto práci žiadnym spôsobom limitovaná. V ukončovacom móde, označovaný tiež ako t-mód (*terminating*), vykonáva komponenta derivačné kroky tak dlho, kým neprepíše všetky neterminály, objavujúce sa na ľavej strane pravidiel tejto komponenty (maximálne využitie jeho kompetencie). V momente, keď ďalej prepisovať nemôže, začína svoju činnosť nasledujúca komponenta. Režim $=k$ derivácie odpovedá k krokom priamej derivácie za sebou s využitím i-tej komponenty. Režim $\leq k$ derivácie nesie zo sebou obmedzenie, pretože komponenta môže vykonať najviac k zmien. Naopak, režim $\geq k$ derivácie vyžaduje aby komponenta vykonala aspoň k zmien, takže vyžaduje určité minimálne pôsobenie komponenty. Množinu všetkých derivačných režimov môžeme definovať takto:

$$D = \{*, t\} \cup \{\leq k, =k, \geq k : k \geq 1\}.$$

Definícia 5.3 *Jazyk generovaný CD gramatickým systémom* $\Gamma = (N, T, S, P_1, \ldots, P_n)$ *v derivačnom režime* $f \in D$ *je:*

$$L_f(\Gamma) = \{w \in T^* : S \Rightarrow^f_{P_{i_1}} w_1 \Rightarrow^f_{P_{i_2}} \ldots \Rightarrow^f_{P_{i_\ell}} w_\ell = w, \ell \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq \ell\}.$$

Príklad 5.1 *Nech* $\Gamma = (\{S, A, A', B, B'\}, \{a, b, c\}, S, P_1, P_2)$ *je gramatický systém kde:*

$$P_1 = \{S \to S, S \to AB, A' \to A, B' \to B\},$$

$$P_2 = \{A \to aA'b, B \to cB', A \to ab, B \to c\}.$$

Pozrime sa detailnejšie na to, ako Γ pracuje v k-krokovom režime, kde *k=2*. Keďže počiatočným symbolom je S, tak z tohto stavu sa môže použiť iba komponenta P_1. Ak by sme aplikovali dvakrát pravidlo $S \to S$, nič by sa tým nezmenilo, takže vykonáme postupne kroky $S \Rightarrow_{P_1} S \Rightarrow_{P_1} AB$. Pravidlo $S \to S$ nám slúži na dodržanie podmienky dvoch derivačných krokov. Od tejto chvíle sa S nikde neobjavuje, takže P_1 končí svoju činnosť a nasleduje komponenta P_2. Ak aplikujeme neterminálne pravidlá $A \to aA'b$ a $B \to cB'$ dostávame $AB \Rightarrow_{P_2} aA'bB \Rightarrow_{P_2} aA'bcB'$. Na reťazec AB by sme ešte mohli aplikovať pravidlá $A \to ab$ a $B \to c$ a vtedy by bol výsledný reťazec v tvare abc. Žiadna iná kombinácia pravidiel komponenty P_2 nie je prípustná. Ak by sme aplikovali jedno neterminálne pravidlo (nahradenie A alebo B za A' resp. B') a jedno terminálne (odstránenie zostávajúcich symbolov A alebo B), tak na vzniknutý reťazec obsahujúci iba jeden neterminál, ktorý sa líši od S, by sa nedala použiť ani jedna z komponent. Znamená to, že všetky derivácie v Γ v režime *=2* sú v tvare:

$$\begin{aligned} S &\Rightarrow_{P_1} S \Rightarrow_{P_1} AB \\ &\Rightarrow_{P_2} aA'bB \Rightarrow_{P_2} aA'bcB' \\ &\Rightarrow_{P_1} aAbcB' \Rightarrow_{P_1} aAbcB \\ &\Rightarrow_{P_2} aaA'bbcB \Rightarrow_{P_2} aaA'bbccB' \Rightarrow_{P_1} \ldots \Rightarrow_{P_1} a^nAb^nc^nB \\ &\Rightarrow_{P_2} a^{n+1}b^{n+1}c^nB \Rightarrow_{P_2} a^{n+1}b^{n+1}c^{n+1} \end{aligned}$$

Preto pre tento gramatický systém platí: $L_{=2}(\Gamma) = \{a^nb^nc^n\} : n \geq 1$.

Pre režim ≥ 2 platí, že výsledok je rovnaký ako pre režim *=2*. Ak by tento systém pracoval v režime *f*, kde $f \in \{=1, \geq 1, *, t\} \cup \{\leq k : k \geq 1\}$, tak by aktívna komponenta mohla použiť iba jedno pravidlo, z čoho vyplýva, že jazyk generovaný systémom v režime *f* je: $L_f(\Gamma) = \{a^nb^nc^m : n, m \geq 1\}$. Tento výsledný jazyk je bezkontextový.

Ak by sme uvažovali režimy *=3*, $\geq k$ pre $k \geq 3$, tak pre tieto režimy neexistuje riešenie. Sekvencia derivačných krokov by opäť začala komponentou P_1, ktorá by bola schopná vykonať tri kroky, pretože by dvakrát použila pravidlo $S \to S$ a raz $S \to AB$, potom by svoju činnosť ukončila a predala riadenie komponente P_2, ktorá by však po použití dvoch pravidiel nemala k dispozícii ďalšie použiteľné pravidlo a tak by podmienka k, resp. $\geq k$ krokov nebola splniteľná. Takže platí: $L_{=3}(\Gamma) = L_{\geq 3}(\Gamma) = \varnothing$.

5.2 PC gramatické systémy

Paralelne komunikujúce (PC) gramatické systémy, boli zavedené v [16] s cieľom preskúmať koncepty ako paralelizmus, synchronizácia a komunikácia vo formálnych jazykoch. Tieto systémy, podobne ako CD gramatické systémy, pozostávajú z niekoľkých komponent, ktoré paralelne vytvárajú svoje vlastné vetné formy a ich práca je organizovaná tak, aby vytvorili jeden jazyk zložený iba z terminálnych symbolov. Jedna z komponent systému je označená ako *master* a jazyk ktorý generuje, je jazyk vygenerovaný systémom. Ostatné komponenty, ktoré sa nazývajú *slaves*, sa tiež podieľajú na derivácii, ale nemusia mať vplyv na generovanie výsledného reťazca. Kľúčovou vlastnosťou PC gramatického systému je komunikácia prostredníctvom komunikačných (*query*) symbolov. Tieto symboly umožňujú, aby si komponenty v systéme navzájom zdieľali reťazce. Ak nejaká komponenta zavedie komunikačný symbol vo svojej vetnej forme, proces prepisovania sa zastaví a vykoná sa jeden alebo niekoľko komunikačných krokov, v ktorých sa nahradia všetky výskyty komuni-

kačných symbolov aktuálnymi vetnými formami dopytovaných komponentov. Definície boli prebraté z [5] a [4].

Definícia 5.4 *PC gramatický systém stupňa n, $n \geq 1$ je $(n+3)$-tica [18]:*

$$\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n)),$$

kde

- *N je abeceda neterminálov;*
- *T je abeceda terminálov;*
- *$K = \{Q_1, Q_2, Q_3, \ldots, Q_n\}$ je konečná množina query symbolov, pričom musí platiť $N \cap T \cap K = \varnothing$;*
- *P_i je konečná množina prepisovacích pravidiel v tvare $A \to x$, kde $x \in (N \cup T \cup K)^*$ a $A \in N$, pre všetky i, $1 \leq i \leq n$;*
- *S_i je počiatočný symbol i-tej komponenty, $S_i \in N$ pre všetky $i, 1 \leq i \leq n$.*

Množiny $P_i, 1 \leq i \leq n$, sa nazývajú *komponenty systému*. Index i v Q_i ukazuje na *i-tú* komponentu P_i v Γ. Ak chceme explicitne uviesť gramatiku ako komponentu v Γ, môžeme napísať $\Gamma = (N, K, T, G_1, \ldots, G_n)$, kde $G_i = (N \cup K, T, S_i, P_i), 1 \leq i \leq n$. Jedna z komponent G_i, sa nazýva *master* gramatika systému Γ.

N-tica $(x_1, x_2, \ldots, x_n)$, kde $x_i \in (N \cup T \cup K)^*, 1 \leq i \leq n$, sa nazýva *konfigurácia* systému Γ. Konfiguráciu $(S_1, \ldots, S_n)$ nazývame *počiatočnou*.

Derivácia v PC gramatických systémoch pozostáva zo série komunikačných a prepisovacích krokov. Prepisovací krok nie je možný, ak je požadovaná komunikácia, ktorá nastane v prípade, že sa v niektorej konfigurácii objaví komunikačný symbol.

Definícia 5.5 *Nech $\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n)), n \geq 1$ je gramatický systém a nech $(x_1, x_2, \ldots, x_n)$ a $(y_1, y_2, \ldots, y_n)$ sú dve konfigurácie systému Γ. Potom môžeme povedať, že $(x_1, x_2, \ldots, x_n)$ priamo derivuje $(y_1, y_2, \ldots, y_n)$, značíme $(x_1, x_2, \ldots, x_n) \Rightarrow (y_1, y_2, \ldots, y_n)$, ak platí jedna z nasledujúcich prípadov:*

1. *Neexistuje žiadne x_i, ktoré by obsahovalo nejaký komunikačný symbol, t.j $x_i \in (N \cup T)^*$ pre $1 \leq i \leq n$. Pre každé i, $1 \leq i \leq n, x_i \Rightarrow_{P_i} y_i$ (y_i sa získa z x_i priamym derivačným krokom podľa i-tej komponenty) alebo $x_i = y_i$ pre $x \in T^*$.*
2. *Existuje nejaké $x_i, 1 \leq i \leq n$, ktoré obsahuje aspoň jeden výskyt komunikačného symbolu. Potom pre všetky $x_i, 1 \leq i \leq n$, pre ktoré platí $|x_i|_k$[1] $\neq 0, 2 \leq i \leq n$, môžeme písať $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}, t \geq 1$, kde $z_j \in (N \cup T)^*, 1 \leq i \leq n$ a $Q_{i_l} \in K, 1 \leq l \leq t$. Ak $|x_i|_k = 0$, pre všetky $j, 1 \leq j \leq t$, tak $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ a v systémoch:*

   (a) s návratom máme $y_{i_j} = S_{i_j}, 1 \leq j \leq t$.

   (b) bez návratu máme $y_{i_j} = x_{i_j}, 1 \leq j \leq t$

   Ak pre nejaké j, $1 \leq j \leq t, |x_{i_j}|_k \neq 0$, potom $y_i = x_i$. Pre všetky i, $1 \leq i \leq n$, také, že y_i nie je špecifikované vyššie, máme $y_i = x_i$.

[1] Zápis $|x_i|_k$ označuje počet komunikačných symbolov v x_i

Prvým prípadom je opis prepisovacieho kroku: ak nie je prítomný žiaden komunikačný symbol v niektorej vetnej forme, potom každá komponenta použije jedno zo svojich pravidiel prepisovania, s výnimkou tých komponent, ktoré už vytvorili koncový reťazec. V tomto prípade však môže dojsť k zablokovaniu derivácie, ak vetná forma niektorej z komponent nie je zložená výlučne z terminálov, ale nemôže sa na ňu uplatniť ani jedno z pravidiel.

