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 **FILIP LUPTÁK**
AUTHOR

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

BRNO 2017

Zadání bakalářské práce/20193/2016/xlupta03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Lupták Filip**
Obor: Informační technologie
Téma: **Systémy převodníků a jejich aplikace**
Transducer Systems and Their Applications
Kategorie: Teoretická informatika

Pokyny:

1. Seznamte se detailně s převodníky.
2. Zaveďte systémy převodníků dle instrukcí vedoucího.
3. Studujte vlastnosti převodníků zavedených v bodě 2.
4. Aplikujte převodníky navržené v bodě 2 v oblasti syntaxí řízeného překladu. Tuto aplikaci proveďte tak, aby zahrnovala překlad struktur, které nejsou bezkontextové.
5. Implementujte aplikaci navrženou v předchozím bodě a testujte ji.
6. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Rozenberg, G. and Salomaa, A. (eds.): Handbook of Formal Languages, Volume 1 through 3, Springer, 1997, ISBN 3-540-60649-1

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Meduna Alexander, prof. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Táto práca zavádza nový formálny model „systém prevodníkov.“ Systém prevodníkov sa skladá z niekoľkých konečných prevodníkov, ktoré spolupracujú. Práca skúma vyjadrovaciu silu tohto formálneho modelu a ukazuje, že je silnejší ako samotný konečný prevodník. Pomocou systému prevodníkov modeluje parser aritmetických výrazov, ktorý výrazy prekladá do postfixovej notácie.

Abstract

This thesis defines a new formal model „transducer system.“ Transducer system is composed of a number of finite transducers, which cooperate. The thesis examines expressive power of this formal model and shows that transducer system is more powerful than isolated finite transducer. This model is then used to design a parser of arithmetic expressions, which translates them to postfix notation.

Kľúčové slová

Konečný prevodník, Systém prevodníkov, Syntaxou riadený preklad, Aritmetické výrazy, Postfixová notácia, Formálne jazyky

Keywords

Finite transducer, Transducer system, Syntax directed translation, Arithmetic expressions, Postfix notation, Formal languages

Citácia

LUPTÁK, Filip. *Systémy převodníků a jejich aplikace*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexander Meduna, CSc.

Systémy převodníků a jejich aplikace

Prehlásenie

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

.....
Filip Lupták
18. mája 2017

Poďakovanie

Ďakujem pánovi profesorovi Medunovi za jeho odborné rady, inšpiráciu, trpezlivosť a pozitívny prístup.

Obsah

1 Úvod **3**

2 Konečné prevodníky **4**

3 Systém prevodníkov **8**
3.1 Príklad – systém prijímajúci bezkontextový jazyk 10
3.2 Príklad – systém prijímajúci kontextový jazyk 13
3.3 Systém prevodníkov – záver 18

4 Systém prevodníkov II. **19**
4.1 Analýza systému S_3 a porovnanie systémov I. a II. 25

5 Prevod aritmetických výrazov **28**
5.1 Konceptuálny návrh 28
5.2 Prevodník M_1 30
5.3 Prevodník M_2 34
5.4 Redukcia zátvoriek: Prevodník M_3 36
5.5 Príklad – prevod do postfixovej notácie 38

6 Implementácia a testovanie **40**

7 Záver **45**

Literatúra **46**

Zoznam obrázkov

2.1 Prevodník M_1 5

3.1 Prevodník M_2 10
3.2 Prevodník M_1 12
3.3 Schéma systému S_1 13
3.4 Prevodník M_1 14
3.5 Prevodník M_2 15
3.6 Systém S_2 15

4.1 Prevodníky M_1 (vľavo) a M_2 21
4.2 Prevodníky M_3 (vľavo) a M_4 22
4.3 Schéma systému S_3 27

5.1 Zlučenie hrán v obrázku grafu 29
5.2 Prevodník M_1 31
5.3 Prevodník M_2 34
5.4 Aktualizácia grafu prevodníku M_1 36
5.5 Prevodník M_3 37

6.1 Diagram tried 41

Kapitola 1

Úvod

Účelom tejto práce je definovať formálny model „systém prevodníkov“ a pokúsiť sa aplikovať tento koncept v oblasti prekladu zdrojového kódu, konkrétne aritmetických výrazov.

Systém prevodníkov vznikne spojením niekoľkých konečných prevodníkov. Ľubvoľný prevodník v systéme môže po preložení celého reťazca symbolov, ktorý dostane na vstup, prípadne jeho časti, odoslať výsledok svojho prekladu na vstup iného prevodníku v systéme a nechať ho reťazec znovu preložiť. Vďaka spolupráci prevodníkov môže byť výsledný systém silnejší než samotný konečný prevodník a potenciálne prijímať zložitejšie jazyky.

Kapitola Konečné prevodníky obsahuje definíciu konečného prevodníku ako teoretické východisko pre zvedenie systému prevodníkov. Kapitola Systém prevodníkov zavádza prvotnú definíciu systému prevodníkov a skúma vlastnosti systémov. Obsahuje príklad systému, ktorý prijíma bezkontextový jazyk $L = \{a^n b^n c^n : n \geq 0\}$ a príklad systému, ktorý prijíma kontextový jazyk $L = \{a^{2^n} : n \geq 0\}$.

Kapitola Systém prevodníkov II. zavádza rafinovanejšiu, optimalizovanú definíciu systému prevodníkov. Odstraňuje nedostatky prvotnej definície, ktoré sú patrné na príkladoch. Obsahuje príklad systému podľa novej definície, ktorý tiež prijíma jazyk $L = \{a^n b^n c^n : n \geq 0\}$ a porovnáva efektivitu starého a nového systému.

Kapitola Prevod aritmetických výrazov využíva koncept systému prevodníkov a modeluje pomocou neho jednoduchý parser aritmetických výrazov, ktorý výrazy zároveň prekladá do postfixovej notácie. Kapitola obsahuje jeden demonštračný príklad prevodu aritmetického výrazu. Validita navrhnutého systému je potom testovaná pomocou aplikácie, ktorá tento systém implementuje, v kapitole Implementácia a testovanie. Táto kapitola tiež obsahuje popis objektovej realizácie aplikácie a stručný popis implementácie.

Kapitola 2

Konečné prevodníky

Táto kapitola definuje konečné prevodníky – základné komponenty, z ktorých budú poskladané systémy prevodníkov.

Definícia 2.1. Konečný prevodník M je pätica [1]

$$M = (Q, \Sigma, R, s, F)$$

kde

- Q je konečná množina stavov;
- Σ je abeceda symbolov, pre ktorú platí, že $\Sigma = \Sigma_I \cup \Sigma_O$, 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 reprezentujúca množinu pravidiel;
- s je počiatočný stav a
- $F \subseteq Q$ je množina koncových stavov.

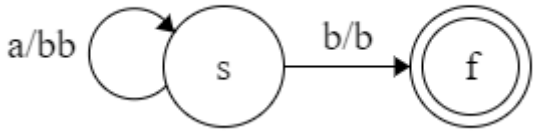
Majme pravidlo $r = (pw, qz); p, q \in Q; w \in (\Sigma_I \cup \{\varepsilon\}); z \in \Sigma_O^*$. Potom pw nazývame lhs(r) (left hand side), q nazývame rhs(r) (right hand side) a z nazveme out(r) (output). V tejto práci bude okrem zápisu pravidla r ako dvojice slov (pw, qz) používaný aj zápis $pw \to qz$. Tieto zápisy sú ekvivalentné.

Prevodník si môžeme predstaviť ako stroj s *čítacou* a *zápisovou hlavou*. Čítacia hlava číta symboly zo *vstupnej pásky* a zápisová hlava zapisuje symboly na *výstupnú pásku*. Keď prevodník začína v počiatočnom stave s, na vstupnej páske je vstupný reťazec a výstupná páska je prázdna. Čítacia hlava je nastavená na začiatku vstupnej pásky a zápisová hlava na začiatku výstupnej pásky.

Pravidlo možno chápať ako možnosť prejsť z jedného stavu prečítaním *práve* jedného vstupného symbolu do iného stavu a súčasne výpis všeobecne niekoľkých (aj nula) výstupných symbolov. Týmto sa čítacia hlava posunie o jednu a zápisová hlava o niekoľko pozícií doprava.

Príklad 2.2. Nech M_1 je konečný prevodník:

$$M_1 = (\{s, f\}, \{a, b\}, \{(sa, sbb), (sb, fb)\}, s, \{f\})$$



Obr. 2.1: Prevodník M_1

Množina stavov obsahuje stavy s a f, množina symbolov Σ symboly a a b, množina pravidiel pravidlá $r_1 = (sa, sbb)$ a $r_2 = (sb, fb)$, počiatočný stav je s a množina koncových stavov obsahuje jeden stav f.

Lavá strana pravidla r_1 [lhs(r_1)] je slovo sa, pravá strana [rhs(r_1)] je stav s a výstup [out(r_1)] je slovo bb.

Prevodník môže byť znázornený ako graf a to tak, že stavy budú uzly grafu a pravidlá budú hrany grafu označené ako <prečítaný symbol> / <výstup pravidla>. Koncové stavy sú zakrúžkované. Na obrázku 2.1 je grafické znázornenie prevodníku M_1.

Definícia 2.3. Nech $M = (Q, \Sigma, R, s, F)$ je konečný prevodník. Konfigurácia χ prevodníku M je slovo

$$\chi \in Q\Sigma_I^*|\Sigma_O^*$$

kde $| \notin \Sigma$ je špeciálny symbol.

Konfigurácia je slovo, ktoré reprezentuje okamžitý stav prevodníku. Prvý symbol slova je stav, v ktorom prevodník práve je. Za ním nasledujú doposiaľ neprečítané symboly na vstupnej páske, oddeľovač „|“ a symboly zapísané výstupnej páske.

Príkladom konfigurácie prevodníku M_1 z príkladu 2.2. je slovo $\chi_1 = saab\ |bbbb$. χ_1 naznačuje, že prevodník je v stave s, na vstupnej páske je (od pozície kam ukazuje čítacia hlava smerom doprava) reťazec aab a na výstupnej páske je vypísaný reťazec $bbbb$.

Definícia 2.4. Nech $M = (Q, \Sigma, R, s, F)$ je konečný prevodník. Nech existuje pravidlo $r \in R$. Majme dve konfigurácie χ a χ':

$$\chi = \text{lhs}(r)w|y$$
$$\chi' = \text{rhs}(r)w|y\,\text{out}(r)$$

kde $w \in \Sigma_I^*, y \in \Sigma_O^*$.
Potom M môže urobiť prechod z χ do χ' podľa pravidla r. Značíme

$$\chi \Vdash \chi' \quad [r]$$

Pokiaľ je jednoznačné, podľa ktorého pravidla prevodník prechod urobil, použijeme skrátený zápis

$$\chi \Vdash \chi'$$

Príklad 2.5. Uvažujme prevodník M_1 z príkladu 2.2. a pravidlo $r_1 = (sb, fb)$. Majme dve konfigurácie

$$\chi_1 = sb\ |bb \text{ a } \chi_2 = f\ |bbb$$

Potom môže M_1 podľa pravidla r_1 urobiť prechod z konfigurácie χ_1 do konfigurácie χ_2. Značíme

$$\chi_1 \Vdash \chi_2$$

Tento prechod spôsobí, že M_1 prejde zo stavu s do stavu f, prečíta symbol b zo vstupnej pásky z pozície na ktorej je nastavená čítacia hlava a posunie čítaciu hlavu o jednu pozíciu do prava. Ďalej na výstupnú pásku zapíše symbol b a posunie zápisovú hlavu o jednu pozíciu do prava.

Definícia 2.6. Nech $M = (Q, \Sigma, R, s, F)$ je konečný prevodník.

1. Nech χ ľubovolná konfigurácia M. Potom M urobí 0 prechodov z χ do χ

$$\chi \Vdash^0 \chi$$

2. Nech existujú konfigurácie $\chi_0, \chi_1, \ldots, \chi_n$ a nech platí

$$\chi_{i-1} \Vdash \chi_i$$

pre všetky $i = 1, \ldots, n$. Potom M urobí n prechodov z χ_0 do χ_n podľa pravidiel $r_1, \ldots, r_n$

$$\chi_0 \Vdash^n \chi_n \quad [r_1, \ldots, r_n]$$

Pokiaľ je jednoznačné, podľa ktorých pravidiel prechody urobil, môžeme použiť skrátený zápis bez uvedenia postupnosti pravidiel.

Navyše, ak existuje $n \geq 0$ také, že $\chi_0 \Vdash^n \chi_n$, potom môžeme všeobecne napísať

$$\chi_0 \Vdash^* \chi_n$$

Príklad 2.7. Uvažujme prevodník M_1 z príkladu 2.2.. Pravidlo $(sa, sbb) \in R_1$ označme r_1 a pravidlo $(sb, fb) \in R_1$ označme r_2. Na jeho vstup dajme reťazec aab:

$$saab \mid \Vdash sab \mid bb \Vdash sb \mid bbbb \Vdash f \mid bbbbb \quad [r_1, r_1, r_2]$$

M_1 urobil tri prechody, ktoré na seba navzájom naväzujú. To znamená, že môže urobiť sekvenciu prechodov dĺžky tri z konfigurácie $saab \mid$ do konfigurácie $f \mid bbbbb$:

$$saab \mid \Vdash^3 f \mid bbbbb \text{ alebo}$$
$$saab \mid \Vdash^* f \mid bbbbb$$

Definícia 2.8. Nech $M = (Q, \Sigma, R, s, F)$ je konečný prevodník. M *preloží* x na y práve vtedy, ak

$$sx| \Vdash^* f|y$$

kde $x \in \Sigma_I^*, y \in \Sigma_O^*, f \in F$. Reťazec y v tomto prípade nazývame *výstup* M.

To znamená, že M *preloží* reťazec x na reťazec y práve vtedy, ak existuje sekvencia prechodov z počiatočného stavu do niektorého z koncových stavov, pri ktorej M prečíta celý reťazec x a po prechode do koncového stavu na konci tejto sekvencie bude na výstupnej páske reťazec y.

Pokiaľ výstup nie je dôležitý, môžeme tiež povedať, že M *prijíma* x. Platí, že ak M *preloží* x na y, tak zároveň M *prijíma* x.

Preklad $T(M)$ definovaný prevodníkom M je množina

$$T(M) = \{(x, y) : x \in \Sigma_I^*, y \in \Sigma_O^* \text{ a } M \textit{ preloží } x \text{ na } y\}$$

Preklad definovaný prevodníkom M je teda množina všetkých dvojíc reťazcov (x, y), pre ktoré platí, že M preloží x na y.