Druhý prípad opisuje komunikačný krok. Ak sa vo vetnej forme x_i objaví komunikačný symbol Q_j, potom sa prepisovanie zastaví a je nutné vykonať komunikáciu. Každý výskyt Q_j v x_i sa nahradí aktuálnou vetnou formou x_j komponenty P_j, ale iba v prípade, že x_j neobsahuje komunikačné symboly. Ak by jeden z týchto vetných foriem x_j obsahoval komunikačný symbol, potom musí byť prvý x_j oslobodený od týchto symbolov a až potom môže vykonať zmenu v x_i. V komunikačnom kroku reťazec x_j nahradí komunikačný symbol Q_j. Následne môže komponenta pracovať v práci dvoma spôsobmi. V systéme s návratom sa komponenta musí vrátiť k počiatočnému symbolu a začať vytvárať novú vetnú formu, kým v systéme bez návratu môžu komponenty pokračovať v generovaní svojich súčasných vetných foriem. Komunikácia má vždy prednosť pred prepisovaním. Ak niektoré komunikačné symboly nie sú oslobodené v danom komunikačnom kroku, môžu byť oslobodené v ďalšom kroku. Aj v tomto prípade môže dojsť k zablokovaniu derivácie. Ak v konfigurácii nemôže byť žiadna vetná forma s výskytom komunikačného symbolu oslobodená od tohto symbolu vyššie uvedeným spôsobom, hovoríme že nastala kruhová komunikácia.

Definícia 5.6 *Nech* $\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n)), n \geq 1$ *je gramatický systém. Jazyk generovaný systémom* Γ *je definovaný ako [18]:*

$$L(\Gamma) = \{x \in T^* : (S_1, S_2, \ldots, S_n) \Rightarrow^* (x, \alpha_2, \ldots, \alpha_n), \alpha_i \in (N \cup T \cup K)^*, \text{ pre všetky } i = 2, \ldots, n\}.$$

Začína sa od n-tice axiómov $(S_1, \ldots, S_n)$ a pokračuje sa opakovanými prepisovacími a komunikačnými krokmi až kým komponenta P_1(označovaná ako *master*) nevytvorí koncový reťazec. Po vytvorení tohto reťazca nás nezaujímajú reťazce vytvorené ostatnými komponentami. Dva PC gramatické systémy sú ekvivalentné ak generujú rovnaký jazyk.

Definícia 5.7 *Nech* $\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n)), n \geq 1$ *je gramatický systém. Ak iba komponenta* P_1 *má povolené zavádzať komunikačné symboly (formálne zapísané ako* $P_i \subseteq (N \cup T)^* \times (N \cup T)^*$ *pre* $2 \leq i \leq n$*), tak hovoríme, že* Γ *je centralizovaný PC gramatický systém, v opačnom prípade je* Γ *necentralizovaný.*

Príklad 5.2 *Nech* $\Gamma = (\{S_1, S_1', S_2, S_3\}, K, \{a, b\}, (S_1, P_1), (S_2, P_2), (S_3, P_3))$ *je PC gramatický systém kde:*

$$P_1 = \{S_1 \to abc, S_1 \to a^2b^2c^2, S_1 \to aS_1', S_1' \to a^3Q_2, S_1' \to aS_1', S_1' \to a^3Q_2, S_2 \to b^2Q_3, S_3 \to c\},$$

$$P_2 = \{S_2 \to bS_2\},$$

$$P_2 = \{S_3 \to cS_3\}.$$

Preskúmajme ako pracuje systém Γ. Jedná sa o centralizovaný systém, pretože iba *master* komponenta P_1 obsahuje komunikačné symboly. Začíname s (S_1, S_2, S_3). Použitím pravidla $S_1 \to aS_1'$ v komponente P_1 a pravidiel v P_2 a P_3 prevedieme derivačný krok a následne dostaneme: $(S_1, S_2, S_3) \Rightarrow_r (aS_1', bS_2, cS_3)$, kde zápis $\Rightarrow_r$ značí derivačný krok s návratom.

Následne opäť vykonáme derivačný krok použitím pravidla $S_1' \to a^3 Q_2 s$ a tiež pravidiel v P_2 a P_3, čím dostaneme: $(aS_1', bS_2, cS_3) \Rightarrow_r (a^4 Q_2, b^2 S_2, c^2 S_3)$. Pretože je prítomný komunikačný symbol Q_2, je nutné vykonať komunikačný krok: $b^2 S_2$ sa pošle do komponenty P_1, čím sa nahradí Q_2. Keďže sa jedná o systém s návratom, tak komponenta P_2 začína opäť s počiatočným symbolom S_2. Výsledný tvar po vykonaní komunikačného kroku je: $(a^4 Q_2, b^2 S_2, c^2 S_3) \Rightarrow_r (a^4 b^2 S_2, S_2, c^2 S_3)$. Použitím pravidla $S_2 \to b^2 Q_3$ sa dorovná počet symbolov b vo vetnej forme komponenty P_1 – $(a^4 b^2 S_2, S_2, c^2 S_3) \Rightarrow_r (a^4 b^4 Q_3, bS_2, c^3 S_3)$. Opäť máme v komponente P_1 komunikačný symbol, takže vetnú formu P_3 skopírujeme namiesto Q_3 čím dostaneme: $(a^4 b^4 Q_3, bS_2, c^3 S_3) \Rightarrow_r (a^4 b^4 c^3 S_3, bS_2, S_3)$. Vykonaním ešte jedného derivačného kroku dostaneme výsledný reťazec terminálov v tvare $a^4 b^4 c^4$. Takže, všetky reťazce tvaru $a^n b^n c^n$ kde $n \geq 4$ je možné vyprodukovať týmto spôsobom.

Deriváciou v tvare:

$$(S_1, S_2, S_3) \Rightarrow_r (a^3 Q_2, bS_2, cS_3) \Rightarrow_r (a^3 bS_2, S_2, cS_3) \Rightarrow_r (a^3 b^3 Q_3, bS_2, c^2 S_3) \Rightarrow_r (a^3 b^3 c^2 S_3, bS_2, S_3) \Rightarrow_r (a^3 b^3 c^3, b^2 S_2, cS_3),$$

získame reťazec $a^3 b^3 c^3$, zatiaľ čo reťazce $a^2 b^2 c^2, abc$ sú vyprodukované priamo master komponentou P_1. Z tohto vyplýva, že jazyk generovaný týmto systémom je:

$$L_r(\Gamma) = L_{nr}(\Gamma) = \{a^n b^n c^n : n \geq 1\}.$$

$L_r(\Gamma)$ značí jazyk vygenerovaný systémom s režime s návratom a $L_{nr}(\Gamma)$ značí jazyk vygenerovaný systémom bez návratu. Pretože existuje iba jeden komunikačný symbol z P_1 do P_2 a P_3, tak sa jazyky vygenerované v oboch spomínaných režimoch zhodujú.

Kapitola 6

Systémy prevodníkov

Táto kapitola zavádza nový formálny model – systém zásobníkových prevodníkov, ktorých práca je veľmi blízka práci CD gramatických systémov. Tieto systémy pozostávajú z niekoľkých zásobníkových prevodníkov, ktoré spolupracujú na základe protokolov, podobných tým, podľa ktorých pracujú gramatické systémy. Aktivácia/deaktivácia komponentov je takmer identická ako v gramatických systémoch, s tým rozdielom, že prevodníky nemajú určený presný, minimálny alebo maximálny počet krokov ktoré musia vykonať, ale prepnutie na ďalšiu komponentu sa deje v momente, keď práve aktívny prevodník úspešne dokončí svoj preklad, čiže sa nachádza v koncovom stave. V jednom momente je tak v systéme aktívny iba jeden prevodník, ostatné sú „zamrznuté".

Celý systém si môžeme predstaviť ako konečnú sekvenciu prevodníkov, kde každý prevodník má vlastnú vstupnú aj výstupnú pásku. Preklad systému začína prvý prevodník, čiže jeho vstupná páska je aj vstupom celého systému. V našom prípade bude tento prevodník vždy označený ako M_1. Ak tento prevodník, na základe jeho pravidiel úspešne preloží vstupný reťazec na výstupný, predá tento výstup ako vstup pre nasledujúcu komponentu. Protokol o spolupráci teda spočíva v tom, že výstup jedného prevodníka je „pripojený" na vstup ďalšieho, takže na preklad jedného prevodníka naviaže svoj preklad iný prevodník v sekvencii. Na to, aby bola spolupráca možná, musia mať prevodníky v systéme kompatibilné vstupné a výstupné abecedy. V momente, keď posledná komponenta v sekvencii dokončí svoj preklad, tak jeho výstup môžeme považovať za výstupný reťazec celého systému.

V takto definovanom systéme nie je možný paralelizmus, pretože nasledujúci prevodník v sekvencii má k dispozícii celý vstup až v momente, keď práve aktívny prevodník úspešne vyprodukuje celý svoj výstup.

6.1 Základné definície

Definícia 6.1 *Systém zásobníkových prevodníkov stupňa n, je (n+3)-tica:*

$$\mathcal{T} = (\Sigma, \Gamma, \Delta, M_1, M_2, \ldots, M_n)$$

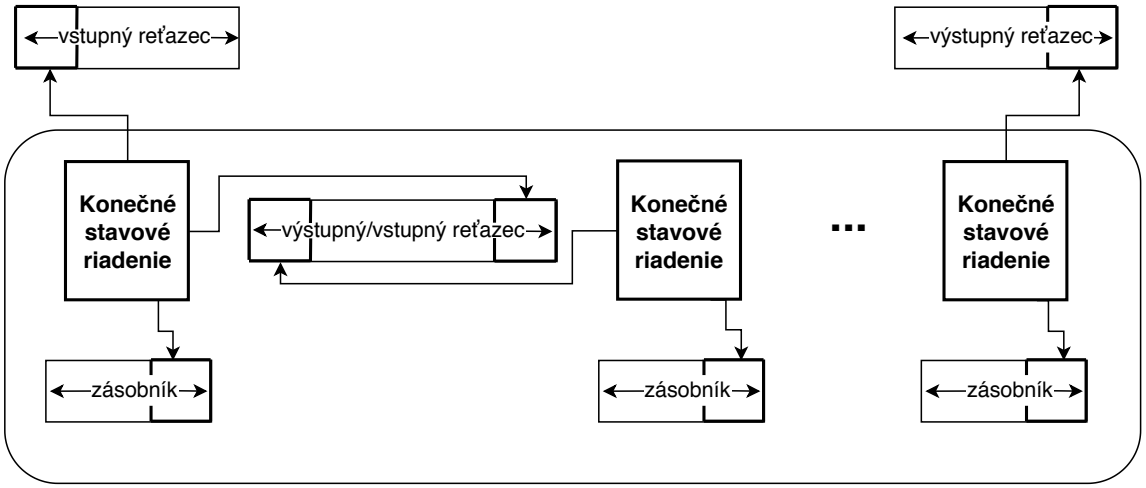
kde:

- Σ *je vstupná abeceda systému;*
- Γ *je zásobníková abeceda systému;*
- Δ *je výstupná abeceda systému;*

- *pre každé* $i = 1, \ldots, n$ *je* $M_i = (Q_i, \Sigma_i, \Gamma_i, \Delta_i, \delta_i, q_i, Z_i, F_i)$ *zásobníkový prevodník kde* Q_i *je konečná množina stavov,* $q_i \in Q_i$ *je počiatočný stav i-tého prevodníka* M_i*,* Σ_i *je vstupná abeceda,* Γ_i *je zásobníková abeceda,* Δ_i *je výstupná abeceda prevodníka* M_i*,* $Z_i \in \Gamma$ *je počiatočný symbol na zásobníku,* $F_i \subseteq Q$ *je množina koncových stavov prevodníka* M_i *a* δ *je prechodová relácia z* $Q_i \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ *do* $Q_i \times \Gamma^* \times \Delta^*$*.*

Zásobníkové prevodníky $M_1, M_2, \ldots, M_n$ *sa nazývajú komponenty systému* $\mathcal{T}$*.*

Vstupná a výstupná abeceda systému obsahujú symboly, ktoré sa objavujú na vstupe alebo výstupe prvej (sú vstupom celého systému), resp. poslednej komponenty (sú výstupom celého systému). Tieto abecedy nemusia byť rovnaké ako vstupné a výstupné abecedy ostatných komponentov, nakoľko v medzivýsledkoch spracovania sa môžu objaviť aj iné, pomocné symboly, ktoré sa však na celkovom výstupe systému neobjavia. Systém zásobníkových prevodníkov je schematický znázornený na obrázku 6.1.



Obr. 6.1: Systém zásobníkových prevodníkov

Definícia 6.2 *Konfigurácia systému prevodníkov* $\mathcal{T} = (\Sigma, \Gamma, \Delta, M_1, M_2, \ldots, M_n)$ *stupňa n, je štvorica* $M_i(q, x, \alpha, y)$ *kde:*

- $M_i = 1, \ldots, n$ *reprezentuje aktuálne pracujúci prevodník v sekvencii;*
- $q \in Q_i$ *reprezentuje aktuálny stav i-tej komponenty;*
- $x \in \Sigma_i^*$ *reprezentuje doposiaľ neprečítaný reťazec i-tej komponenty s najľavejším symbolom x aktuálne na vstupe;*
- $\alpha \in \Gamma_i^*$ *reprezentuje obsah zásobníka, pričom za jeho vrchol je považovaný najľavejší symbol z* α*;*
- $y \in \Delta_i^*$ *reprezentuje doposiaľ vygenerovaný výstup i-tej komponenty.*

Konfigurácia systému prevodníkov je takmer identická ako konfigurácia samotného zásobníkového prevodníka. Je to z dôvodu, že v systéme je v jednom okamihu aktívna iba jedna komponenta, takže konfigurácia celého systému je rovná konfigurácii aktuálne pracujúceho prevodníka. V prípade, že sa jedná o prvý prevodník v sekvencii, platí $\Sigma_i^* = \Sigma^*$ a obdobne

pre posledný prevodník platí $\Delta_i^* = \Delta^*$. Index M_i udáva o ktorý prevodník v sekvencii sa jedná.

Okamžitým popisom stavu systému prevodníkov $\mathcal{T}$ stupňa n, je $4n$-tica:

$$(q_1, x_1, \alpha_1, y_1, q_2, x_2, \alpha_2, y_2, \ldots, q_n, x_n, \alpha_n, y_n)$$

pre všetky $i, 1 \leq i \leq n$. Význam jednotlivých symbolov je rovnaký ako v prípade konfigurácie systému. Ak by sme brali do úvahy tento formát zápisu, tak jednotlivé symboly by sa pri prechode menili iba v prípade aktívneho prevodníka. Komponenty, ktoré už dokončili svoju činnosť by sa nachádzali v niektorom z ich koncových stavov, vstupný reťazec by bol rovný ε (celý vstup je prečítaný) a výstupná páska by obsahovala vyprodukovaný reťazec. Naopak, komponenty ktoré ešte nezačali svoju činnosť, by sa nachádzali v počiatočnom stave, na zásobníku by mali počiatočný symbol a ich výstupná páska by bola prázdna. Takéto značenie by bolo efektívne v prípade, ak by sme uvažovali paralelne pracujúce zásobníkové prevodníky [2].

Definícia 6.3 *Nech $\mathcal{T} = (\Sigma, \Gamma, \Delta, M_1, M_2, \ldots, M_n)$ je systém prevodníkov stupňa n, a nech $C_1 = M_i(q, ax, Z\gamma, y)$ a $C_2 = M_i(r, x, \alpha\gamma, yz)$ sú dve konfigurácie systému $\mathcal{T}$ pre $i = 1, \ldots, n$, kde $q, r \in Q_i, a \in (\Sigma_i \cup \{\varepsilon\}), x \in \Sigma_i^*, y, z \in \Delta_i^*, Z \in (\Gamma_i \cup \{\varepsilon\})$ a $\alpha, \gamma \in \Gamma_i^*$. Krok systému $\mathcal{T}$, vykonaný aktívnou komponentou i (reprezentovaný binárnou reláciou $\Vdash$) z konfigurácie C_1 do C_2 je definovaný ako:*

$$M_i(q, ax, Z\gamma, y) \Vdash M_i(r, x, \alpha\gamma, yz)$$

ak δ_i obsahuje (q, a, Z, r, α, z).

Platí, že ak aktívny prevodník M_i urobí krok z konfigurácie C_1 do C_2, tak aj celý systém $\mathcal{T}$ vykoná jeden krok. Aj tu existujú dve možnosti prechodu z jednej konfigurácie do druhej. Prvou je, že $a \neq \varepsilon$, čiže prevodník prečíta vstup symbol a posunie svoju čítaciu hlavu doprava o jeden symbol. Druhou možnosťou je, že $a = \varepsilon$, takže vstupný symbol sa neberie do úvahy a pracuje sa len so symbolom na vrchole zásobníka. Túto možnosť môžeme formálne zapísať ako

$$M_i(q, x, Z\gamma, y) \Vdash M_i(r, x, \alpha\gamma, yz)$$

ak δ_i obsahuje $(q, \varepsilon, Z, r, \alpha, z)$.

Okrem kroku v rámci aktívnej komponenty v systéme, je však nutné definovať aj taký typ kroku, v ktorom je aktívny prevodník v koncovom stave a musí tak dojsť k aktivácii nasledujúceho prevodníka. Tento typ kroku je definovaný nasledovne:

Definícia 6.4 *Nech $C = M_i(q, a, A, y)$ a $C' = M_{i+1}(r, x, B, z)$ sú dve konfigurácie systému $\mathcal{T}$. Pri kroku, keď dochádza k aktivácii nasledujúcej komponenty v sekvencii platí:*

- *$a = \varepsilon$ – vstupný reťazec komponenty M_i je celý prečítaný;*
- *$x = y$ – výstupný reťazec komponenty M_i je vstupom pre komponentu M_{i+1};*
- *$q \in F_i$ – komponenta M_i sa nachádza v jednom z jeho koncových stavov;*
- *$r = q_{0_{i+1}}, B = Z_{0_{i+1}}, z = \varepsilon$ – práve aktivujúca sa komponenta M_{i+1} sa nachádza v počiatočnom stave, na zásobníku má počiatočný symbol a má prázdnu výstupnú pásku.*

Aby sme odlíšili typy prechodov v systéme $\mathcal{T}$, tak si binárnu reláciu značiacu prechod, pri ktorom dochádza k aktivácii nasledujúcej komponenty označíme ako $\vdash^a$, kde a je skratka slova activation. *Aktivačný krok má teda tvar:*

$$M_i(q, a, A, y) \vdash^a M_{i+1}(r, x, B, z)$$

Definícia 6.5 *Nech $\mathcal{T} = (\Sigma, \Gamma, \Delta, M_1, M_2, \ldots, M_n)$ je systém prevodníkov stupňa n a majme sekvenciu konfigurácií aktívneho prevodníka $C_0, \ldots, C_k$, pre $k \geq 1$ také, že platí $C_{j-1} \vdash C_j$ pre všetky $j = 1, \ldots, k$. C_0 značí počiatočnú a C_k akceptujúcu konfiguráciu prevodníka M_i. Môžeme povedať, že prevodník M_i v systéme $\mathcal{T}$ urobí k prechodov z C_0 do C_k. Zapisujeme:*

$$C_0 \vdash^k C_k$$

Ak $C_0 \vdash^k C_k$ pre nejaké $k \geq 1$, potom môžeme napísať $C_0 \vdash^+ C_k$.
Ak $C_0 \vdash^k C_k$ pre nejaké $k \geq 0$, potom môžeme napísať $C_0 \vdash^* C_k$.

Ako v prípade jedného kroku, tak aj v prípade sekvencie krokov platí, že ak aktívny prevodník vykoná k krokov, tak aj celý systém $\mathcal{T}$ vykoná k krokov. Avšak sekvencia krokov systému $\mathcal{T}$ nepozostáva iba z krokov aktívneho prevodníka, ale je nutné zahrnúť aj aktivačné kroky. Sekvenciu krokov systému $\mathcal{T}$ stupňa n, je teda možné definovať ako:

$$1 : C_0 \vdash^k C_k \vdash^a 2 : C_0 \vdash^k C_k \vdash^a \ldots \vdash^a n : C_0 \vdash^k C_k$$

pričom počet vykonaných prechodov k v jednotlivých komponentoch nemusí byť rovnaký. Jednotlivé čísla $1, \ldots, n$ značia o ktorý prevodník v sekvencii sa jedná.

Definícia 6.6 *Ak máme systém prevodníkov $\mathcal{T} = (\Sigma, \Gamma, \Delta, M_1, M_2, \ldots, M_n)$ stupňa n, tak platí, že $y \in \Delta^*$ je výstupom celého systému $\mathcal{T}$ pre vstup $x \in \Sigma^*$, ak $M_1(q_0, x, Z_0, \varepsilon) \vdash^* M_n(q, \varepsilon, \alpha, y)$ pre nejaké $q \in F_n, \alpha \in \Gamma_i^*$. Platí, že q_0 je počiatočným stavom prvej komponenty a q je koncovým stavom poslednej komponenty v sekvencii.*

Definícia 6.7 *Preklad definovaný systémom prevodníkov $\mathcal{T}$ stupňa n, označovaný ako $T(\mathcal{T})$ je:*

$$T(\mathcal{T}) = \{(x, y) : x \in \Sigma^*, y \in \Delta^*, M_1(q_0, x, Z_0, \varepsilon) \vdash^* M_k(q, \varepsilon, \alpha, y) \text{ pre nejaké } q \in F_k, k \leq n \text{ a } \alpha \in \Gamma^*\}.$$

Prekladová množina systému $\mathcal{T}$ je tvorená dvojicou x a y, pre ktoré platí, že systém $\mathcal{T}$ urobí sekvenciu prechodov z počiatočného stavu prvého prevodníka v systéme do niektorého koncového stavu ľubovolného prevodníka. Prevodník nachádzajúci sa v koncovom stave má prázdnu vstupnú pásku a jeho výstupom je reťazec y, zatiaľ čo vstupom prvého prevodníka v systéme je reťazec x.

Definícia 6.8 *Vstupný jazyk L_I, korešpondujúci prekladu $T(\mathcal{T})$, je definovaný ako:*

$$L_I(\mathcal{T}) = \{x : (x, y) \in T(\mathcal{T}) \text{ pre nejaké } y \in \Delta^*\}$$

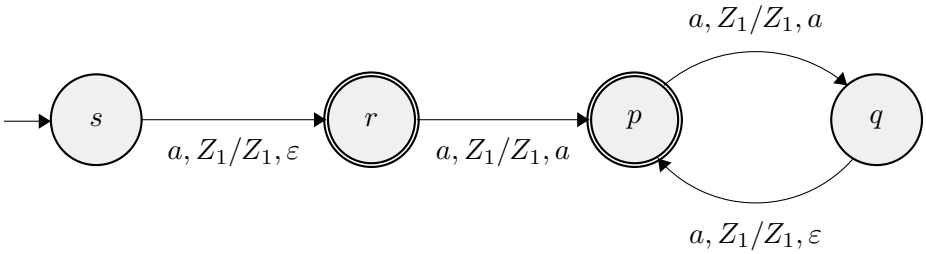
Definícia 6.9 *Výstupný jazyk L_O, korešpondujúci prekladu $T(\mathcal{T})$, je definovaný ako:*

$$L_O(\mathcal{T}) = \{y : (x, y) \in T(\mathcal{T}) \text{ pre nejaké } x \in \Sigma^*\}$$

6.2 Demonštračné príklady

Príklad 6.1 *Nech* $\mathcal{T}_1 = (\Sigma, \Gamma, \Delta, M_1)$ *je systém prevodníkov s jedným typom komponenty. V tomto systéme platí:*

- $\Sigma = \{a\}$;
- $\Gamma = \{Z_1\}$;
- $\Sigma = \{\varepsilon\}$;
- $M_1 = (Q_1, \Sigma_1, \Gamma_1, \Delta_1, \delta_1, q_1, Z_1, F_1)$ *kde:*
  - $Q_1 = \{s, r, p, q\}$;
  - $\Sigma_1 = \{a\}$;
  - $\Gamma_1 = \{Z_1\}$;
  - $\Delta_1 = \{a, \varepsilon\}$;
  - $\delta_1 = \{(s, a, Z_1, r, Z_1, \varepsilon), (r, a, Z_1, p, Z_1, a), (p, a, Z_1, q, Z_1, a), (q, a, Z_1, p, Z_1, \varepsilon)\}$;
  - $q_1 = \{s\}$;
  - $Z_1 = \{Z_1\}$;
  - $F_1 = \{p, r\}$;



Obr. 6.2: Prevodník M_1 zo systému $\mathcal{T}_1$

Systém $\mathcal{T}_1$ prijíma vstupný jazyk v tvare $L_i(\mathcal{T}_1) = \{a^{2^k} : k \geq 0\}$ a definuje preklad $T(\mathcal{T}_1) = \{(x, \varepsilon) : x \in L_i(\mathcal{T}_1)\}$. Tento systém je zaujímavý tým, že počet komponentov závisí od hodnoty k vo vstupnom reťazci. Keďže sa celá sekvencia skladá z jedného typu komponenty, ktorý je zobrazený na obrázku 6.2, tak si tento systém môžeme predstaviť ako systém s jednou komponentou, ktorá dookola spracováva svoj výstup, až kým neskončí v jednom z jeho koncových stavov. V činnosti systému $\mathcal{T}_1$ je teda možné identifikovať cyklus. Každým prechodom komponentou sa zníži počet symbolov a vo vstupnom reťazci na polovicu. Systém teda prijíma iba reťazce $a, a^2, a^4, a^8, \ldots$. Počet prechodov komponentou je vždy rovný $k + 1$, (napr. ak $k = 2$, počet prechodov je $k + 1 = 3$).