Príklad 2.9. Uvažujme prevodník M_1 z príkladu 2.2.. V príklade 2.7. bola uvedená sekvecia prechodov, pri ktorej M_1 prešiel zo stavu s do stavu f, prečítal pri tom celý vstup (reťazec aab) a na výstupnú pásku zapísal reťazec $bbbbb$. Môžeme povedať, že M_1 preloží aab na $bbbbb$ a dvojica $(aab, bbbbb)$ patrí do $T(M_1)$.

Skúmaním M_1 rýchlo zistíme, že prijíma reťazce, ktoré majú v prefixe ľubovoľný počet symbolov a a sú zakončené jedným symbolom b. Každé a pri preklade prepíše na dve b a symbol b na konci reťazca prepíše na výstup bez zmeny. Z toho usúdime, že

$$T(M_1) = \{(a^n b, b^{2n+1}) : n \geq 0\}.$$

Kapitola 3

Systém prevodníkov

V táto kapitola zavádza definíciu systému prevodníkov: prezentuje jeden z možných spôsobov, ako definovať prepojenie niekoľkých prevodníkov do systému tak, aby spolupracovali. Ďalej skúma vlastnosti takto vzniknutého systému. Systém definuje klasicky, pomocou množiny prvkov, ktoré ho tvoria (prevodníkov), a zobrazenia, ktoré určí vzťahy medzi prvkami.

Definícia 3.1. Systém prevodníkov S je dvojica

$$S = (U, f_r)$$

kde

- $U = \{M_1, M_2, \ldots, M_n\}$ je množina *konečných* prevodníkov $M_i = (Q_i, \Sigma_i, R_i, s, F_i)$;
- $f_r : U \to U$ je *ireflexívne zobrazenie* určújúce vzťahy medzi prevodníkmi a
- $M_1 \in U$ je *vstupný bod systému.*

Konvencie značenia: Prevodníky v množine U budú značené $M_1, M_2, \ldots, M_n$. Množina stavov, abeceda symbolov, množina pravidiel a množina koncových stavov jednotlivých prevodníkov budú indexované rovnakým číslom ako prevodník. Množina stavov prevodníku M_1 bude označená Q_1, abeceda bude označená Σ_1 atd. Počiatočný stav bude v každom prevodníku označený rovnako, a síce s.

V nasledujúcich odstavcoch bude stručne načrtnutý princíp činnosti systému, aby bolo možné lepšie porozumieť významu ďalšej časti definície. Presný popis fungovania je v definícii 3.2..

Preklad začína vstupný bod systému. Vstupný bod systému je implicitne prevodník uvedený v množine U ako prvý. Tento prevodník prekladá reťazec podľa svojich pravidiel. Ak reťazec prijme, podľa toho, v ktorom koncovom stave skončí, systém rozhodne, čo ďalej s výsledkom prekladu urobí: buď reťazec, ktorý prekladom vznikol, opäť spracuje niektorý prevodník v systéme alebo systém ukončí preklad.

V prípade, že má preklad pokračovať bude vybraný prevodník znovu prekladať vzniknutý medzivýsledok. Ak aj tento prevodník reťazec prijme, znovu prebieha rozhodovanie. V ľubovoľnom okamihu prekladu je teda aktívny vždy jeden prevodník v systéme a predávanie riadenia prebieha po prechode aktívneho prevodníku do koncového stavu.

Aby bolo možné rozhodovať, čo sa má pri prechode prevodníku v systéme do koncového stavu stať, budeme rozlišovať tri druhy koncových stavov. Pre všetky prevodníky v systéme M_i platí

$$F_i = F_{iA} \cup F_{iN} \cup F_{iF}$$

kde

- F_{iA} (again) sú koncové stavy, v ktorých keď M_i skončí, sám spracuje svoj výstup znovu ako vstup.
- F_{iN} (next) sú koncové stavy, v ktorých keď M_i skončí, pošle svoj výstup na vstup prevodníku $M_{next} = f_r(M_i)$. Ak je množina F_{iN} neprázdna, musí v zobrazení f_r existovať dvojica $(M_i, M_x); M_x \in U$.
- F_{iF} (final) sú koncové stavy, v ktorých keď M_i skončí, činnosť systému sa zastaví a výstup považujeme za konečný.

F_{iA}, F_{iN}, F_{iF} sú po dvoch disjunktné.

Predávanie riadenia a determinizmus

Zobrazenie f_r priraďuje každému prevodníku v systéme najviac jeden prevodník. Výstup jedného prevodníku (vzor) je tým „pripojený“ na vstup *iného* (obraz). Zobrazenie je ireflexívne. Pripojenie výstupu prevodníku na svoj vlastný vstup možno dosiahnuť pridaním koncového stavu do množiny F_{iA}.

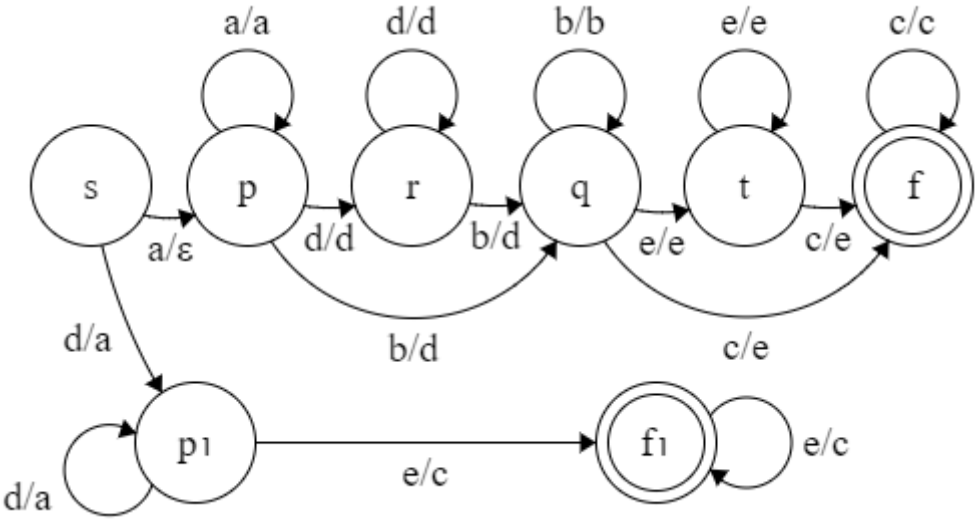
Množiny F_{iA}, F_{iN} a F_{iF} sú vzájomne disjunktné, takže každý koncový stav patrí do práve jednej z nich. Vo výsledku tak môže byť výstup každého prevodníku v systému pripojený na vstupy *najviac* dvoch prevodníkov a to tak, že po prechode do aktívneho prevodníku do ľubovoľného koncového stavu je *vždy* jednoznačné, aký krok systém ďalej urobí.

Definícia 3.2. Nech $S = \{U, f_r\}$ je systém prevodníkov. Preklad $T(S)$, definovaný systémom S, je množina

$$\begin{aligned} T(S) = \{&(x, y) : \exists\ v_0, v_1, \ldots, v_n \text{ a } \exists\ M_{j_1}, M_{j_2}, \ldots, M_{j_n} \in U \text{ také, že} \\ &sv_{i-1}| \Vdash^* \mathbf{f}|v_i \text{ v } M_{j_i};\ v_{i-1}, v_i \in \Sigma^*_{j_i}; \\ &(\mathbf{f} \in F_{j_iA} \text{ a } M_{j_{i+1}} = M_{j_i}) \textit{ alebo } (\mathbf{f} \in F_{j_iN} \text{ a } M_{j_{i+1}} = f_r(M_{j_i})) \\ &\text{pre } i = 1, \ldots, n-1;\ n \in N \text{ a} \\ &sv_{n-1}| \Vdash^* \mathbf{f}|v_n;\ \mathbf{f} \in F_{j_nF} \text{ v } M_{j_n};\ v_0 = x;\ v_n = y\} \end{aligned}$$

Definícia presne vyjadruje, ako systém prevodníkov funguje. Uvažujme, že systém má preložiť reťazec x na y (tozn. dvojica x, y patrí do množiny prekladu). Keď predložíme systému reťazec x, bude postupne prekladaný radou prevodníkov $M_{j_1}, \ldots, M_{j_n}$ z množiny U. M_{j_1} je vstupný bod systému. Zvyšok rady $M_{j_2}, \ldots, M_{j_n}$ je premenlivý a závisí na vstupnom reťazci.

Reťazce $v_0, \ldots, v_n$ v definícií označujú medzivýsledky spracovania. Reťazec $v_0 = x$ je preložený prevodníkom M_{j_1} na v_1, v_1 prevodníkom M_{j_2} na v_2 atd. Po spracovaní reťazca v_{i-1} môže prevodník M_{j_i} skončiť buď v stave $f_a \in F_{j_iA}$ a sám bude svoj výstup znovu spracovávať ako vstup, takže $M_{j_{i+1}} = M_{j_i}$, alebo v stave $f_n \in F_{j_iN}$ a ďalší prevodník v rade $M_{j_{i+1}}$ bude určený ako obraz M_{j_i} v zobrazení f_r. Posledný prevodník v rade M_{j_n} preloží v_{n-1} na y, skončí v stave $f \in F_{j_nF}$ a systém ukončí preklad.



Obr. 3.1: Prevodník M_2

3.1 Príklad – systém prijímajúci bezkontextový jazyk

Nasledujúce príklady demonštrujú princíp fungovania systémov a skúmajú ich vlastnosti. V tomto príklade sa pokúsime zostrojiť systém S_1 s prekladom

$$T(S_1) = \{(a^n b^n c^n, a^n c^n) : n \geq 0\}$$

To znamená, že systém poskladaný z konečných prevodníkov bude prijímať bezkontextový jazyk

$$L = \{a^n b^n c^n : n \geq 0\}.$$

Definujme systém $S_1 = (U, f_r)$ nasledovne:

$$U = \{M_1, M_2\}.$$

Ako prvý definujeme prevodník M_2:

$$M_2 = (\{s, p, p_1, q, r, t, f, f_1\}, \{a, b, c, d, e\}, R_2, s, F_2)$$

kde

$$\begin{aligned} R_2 \ &= \{sa \to p,\ pa \to pa,\ pd \to rd,\ pb \to qd,\ rd \to rd,\ rb \to qd,\ qb \to qb,\ qe \to te,\\ &\quad qc \to fe,\ te \to te,\ tc \to fe,\ fc \to fc,\ sd \to p_1 a,\ p_1 d \to p_1 a,\ p_1 e \to f_1 c,\\ &\quad f_1 e \to f_1 c\} \\ F_{2A} &= \{f\},\ F_{2N} = \emptyset,\ F_{2F} = \{s, f_1\} \end{aligned}$$

Analýza prevodníku M_2

Na prvý pohľad prekvapivo pôsobí abeceda symbolov Σ_2, ktorá okrem symbolov a, b, c obsahuje aj d a e. Je treba poznamenať, že symboly slúžia ako pomocné a bude ich do reťazca pridávať systém.

Graf prevodníku M_2 budeme analyzovať ako dva podgrafy (ďalej budú označované ako *vetvy*). Prvá vetva je tvorená uzlami (stavy) p, r, q, t, f a druhá vetva uzlami p_1, f_1.

V prvej vetve (hornej) je skrytý princíp prijímania jazyka $\{a^n b^n c^n : n \geq 0\}$. Do tejto vetvy prejde M_2 z počiatočného stavu s prečítaním symbolu a. Vetva je zakončená stavom $f \in F_{2A}$. Túto konštrukciu možno chápať ako cyklus. Podmienkou pre vstup do tela cyklu je, že v prefixe spracovávaného reťazca je aspoň jeden symbol a a skok na vyhodnotenie podmienky nastane, keď M_2 v prečíta celý vstup a prejde do stavu f, kedy podľa definície začne svoj výstup čítať od začiatku ako vstup.

To ako je horná vetva navrhnutá, udáva, že do koncového stavu f sa M_2 dostane, len ak má vstupný reťazec tvar

$$a^k d^i b^l e^j c^m : \ k, l, m \geq 1; \ i, j \geq 0$$

V tejto vetve (tele cyklu) sa M_2 snaží odstrániť najľavejší symbol symbol a (prechod do stavu p), nahradiť najľavejší symbol b za pomocný symbol d (prechod zo stavu r alebo p do stavu q) a nahradiť najľavejší symbol c za pomocný symbol e (prechod zo stavu t alebo q do stavu f). Ostatné symboly prepisuje na výstup bez zmeny. V reťazci postupne pribúdajú pomocné symboly a M_2 ich preto musí v nasledujúcich priechodoch zohľadniť.

M_2 teda opakovane prechádza celým vstupným reťazcom zľava doprava, až kým neodstráni všetky symboly a v prefixe. V prípade, že M_2 odstránením symbolu a vstúpi do hornej vetvy, ale už nenájde symbol b alebo symbol c ($k > l$ alebo $k > m$), M_2 reťazec odmietne, pretože nebude môcť urobiť žiadny prechod zo stavu r, resp. t.

V momente, keď M_2 všetky symboly a z prefixu odstráni, vieme, že reťazec má tvar:

$$d^i b^l e^j c^m : \ l, m \geq 0; i, j \geq 1$$

Ak M_2 v stave s prečíta pomocný symbol d na začiatku reťazca, prejde do druhej (spodnej) vetvy. Vieme, že v predošlých priechodoch M_2 odstránil s každým symbolom a aj jeden symbol b a c. Označme počet symbolov a v *pôvodnom* reťazci ako k, počet symbolov b ako l a počet symbolov c ako m. Je isté, že keď M_2 vstúpi do spodnej vetvy, platí

$$k \leq l \text{ a súčasne } k \leq m.$$

V spodnej vetve sa M_2 pokúsi sa prepísať všetky pomocné symboly d na a a e na c. Ak v reťazci zostali symboly b alebo c, odmietne ho. Tým overí, že l *nie je väčšie než* k a m *nie je väčšie než* k, to znamená

$$l \leq k \text{ a } m \leq k.$$

Vetva je zakončená koncovým stavom $f_1 \in F_{2F}$. Ak M_2 prečíta celý reťazec a skončí v tomto stave, systém ukončí preklad. V momente keď sa to stane, platí, že *pôvodný* vstupný reťazec mal tvar:

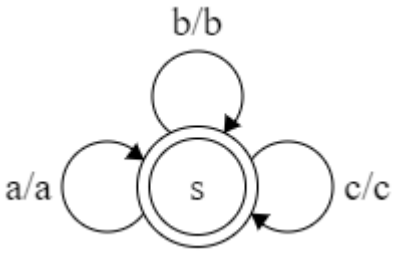
$$a^k d^i b^l e^j c^m : \ k, l, m, i, j \geq 0,$$

kde platí

$$k \leq l \wedge l \leq k \wedge k \leq m \wedge m \leq k,$$

a z toho vyplýva

$$k = l \wedge k = m.$$



Obr. 3.2: Prevodník M_1

Takže M_2 dokáže overiť, že počet symbolov a je rovnaký ako počet symbolov b a c. Kvôli tomu, že M_2 musí zohľadňovať pomocné symboly pridané v predošlých priechodoch však prijímaný jazyk zahŕňa aj reťazce $a^k d^i b^k e^j c^k : \ i,j,k \geq 1$ alebo dokonca $d^i e^j : \ i,j \geq 1$. Keby systém tvoril iba prevodník M_2, preklad systému by bol

$$T(S_1) = \{(a^k d^i b^k e^j c^k, a^{k+i} c^{k+j}) : k,i,j \geq 0, \text{ ale } ((k = 0 \wedge i \neq 0) \Rightarrow j \neq 0)\}$$

Ak položíme $i = 0$ a $j = 0$, vidíme, že všetky dvojice

$$(a^k b^k c^k, a^k c^k) \in T(S_1)$$

Ak prevodníku M_2 v systéme predradíme prevodník M_1, ktorý na jeho vstup prepustí len reťazce, ktoré neobsahujú symboly d a e, získame požadovaný preklad. Ďalej teda definujeme prevodník M_1:

$$M_1 = (\{s\}, \{a,b,c\}, R_1, s, F_1)$$

kde

$R_1 \ = \{sa \to sa, \ sb \to sb, \ sc \to sc\}$
$F_1 \ = F_{1A} \cup F_{1N} \cup F_{1F}$
$F_{1A} = \emptyset, \ F_{1N} = \{s\}, \ F_{1F} = \emptyset$

Prevodník M_1 je jednoduchý filter, ktorý preloží ľubovoľný reťazec zložený zo symbolov a, b a c sám na seba. Jeho jediný stav patrí do množiny F_{1N}, takže výsledok prekladu vždy predá ďalšiemu prevodníku v systéme. Toto stačí na to, aby sme zaručili, že pomocné symboly nebudú primiešané vo vstupnom reťazci od začiatku. Kvôli tomu, že prevodníku M_2 predradíme M_1, sa nežiaduce reťazce nedostanú na vstup M_2 a vylúčime ich z prekladovej množiny systému:

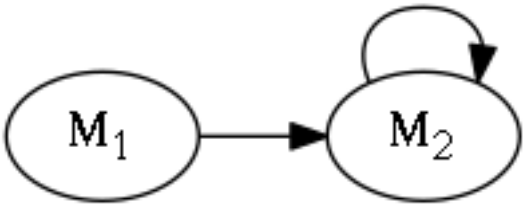
$$T(S_1) \cap \{(a^k d^i b^k e^j c^k, a^{k+i} c^{k+j}) : k,i,j \in N; \ \mathbf{i \neq 0 \vee j \neq 0}\} = \emptyset$$

Keďže je množina F_{1N} neprázdna, musíme do zobrazenia f_r pridať dvojicu, ktorá určí, ktorému prevodníku bude výsledok prekladu M_1 predaný:

$$f_r = \{(M_1, M_2)\}$$

Zobrazenie obsahuje jedinú dvojicu (M_1, M_2), teda výstup M_1 je „pripojený“ na vstup M_2. Okrem toho množina F_{2A} je neprázdna, takže výstup M_2 je pripojený na jeho vlastný vstup. Schéma systému S je na obrázku 3.3. Výsledná prekladová množina definovaná systémom S_1, je

$$T(S_1) = \{(a^n b^n c^n, a^n c^n) : n \geq 0\}.$$



Obr. 3.3: Schéma systému S_1

Príklad spracovania konkrétneho reťazca

Ďalšia sekcia predvedie, ako S_1 spracuje konkrétny reťazec. Činnosť systému je znázornená po krokoch. Jeden krok odpovedá sekvencii prechodov, ktorá začína v počiatočnom stave aktívneho prevodníku, a pri ktorej prevodník prečíta celý vstup. Systém S_1 bude prekladať reťazec $a^3b^3c^3$. Preklad začína prevodník M_1 – implicitný vstupný bod:

1. $M_1: \ saaabbbccc| \vdash^* s|aaabbbccc$

Prevodník M_1 prečítal celý reťazec a prepísal ho na výstup bez zmeny. Ako už bolo zmienené, M_1 plní funkciu filtra. M_1 skončil v stave $s \in F_{1N}$. Podľa zobrazenia f_r sa vyberie prevodník, ktorý bude prekladať výstup M_1. Platí $f_r(M_1) = M_2$, preto výstup prevezme M_2.

2. $M_2: \ saaabbbccc| \vdash paabbbccc| \vdash^* pbbbccc|aa \vdash qbbccc|aad \vdash^* f|aadbbecc$

M_2 prečítal a zahodil symbol a a prešiel do stavu p (horná vetva grafu z obrázku 3.1). Potom nahradil najľavejší symbol b za d a najľavejší symbol c za e. Skončil v stave $f \in F_{2A}$. Podľa definície bude svoj výstup čítať znovu ako vstup.

3. $M_2: \ saadbbecc| \vdash^* f|addbeec; \ f \in F_{2A}$

4. $M_2: \ saddbeec| \vdash^* f|dddeee; \ f \in F_{2A}$

M_2 ešte dvakrát preložil reťazec. Pri každom preklade odstránil jeden symbol a a nahradil jedno b a jedno c. V prefixe už nezostali žiadne a. Overil, že symbolov a nebolo viac než symbolov b alebo c.

5. $M_2: \ sdddeee| \vdash p_1ddeee|a \vdash^* f_1|aaaccc$

M_2 prešiel do spodnej vetvy, spätne prepísal pomocné symboly. Nenatrafil na žiadne symboly b ani c a úspešne prečítal celý reťazec. Symbolov a nebolo ani menej než symbolov b alebo c, takže počty boli rovnaké. M_2 skončil v stave $f_1 \in F_{2F}$ a systém ukončil preklad. Na výstupe je reťazec a^3c^3, takže $(a^3b^3c^3, a^3c^3) \in T(S_1)$.

3.2 Príklad – systém prijímajúci kontextový jazyk

Vlastnosti systémov prevodníkov budú demonštrované ešte na jednom príklade. Systém S_2 bude prijímať kontextový jazyk

$$L = \{a^{2^k} : k \geq 0\}$$

Nech $S_2 = (U, f_r)$ je systém prevodníkov. Množina U je daná

$$U = \{M_0, M_1, M_2\}$$

Nech $M_0 \in U$ je konečný prevodník

$$M_0 = (\{s\}, \{a\}, \{sa \to sa\}, s, F_0)$$

kde

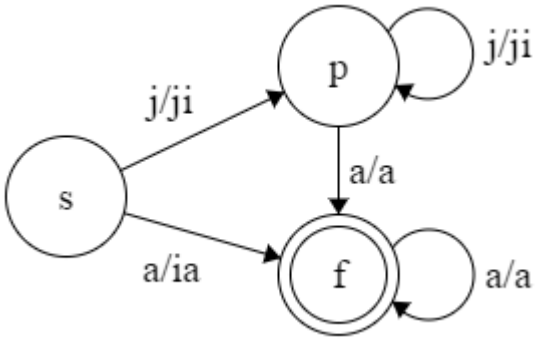
$F_0 \ = F_{0A} \cup F_{0N} \cup F_{0F}$
$F_{0A} = \emptyset,\ F_{0N} = \{s\},\ F_{0F} = \emptyset$

Prevodník M_0 je triviálny. Prekladá reťazec sám na seba a jeho úlohou je (podobne ako prevodník M_1 v predošlom príklade) zaistiť, aby sa na vstup ďalšieho prevodníku, M_1, nedostali reťazce už obsahujúce pomocné symboly, ktoré má pridávať iba systém a ktoré budú riadiť spracovávanie. Ďalej, nech $M_1 \in U$ je konečný prevodník

$$M_1 = (\{s, p, f\}, \{a, i, j\}, R_1, s, F_1)$$

kde

$R_1 \ = \{sa \to fia,\ sj \to pji,\ pj \to pji,\ pa \to fa,\ fa \to fa\}$
$F_1 \ = F_{1A} \cup F_{1N} \cup F_{1F}$
$F_{1A} = \emptyset,\ F_{1N} = \{f\},\ F_{1F} = \emptyset$



Obr. 3.4: Prevodník M_1

Z grafu na obrázku 3.4 sa dá vyčítať, že M_1 prijíma dva typy reťazcov. Reťazce $a^n; n \geq 1$ preloží na ia^n a reťazce $j^m a^n; m, n \geq 1$ preloží na $(ji)^m a^n$. Formálne

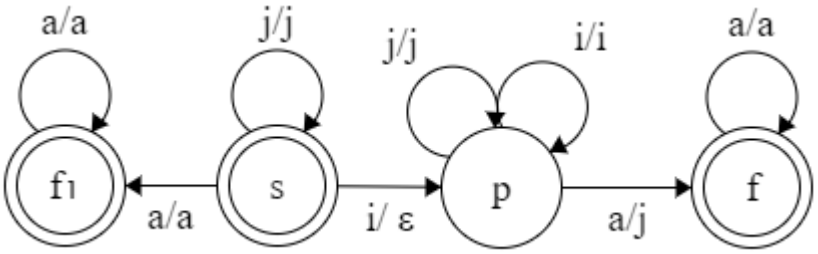
$$T(M_1) = \{(a^n, ia^n), (j^m a^n, (ji)^m a^n) : m, n \geq 1\}$$

Význam tohoto prevodníku bude zrejmý potom, ako definujeme celý systém. Systém používa pomocné symboly i a j, ktoré umiestňuje do prefixu reťazca. Zatiaľ si všimnime, že M_1 pridá buď jeden symbol i na začiatok v prípade, že v prefixe ešte žiadny pomocný symbol nie je, alebo ak v prefixe sú symboly j, ku každému z nich pridá symbol i a súhrnný počet pomocných symbolov v prefixe sa tak *zdvojnásobí.* Konečne, nech $M_2 \in U$ je konečný prevodník

$$M_2 = (\{s, p, f, f_1\}, \{a, i, j\}, R_2, s, F_2)$$

kde

$R_2 = \{sj \to sj,\ si \to p,\ pi \to pi,\ pj \to pj,\ pa \to fj,\ fa \to fa,\ sa \to f_1a,\ f_1a \to f_1a\}$
$F_2 = F_{2A} \cup F_{2N} \cup F_{2F}$
$F_{2A} = \{f\},\ F_{2N} = \{f_1\},\ F_{2F} = \{s\}$



Obr. 3.5: Prevodník M_2

Nech sú vzťahy medzi prevodníkmi nasledovné:

$$f_r = \{(M_0, M_1), (M_1, M_2), (M_2, M_1)\}$$

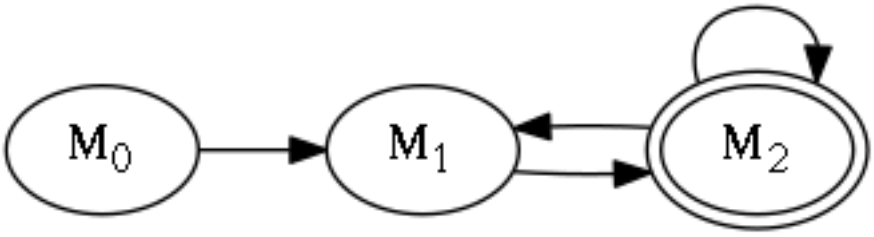
Cieľom prekladu je spočítať, či je počet symbolov a rovný niektorej mocnine čísla 2. Systém nad reťazcom iteruje, symboly postupne prepisuje a značí si pomocné symboly.

Z grafu systému na obrázku 3.6 vyčítame, že výstup M_1 je pripojený na vstup M_2, potom naopak výstup M_2 je pripojený na vstup M_1 a M_2 má pripojený výstup na svoj vlastný vstup. V činnosti systému S_2 je možné identifikovať cyklus. Priebeh jedného priechodu cyklom (na vyššej úrovni) je nasledovný:

1. M_1 pridá do prefixu reťazca niekoľko symbolov i a predá reťazec M_2.
2. M_2 opakovane prechádza reťazcom, až kým nevymaže všetky i z prefixu.
3. M_2 predá reťazec späť prevodníku M_1.

V rámci jedného priechodu cyklom vyššej úrovne prebehne cyklus nižšej úrovne. M_2 opakovane prechádza reťazcom a počíta, či je v reťazci *aspoň* toľko symbolov a ako symbolov i.

Cyklus nižšej úrovne možno identifikovať v obrázku grafu M_2 (3.5). Vetvenie prebieha v stave s. Do tela cyklu prechádza M_2 tým, že nájde symbol i v prefixe (prechod do stavu p). Telo cyklu je pozostáva z prečítania zvyšnej časti prefixu a prepísania najľavejšieho symbolu



Obr. 3.6: Systém S_2

a na j (prechod do stavu f). Skok na opätovné vyhodnotenie podmienky nastane, keď M_2 dočíta zvyšok reťazca a skončí v stave $f \in F_{2A}$.

Keď M_2 prejde do tela cyklu prečítaním a odstránením jedného i, musí sa v reťazci ešte nachádzať aspoň jedno a, inak sa M_2 zasekne v stave p. Tým je zaistené, že preklad neskončí úspešne, ak je počet symbolov a menší než počet i.

O tom, koľko priechodov sa bude M_2 snažiť urobiť, rozhoduje M_1 tým, koľko pridá symbolov i do prefixu. Všeobecne, v momente, keď M_2 predáva riadenie M_1 (tj. na prelome dvoch prechodov cyklom vyššej úrovne), má reťazec tvar

$$j^n a^m : n, m \geq 1$$

Symboly j zastupujú už spracované (prepísané) symboly a. Ako už bolo naznačené, prevodník M_1 pridá ku každému symbolu j jeden symbol i a reťazec bude mať tvar

$$(ji)^n a^m : n, m \geq 1$$

M_2 tento reťazec prevezme a odstraňuje symboly i a prepisuje symboly a zľava. Po prvom priechode cyklom nižšej úrovne bude mať reťazec tvar

$$j^1 (ji)^{n-1} j^1 a^{m-1},$$

po druhom

$$j^2 (ji)^{n-2} j^2 a^{m-2}$$

a tak ďalej, až kým M_2 neodstráni všetky symboly i. Reťazec tak nadobudne tvar

$$j^n (ji)^{n-n} j^n a^{m-n} = j^{2n} a^{m-n}$$

Tomuto priebehu je prispôsobená štruktúra prevodníku M_2. Vidíme, že po jednom priechode cyklom vyššej úrovne sa počet prepísaných symbolov a zdvojnásobil. Je badateľné, že nie je možné, aby $m > n$.