Činnosť tohto systému si ukážeme na vstupnom reťazci v tvare a^8:

$$\begin{aligned}
M_1(s, a^4, Z_1, \varepsilon) &\Vdash M_1(r, a^3, Z_1, \varepsilon) \\
&\Vdash M_1(p, a^2, Z_1, a^1) \\
&\Vdash M_1(q, a^1, Z_1, a^2) \\
&\Vdash M_1(p, \varepsilon, Z_1, a^2) \qquad M_1 \Vdash^a M_2 \text{ (v tomto prípade } M_2 = M_1) \\
&\Vdash M_1(s, a^2, Z_1, \varepsilon) \\
&\Vdash M_1(r, a^1, Z_1, \varepsilon) \\
&\Vdash M_1(p, \varepsilon, Z_1, a^1) \qquad M_1 \Vdash^a M_1 \\
&\Vdash M_1(s, a^1, Z_1, \varepsilon) \\
&\Vdash M_1(r, \varepsilon, Z_1, \varepsilon)
\end{aligned}$$

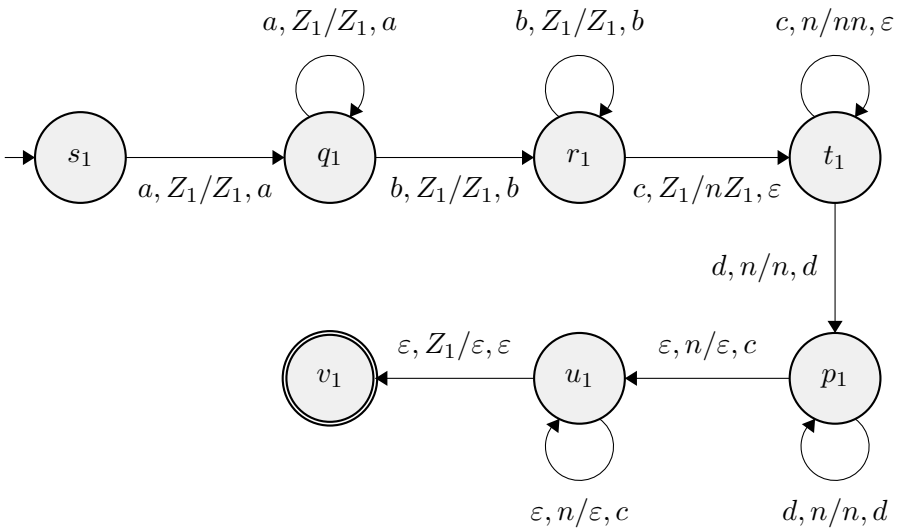
Tento systém je možné chápať aj tak, akoby bol zložený z konečných automatov, keďže celkovým výstupom je prázdny reťazec a nepracuje sa ani so zásobníkom. Avšak pridaním ďalších komponentov, by sa vstup dal prekladať na zaujímavejší výstupný reťazec. Cieľom príkladu však bolo ukázať, že systémy prevodníkov dokážu spracovať aj také jazyky, ktoré konečné alebo zásobníkové prevodníky samostatne nedokážu.

Príklad 6.2 *Nech* $\mathcal{T}_2 = (\Sigma, \Gamma, \Delta, M_1, M_2, M_3)$ *kde:*

- $\Sigma = \{a, b, c, d\}$;
- $\Gamma = \{Z_1, Z_2, Z_3, n, m, a, b\}$;
- $\Sigma = \{a\}$;
- $M_1 = (\{s_1, q_1, r_1, t_1, p_1, u_1, v_1\}, \{a, b, c, d\}, \{n, m\}, \{a, b, c, d\}, \delta_1, \{s_1\}, \{Z_1\}, \{v_1\})$;
  - $\delta_1 = \{(s_2, a, Z_1, q_1, Z_1, a), (q_1, a, Z_1, q_1, Z_1, a), (q_1, b, Z_1, r_1, Z_1, b),$
    $(r_1, b, Z_1, r_1, Z_1, b), (r_1, c, Z_1, t_1, nZ_1, \varepsilon), (t_1, c, n, t_1, nn, \varepsilon), (t_1, d, n, p_1, n, d),$
    $(t_1, d, n, t_1, n, d), (t_1, \varepsilon, n, u_1, \varepsilon, c), (u_1, \varepsilon, n, u_1, \varepsilon, c), (u_1, \varepsilon, Z_1, v_1, \varepsilon, \varepsilon)\}$;
- $M_2 = (\{s_3, q_3\}, \{a\}, \{a\}, \{Z_3\}, \delta_3, \{s_3\}, \{Z_3\}, \{q_3\})$;
  - $\delta_2 = \{(s_1, a, Z_2, q_2, aZ_2, \varepsilon), (q_2, a, aZ_1, q_2, aa, \varepsilon), (q_2, b, a, r_2, bb, \varepsilon),$
    $(r_2, b, b, r_2, bb, \varepsilon), (r_2, d, b, t_2, \varepsilon, a), (t_2, d, b, t_2, \varepsilon, a),$
    $(t_2, c, a, p_2, \varepsilon, a), (p_2, c, a, p_2, \varepsilon, a), (p_2, \varepsilon, Z_2, u_2, \varepsilon, \varepsilon)\}$;
- $M_3 = (\{s_2, q_2, r_2, t_2, p_2, u_2\}, \{a, b, c, d\}, \{a, b\}, \{a\}, \delta_2, \{s_2\}, \{Z_2\}, \{u_2\})$;
  - $\delta_3 = \{(s_3, a, Z_3, s_3, Z_3, aa), (s_3, \varepsilon, Z_3, q_3, \varepsilon, \varepsilon)\}$;

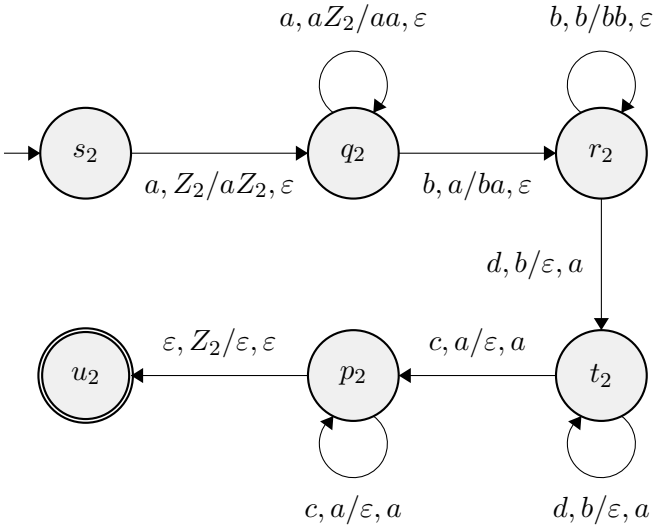
V druhom príklade máme systém, ktorého preklad je definovaný nasledovne:

$$T(\mathcal{T}_2) = \{(a^n b^m c^n d^m, a^{2*(m+n)} : m, n \geq 1)\}$$



Obr. 6.3: Prevodník M_1 v systéme $\mathcal{T}_2$

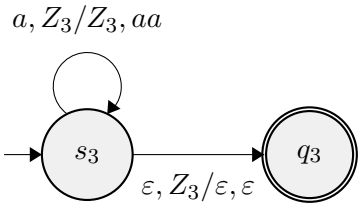
Tento systém prijíma jazyk, ktorý opäť nie je možné spracovať jednou komponentou. Všimnime si, že náš vstupný jazyk je veľmi podobný jazyku $a^n b^m c^m d^n, n, m \geq 1$, ktorý je bezkontextový, čiže je akceptovateľný zásobníkovým prevodníkom. Celý trik tohto systému spočíva v tom, že si vstupný reťazec prekonvertuje na vyššie spomínaný bezkontextový jazyk, ktorý je následne ľahko prijateľný. Túto konverziu vykonáva prevodník M_1, zobrazený na obrázku 6.3. Prevodník postupne vypisuje symboly a, b na výstup v poradí akom sa nachádzajú na vstupe. Symboly c si uloží na zásobník a symboly d vypíše na výstup. Nakoniec vypíše aj symboly c zo zásobníku, čím sa zrealizuje výmena symbolov c a d. Tento výstup následne predá komponente M_2 ako vstup.



Obr. 6.4: Prevodník M_2 v systéme $\mathcal{T}_2$

Komponenta M_2 (na obrázku 6.4) postupne ukladá vstupné symboly a, b na zásobník. Na vrchole zásobníka sú symboly b, ktorých počet je rovný hodnote m. Ak prevodník prečíta

prvý symbol c, tak zo zásobníka začne odstraňovať symboly b, čím skontroluje či je ich počet rovný. Rovnako to následne vykoná aj so symbolmi a a c. Pri odstraňovaní symbolov zo zásobníka postupne vypisuje symboly a na výstup, ktorý je predaný komponente M_3.



Obr. 6.5: Prevodník M_3 v systéme $\mathcal{T}_2$

Komponenta M_3 je veľmi jednoduchá a iba zdvojnásobí počet symbolov a vo vstupnom reťazci, takže jeho výstup je v tvare $a^{2*(m+n)}$. Táto komponenta by mohla byť nahradená inou komponentou, ktorá by produkovala iný ľubovoľný výstup. Činnosť systému si ukážeme na príklade, kde vstupom je reťazec *abbcdd*.

$M_1(s_1, abbcdd, Z_1, \varepsilon) \vdash M_1(q_1, bbcdd, Z_1, a) \vdash M_1(r_1, bcdd, Z_1, ab) \vdash M_1(r_1, cdd, Z_1, abb) \vdash$
$M_1(t_1, dd, nZ_1, abb) \vdash M_1(p_1, d, nZ_1, abbd) \vdash M_1(p_1, \varepsilon, nZ_1, abbdd) \vdash M_1(u_1, \varepsilon, Z_1, abbddc) \vdash$
$M_1(v_1, \varepsilon, \varepsilon, abbddc) \quad \mathbf{M_1} \vdash^{\mathbf{a}} \mathbf{M_2} \quad M_2(s_2, abbddc, Z_2, \varepsilon) \vdash M_2(q_2, bbddc, aZ_2, \varepsilon) \vdash$
$\vdash M_2(r_2, bddc, baZ_2, \varepsilon) \vdash M_2(r_2, ddc, bbaZ_2, \varepsilon) \vdash M_2(t_2, dc, baZ_2, a) \vdash M_2(t_2, c, aZ_2, aa) \vdash$
$\vdash M_2(p_2, \varepsilon, Z_2, aaa) \vdash M_2(u_2, \varepsilon, \varepsilon, aaa) \quad \mathbf{M_2} \vdash^{\mathbf{a}} \mathbf{M_3} \quad \vdash M_3(s_3, aaa, Z_3, \varepsilon) \vdash$
$\vdash M_3(s_3, aa, Z_3, a^2) \vdash M_3(s_3, a, Z_3, a^4) \vdash M_3(s_3, \varepsilon, Z_3, a^6) \vdash M_3(s_3, \varepsilon, \varepsilon, a^6)$

6.3 Systémy prevodníkov – zhrnutie

Z príkladov 6.1 a 6.2 vidíme, že stačí pridanie jedného alebo dvoch komponent k samotnému zásobníkovému prevodníku a vznikne nám systém prevodníkov, ktorý je schopný prijímať zložitejšie jazyky a teoreticky aj produkovať zložitejšie výstupy. Výhodou tohto systému je fakt, že prevodníky môžu do vstupného reťazca pridávať alebo odoberať symboly a následne vzniknutý reťazec môže spracovávať iná, alebo aj rovnaká komponenta. Z príkladu 6.1 vidíme, že stačí iba jeden typ komponenty v systéme a sme schopní namodelovať cyklus, čo je ďalšou výhodou týchto systémov. Na druhej strane sa môže zvýšiť počet prechodov jednotlivými komponentami, avšak tento efekt je neodvrátiteľný, ak chceme spracovávať zložitejšie jazyky alebo vytvárať komplikovanejšie programovacie konštrukcie.

Kapitola 7

Implementácia

V kapitole 6 sme preskúmali systémy zásobníkových prevodníkov z teoretického hľadiska a v tejto kapitole si ukážeme aj ich praktické využitie. Zavedieme si systém prevodníkov, ktorý identifikuje výrazy v zdrojovom kóde zapísanom v zdrojovom jazyku IFJ18[9] a prevedie ich z infixovej do postfixovej notácie. Aj keď na samotný prevod výrazu stačí jeden zásobníkový prevodník, pri potrebe lexikálnej a syntaktickej analýzy je nutné pridať ďalšie prevodníky, ktoré túto činnosť vykonajú a vznikne tak výsledný systém.

7.1 Návrh aplikácie

Na prevod výrazu z infixovej do postfixovej notácie je nutné postupné presunutie operátorov za ich operandy. Prvým krokom na splnenie tohto cieľa je identifikácia jednotlivých lexém v zdrojovom kóde a vytvorenie príslušných tokenov. Túto činnosť v našom systéme vykonáva prevodník M_1, ktorý je vstupným bodom celého systému. Tento prevodník pracuje ako lexikálny analyzátor.

Jednotlivé tokeny sú postupne predávané prevodníku M_2, ktorý vždy na základe dvoch za sebou prijatých tokenov zistí, či je ich kombinácia povolená. Taktiež umožňuje zistiť či postupnosť prijatých tokenov tvorí matematický výraz. Ak sa o výraz jedná, prijaté a skontrolované tokeny sa umiestnia na výstup, odkiaľ budú predané ako vstup pre ďalšiu komponentu. Prevodník M_2 teda kontroluje syntaktickú správnosť výrazu. Ak sa vo výraze objavia zátvorky, tento prevodník nie je schopný skontrolovať či sú vyvážené.

Kontrolu vyváženosti zátvoriek vykonáva zásobníkový prevodník M_3. Zakaždým, ak je aktuálnym vstupným symbolom ľavá zátvorka, umiestni ju na zásobník a ak je aktuálnym symbolom pravá zátvorka, odstráni jeden symbol z vrcholu zásobníka. Ak je po spracovaní celého vstupu zásobník prázdny, zátvorky sú vo výraze vyvážené.

Poslednou, najdôležitejšou súčasťou systému je zásobníkový prevodník M_4. Jeho úlohou je samotné prekonvertovanie výrazu. Ak pri čítaní vstupného reťazca narazí na operand, jednoducho ho umiestni na výstup a ak narazí na operátor alebo ľavú zátvorku, umiestni ich na zásobník pričom berie do úvahy prioritu operátorov.

Formálne je možné tento systém definovať nasledovne:

$$\mathcal{T} = (\Sigma, \Gamma, \Delta, M_1, M_2, M_3, M_4) \text{ kde:}$$

- $\Sigma = \{+, -, *, /, i, (,), <, >, <=, >=, !=, =\}$;
- $\Gamma = \{+, -, *, /, (, <, >, <=, >=, !=, =\}$;
- $\Delta = \{+, -, *, /, i, <, >, <=, >=, !=, =\}$.

Platí, že i je ľubovoľný identifikátor, celé alebo desatinné číslo. V nasledujúcich podsekciách bude činnosť jednotlivých komponent bližšie preskúmaná.

7.1.1 Prevodník M_1 – lexikálny analyzátor

Keďže prevodník M_1 je vstupným bodom systému, jeho abeceda vstupných symbolov je identická so vstupnou abecedou systému. Okrem vyššie spomínaných typov. môže byť symbol i v tomto prípade aj kľúčovým slovom alebo textovým literálom. Lexikálny analyzátor je jednoduchá komponenta, ktorého hlavnou úlohou je prechádzať vstupný reťazec znak po znaku, nájsť a rozpoznať jednotlivé lexémy a vytvoriť z nich im prislúchajúce tokeny. Aj keď hovoríme o systéme zásobníkových prevodníkov, tento prevodník nepotrebuje zásobník na svoju činnosť, čiže sa jedná o konečný prevodník. Vyprodukovaný token je vo formáte (typ, alias, hodnota), kde typ tokenu je vždy zhodný s typom lexémy. Alias vyjadruje číselnú skratku typu tokenu a hodnota je postupnosť symbolov vyjadrujúca identifikovaný operátor, reťazec alebo číslo.

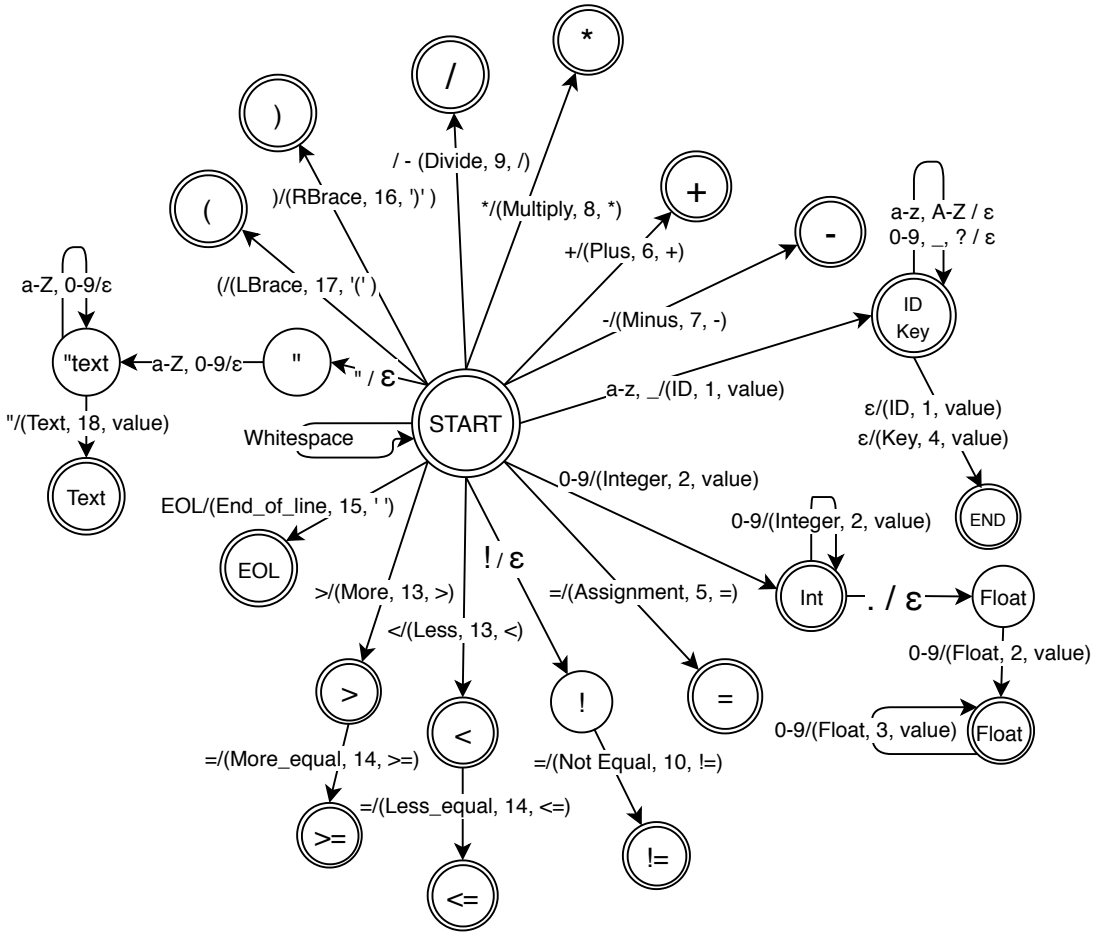
Lexikálny analyzátor je schopný rozlíšiť tieto elementy zdrojového jazyka: **identifikátory, kľúčové slová, celé a desatinné čísla, textové literály, operátory a zátvorky**. Identifikátor je v jazyku IFJ18 definovaný ako neprázdna postupnosť písmen, číslic a znaku podčiarkovníka ('_') začínajúci malým písmenom alebo podčiarkovníkom. Jazyk IFJ18 obsahuje kľúčové slová, ktoré sa nesmú vyskytovať ako identifikátory. Sú to: *def, do, else, end, if, not, nil, then, while*. Medzi slová ktoré sa nesmú vyskytnúť ako identifikátory patria taktiež názvy vstavaných funkcií. Celé, nezáporné číslo v desiatkovej sústave je tvorené postupnosťou číslic. Desatinné číslo taktiež vyjadruje nezáporné číslo v desiatkovej sústave a je zložené z celej a desatinnej časti, pričom tieto časti musia byť od seba oddelené bodkou. Obidve tieto časti sú tvorené neprázdnou postupnosťou číslic. Textový literál je obojstranne ohraničený úvodzovkami (") a tvorí ju ľubovoľný počet znakov, pričom je povolený aj prázdny reťazec.

Grafickú reprezentáciu konečného prevodníka potrebného k lexikálnej analýze môžeme vidieť na obrázku 7.1.

Z tohto obrázka vidíme, že začíname v počiatočnom stave *start* a po prečítaní prvého vstupného symbolu je možné deterministicky sa rozhodnúť do ktorého nasledujúceho stavu pôjdeme. V prípade, že nasledujúci stav ešte nie je koncovým, je nutné prečítať ďalší, resp. ďalšie vstupné symboly. Ak skončíme v niektorom z koncových stavov, vraciame príslušný token v tvare, ktorý sme definovali vyššie. Z jazyka IFJ18 sme vynechali niektoré jeho vlastnosti, ako napr. číslo v exponenciálnom tvare, pretože cieľom aplikácie nie je vykonanie dôkladnej lexikálnej analýzy.