Platí, že vstupný reťazec pozostávajúci z niekoľkých symbolov a, sa v prvom priechode transformuje nasledovne:

$$a^m \xrightarrow{M_1} ia^m \xrightarrow{M_2} ja^{m-1}; m \geq 1$$

Systém teda začína prepísaním jedného symbolu a potom počet prepísaných symbolov v jednotlivých priechodoch vyššej úrovne zdvojnásobuje. Systém skutočne postupne porovnáva pôvodný počet symbolov a s mocnimami čisla 2. Pokiaľ ide o vetvenie, resp. rozhodovanie prevodníku M_2, môže nastať jedna z troch situácií:

1. M_2 nájde symbol i v prefixe reťazca. V tomto prípade sa pokúsi urobiť ďalšiu itéráciu nižšej úrovne. Symbol i vymaže a hľadá symbol a. Ak ho nájde, prejde do stavu $f \in F_{2A}$ a rozhodovanie prebieha znovu. Ak ho nenájde, znamená to, že počet symbolov a sa nevyrovnal počtu i, teda nebol rovný žiadnej mocnine čísla 2 a systém sa zasekne.

2. M_2 už nenájde symbol i, to znamená, že v prefixe sú len symboly j. Za prefixom prečíta symbol a. To znamená, že počet symbolov a je väčší než mocnina čísla 2, s ktorou počet aktuálne porovnával. V tomto prípade prejde do stavu $f_1 \in F_{2N}$ a predá riadenie M_1.

3. M_2 nenájde symbol i v prefixe. Za prefixom nenájde žiadny symbol a. To znamená, že počet symbolov a sa presne vyrovnal niektorej mocnine čisla 2. M_2 skončí v stave $s \in F_{2F}$ a systém prijme reťazec.

Príklady spracovania konkrétnych reťazcov

Táto sekcia na konkrétnom reťazci predvedie, ako systém pracuje. Vložme na vstup M_0 reťazec a^4 a pozorujme činnosť S_2:

1. $M_0: \ saaaa| \vdash^* s|aaaa; \ s \in F_{0N}, \ f_r(M_0) = M_1$
2. $M_1: \ saaaa| \vdash^* f|iaaaa; \ f \in F_{1N}, \ f_r(M_1) = M_2$

M_0 plní len funkciu filtra a nespôsobí žiadne zmeny. Prečítal reťazec a predal ho prevodníku M_1. M_1 pridal jeden pomocný symbol i a predal reťazec prevodníku M_2.

3. $M_2: \ siaaaa| \vdash^* f|jaaa; \ f \in F_{2A}$
4. $M_2: \ sjaaa| \vdash^* f_1|jaaa; \ f_1 \in F_{2N}, \ f_r(M_2) = M_1$

V kroku 3 M_2 odstránil i z prefixu a prepísal najľavejšie a. V kroku 4 zisťuje, že v prefixe už nie sú symboly i, ale v sufixe sú stále neprepísané a. Skončil prvý priechod cyklom (vyššej úrovne), pri čom systém zistil, že symbolov a bude viac než jeden. Začína druhý priechod: reťazec je predaný prevodníku M_1.

5. $M_1: \ sjaaa| \vdash^* f|jiaaa; f \in F_{1N}, \ f_r(M_1) = M_2$
6. $M_2: \ sjiaaa| \vdash^* f|jjaa; \ f \in F_{2A}$
7. $M_2: \ sjjaa| \vdash^* f_1|jjaa; \ f_1 \in F_{2N}, \ f_r(M_2) = M_1$

V kroku 5 M_1 pridal jedno i do prefixu a predal reťazec prevodníku M_2. Ten opäť prepísal jedno a. V kroku 7 zisťuje, že v prefixe už nie sú symboly i a v reťazci stále sú neprepísané a. Systém zistil, že počet symbolov a bude viac než dva. Končí druhý priechod vyššej úrovne a reťazec je predaný M_1.

9. $M_1: \ sjjaa| \vdash^* f|jijiaa; \ f \in F_{2N}, \ f_r(M_1) = M_2$
10. $M_2: \ sjijiaa| \vdash^* f|jjija; \ f \in F_{2A}$
11. $M_2: \ sjjija| \vdash^* f|jjjj; \ f \in F_{2A}$
12. $M_2: \ sjjjj| \vdash^* s|jjjj$

M_1 v kroku 9 pridal dva symboly i. M_2 potom v dvoch priechodoch prepísal dve a. Počet spracovaných symbolov a sa tak zdvojnásobil na štyri. V posledom priechode zisťuje, že odstránil všetky i z prefixu a zároveň prepísal všetky a. Skončil v stave $s \in F_{2F}$. Systém prijal reťazec a dvojica $(aaaa, jjjj) \in T(S_2)$.

Keby sme uvažovali reťazec a^5, postup by bol do bodu 12 podobný. Posledný symbol a na konci reťazca by zostal nedotknutý. V bode 12 by potom systém neskončil. Prečítaním symbolu a na konci by rozhodol, že počet a bude väčší než 4 a poslal by výstup znovu prevodníku M_1:

12. $M_2: \ sjjjja| \vdash^* f_1|jjjja; \ f_1 \in F_{2N}, \ f_r(M_2) = M_1$
13. $M_1: \ sjjjja| \vdash^* f|jijijija; \ f \in F_{2N}, \ f_r(M_1) = M_2$

M_1 pridal štyri symboly i a prevodníku M_2 tak určil, aby prepísal ďalšie štyri a.

14. $M_2: \ sjijijija| \vdash^* f|jjijijij; \ f \in F_{2A}$

15. $M_2: \ sjjijijij| \vdash^* p|jjjijij; p \notin F_2$

M_2 prepísal zostávajúce (v poradí piate) a. V ďalšom priechode M_2 zistil, že v prefixe sú symboly i, ale v reťazci už nie sú symboly a, ktoré by prepísal. Systém overil, že symbolov a bolo viac než 4 ale menej než 8, preto reťazec odmietne.

Príklad ukazuje, že počet prepísaných symbolov sa v každom priechode vyššej úrovne buď zdvojnásobí, alebo systém reťazec odmietne. Systém S_2 tak prijme iba reťazce $a,\ a^2,\ a^4,\ a^8, \ldots$. Preklad $T(S_2)$, definovaný systémom S_2, je

$$T(S_2) = \{(a^{2^k}, j^{2^k}) : k \geq 0\}$$

Preklad ako taký nie je príliš zaujímavý. Samozrejme, pridaním ďalších prevodníkov do systému by sme jednoducho mohli prekladať i na niečo zaujímavejšie. Podstatné je, že príklad naznačuje, že systémy prevodníkov by mohli prijímať i kontextové jazyky.

3.3 Systém prevodníkov – záver

Hlavnou vlastnosťou systémov, ktorá im pridáva na sile v porovnaní so samostatnými prevodníkmi, je posun čítacej hlavy doľava. Samotné prevodníky dokážu ľubovolne prepisovať, či pridávať symboly do reťazca. Vďaka týmto vlastnostiam môže prevodník počas prekladu čítať symboly, ktoré do reťazca predtým sám pridal, resp. pridal iný prevodník v systéme. Prvky systému tak môžu mezi sebou komunikovať a riadiť preklad. Pomocou systému prevodníkov možno namodelovať vetvenie i cyklus.

V príkladoch je možné identifikovať dva výrazné problémy:

1. Prevodníky vždy musia prekladať reťazec až dokonca, aj keď už zvyšok reťazca nie je pre preklad podstatný.

2. Návrat čítacej hlavy je možný len na začiatok reťazca; nie je možné posunúť hlavu o ľubovolný počet pozícií doľava. To znamená, že prevodník, ktorému je predaný reťazec, musí prekladať symboly na začiatku reťazca, ktoré z hľadiska jeho funkcie nie sú vždy podstatné.

Kapitola 4

Systém prevodníkov II.

Táto kapitola zavádza novú definíciu systému prevodníkov, ktorá sa pokúša eliminovať nedostatky tej pôvodnej a zvýšiť tak ich efektivitu.

Definícia 4.1. Systém prevodníkov S je množina

$$S = \{M_1, M_2, \ldots, M_n\}$$

kde

- $M_1, M_2, \ldots, M_n$ sú *konečné* prevodníky a
- M_1 je *vstupný bod* systému. Vstupný bod je opäť určený implicitne ako prevodník zpísaný v množine ako prvý.

Pre účely definovania systému prevodníkov rozšírime pôvodnú definíciu konečného prevodníku. Konečný prevodník M je osmica

$$M = (Q, \Sigma, R, s, F, Q_K, f_K, d)$$

kde

- Q, Σ, R, s, F sú rovnaké ako v definícii 2.1. z kapitoly o konečných prevodníkoch;
- $Q_K \subset Q$ je množina *komunikačných* stavov;
- $f_K : Q_K \to S$ je zobrazenie z množiny komunikačných stavov do množiny prevodníkov tvoriacich systém a
- $d = \{\} \vee d = \{\emptyset\}$ je príznak, určujúci smer posunu čítacej hlavy.

Navyše pre množinu Q_K a zobrazenie f_K platí:

$$t \in Q_K \Rightarrow (t, M_i) \in f_K;\ M_i \in S$$

Znamená to, že každý stav z množiny Q_K musí mať svoj obraz v zobrazení f_K.

Nové položky v definícii konečného prevodníku Q_K a f_K určujú ktoré, stavy spôsobia predanie riadenia a ktorému prevodníku bude riadenie predané. V novej definícii je systém prevodníkov iba množina. Vzťahy medzi prvkami sú určené zobrazeniami, ktoré sú súčasťou definícií jednotlivých konečných prevodníkov. Navyše, nové systémy dokážu čítať reťazec aj

sprava doľava (príznak d). Podrobne budú tri nové položky Q_K, f_K a d vysvetlené v definícii 4.4.b.

Menné konvencie zostávajú rovnaké ako v predošlej kapitole. Prevodníky v systéme sú značené $M_1, M_2, \ldots, M_n$. Množiny stavov, symbolov, pravidiel, koncových stavov, komunikačných stavov, zobrazenie a príznak prevodníku M_1 majú dolný index jedna, prevodníku M_2 dolný index dva atď. Počiatočný stav je v každom prevodníku pomenovaný s:

$$\begin{aligned} S &= \{M_1, M_2, \ldots, M_n\} \\ M_1 &= (Q_1, \Sigma_1, R_1, s, F_1, Q_{K1}, f_{K1}, d_1) \\ M_2 &= (Q_2, \Sigma_2, R_2, s, F_2, Q_{K2}, f_{K2}, d_2) \\ &\ldots \\ M_n &= (Q_n, \Sigma_n, R_n, s, F_n, Q_{Kn}, f_{Kn}, d_n) \end{aligned}$$

Jeden spôsob, ako si predstaviť systém prevodníkov, je ako sieť navzájom prepojených prevodníkov, kde každý má vlastnú vstupnú i výstupnú pásku a prevodníky medzi sebou dokážu predávať riadenie, to znamená predávať si spracovávaný (prekladaný) reťazec. Týmto spôsobom je o systéme vhodnejšie uvažovať pri analýze vzťahov medzi prvkami, tj. pri skúmaní ako sú prevodníky v systéme prepojené. Toto znázornenie bolo použité už v predošlej kapitole.

Druhý možný spôsob intuitívnej predstavy je ako jedna vstupná a jedna výstupná páska, nad ktorými operuje niekoľko prevodníkov. Každý prevodník má vlastnú logiku. V každom momente prekladu je aktívny jeden z nich a systém určuje ako si predávajú riadenie. Táto koncepcia je vhodnejšia pri analýze jednotlivých krokov prekladu.

Príklad 4.2. Ako príklad bude uvedený systém, zložený až z šiestich prevodníkov. Slovne bude popísaný len jeden z nich. To postačí na ilustráciu definície, ale zvyšné prevodníky budú mať význam v ďalších príkladoch, ktoré sa tento systém odkazujú.

Nech S_3 je systém prevodníkov:

$$S_3 = \{M_1, M_2, M_3, M_4, M_5, M_6\}$$

kde

$$\begin{aligned} M_1 &= (\{s, p, p_1, q, r, t, t_1\}, \{a, b, c, d, e, \#\}, R_1, s, \{\}, Q_{K1}, f_{K1}, \{\}) \\ R_1 &= \{s\# \to s\#,\ sa \to pd,\ pa \to pa,\ pb \to q,\ qb \to qb,\ qc \to rc,\ rc \to rc,\ re \to te, \\ &\quad r\# \to t\#,\ se \to p_1e,\ p_1e \to p_1e,\ p_1\# \to t_1\#\} \\ Q_{K1} &= \{t, t_1\} \\ f_{K1} &= \{(t, M_2), (t_1, M_5)\} \\ M_2 &= (\{s, p, t\}, \{c, e, \#\}, R_2, s, \{\}, \{t\}, \{(t, M_3)\}, \{\emptyset\}) \\ R_2 &= \{se \to pe, s\# \to p\#, pc \to te\} \end{aligned}$$

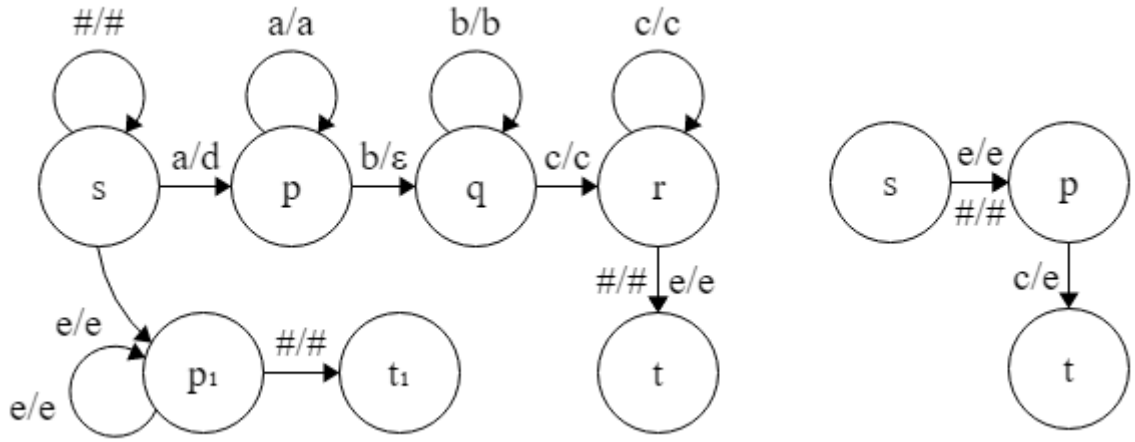
Prevodník M_1 má množinu stavov $Q_1 = \{s, p, p_1, q, r, t, t_1\}$, abecedu symbolov $\Sigma_1 = \{a, b, c, d, e, \#\}$, množinu pravidiel R_1, počiatočný stav s a množinu koncových stavov F_1, ktorá je *prázdna*. Množina komunikačných stavov Q_{K1} obsahuje stavy t a t_1. Zobrazenie f_{K1} musí oba stavy zobraziť na niektorý z prevodníkov. Stav t zobrazuje na prevodník M_2 a stav t_1 na prevodník M_5. Príznak d_1 je prázdna množina.

U prevodníku M_2 je treba vyzdvihnúť, že príznak d_2 je množina obsahujúca symbol prázdnej množiny. Príznaky d_1 a d_2 sú rôzne, čo spôsobí, že M_2 číta opačným smerom ako M_1.

Ďalej

$$
\begin{aligned}
M_3 &= (\{s,p,p_1,q,r,t,t_1\},\{a,b,c,d,e,\#\},R_3,s,\{\},\{t,t_1\},\{(t,M_4),(t_1,M_6)\},\{\emptyset\})\\
R_3 &= \{sc \to pe, pc \to pc, pb \to q, qb \to qb, qa \to ra, ra \to ra, rd \to td, sd \to p_1d, p_1d \to p_1d,\\
&\quad p_1\# \to t_1\#\}\\
M_4 &= (\{s,p,t\},\{a,d\},\{sd \to pd, pa \to td\},s,\{\},\{t\},\{(t,M_1)\},\{\})\\
M_5 &= (\{s,p,q,f\},\{a,c,d,e,\#\},R_5,s,\{f\},\{\},\{\},\{\emptyset\})\\
R_5 &= \{s\# \to p\#,\ pe \to pa,\ pd \to qc,\ qd \to qc,\ q\# \to f\}\\
M_6 &= (\{s,p,q,f\},\{a,c,d,e,\#\},R_6,s,\{f\},\{\},\{\},\{\})\\
R_6 &\ \{s\# \to p\#,\ pd \to pa,\ pe \to qc,\ qe \to qc,\ q\# \to f\}
\end{aligned}
$$

Grafy prevodníkov M_1 a M_2 sú na obrázku 4.1 a grafy prevodníkov M_3 a M_4 na obrázku 4.2.



Obr. 4.1: Prevodníky M_1 (vľavo) a M_2

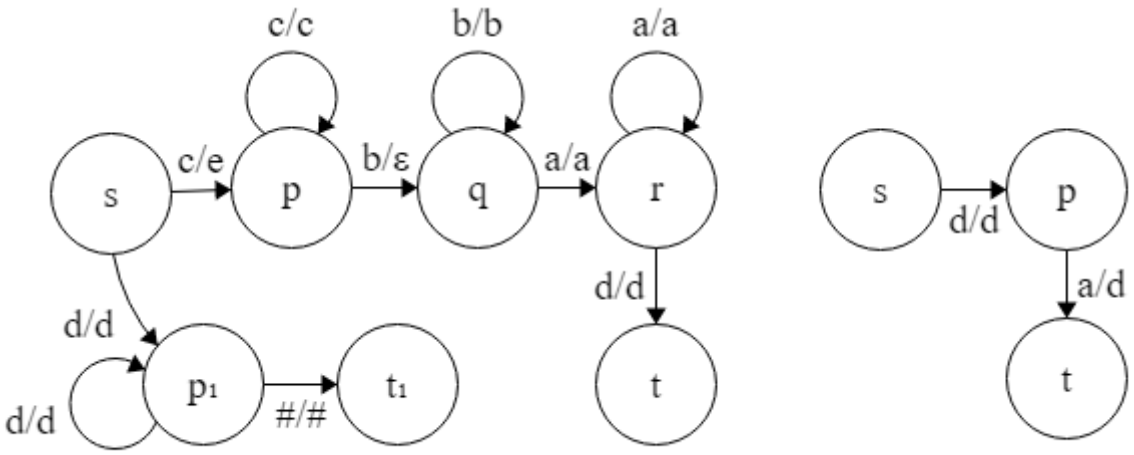
Definícia 4.3. Nech $S = \{M_1, M_2, \ldots, M_n\}$ je systém prevodníkov. Konfigurácia χ' systému S je slovo

$$\chi' = M_i : \chi_i$$

kde

- $M_i = (Q_i, \Sigma_i, R_i, s, F_i, Q_{Ki}, f_{Ki}, d_i) \in S$ je prevodník, ktorý je súčasťou systému;
- : je špeciálny symbol a
- $\chi_i \in Q_i\Sigma_i^*|\Sigma_i^*$ je konfigurácia prevodníku M_i, ako je definovaná v definícii 2.3..

Systémy prevodníkov sú koncipované tak, že v priebehu prekladu je v systéme aktívny vždy jeden prevodník. Konfiguráciu systému preto možno jednoducho vyjadriť ako konfiguráciu aktívneho prevodníku, ku ktorej pripojíme jeho identifikátor.



Obr. 4.2: Prevodníky M_3 (vľavo) a M_4

Definícia 4.4.a. Nech $S = \{M_1, \ldots, M_n\}$ je systém prevodníkov a nech χ_1' a χ_2' sú konfigurácie S

$$\chi_1' = M_i : \chi_1$$
$$\chi_2' = M_i : \chi_2$$

kde

- $M_i \in S$ je prevodník zo systému S a
- χ_1, χ_2 sú konfigurácie prevodníku M_i.

Platí

$$\chi_1 \Vdash \chi_2 \Rightarrow \chi_1' \Vdash \chi_2'$$

Systém prevodníkov robí prechody medzi konfiguráciami podľa definície 2.4.. Platí, že ak prevodník M_i robí prechod z χ_1 do χ_2, tak aj systém S robí prechod z χ_1' do χ_2'. Takýto prechod nazveme prechod *v rámci jedného prevodníku.*

Definícia 4.4.b. Nech $S = \{M_1, \ldots, M_n\}$ je systém prevodníkov. Nech χ_1 je konfigurácia systému S

$$\chi_1 = M_i : tu|v;\ M_i \in S;\ \mathbf{t} \in \mathbf{Q_{Ki}};\ u, v \in \Sigma_i^*$$

Podľa definície musí zobrazenie f_{Ki} priradiť komunikačnému stavu $t \in Q_{Ki}$ niektorý prevodník v systéme. Nech

$$(t, M_j) \in f_{Ki};\ M_j \in S$$

Vždy keď systém urobí prechod podľa definície 4.0.4.a z ľubovoľnej konfigurácie **do** konfigurácie χ_1, automaticky nadviaže urobením prechodu do konfigurácie χ_2

$$\chi_1 \Vdash \chi_2$$

Pre konfiguráciu χ_2 platí

- Ak $d_i = d_j$, potom $\chi_2 = \mathbf{M_j} : su\ |v$
- Ak $d_i \neq d_j$, potom $\chi_2 = \mathbf{M_j} : s\ \text{reversal}(v)\ |\text{reversal}(u)$

Okrem prechodov v rámci jedného prevodníku robí systém navyše aj prechody, kedy aktívny prevodník predá riadenie inému prevodníku v systéme. Takýto prechod nazveme *komunikačný* prechod. Komunikačný prechod robí systém automaticky, bez prečítania či výpisu symbolu, vždy po prechode aktívneho prevodníku do komunikačného stavu.

Prevodník, ktorému je predané riadenie je určený zobrazením f_K aktívneho prevodníku. Po prevzatí riadenia prevodník začína v počiatočnom stave s. Prevodník buď pokračuje v čítaní symbolov rovnakým smerom alebo zmení smer, v závislosti na príznakoch d_i a d_j.

Ak prevodník pokračuje v čítaní rovnakým smerom, znamená to, že prevezme vstupnú i výstupnú pásku od prevodníku, ktorý bol aktívny pred ním a pokračuje v čítaní i zápise na pozíciách, kde predchádzajúci prevodník skončil.

Zmena smeru čítania je simulovaná výmenou vstupnej a výstupnej pásky a prevrátením ich obsahu. Prevodník, ktorý preberá riadenie tak bude mať na začiatku vstupnej pásky symbol, ktorý predchádzajúci prevodník vypísal ako posledný. Na výstupnej páske už bude prevrátený zvyšok vstupu predchádzajúceho prevodníku a ďalej sa bude zapisovať štandardne na jej koniec.

Systém prevodníkov robí prechod medzi dvomi konfiguráciami χ_1' a χ_2' práve vtedy, ak medzi nimi môže urobiť prechod *v rámci jedného prevodníku* alebo *komunikačný* prechod. V oboch prípadoch značíme

$$\chi_1' \Vdash \chi_2'$$

Nech $\chi_1' = M_i : \chi_1$ a $\chi_2' = M_j : \chi_2$, kde χ_1 a χ_2 sú konfigurácie prevodníkov M_i, resp. M_j. Ak systém robí prechod v rámci jedného prevodníku z χ_1' do χ_2' (tozn. $M_i = M_j$), potom môže byť zápis prechodu skrátený a identifikátor prevodníku v χ_2' vynechaný:

$$M_i : \chi_1 \Vdash \chi_2$$

Definícia sekvencie prechodov systému prevodníkov je rovnaká ako definícia 2.6., len namiesto $M = (Q, \Sigma, R, s, F)$ dosadíme $S = \{M_1, \ldots, M_n\}$.

Príklad 4.5. Uvažujme systém prevodníkov S_3 z príkladu 4.2.. Nechajme ho prekladať reťazec $\#aaabbbccc\#$. Preklad naštartuje implictný vstupný bod, prevodník M_1. Systém začína v konfigurácii

$$\chi_1 = M_1 : s\#aaabbbccc\#\ |$$

Z konfigurácie χ_1 urobí systém S podľa pravidla $(s\#, s\#) \in R_1$ prechod:

$$M_1 : s\#aaabbbccc\#\ | \Vdash saaabbbccc\#\ |\#$$

a následne sekvenciu niekoľkých ďalších prechodov:

$$M_1 : saaabbbccc\#\ |\# \Vdash^* r\#\ |\#daabbccc \Vdash t\ |\#daabbccc\#$$

Všetky prechody urobil v rámci prevodníku M_1. Posledný prechod bol prechodom do komunikačného stavu $t \in Q_{K1}$. Ďalej podľa definície systém nadviaže komunikačným prechodom. Platí

$$f_{K1}(t) = M_2$$
$$d_1 \neq d_2$$

Zobrazenie f_{K1} priraďuje stavu t prevodník M_2 a ten preberá riadenie. Príznaky d_1 a d_2 sú rôzne, takže dochádza k výmene vstupnej a výstupnej pásky a prevráteniu ich obsahu:

$$M_1 : t \mid\#daabbccc\# \Vdash M_2 : s\#cccbbaad\# \mid$$

Celú túto postupnosť krokov možno zhrnúť do jedného riadku:

$$M_1 : s\#aaabbbccc\# \mid \Vdash^* M_2 : s\#cccbbaad\# \mid$$

Definícia 4.6. Nech $S = \{M_1, \ldots, M_n\}$ je systém prevodníkov. Preklad $T(S)$ definovaný systémom S je

$$T(S) = \{(x, y) : \ M_1 : sx \mid \Vdash^* M_x : f \mid y;\ M_1, M_x \in S;\ f \in F_x;\ x, y \in (\bigcup_{i=1}^{n} \Sigma_i)^*\}$$

Vďaka tomu, že sme definovali sekvenciu prechodov na úrovni celého systému, je možné vyjadriť prekladovú množinu veľmi jednoducho. Prekladovú množinu tvoria dvojice reťazcov x a y, pre ktoré platí, že systém môže urobiť sekvenciu prechodov z konfigurácie, kde je prevodník, označený ako vstupný bod, v počiatočnom stave a na vstupe má reťazec x, do konfigurácie, kde je ľubovoľný prevodník v systéme v koncovom stave, vstupná páska je prázdna a na výstupnej páske je reťazec y.

Príklad 4.7. Nadviažme na postup systému S_3 v príklade 4.5.:

$$M_1 : s\#aaabbbccc\# \mid \Vdash^* M_2 : s\#cccbbaad\# \mid$$

Reťazec ďalej prekladá prevodník M_2:

$$M_2 : s\#cccbbaad\# \mid \Vdash pcccbbaad\# \mid\# \Vdash tccbbaad\# \mid\#e \Vdash$$

M_2 urobil dva prechody a dostal sa do stavu $t \in Q_{K2}$. Nasleduje komunikačný prechod. Riadenie preberá prevodník $f_{K2}(t) = M_3$. Príznaky d_2 a d_3 sú rovnaké, takže nedochádza k zmene smeru čítania. M_3 začne v počiatočnom stave s. Na vstupe aj výstupe bude mať tie isté symboly, ako mal na vstupe resp. výstupe M_2 predtým, než systém urobil komunikačný prechod.

$$\Vdash M_3 : sccbbaad\# \mid\#e \Vdash^* t\# \mid\#eecbaad \Vdash$$

M_3 urobil sekvenciu prechodov do komunikačného stavu t. Zobrazenie f_{K3} zobrazuje stav t na prevodník M_4, takže M_4 preberá riadenie. Príznaky d_3 a d_4 sú rôzne, preto dochádza k zmene smeru čítania.

$$\Vdash M_4 : sdaabcee\# \mid\# \Vdash^2 tabcee\# \mid\#dd \Vdash$$

M_4 prešiel do komunikačného stavu t. Platí $f_{K4}(t) = M_1$, preto riadenie preberá M_1. Príznaky d_4 a d_1 sú rovnaké, preto M_1 pokračuje v čítaní doprava.

$$\begin{aligned}
&\Vdash M_1 : sabcee\# \mid\#dd \Vdash^* te\#\mid\#dddce \Vdash \\
&\Vdash M_2 : secddd\# \mid\#e \Vdash^2 tddd\# \mid\#eee \Vdash \\
&\Vdash M_3 : sddd\# \mid\#eee \Vdash^* t_1 \mid\#eeeddd\# \Vdash \\
&\Vdash M_6 : s\#dddeee\# \mid \Vdash^* f \mid aaaccc
\end{aligned}$$

Sekvencia je rozpísaná tak, aby boli zvýraznené komunikačné prechody. Môžeme ju zhrnúť do kompaktnejšej podoby:

$$M_1 : s\#aaabbbccc\# \mid \Vdash^* M_6 : f \mid aaaccc$$

Celá sekvencia začínala v počiatočnom stave vstupného bodu (prevodník M_1) a preklad postupne dospel až k tomu, že prevodník M_6 prečítal celý vstup a prešiel do koncového stavu f. Systém S_3 preložil reťazec $\#aaabbbccc\#$ na $aaaccc$, preto dvojica $(\#aaabbbccc\#, aaaccc)$ patrí do prekladovej množiny $T(S_3)$.