7.1.2 Prevodník M_2 – syntaktický analyzátor

Prevodník M_2 patrí medzi najdôležitejšie súčasti systému. Jeho úlohou je skontrolovať syntaktickú správnosť daného aritmetického výrazu. Tento prevodník postupne obdrží na vstupe prúd tokenov, ktoré získa od lexikálneho analyzátoru, a v súlade s definíciou syntaxe jazyka vytvára derivačný strom. Ak sa mu tento derivačný strom podarí vytvoriť, zdrojový kód je syntakticky správny. Prevodník M_2 je možné podobne ako M_1 vyjadriť graficky, ale keďže syntaktický analyzátor pracuje na úrovni bezkontextových jazykov, tak vyjadrenie M_2 pomocou bezkontextovej gramatiky nám poskytne efektívnejší prvotný pohľad na syntax výrazov v našom jazyku.
Nech $G = (\{E, T, Op\}, \{i, +, -, *, /, (,)\}, P, E)$ je bezkontextová gramatika s nasledujúcimi



Obr. 7.1: Grafická reprezentácia konečného prevodníku špecifikujúci lexikálny analyzátor

pravidlami:

$$\begin{aligned} P = \{ & 1 : E \to (E \\ & 2 : E \to iT \\ & 3 : E \to OpE \\ & 4 : T \to)T \\ & 5 : T \to \varepsilon \\ & 6 : OP \to + \\ & 7 : OP \to - \\ & 8 : OP \to * \\ & 9 : OP \to / \} \end{aligned}$$

Gramatika G obsahuje tri neterminálne symboly – E, T a Op, pričom E je počiatočným symbolom tejto gramatiky. Ďalej obsahuje sedem terminálnych symbolov – $+, -, *, /, i, (,)$, kde i značí ľubovoľný povolený operand. Pravidlá P určujú, čo musí byť splnené aby mohol byť daný výraz akceptovaný prevodníkom M_2.

Ak sa na tieto pravidlá pozrieme bližšie, zistíme, že výraz musí začínať operandom alebo ľavou zátvorkou. Za touto ľavou zátvorkou musí nasledovať operand alebo ďalšia

ľavá zátvorka, keďže je povolené zanorovanie zátvoriek. Za operandom musí nasledovať operátor z množiny terminálnych symbolov alebo pravá zátvorka. Pravá zátvorka môže udávať, že sa jedná o koniec výrazu, ale v prípade zanorených zátvoriek môže za touto zátvorkou nasledovať operátor alebo ďalšia pravá zátvorka. Za operátorom musí nasledovať operand alebo ľavá zátvorka. Ak nastane akýkoľvek iný prípad, nastane chyba, prevodník M_2 aritmetický výraz neakceptuje a systém $\mathcal{T}$ neprevedie konverziu.

Táto bezkontextová gramatika nám poskytla jednoduchý pohľad na aritmetické výrazy v zdrojovom jazyku IFJ18. Keďže však pracujeme so systémom zásobníkových prevodníkov, zostrojíme si zásobníkový prevodník pre gramatiku G.
Nech $M_2 = (\{q\}, \{i, +, -, *, /, (,)\}, \{E, T, Op, i, +, -, *, /, (,)\}, \{1, \ldots, 9), \delta, q, E, \varnothing)$ je zásobníkový prevodník, pričom platí:

$$\begin{aligned}
\delta(q, \varepsilon, E) &= \{(q, (E, 1), (q, iT, 2)\} \\
\delta(q, \varepsilon, T) &= \{(q, Op\ E, 3), (q,)T, 4), (q, \varepsilon, 5)\} \\
\delta(q, \varepsilon, Op) &= \{(q, +, 6), (q, -, 7), (q, *, 8), (q, /, 9)\} \\
\delta(q, b, b) &= \{(q, \varepsilon, \varepsilon)\}\ \textit{pre každé}\ b \in \{i, +, -, *, /, (,)\}
\end{aligned}$$

Takto zostrojený zásobníkový prevodník sa nazýva ľavý analyzátor [10].

Príklad 7.1 *Činnosť ľavého analyzátoru si ukážeme na jednoduchom príklade so vstupným reťazcom v tvare* $i + i * i$*:*

$$\begin{aligned}
(q, i + i * i, E, \varepsilon) &\vdash (q, i + i * i, iT, 2) \\
&\vdash (q, +i * i, T, 2) \\
&\vdash (q, +i * i, Op\ E, 23) \\
&\vdash (q, +i * i, +E, 236) \\
&\vdash (q, i * i, +E, 236) \\
&\vdash (q, i * i, iT, 2362) \\
&\vdash (q, *i, T, 2362) \\
&\vdash (q, *i, Op\ E, 23623) \\
&\vdash (q, *i, *E, 236238) \\
&\vdash (q, i, E, 236238) \\
&\vdash (q, i, iT, 2362382) \\
&\vdash (q, \varepsilon, T, 2362382) \\
&\vdash (q, \varepsilon, \varepsilon, 23623825)
\end{aligned}$$

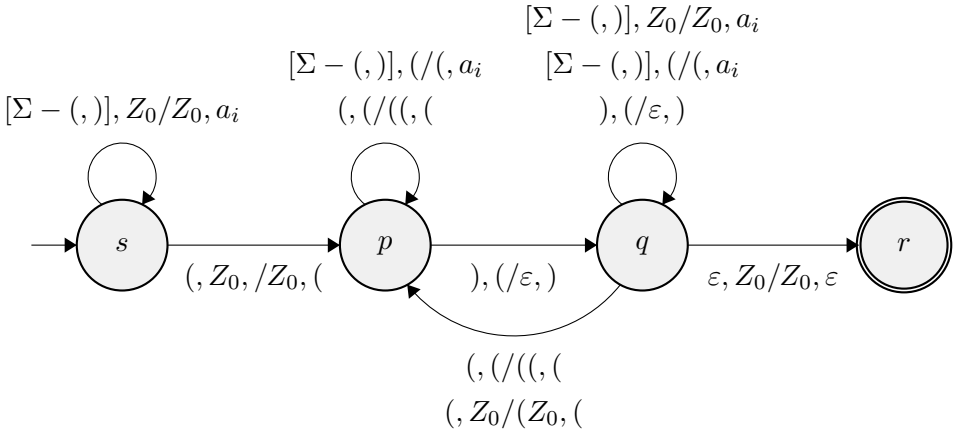
Z výsledku vidíme, že výstupom prevodníka je ľavý rozbor vstupného reťazca. Avšak, v aplikácii nebude syntaktická analýza výrazu posledným krokom, tým pádom bude programová implementácia odlišná od tohto príkladu. Rozdiel bude spočívať v tom, že výstupom prevodníka nebude ľavý rozbor, ale bude ním rovnaký výraz aký bol na jeho vstupe, avšak už bude skontrolovaný. Bližšie bude táto implementácia opísaná v sekcii 7.2.2.

Prevodník M_2 dokáže overiť či je pozícia zátvoriek vo výraze vyhovujúca vzhľadom k okolitým symbolom, avšak nedokáže overiť či sú tieto zátvorky správne vyvážené, t.j či je počet ľavých zátvoriek rovný počtu pravých. Kvôli tejto skutočnosti je nutné zaviesť prevodník M_3, ktorý bude slúžiť na overenie vyváženosti zátvoriek.

7.1.3 Prevodník M_3 – kontrola zátvoriek

Na vyriešenie problému vyváženosti zátvoriek je dôležité si uvedomiť jednu vec. Pri spracovávaní vstupného reťazca sa musí posledná ľavá zátvorka zhodovať s prvou pravou zátvorkou vo výraze. Tiež bude potrebné, aby prvá spracovaná ľavá zátvorka počkala na poslednú pravú zátvorku. Tieto podmienky sú vodítkom, že na riešenie je nutné použiť zásobník. Formálne je teda možné tento prevodník definovať nasledovne:

Nech $M_3 = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$ je zásobníkový prevodník, ktorý je zobrazený na obrázku 7.2. Symbol $[\Sigma - (,)]$[1] označuje všetky vstupné symboly okrem ľavej a pravej zátvorky a a_i predstavuje aktuálne prečítaný vstupný symbol.



Obr. 7.2: Prevodník M_3 v systéme $\mathcal{T}$

Jedná sa o prevodník, ktorý sa aktivuje iba v prípade že sa vo výraze objavujú zátvorky. Platí teda, že sa nemusí vôbec zapojiť do celkového prekladu systému. V prípade aktivácie, je algoritmus jeho činnosti nasledovný: ak je aktuálne na vstupe jeden zo symbolov $+, -, *, /$ alebo ľubovoľný operand, iba sa umiestni na výstup. Ak je vstupným symbolom ľavá zátvorka, uloží ho na zásobník ako signál, že sa neskôr musí objaviť odpovedajúca pravá zátvorka. V opačnom prípade, ak je na vstupe pravá zátvorka, odstráni sa jedna ľavá zátvorka z vrcholu zásobníka. Obidve tieto zátvorky sa taktiež umiestnia na výstup. Ak nastane situácia, že sa prevodník pokúsi odstrániť symbol z prázdneho zásobníka, je to znamenie, že zátvorky nie sú vyvážené – končí sa chybou. Aby boli zátvorky správne vyvážené, na konci reťazca, keď už sú všetky vstupné symboly spracované, musí byť zásobník prázdny.

Opäť platí, že ak je všetko v poriadku, výstup prevodníka je rovnaký ako jeho vstup. Po ukončení kontroly predáva M_3 riadenie prevodníku M_4, ktorý vykoná prevod výrazu z infixovej do postfixovej notácie.

7.1.4 Prevodník M_4 – prevod výrazu

Prevodník M_4 je najdôležitejšou komponentou systému $\mathcal{T}$. Ako bolo vyššie spomenuté, jeho úlohou je vykonať prevod výrazu z infixovej do postfixovej notácie. Celý algoritmus prevodu je založený na použití zásobníka. Pri tomto prevode platí, že operandy zostávajú vo svojich relatívnych polohách a iba operátory menia svoje pozície (posúvajú sa smerom dozadu vo výraze). Veľmi dôležitú úlohu zohráva priorita operátorov. Tabuľka 7.1 udáva prioritu operátorov v našom zdrojovom jazyku (priorita s číslom 4 je najvyššia).

[1] Hranaté zátvorky sú použité iba pre väčšiu prehľadnosť

Priorita	Operátory
1	= !=
2	< > <= >=
3	+ -
4	* /

Tabuľka 7.1: Priorita operátorov v zdrojovom jazyku

Pri spracovávaní výrazu sa musia operátory uložiť na zásobník, pretože ich príslušný pravý operand sa spracuje až v nasledujúcich krokoch. V niektorých prípadoch môže byť potrebné zmeniť poradie operátorov z dôvodu ich priority. Platí, že operátory s nižšou prioritou sa musia na výstupe objaviť až po operátoroch s vyššou prioritou. Symbol na vrchole zásobníka udáva naposledy spracovaný operátor. Ak po prečítaní operátoru už na zásobníku nejaké sú, najskôr sa odstránia tie, ktoré majú vyššiu alebo rovnakú prioritu ako práve prečítaný operátor a až potom sa aktuálny operátor umiestni na zásobník.

Ak je aktuálnym vstupným symbolom ľavá zátvorka, umiestni sa na zásobník a taktiež aj operátory nachádzajúce sa za touto zátvorkou. Tieto operátory budú musieť na zásobníku čakať, kým sa neobjaví odpovedajúca pravá zátvorka. Ak sa objaví, odstránia sa všetky operátory až po odpovedajúcu ľavú zátvorku na zásobníku. Odstránené operátory sa pripoja na koniec výstupnej pásky. Keďže prevodník M_4 je poslednou komponentou v systéme $\mathcal{T}$, tak jeho výstupná páska je rovnaká ako výstupná páska systému $\mathcal{T}$.