4.1 Analýza systému S_3 a porovnanie systémov I. a II.

Porovnané budú dva systémy s (takmer) rovnakou prekladovou množinou. Systém S_1 z príkladu 3.1 v predchádzúcej kapitole, ktorého preklad je $T(S) = \{(a^n b^n c^n, a^n c^n) : n \geq 0\}$ a systém S_3 definovaný v tejto kapitole.

Systém S_3 sa snaží dosiahnuť rovnaký preklad a je založený na rovnakom princípe ako systém z predchádzajúcej kapitoly s rozdielom, že využíva vlastnosti novej definície systému prevodníkov, aby sa znížil celkový počet prechodov pri preklade. Pre pripomenutie, základná myšlienka je, že systém opakovane prechádza reťazcom a v každom priechode sa pokúša prepísať jeden symbol a, jedno b a jedno c na pomocné symboly. Ak sa systému v niektorom priechode nepodarí nájsť všetky tri, reťazec neprijme. Ak sa mu podarí prepísať všetky pôvodné symboly, tak bol počet symbolov a, b a c rovnaký. Predtým než bude kvantifikovaný rozdiel v efektivite je však nutné vyvetliť či vôbec a ako nový systém S_3 funguje.

Fungovanie systému S_3

Systém S_1 prechádzal vždy zľava doprava a vždy od začiatku až do konca. V každom priechode prepísal jednu trojicu symbolov a, b a c (najľavejší z každého) na pomocné symboly. Systém S_3 prechádza reťazcom striedavo zľava doprava a sprava doľava. V každom priechode prepíše najľavejšie a, najpravejšie c a jeden zo symbolov b uprostred. Pomocné symboly d pribúdajú zľava a e sprava. Každým priechodom sa tak znižuje počet symbolov ktoré musí systém S_3 v nasledujúcom priechode prečítať, pretože pomocné symboly už prechádzať nemusí.

Prekladová množina systému S_3 sa trochu líši. Systém potrebuje spôsob ako rozoznať začiatok a koniec reťazca. Z tohoto dôvodu musí byť na začiatku a na konci vstupného reťazca zarážka $\#$. Namiesto reťazcov $a^n b^n c^n$ tak bude prijímať reťazce $\#a^n b^n c^n\#$.

Priechod zľava doprava je realizovaný dvojicou prevodníkov M_1 a M_2. M_1 prečíta zarážku a prepíše prvý symbol a na d. Potom hľadá najľavejší symbol b, ktorý odstráni. Aby sa dokončil celý priechod je potrebné prepísať *najpravejší* symbol c na e. Najpravejší symbol c leží pred zarážkou na konci (v prípade prvého priechodu), alebo pred najľavejším symbolom e. M_1 preto pokračuje hľadaním zarážky na konci alebo prvého symbolu e. Prečítaním jedného z týchto symbolov prejde M_1 do komunikačného stavu t a riadenie preberie prevodník M_2.

$$M_1 : s\#aaabbbccc\# \mid \Vdash^* paabbbccc\# \mid \#d \Vdash^* t \mid \#daabbccc\# \Vdash M_2 : s\#cccbbaad\#$$

M_2 už číta reťazec sprava doľava. M_1 sa pred predaním riadenia nastavil tak, že keď sa M_2 vráti o jeden symbol doľava, bude na pozícii, kde sa nachádza najpravejší symbol c. M_2 vždy prečíta 2 symboly: Zarážku alebo symbol e a za ním symbol c, ktorý prepíše na e. Až

týmto krokom je dokončený jeden priechod zľava doprava. M_2 prechádza do komunikačného stavu t a odovzdáva riadenie prevodníku M_3.

$$M_2 : s\#cccbbaad\# \vdash^2 tccbbaad\# \mid\#e \vdash M_3 : sccbbaad\# \mid\#e$$

Prevodníky M_3 a M_4 realizujú priechod sprava doľava. Robia to isté ako M_1 a M_2, len prevrátene. M_3 nadväzuje na M_2. Začína na pozícii najpravejšieho symbolu c, ktorý prepíše na e a pokračuje v čítaní sprava doľava. Ďalej vymaže najpravejší symbol b (z pohľadu M_3 je to prvý symbol b, ktorý prečíta). K dokončeniu priechodu sprava doľava zostáva prepísať najľavejší symbol a. M_3 sa ďalej nastaví na pozíciu prvého (najpravejšieho) symbolu d, prechádza do komunikačného stavu a predáva riadenie prevodníku M_4. Ak preklad dospel do tohoto štádia, symbol d v reťazci zaručene bude.

$$M_3 : sccbbaad\# \mid\#e \vdash pcbbaad\# \mid\#ee \vdash^* t\# \mid\#eecbaad \vdash M_4 : sdaabcee\# \mid\#$$

M_4 mení smer a číta reťazec zľava doprava. Prečíta symbol d (tu skončil prevodník M_3). Napravo od neho sa nachádza najľavejší symbol a, ktorý prepíše na d. Tým je zakončený jeden priechod sprava doľava. M_4 prechádza do komunikačného stavu t a predáva riadenie M_1. Cyklus je tak uzatvorený. Zapojenie prevodníkov do cyklu je badateľné na obrázku 4.3. M_1 začína ďalšiu iteráciu na najľavejšom symbole a.

$$M_4 : sdaabcee\# \mid\# \vdash^2 tabcee\# \mid\#dd \vdash M_1 : sabcee\# \mid\#dd$$

Cyklus môže byť ukončený po tom, ako systém dokončí kompletný priechod zľava doprava (v takom prípade to detekuje prevodník M_3) alebo priechod sprava doľava (detekuje M_1). Cyklus ukončuje M_3, ak je počet n nepárny. Keď M_3 preberá riadenie, očakáva na vstupe najpravejší symbol c. Ak už sú všetky pôvodné symboly c prepísané a ich počet bol rovnaký ako počet symbolov a a b, nebudú v reťazci ani žiadne a ani b. M_3 prečíta rovno symbol d. Reťazec má v tomto okamihu tvar $\#d^n e^n\#$. M_3 prečíta všetky symboly d a tým sa presunie sa na začiatok reťazca a predá riadenie prevodníku M_6.

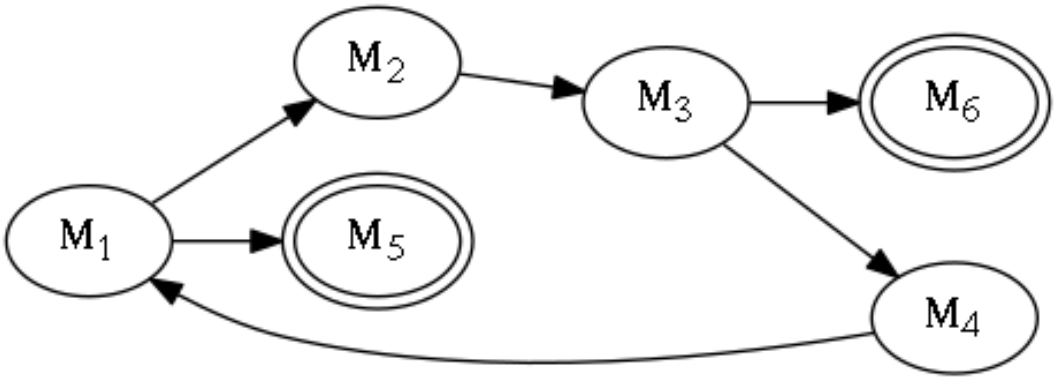
$$\begin{aligned} &M_1 : sabcee\# \mid\#dd \vdash^* M_2 : tddd\# \mid\#eee \vdash \\ &\vdash \mathbf{M_3} : sddd\# \mid\#eee \vdash \mathbf{t_1} \mid\#eeeddd\# \vdash \\ &\vdash M_6 : s\#dddeee\# \mid \vdash^* f \mid aaaccc \end{aligned}$$

Naopak, ak bol počet n párny, cyklus ukončuje M_1. Očakáva na vstupe najľavejší symbol a, ale kvôli tomu, že už boli všetky a a spolu s nimi aj všetky b a c prepísané, prečíta symbol e. Presunie sa na koniec reťazca a predá riadenie prevodníku M_5.

Prevodníky M_1 až M_4 prepíšu reťazec $\#a^n b^n c^n\#$ na $\#d^n e^n\#$ a prevodníky M_5 a M_6 ho potom prepíšu na $a^n c^n$. Musia byť dva, pretože ak cyklus ukončuje M_1, zostane čítacia hlava na konci reťazca a ak cyklu ukončuje M_3, hlava zostane na začiatku reťazca. Každý s prevodníkov M_5 a M_6 je prispôsobený na jeden z týchto prípadov.

Porovnanie efektivity

Systém podľa novej definície je oproti systému podľa definície I. zložitejší. Je poskladaný z viacerých prevodníkov a navýšila sa réžia: systém potrebuje zarážky a používa návrat, aby mohol pri prechode zľava doprava nájsť najpravejší symbol c (aby zistil, či je najpravejší, musí prečítať nasledujúci symbol) a obrátene pri priechode sprava doľava nájsť najľavejší symbol a.



Obr. 4.3: Schéma systému S_3

	Počet prechodov	
n	**Systém I.**	**Systém II.**
1	5	13
2	15	25
3	30	40
4	50	58
5	75	79
6	105	103
7	140	130
8	180	160
9	225	193

Tabuľka 4.1: Celkový počet prechodov pri preklade reťazca $a^n b^n c^n$ v závislosti na n

Zvýšená réžia spôsobila, že je celkový počet prechodov pri preklade reťazcov s malou hodnotou n vyšší, než u systému podľa starej definície. S narastajúcou hodnotou n je však nový systém efektívnejší a rodzdiel v celkovom počte prechodov sa zvyšuje.

V tabuľke 4.1 je vidieť narastajúci rozdiel v počte prechodov. Dá sa vyjadriť, že pre reťazec $a^n b^n c^n$ bude rozdiel $n^2 - 5n - 4$. Táto práca neobsahuje väčšie množstvo demoštračných príkladov, ani odvodenie vzťahu, pretože sú príliš rozsiahle.

Kapitola 5

Prevod aritmetických výrazov

Táto kapitola prezentuje návrh systému prevodníkov podľa defnície II., ktorý prevedie syntakticky správny aritmetický výraz z infixovej notácie do postfixovej. Definícia II. sa hodí na preklad aritmetických výrazov viac, kvôli možnosti flexibilnejšieho návratu, ktorý bude pri porovnávaní priorit operátorov potrebný.

Konvencie značenia

Budeme uvažovať aritmetické výrazy iba s binárnymi operátormi troch úrovní priorit. Prvá úroveň priorit bude zastupovaná symbolom $<$, druhá $+$ a tretia (najvyššia) $*$. Ďalej budú výrazy obsahovať zátvorky plniace obvyklú funkciu. Symbol i (identifier) v reťazci bude reprezentovať ľubovoľný operand vo výraze: identifikátor premennej alebo číselnú konštantu. Symbol # plní funkciu zarážky. Kvôli tomu, že systém prechádza reťazcom obojsmerne, je potrebné, aby zarážka bola na začiatku aj na konci reťazca.

Systém bude postupne premiestňovať operátory za príslušné dvojice operandov a bude potrebovať rozlíšiť, ktoré časti reťazca sú v konkrétnom okamihu prekladu už spracované. Tento účel plnia symboly $E, 1, 2$ a 3. E reprezentuje spracovaný operand, symbol 1 spracovaný operátor 1. úrovne, symbol 2 spracovaný operátor 2. úrovne a symbol 3 spracovaný operátor 3. úrovne.

Grafy prevodníkov budú obsahovať dvojice uzlov prepejené viacerími hranami. Kvôli prehľadnosti bude výhodné zjednotiť tieto hrany do jednej. Takto vzniknutú hranu označíme značkami jednotlivých hrán zaradenými za seba a oddelenými čiarkou. Nech S je systém prevodníkov a $M = (Q, \Sigma, R, s, F, Q_K, F_K, d) \in S$ prevodník v systéme S. Napríklad ak

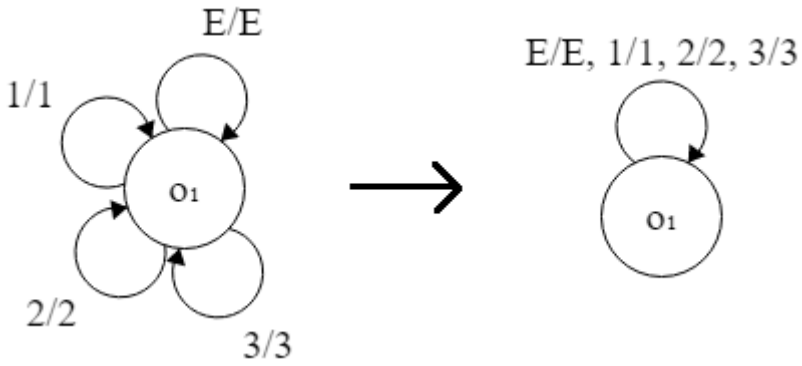
$$\{o_1E \to o_1E,\ o_11 \to o_11,\ o_12 \to o_12,\ o_13 \to o_13\} \subseteq R$$

potom bude v grafe zo stavu o_1 do o_1 viesť jedna hrana označená $E/E,\ 1/1,\ 2/2,\ 3/3$.

5.1 Konceptuálny návrh

Základná myšlienka je, že systém postupne po jednom presúva operátory a pretvára výraz z infixovej notácie do postfixovej. Systém prechádza reťazcom a zľava hľadá operátor, ktorý má prednosť. Keď ho nájde, premiestni ho, označí ako spracovaný. Potom hľadá operátor, ktorého vyhodnotenie má vzľadom na priority nasledovať. To opakuje, až kým nepremiestni všetky operátory.

Pri prechádzaní reťazcom musí systém porovnávať každý operátor s nasledujúcim až kým nenájde dvojicu, kde nasledujúci má menšiu alebo *rovnakú* (uvažujeme operátory s



Obr. 5.1: Zlučenie hrán v obrázku grafu

ľavou asociativitou) prioritu než predchádzajúci, alebo nenatrafí na koniec reťazca. Tento krok bude realizovať jeden prevodník (M_1).

Kvôli tomu, že M_1 dokáže rozhodnúť, či má operand prednosť, až keď prečíta nasledujúci („look-ahead“), systém potrebuje druhý prevodník (M_2), ktorý zrealizuje návrat. Súčasne s návratom môže prehodiť operátor a príslušné operandy do postfixovej formy a poznačiť, že daná časť výrazu je už prevedená.

Posledným problémom sú zátvorky. Vlastnosťou postfixového výrazu je, že k určeniu priority operátorov zátvorky nepoužíva. Súčasťou systému bude ešte tretí prevodník, ktorého zodpovednosť bude redukcia zátvoriek.

Nech je teda S systém prevodníkov, ktorý prevádza aritmetické výrazy do postfixovej notácie

$$S = \{M_1, M_2, M_3\}$$

Pre jednoduchosť bude abeceda symbolov všetkých prevodníkov rovnaká

$$\Sigma = \{i, <, +, *, (,), E, 1, 2, 3, \#\}$$

V nasledujúcich sekciách budú definované a diskutované jednotlivé prevodníky.

5.2 Prevodník M_1

Formálna definícia

Nech $M_1 \in S$ je konečný prevodník

$$M_1 = (Q_1, \Sigma, R_1, s, \{f\}, Q_{K1}, f_{K1}, \{\})$$

kde

$$\begin{aligned}
Q_1 &= \{s, z, o_1, zo_1, l_1, l_2, l_3, o_{21}, o_{22}, o_{23}, t, t_1, f\} \\
R_1 &= \{s\# \to s\#,\ s(\to z(,\ si \to o_1 i,\ sE \to o_1 E,\ s1 \to o_1 1,\ s2 \to o_1 2,\ s3 \to o_1 3, \\
&\quad z(\to z(,\ zi \to zo_1 i,\ zE \to zo_1 E, \\
&\quad o_1 <\!\to l_1 <,\ o_1 + \to l_2 +,\ o_1 * \to l_3 *,\ o_1 \# \to f\#, \\
&\quad zo_1 <\!\to l_1 <,\ zo_1 + \to l_2 +,\ zo_1 * \to l_3 *,\ zo_1) \to t_1), \\
&\quad l_1 i \to o_{21} i,\ l_1 E \to o_{21} E,\ l_2 i \to o_{22} i,\ l_2 E \to o_{22} E,\ l_3 i \to o_{23} i,\ l_3 E \to o_{23} E, \\
&\quad l_1(\to z(,\ l_2(\to z(,\ l_3(\to z(, \\
&\quad o_{21} <\!\to t <,\ o_{21} + \to l_2 +,\ o_{21} * \to l_3 *,\ o_{21}) \to t),\ o_{21} \# \to t\#, \\
&\quad o_{21} E \to o_{21} E,\ o_{21} 1 \to o_{21} 1,\ o_{21} 2 \to o_{21} 2,\ o_{21} 3 \to o_{21} 3, \\
&\quad o_{22} <\!\to t <,\ o_{22} + \to t +,\ o_{22} * \to l_3 *,\ o_{22}) \to t),\ o_{22} \# \to t\#, \\
&\quad o_{22} E \to o_{22} E,\ o_{22} 1 \to o_{22} 1,\ o_{22} 2 \to o_{22} 2,\ o_{22} 3 \to o_{22} 3, \\
&\quad o_{23} <\!\to t <,\ o_{23} + \to t +,\ o_{23} * \to t *,\ o_{23}) \to t),\ o_{23} \# \to t\#, \\
&\quad o_{23} E \to o_{23} E,\ o_{23} 1 \to o_{23} 1,\ o_{23} 2 \to o_{23} 2,\ o_{23} 3 \to o_{23} 3\} \\
Q_{K1} &= \{t, t_1\} \\
f_{K1} &= \{(t, M_2), (t_1, M_3)\}
\end{aligned}$$

Syntax prijímaných reťazcov

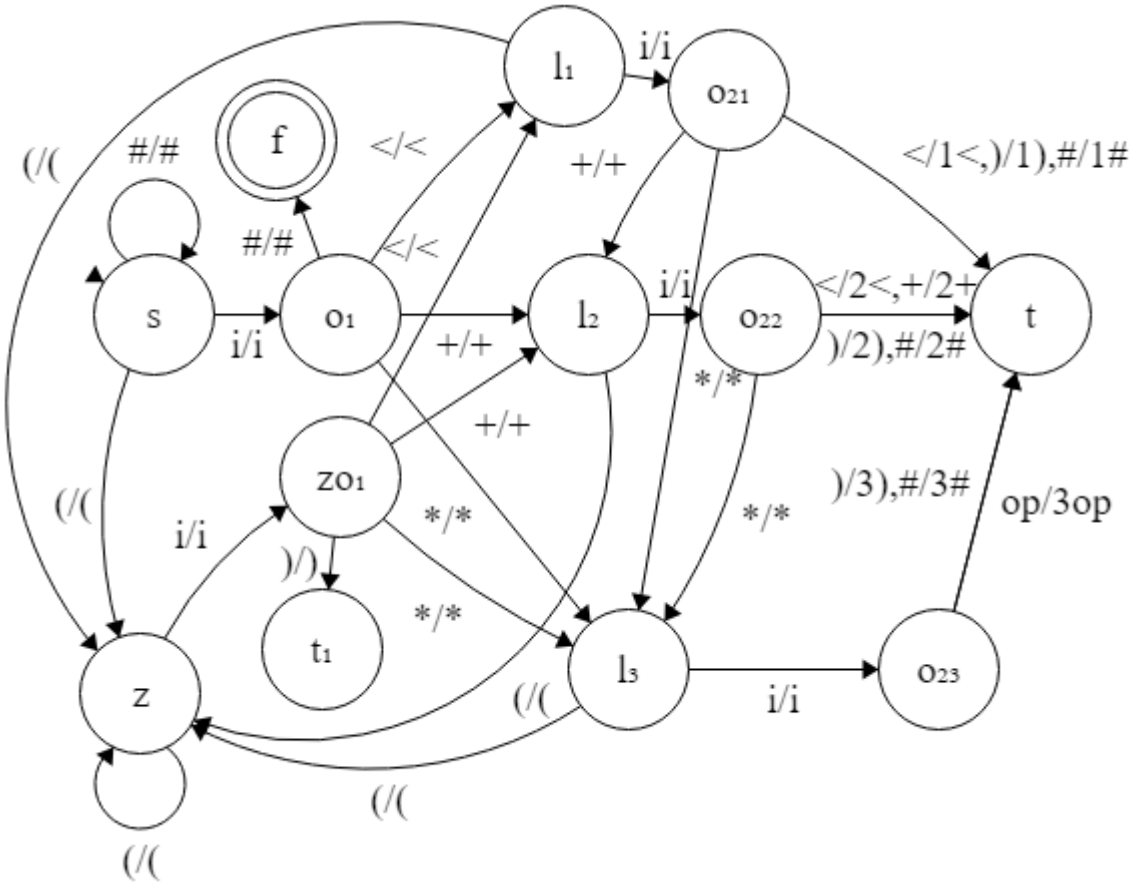
Prevodník M_1 je vstupný bod systému a zároveň najzložitejšia súčasť. Jeho graf je na obrázku 5.2. Definícia II. umožňuje systému počas predkladu rôzne predávať riadenie prevodníkmi, prechádzať reťazcom striedavo zľava doprava aj sprava doľava. Kvôli tomu stráca zmysel prijímaný jazyk na úrovni jednotlivých prevodníkov. Vyjdrenie pomocou regulárneho výrazu nie je v pripade aritmetických výrazov smerodajné. Vyjadrenie (regulárnou) gramatikou by však mohlo poskytnúť prvotný pohľad na M_1 a pomôcť pri analýze vlastností. Nech G_1 je regulárna gramatika:

$$G_1 = \{\{S, V, O\}, \{i, op,), (, \#\}, S, V\}$$

kde

$$\begin{aligned}
P = \{&0 : S \to \#V, \\
&1 : V \to (V, \\
&2 : V \to iO, \\
&3 : O \to opV, \\
&4 : O \to)O, \\
&5 : O \to \#\}
\end{aligned}$$

Gramatika G_1 určuje pravidlá, ktoré musí aritmetický výraz splniť, aby ho M_1 prijal. Charakterizuje ako M_1 overuje *syntaktickú* správnosť aritmetického výrazu. Jedná sa o akýsi zjednodušený pohľad na M_1.



Obr. 5.2: Prevodník M_1

Gramatiku G_1 tvoria neterminálne symboly S, V a O a terminálne symboly i (operand), *op* (operátor), ľavá a pravá zátvorka a zarážka na konci reťazca #. Jej pravidlá poukazujú na to, že (čísla v zátvorkách sú odpovedájúci sled použitých pravidiel):

- Výraz (symbol V) môže záčínať niekoľkými **ľavými zátvorkami** (0,1) alebo **operandom** (0,2),
- za **ľavou zátvorkou** nasleduje ďalšia ľavá zátvorka (1, 1) alebo operand (1, 2),
- za **operandom** nasleduje **operátor** (2, 3), **pravá zátvorka** (2, 4) alebo koniec reťazca (2, 5),
- za **operátorom** nasleduje ľavá zátvorka (3, 1) alebo operand (3, 2).

M_1 sám o sebe neoverí, či ku každej ľavej zátvorke existuje odpovedajúca pravá zátvorka. Overí len, či sú zátvorky správne umiestnené v reťazci vzhľadom k okolitým symbolom. Párovanie zátvoriek zaistí prevodník M_3 pri redukcii. Pri analýze syntaxe nie je dôležitá priorita operátorov, preto G_1 kvôli jednoduchosti používa všeobecný terminálny symbol *op*, ktorý zastupuje ľubovoľný operátor. Porovnávanie priorít bude diskutované v ďalšej sekcii.

Chýba ešte zmienka o tom, aký symbol môže nasledovať za pravou zátvorkou. Keď sa však pozrieme na obrázok grafu M_1 zistíme, že s prečítaním pravej zátvorky pravodník vždy prechádza do komunikačného stavu a vzdáva sa riadenia prekladu.

Funkcia prevodníku M_1

Ako už bolo zmienené, úlohou M_1 je prechádzať reťazcom zľava doprava a nájsť operátor, ktorý má prednosť, vhľadom k predchádzajúcemu operátoru a zátvorkám vo výraze. V tejto časti sa zameriame na to, ako M_1 priraďuje symbolom *význam*. Začneme identifikáciou jednotlivých krokov, ktoré M_1 pri porovnávaní priorit robí:

1. Prečítaj prvý operand (prechody do stavu o_1 alebo do stavu zo_1).
2. Ak za prvým operandom nasleduje pravá zátvorka a bezprostredne pred ním je ľavá zátvorka (stav zo_1), predaj reťazec prevodníku $\mathbf{M_3}$, ktorý sa pokúsi zredukovať zátvorky.
3. Ak za prvým operandom nasleduje koniec reťazca (zarážka #) a bezprostredne pred ním nie je ľavá zátvorka (stav o_1), skonči. Výraz bol zredukovaný kvázi do jedného operandu a reťazec na výstupe je výsledný výraz v postfixovej notácii.
4. Prečítaj operátor, ktorý nasleduje za prvým operandom. Zapamätaj si jeho úroveň priority.
5. Ak za operátorom nasleduje ľavá zátvorka, vráť sa na krok 1.
6. Prečítaj ďalší operand (prechody do o_{21}, o_{22} alebo o_{23}).
7. Prečítaj ďalší symbol.
   (a) Ak je ďalší symbol pravá zátvorka, koniec reťazca # alebo operátor s nižšou či rovnakou prioritou, poznač na aktuálnu pozíciu číslom prioritu predošlého operátora a predaj reťazec prevodníku $\mathbf{M_2}$.
   (b) Ak je ďalší symbol operátor s vyššou prioritou, zapamätaj si úroveň jeho priority a pokračuj bodom 5.

Poznámka: Táto postupnosť krokov obsiahla všetky možnosti, ktoré môžu nastať v reťazci (aritmetickom výraze), spĺňajúcom pravidlá sytaxe, stanovené v predchádzajúcej sekcii. Napríklad v prípade posledného bodu, za operandom nasleduje pravá zátvorka, koniec reťazca alebo operátor. Akýkoľvek iný symbol spôsobí, že S reťazec neprijme – dojde k chybe.

Prečítaniu prvého operandu odpovedá na obrázku grafu M_1 (obrázok 5.2) prechod do stavu o_1, prečítaním jedného symbolu i. Tento krok však v skutočnosti bude zložitejší. Formálny zápis je úplný, zatiaľ čo obrázok grafu (kvôli zjenodušeniu) nie je. Graf bude doplnený po tom, ako zavedieme prevodník M_2. Navyše, operandu môže predchádzať niekoľko ľavých zátvoriek. V takom prípade prechádza prevodník M_1 najprv do stavu z, kde číta ľavé zátvorky, až kým sa nedostane k operandu vo vnútri. Zo stavu z potom prečítaním symbolu i prechádza do stavu zo_1.

V prípade že M_1 neskončí v bode 2 alebo 3, prečítaním operátoru prechádza do jedného zo stavov l_1, l_2 alebo l_3. Graf sa tu rozdeľuje do troch vetiev. Každá vetva je špecifická pre daný operátor. M_1 tak neskôr vie, aký operátor naposledy prečítal podľa toho, v ktorej vetve sa nachádza.

Ak za operátorom nasleduje ľavá zátvorka (bod 5), výraz v zátvorke má prednosť. M_1 sa vráti do stavu z, kde sa bude pokúšať prečítať operand v zátvorke. Bude s ním nakladať ako s prvým operandom. Inak (ak hneď za operátorom nasleduje druhý operand) M_1 prečíta druhý operand a prejde do stavu o_{21}, o_{22} resp. o_{23}. Pre prečítanie druhého operandu tiež platí, že bude zložitejšie, ako je zobrazené v grafe.