Príklad 7.2 *Príklad prevodu si ukážeme na vstupnom reťazci v tvare $A * (B + C)$. Príklad budeme zapisovať do tabuľky s tromi stĺpcami. Prvý zobrazuje aktuálny symbol, druhý zobrazuje obsah zásobníka a tretí doposiaľ vyprodukovaný výstupný postfixový výraz. Na zásobník sa zapisuje zľava doprava, spodná časť zásobníka sa nachádza vľavo.*

#	Aktuálny symbol	Zásobník	Výstupný reťazec
1	A		A
2	*	*	A
3	(	* (	A B
4	B	* (	A B
5	+	* (+	A B
6	C	* (+	A B C
7	)	*	A B C +
8			A B C + *

Tabuľka 7.2: Tabuľka zobrazujúca príklad prevodu reťazca $A * (B + C)$

Pretože podvýraz v zátvorkách musí byť prevedený ako prvý, tak po uložení symbolu násobenia ($*$) sa na zásobník uloží aj ľavá zátvorka a poskytne akúsi značku. Ak sa načíta ďalší operátor vo výraze, bude sa so zásobníkom pracovať ako keby bol prázdny a nový operátor ($+$) sa uloží na jeho vrchol. Následne, ak je prečítaná pravá zátvorka, odstraňujú sa symboly zo zásobníka, až kým nie je nájdená odpovedajúca ľavá zátvorka. Pretože výrazy v postfixovej notácie neobsahujú zátvorky, tak sa zátvorky nevypisujú. Ak je načítaným symbolom operand, ihneď sa vypisuje na výstup.

7.2 Programová implementácia

V tejto sekcii sa bližšie pozrieme na programovú implementáciu navrhnutého systému prevodníkov zo sekcie 7.1. Výsledná aplikácia bola implementovaná v programovacom jazyku *Python 3.8* a na tvorbu užívateľského rozhrania bola využitá voľne dostupná knižnica *Tkinter*. Jednotlivé podsekcie obsahujú popis tried a verejných metód, ktoré tieto triedy obsahujú.

7.2.1 Trieda LexicalAnalyzer

Aby bol systém prevodníkov $\mathcal{T}$ schopný identifikovať výraz a následne vykonať prevod identifikovaného výrazu z infixovej do postfixovej notácie, potrebuje identifikovať vo vstupnom reťazci jednotlivé symboly. Na identifikovanie týchto symbolov mu pomáha trieda `LexicalAnalyzer`. Pri jej implementácii sme si ako prvé vytvorili jednoduchý typ výčtu `Enum`, ktorý umožňuje vytvárať výčty, tak ako napr. v jazyku C++. Následne bola implementovaná trieda `Token`, ktorá uchováva informácie o identifikovanom symbole.

Trieda `LexicalAnalyzer` obsahuje štyri metódy, z ktorých najdôležitejšie sú `scan()` a `getChar()`. Metóda `getChar()` vráti jeden znak zo zadaného vstupného reťazca. Vstup aplikácie sa predáva ako argument konštruktoru tejto triedy. Úlohou metódy `scan()` je simulovať prechody podľa obrázka 7.1 a následne vrátiť inštanciu triedy `Token`, ktorá obsahuje tri atribúty: type (tym symbolu), alias (číselný identifikátor typu) a value (hodnota symbolu). Metóda `skipWhiteSpaces()` slúži na preskočenie bielych znakov vo vstupnom reťazci a metóda `findMatch` má pomocný charakter, pretože umožňuje vyhľadať podreťazec na základe zadaného regulárneho výrazu.

7.2.2 Trieda SyntaxAnalyzer

Ako už bolo vyššie spomenuté, jednou z úloh syntaktického analyzátoru je overiť či získaná kombinácia tokenov tvorí aritmetický výraz. Ak áno, jeho ďalšou úlohou je skontrolovať príslušnosť výrazu v jazyku danej bezkontextovou gramatikou z 7.1.2, tzn. či je výraz syntakticky správne zapísaný.

Syntaktický analyzátor je v aplikácii reprezentovaný triedou `SyntaxAnalyzer`. Zavolaním metódy `getToken()` získa jeden token od lexikálneho analyzátoru. Na overenie či kombinácia za sebou idúcich tokenov vytvára aritmetický výraz, slúži metóda `is_Expression()`. Kontrola syntaxe výrazu prebieha nasledovným spôsobom: prijatý token je uložený na zásobník a následne prijatím nasledujúceho tokenu sa skontroluje či vyhovuje kombinácia typu aktuálne získaného tokenu s typom tokenu na zásobníku. V prípade že vyhovuje, token zo zásobníku sa umiestni na výstup a aktuálne prijatý token sa uloží na zásobník.

Metóda `bracket_occurence` vracia `True`, ak získaný výraz obsahuje zátvorky a tým pádom musí nastať kontrola vyváženosti zátvoriek, ktorú vykonáva trieda `BracketController`.

7.2.3 Trieda BracketController

Jedná sa o veľmi jednoduchú triedu s jedinou metódou `areBracketsBalanced()`. Táto metóda vracia booleovský výsledok a jeho úlohou je skontrolovať, či sú zátvorky vo vstupnom výraze vyvážené. Ak je aktuálnym symbolom ľavá zátvorka, umiestni sa na vrchol zásobníku. Následne, ak je aktuálnym symbolom pravá zátvorka, pomocou vstavanej funkcie `pop()` sa odstráni symbol z vrcholu zásobníka. Táto odstránená hodnota sa ďalej nepou-

žíva, nakoľko vieme, že sa jedná o ľavú zátvorku spracovanú v predchádzajúcich krokoch. Na konci, ak je zásobník vyprázdnený, predstavuje reťazec výraz s vyváženými zátvorkami.

7.2.4 Trieda ExpressionConverter

Hlavnou komponentou systému a aj aplikácie je trieda `ExpressionConverter`, ktorá sa stará o prevod výrazu z infixovej do postfixovej notácie. Táto trieda obsahuje okrem hlavnej metódy `infixToPostfix()` aj niekoľko „pomocných“, ktoré jej pomáhajú pri prevode výrazu. Niekoľko z týchto metód zaisťujú prácu so zásobníkom. Metóda `isEmpty()` vráti `True`, ak je zásobník prázdny, `push()` a `pop()` umožňujú uložiť, resp. odstrániť symbol zo zásobníka a `top()` vráti symbol z vrcholu zásobníka. K „pomocným“ metódam patrí aj `isOperand()`, ktorá zisťuje či aktuálny vstupný symbol nie je operand.

Aby bolo možné porovnať úrovne priorít jednotlivých operátorov, bol využitý slovník, ktorý namapuje každý operátor na celé číslo a následne je toto porovnanie vykonané metódou `notGreater()`, ktorá vracia `True`, ak je priorita aktuálneho vstupného symbolu menšia alebo rovná priorite operátoru na vrchole zásobníka.

Metóda `infixToPostfix()` vykonáva samotný prevod aritmetického výrazu. Pseudokód tejto metódy je nasledovný:

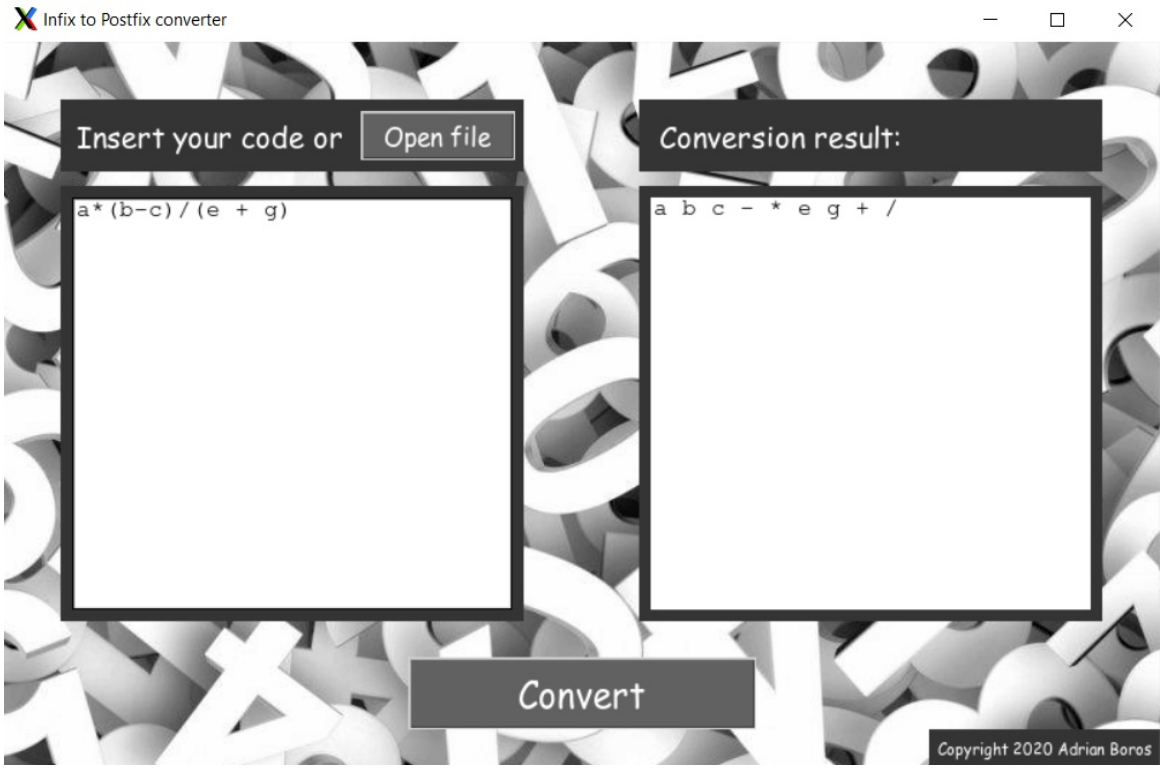
Algorithm 1 Pseudokód metódy `infixToPostfix()`

```
method INFIXTOPOSTFIX()
    for každý symbol ch vo vstupnom reťazci do
        if ch je operand then
            pridaj ch na koniec výstupu
        else if ch = ľavá zátvorka '(' then
            umiestni '(' na zásobník
        else if ch = pravá zátvorka ')' then
            while zásobník nie je prázdny a vrchol zásobníka ≠ '(' do
                odstráň operátor z vrcholu zásobníka a pridaj ho na koniec výstupu
            odstráň '(' zo zásobníka
        else
            while zásobník nie je prázdny a priorita operátoru ch ≤ priorite operátoru
na vrchole zásobníka do
                odstráň operátor z vrcholu a pridaj ho na koniec výstupu
            pridaj nový operátor na zásobník
    while zásobník obsahuje nejaké zostávajúce symboly do
        odstráň operátor z vrcholu a pridaj ho na koniec výstupu
```

7.2.5 Grafické užívateľské rozhranie

Poslednou časťou navrhnutej aplikácie je grafické užívateľské rozhranie, ktoré je viditeľné na obrázku 7.3. Celé toto rozhranie pozostáva z jediného okna. Na ľavej strane okna sa nachádza editovateľné textové pole do ktorého užívateľ zadáva zdrojový kód v jazyku IFJ18. Pomocou tlačítka `Open File` má užívateľ možnosť výberu textového súboru s jeho kódom. Po úspešnom výbere sa obsah vybraného súboru zobrazí v textovom poli. Po stlačení tlačítka `Convert` sa vykonajú prevody nájdených výrazov. Výsledok prevodu je viditeľný na výstupnej časti na pravej strane okna.

Ak počas prevodu nastane chyba, užívateľ je na túto chybu upozornený chybovou hláškou a výraz, ktorý obsahuje chybu nebude prekonvertovaný. Užívateľ je upozornený aj v prípade, že jeho vstupný kód neobsahuje aritmetické výrazy. Celá aplikácia je navrhnutá tak, aby práca s ňou bola jednoduchá a intuitívna.



Obr. 7.3: Grafické užívateľské rozhranie

Kapitola 8

Záver

Preklad je jedným z najdôležitejších aspektov pri práci s počítačovými systémami. Kvôli tejto skutočnosti bola táto bakalárska práca zameraná na štúdium prekladových modelov. Okrem konečných a zásobníkových prevodníkov, boli preskúmané aj ďalšie formálne modely, ktoré sa stali základom nového modelu, systému prevodníkov. Práve tieto systémy boli hlavnou oblasťou záujmu tejto práce. U každého modelu boli uvedené základné definície, slúžiace pre lepšie pochopenie danej problematiky a na praktických príkladoch bola predvedená činnosť týchto modelov.

V prvej časti sme sa zamerali na konečné prevodníky, ktoré vznikli z konečných automatov pridaním výstupnej pásky. Práve v tejto výstupnej páske spočíva rozdiel medzi týmito dvomi modelmi. Pomocou grafického znázornenia bola ukázaná činnosť čítacej a zapisovacej hlavy. Preskúmané boli situácie, ktoré môžu nastať pri prechode prevodníka z jednej konfigurácie do druhej a na jednoduchom príklade bol demonštrovaný preklad daný konečným prevodníkom.

Následne sme sa presunuli na model veľmi blízky konečným prevodníkom, a tým sú zásobníkové prevodníky. Opísali sme, akými rozšíreniami sa zo zásobníkových automatov, príp. konečných prevodníkov, získajú zásobníkové prevodníky. Použitie zásobníka zohráva dôležitú úlohu pri preklade vstupného reťazca na výstupný. Na grafickej reprezentácii boli porovnané označenia prechodov konečného a zásobníkového prevodníka. Keďže sme zistili, že môže nastať situácia, v ktorej je prevodník schopný z jednej konfigurácie vykonať niekoľko prechodov do iných, zadefinovali sme deterministickú verziu zásobníkového prevodníka, ktorá túto nejednoznačnosť odstraňuje.

Druhá časť práce sa zaoberá prekladovými gramatikami a gramatickými systémami. Prekladové gramatiky sú veľmi podobné bezkontextovým gramatikám, rozdiel je však v tom, že prekladové gramatiky generujú dvojicu reťazcov. Po samotnej definícii prekladovej gramatiky sme zaviedli schému syntaktického prekladu, ktorá s ňou úzko súvisí. Na príklade bola predvedená činnosť tejto gramatiky, ktorá prekladala aritmetický výraz z infixovej do prefixovej notácie. Nakoniec sme ukázali postup, ako z prekladovej gramatiky získame vstupnú a výstupnú bezkontextovú gramatiku.

Ďalšími preskúmanými prekladovými modelmi boli gramatické systémy. Prvým z týchto typov boli kooperujúce distribuované (CD) gramatické systémy. Tento typ je význačný tým, že niekoľko spolupracujúcich gramatík vytvára jednu spoločnú vetnú formu. Jednotlivé gramatiky sú spracovávané jedna za druhou a v jednom momente je aktívna iba jedna z nich. Predstavili a opísali sme jednotlivé derivačné režimy. Na príklade sme predviedli, že jazyk generovaný týmto systémom v jednotlivých režimoch nemusí byť rovnaký.

Paralelne komunikujúce (PC) gramatické systémy využívajú toho, že jednotlivé gramatiky pracujú paralelne a vytvárajú tak svoje vlastné vetné formy. Preklad končí v okamihu, keď gramatika označovaná ako master vyprodukuje reťazec terminálov. V týchto systémoch sa objavujú dva typy derivačných krokov. Ako prvý sme opísali prepisovací krok, pri ktorom každá gramatika iba použije jedno zo svojich pravidiel prepisovania. Druhým opísaným typom bol komunikačný krok, ktorý slúži na výmenu doposiaľ vyprodukovaných reťazcov medzi komponentami. V závere kapitoly sme si vysvetlili rozdiel medzi centralizovanými a necentralizovanými PC gramatickými systémami.

Tretia časť obsahuje jadro celej práce. Presnejšie povedané, definuje a skúma nový prekladový model – systém prevodníkov. Tieto systémy sa skladajú z niekoľkých zásobníkových prevodníkov a ich činnosť je veľmi podobná činnosti CD gramatických systémov. Po samotnej definícii systému sme najprv ukázali označenie jeho aktuálnej konfigurácie a potom sme predstavili aj možnosť popisu okamžitého stavu systému. Tento popis je vhodnejší pre systémy v ktorých pracujú prevodníky paralelne. Na príkladoch sme následne dokázali, že tieto systémy sú schopné spracovať a preložiť aj niektoré kontextové jazyky, čím je ich sila väčšia ako sila samotných zásobníkových prevodníkov. Tým, že sme schopní uložiť do sekvencie niekoľko za sebou idúcich (aj rovnakých) komponent, sme schopní namodelovať základné programové štruktúry, ako napr. cyklus.

Uvažovali sme systémy prevodníkov, v ktorých komponenty úspešne dokončia svoj preklad ak sa nachádzajú v jednom z ich koncových stavov. Možnou modifikáciou by mohla byť možnosť, že za úspešný preklad by sa považovalo aj vyprázdnenie zásobníka. V rovnakom duchu by mohlo byť zaujímavé zvážiť aj systémy prevodníkov zložené výlučne z deterministických zásobníkových prevodníkov.

V poslednej časti sme navrhli systém, ktorý sa stal základom implementačnej časti práce. Výsledkom tejto časti je aplikácia na vyhľadanie a prevod aritmetických výrazov z infixovej do postfixovej notácie. Motiváciou pri tvorbe tejto aplikácie je skutočnosť, že aritmetické výrazy v postfixovej notácii nezávisia na zátvorkách pri vyhodnocovaní výrazov. Aplikácia poskytuje jednoduché grafické užívateľské rozhranie pre zadanie zdrojového kódu alebo výber textového súboru s týmto kódom. Toto rozhranie obsahuje aj pole na výpis výsledkov v prípade úspešného prevodu.

Ako prvé sme navrhli a nadefinovali jednotlivé zásobníkové prevodníky, z ktorých je výsledný systém zložený. V prípade, že sa jednalo o zložitejšiu komponentu, bola jej činnosť predvedená na vzorovom príklade alebo bola definovaná pomocou bezkontextovej gramatiky. Neskôr boli popísané hlavné triedy a v nich vytvorené metódy, ale aj prvky grafického užívateľského rozhrania, ich celkové rozloženie a význam.

Práca sa zaoberá iba širokou oblasťou systémov prevodníkov, preto je k dispozícii priestor pre ich dôkladnejšie preskúmanie. Budúcim možným pokračovaním práce je definovanie systémov prevodníkov, ktoré by však namiesto zásobníkových prevodníkov boli zložené z konečných prevodníkov. Taktiež by sa mohli zaviesť paralelne komunikujúce prevodníkové systémy, ktoré by pracovali na podobnom princípe ako PC gramatické systémy. Následne by sa mohla skúmať ich vyjadrovacia sila, ale aj ich výpočetná náročnosť. Do novo nadefinovaného systému prevodníkov by sa mohlo zaviesť obmedzenie na počet krokov, ktoré môžu jednotlivé komponenty vykonať, podobne ako tomu je v CD gramatických systémoch. Niektoré tieto problémy by sa dali vyriešiť jednoduchou úpravou existujúcich definíc, avšak na vyriešenie zložitejších problémov by bolo nutné dôkladnejšie preskúmanie problematickej oblasti.

Literatúra

[1] Aho, A. V. a Ullman, J. D. *The Theory of Parsing, Translation, and Compiling*. 1. vyd. London: Prentice-Hall, Inc., 1972. ISBN 013-914556-7.

[2] Cojocaru, L. a Martin Vide, C. Parallel Communicating Pushdown Transducer System. In: Csuhaj Varjú, E. a Vaszil, G., ed. *Proceedings of Grammar Systems Week 2004*. Budapest: MTA SZTAKI, 2004, s. 126–140.

[3] Crespi Reghizzi, S., Breveglieri, L. a Morzenti, A. *Formal Languages and Compilation*. 2. vyd. London: Springer London, 2013. Texts in Computer Science. ISBN 978-1-4471-5513-3.

[4] Csuhaj Varjú, E. a Vaszil, G. Parallel communicating grammar systems with bounded resources. *Theoretical Computer Science*. Elsevier B.V. 2002, zv. 276, 1-2, s. 205–219. ISSN 0304-3975. Dostupné z: https://www-sciencedirect-com.ezproxy.lib.vutbr.cz/science/article/pii/S0304397501001724.

[5] Dassow, J., Păun, G. a Rozenberg, G. Grammar Systems. In: Rozenberg, G. a Salomaa, A., ed. *Handbook of Formal Languages*. 2. vyd. Berlin, Heidelberg: Springer, 1997, s. 155–213. ISBN 978-3-662-07675-0. Dostupné z: https://doi.org/10.1007/978-3-662-07675-0_4.

[6] Ginsburg, S. Examples of Abstract Machines. *IRE Transactions on Electronic Computers*. 1962, EC-11, č. 2, s. 132–135.

[7] Gurari, E. Pushdown Transducers. *An Introduction to the Theory of Computation* [online]. 1989. Aktualizované 28. 07. 2009 [cit. 6. marca 2020]. Dostupné z: http://dns.uls.cl/~ej/talf_2004/theory-bk/theory-bk-threese2.html#Q1-40002-3.

[8] Karttunen, L. *Finite-State Lexicon Compiler*. Technical Report ISTL-NLTT-1993-04-02. Palo Alto, California: Xerox Palo Alto Research Center, apríl 1993.

[9] Křivka, Z., Regéciová, D., Zobal, L. a Kocman, R. *ZADÁNÍ PROJEKTU Z PREDMĚTU IFJ A IAL* [online]. 2018 [cit. 14. apríla 2020]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIFJ-IT%2Fprojects%2FHistory%2Fifj2018_release_candidate2.pdf&cid=12745.

[10] Meduna, A. a Lukáš, R. *Formální jazyky a překladače: Studijní opora* [online]. Január 2006 [cit. 16. apríla 2020]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIFJ-IT%2Ftexts%2FOporaIFJ.pdf&cid=12745.

[11] Meduna, A. *Automata and languages : theory and applications*. London: Springer, 2000. ISBN 1-85233-074-0.

[12] Meersman, R. a Rozenberg, G. Cooperating grammar systems. In: Winkowski, J., ed. *Mathematical Foundations of Computer Science 1978*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, s. 364–373. ISBN 978-3-540-35757-5.

[13] Melichar, B. Transformations of translation grammars. *Kybernetika*. Institute of Information Theory and Automation AS CR. 1994, zv. 30, č. 1, s. 53–62. Dostupné z: https://www.kybernetika.cz/content/1994/1/53/paper.pdf.

[14] Mohri, M. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*. 1997, zv. 23, č. 2, s. 269–311. Dostupné z: https://www.aclweb.org/anthology/J97-2003.

[15] Nii, H. P. *Blackboard Systems*. Technical Report. Stanford, CA, USA: Stanford University, jún 1986. Dostupné z: https://apps.dtic.mil/dtic/tr/fulltext/u2/a174225.pdf.

[16] Păun, G. a Santean, L. Parallel communicating grammar systems: the regular case. *Analele Universitatii din Timisoara, Seria Matematica - Informatica*. 38. vyd. 1989, č. 2, s. 55–63. Dostupné z: https://cs.uwaterloo.ca/~lila/pdfs/PCGSFirst.pdf.

[17] Rosner, M. *Lecture 5: Finite State Transducers* [online]. Október 2011 [cit. 29. februára 2020]. Dostupné z: http://staff.um.edu.mt/mros1/csa3202/pdf/fst_3.pdf.

[18] Techet, J., Masopust, T. a Meduna, A. *Parallel Communicating Grammar Systems* [online]. 2007 [cit. 21. marca 2020]. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:10-pcgspres.pdf.

[19] Wikipedia contributors. *Finite-state transducer — Wikipedia, The Free Encyclopedia* [online]. 2019. Aktualizované 25. 02. 2020 [cit. 29. februára 2020]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Finite-state_transducer&oldid=931226956.

Príloha A

Obsah priloženého CD

Štruktúra priloženého CD

- **src** – adresár obsahujúci zdrojové súbory
  - *xboros03.py* – zdrojový kód výslednej aplikácie
  - *img1.gif* – pozadie aplikácie
  - *README.md* – manuál a popis aplikácie
- **docs** – adresár obsahujúci písomnú správu vo formáte PDF a jej zdrojové kódy