Stav	**Vstupný symbol**	**Akcia** M_1
o_{21}, o_{22} alebo o_{23}	) alebo #	Predaj riadenie M_2
o_{21}	$<$	Predaj riadenie M_2
o_{21}	$+$	Prejdi do stavu l_2
o_{21}	$*$	Prejdi do stavu l_3
o_{22}	$<$ alebo $+$	Predaj riadenie M_2
o_{22}	$*$	Prejdi do stavu l_3
o_{23}	$<$, $+$ alebo $*$	Predaj riadenie M_2

Tabuľka 5.1: Rozhodovanie M_1

V ďalšom kroku M_1 prečíta symbol za druhým operandom. V tabuľke 5.1 vidíme, ako M_1 rozhoduje, či predá riadenie M_2 (aktuálny operátor má prednosť) alebo či sa posunie v reťazci ďalej doprava. Ak M_1 prečíta za druhým operandom pravú zátvorku alebo zarážku, predáva riadenie prevodníku M_2 bez ohľadu na to, v ktorej vetve sa nachádza. Systém pracuje tak, že keď M_1 natrafí na pravú zátvorku, neposunie sa ďalej doprava, kým nebude celý obsah zátvoriek prevedený a zátvorky zredukované. Pokiaľ ide o zarážku na konci reťazca, ak M_1 prešiel do niektorého zo stavov o_{2*}, musel predtým prečítať aspoň jeden operátor, ktorý ešte nebol prevedený (spracovaný). Je potrebné vrátiť sa doľava a spracovať ho.

Stav o_{21} odpovedá situácii, kedy M_1 prečítal druhý operand a pred ním operátor 1. úrovne ($<$). Z o_{21} preto M_1 prechádza prečítaním operátoru $<$ (menšia alebo rovnaká priorita) do komunikačného stavu t. Prečítaním $+$ (vyššia priorita) prejde do stavu l_2 a prečítaním $*$ do stavu l_3. Tieto prechody do l_2 resp. l_3 možno chápať, akože M_1 zabudne prioritu posledného prečítaného operátora, zapamätá si prioritu nového a pokračuje v prechádzaní výrazom doprava. Analogicky to je so stavom o_{22}. Zo stavu o_{23} potom v každom prípade prechádza do komunikačného stavu t, pretože neexistuje operátor s vyššou prioritou ako $*$.

Príklad

Tento príklad demonštruje, ako prevodník M_1 hľadá vo výraze operátor, ktorý má prednosť. Nechajme M_1 prekladať reťazec $\#i + i * (i + i)\#$:

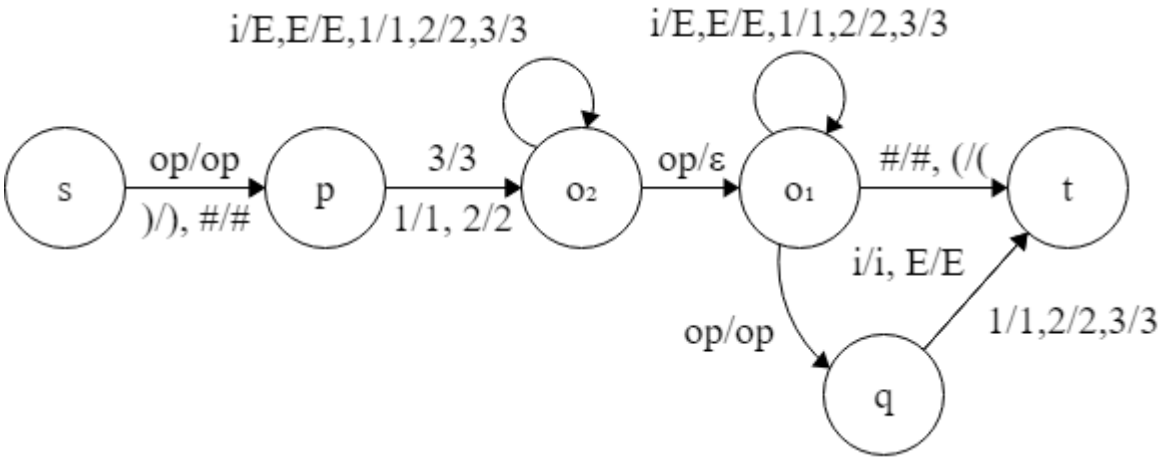
$$M_1 : s\#i + i * (i + i)\# \mid \quad \vdash^2 \; o_1 + i * (i + i)\# \mid \#i \quad \vdash \; l_2 i * (i + i)\# \mid \#i + \quad \vdash$$

M_1 prečítal prvý operand (symbol i) a prešiel do stavu o_1. Potom prečítal operátor $+$ a prešiel do stavu l_2 (priorita 2).

$$\vdash \; o_{22} * (i + i)\# \mid \#i + i \quad \vdash \; l_3 (i + i)\# \mid \#i + i* \quad \vdash$$

Ďalej M_1 prečítal druhý operand a prešiel do stavu o_{22}. Za ním prečítal operátor $*$. M_1 podľa toho, že bol v stave o_{22} vedel, že predošlý operátor mal prioritu 2, teda nižšiu než aktuálne prečítané $*$. Preto prešiel do stavu l_3 a pokračuje v prechádzaní smerom doprava. Za posledný prečítaný operátor je teraz považovaný $*$.

$$\vdash \; zi + i)\# \mid \#i + i * (\quad \vdash$$



Obr. 5.3: Prevodník M_2

M_1 prečítal ľavú zátvorku. Výraz v zátvorkách má mať prednosť. Od tejto chvíle sa chová, ako keby doposiaľ žiadny operátor neprečítal.

$$\vdash\ zo_1 + i)\#\ |\#i + i * (i \quad \vdash\ l_2 i)\#\ |\#i + i * (i+ \quad \vdash\ o_{22})\#\ |\#i + i * (i + i \quad \vdash$$

V ďalších krokoch M_2 prečítal prvý operand v zátvorke, operátor + a druhý operand. Dostal sa do stavu o_{22}.

$$\vdash\ t\#\ |\#i + i * (i + i2)$$

Konečne, M_1 za druhým operandom narazil na pravú zátvorku, čo preň znamená, že posledný (v tomto prípade jediný) operátor v zátvorkách a príslušné operandy majú byť prepísané do postfixovej notácie. Pridal do reťazca symbol 2 a prešiel do komunikačného stavu. Rozhodol, že prednosť ma operátor + v zátvorkách a riadenie preberie M_2.

5.3 Prevodník M_2

Nech $M_2 \in S$ je konečný prevodník

$$M_2 = (\{s,\ p,\ o_1,\ o_2,\ q,\ t\}, \Sigma, R_2, s, \{\}, \{t\}, \{(t, M_1)\}, \{\emptyset\})$$

kde

$$\begin{aligned} R_2 = \{&s <\!\!\rightarrow p <,\ s+ \rightarrow p+,\ s* \rightarrow p*,\ s\# \rightarrow p\#,\ s) \rightarrow p),\ p1 \rightarrow o_2 1,\ p2 \rightarrow o_2 2,\\ &p3 \rightarrow o_2 3,\ o_2 i \rightarrow o_2 E,\ o_2 E \rightarrow o_2 E,\ o_2 1 \rightarrow o_2 1,\ o_2 2 \rightarrow o_2 2,\ o_2 3 \rightarrow o_2 3,\\ &o_2 <\!\!\rightarrow o_1,\ o_2+ \rightarrow o_1,\ o_2* \rightarrow o_1,\\ &o_1 i \rightarrow o_1 E,\ o_1 E \rightarrow o_1 E,\ o_1 1 \rightarrow o_1 1,\ o_1 2 \rightarrow o_1 2,\ o_1 3 \rightarrow o_1 3,\ o_1\# \rightarrow t\#,\ o_1(\rightarrow t(,\\ &o_1 <\!\!\rightarrow q <,\ o_1+ \rightarrow q+,\ o_1* \rightarrow q*,\ qi \rightarrow ti,\ qE \rightarrow tE,\ q1 \rightarrow t1,\ q2 \rightarrow t2,\ q3 \rightarrow t3\} \end{aligned}$$

M_1 prechodom do komunikačného stavu t rozhodne, že predchádzajúci operátor má prednosť a predá riadenie M_2. Úlohou M_2 je vrátiť sa v reťazci doľava, spracovať operátor a k nemu príslušné operandy a označiť túto časť reťazca ako spracovanú. Systém používa symboly $E, 1, 2$ a 3, aby mohol rozlíšiť, ktoré časti reťazca sú spracované. Spracovanie spočíva v tom, že vybraný operátor je presunutý spomedzi príslušných operandov za ne. Operátor sa

prepíše na $1, 2$ respektíve 3 a ak je operandom symbol i prepíše sa na E. Napríklad výraz $i + i$ sa prepíše na $EE2$. Tento výsledok sa dá neskôr ľahko spätne interpretovať na $ii+$.

M_2 preberá riadenie od prevodníku M_1. Príznaky $d_1 = \{\}$ a $d_2 = \{\emptyset\}$ sú rôzne, takže M_2 číta reťazec naopak sprava doľava. Vďaka tomu, že M_2 spätne prechádza symboly, ktoré vypísal M_1, presne vie, čo má na vstupe očakávať. Keď vezmeme do úvahy, že dochádza k reverzácii reťazca, M_2 bude mať vždy na vstupe (po poradí):

- Operátor, pravú zátvorku, alebo zarážku #;
- číslo označujúce prioritu operátora, ktorý bol vybraný prevodníkom M_1;
- druhý operand;
- *vybraný* operátor;
- prvý operand.

V niektorých prípadoch teda bude M_2 čítať dva operátory. Operátor, ktorý je potenciálne na začiatku vstupu, pre M_2 nemá význam a je iba prepísaný na vstup. Je to vlastne ten z dvojice operátorov vzájomne porovnávaných prevodníkom M_1, ktorý mal menšiu alebo rovnakú prioritu. Nadviažme kvôli ukážke na príklad v sekcii 5.2. Systém urobí komunikačný prechod

$$M_1 : t\# \mid\#i + i * (i + i2) \quad \Vdash \quad M_2 : s)2i + i(*i + i\# \mid\#$$

Chceme dosiahnuť, aby sa podreťazec $i + i$ prepísal na $EE2$. M_2 však číta reťazec opačne a keď predá riadenie naspäť prevodníku M_1, jeho výstup sa reverzuje. Preto M_2 musí namiesto $EE2$ vypísať $2EE$. Toto je dôvod prečo M_1 pridáva číslo na danú pozíciu do reťazca. M_2 má takto číslo pripravené na správnej pozícii. Potrebuje už len vymazať vybraný operátor spomedzi dvoch operandov a ak je niektorý z operandov symbol i (nemusí byť vždy – bude vysvetlené ďalej), prepíše ho na E. Pozrime sa ako M_2 postupuje:

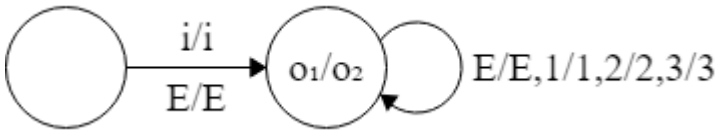
$$M_2 : s)2i + i(*i + i\# \mid\# \quad \Vdash^2 \quad o_2 i + i(*i + i\# \mid\#)2 \quad \Vdash \quad o_2 + i(*i + i\# \mid\#)2E \quad \Vdash$$
$$\Vdash \quad o_1 i(*i + i\# \mid\#)2E \quad \Vdash \quad o_1(*i + i\# \mid\#)2EE$$

Prvý symbol je v tomto prípade pravá zátvorka. Za ňou je poznačený symbol 2. Oba tieto symboly M_2 prepísal na výstup a dostal sa do stavu o_2, kde očakával druhý operand. Operandom je v tomto prípade symbol i. M_2 ho prepísal na E. Ďalej prečítal operátor, ktorý zahodil a prešiel do stavu o_1, kde očakával prvý operand. Prvý operand bol opäť i a M_2 ho prepísal na E.

V tomto momente M_2 vykonal všetky potrebné zmeny. Predtým, než predá riadenie prevodníku M_1, sa musí nastaviť na správnu pozíciu v reťazci, aby mohol M_1 korektne nadviazať a vyhľadať operátor, ktorého spracovanie má podľa precedencie nasledovať. M_2 sa nepotrebuje vrátiť o viac než jeden operátor doľava, pretože časť reťazca od začiatku až do pozície, na ktorej sa nachádza, zostala od posledného porovnávania rovnaká.

Pred prvým operandom môže byť ľavá zátvorka, zrážka na začiatku reťazca alebo operátor. V prípade ľavej zátvorky môže M_2 rovno predať riadenie M_1, pretože výraz v zátvorkách by mal prednosť rovnako ako pri predchádzajúcom porovnávaní. V prípade zarážky tiež odovzdáva riadenie, pretože sa už viac doľava posunúť nemôže.

Ak je pred prvým operandom operátor, musí sa M_2 pred predaním riadenia vrátiť ešte o jeden symbol, tj. na pozíciu operandu pred daným operátorom (stav q). Je to kvôli tomu,



Obr. 5.4: Aktualizácia grafu prevodníku M_1

že M_1 nemôže začínať čítaním operátoru. Ak je týmto operandom sekvencia symbolov o dĺžke viac než jedna, stačí, aby sa M_2 vrátil o jeden symbol.

$$M_2 : o_1(*i + i\# \ |\#)2EE \quad \vdash \ t * i + i\# \ |\#)2EE(\quad \vdash \ M_1 : s(EE2)\# \ |\#i + i*$$

V tomto prípade M_2 prečítal ľavú zátvorku a prešiel rovno do komunikačného stavu t. Systém urobil komunikačný prechod a predal riadenie prevodníku M_1.

Takto postupne v reťazci vznikajú podreťazce tvorené symbolmi $E, 1, 2$ alebo 3. V ďalších krokoch celé tieto podreťazce systém chápe ako jeden operand. Na obrázku grafu M_2 (stavy o_2 a o_1) vidíme, že prevodník akceptuje ako operand ľubovoľne dlhú sekvenciu symbolov $E, 1, 2$ alebo 3.

Po tom, ako M_1 znovu preberie riadenie prekladu, má reťazec tvar $\#i + i * (EE2)\#$. Ak vynecháme redukciu zátvoriek, ktorá bude nasledovať, vidíme, že ďalší operátor, ktorý sa má spracovať je $*$. Prvý operand ktorý sa k nemu viaže je i a druhý celá sekvencia $EE2$. Systém teda bude smerovať k tomu, aby premenil podreťazec $i * (EE2)$ na $EEE23$.

Pokiaľ ide o prechod do komunikačného stavu, M_2 má jediný komunikačný stav t a ten sa v f_{K2} zobrazuje na M_1. Systém S je zostrojený tak, že M_2 sa *vždy* dostane do stavu t. To je, ako už bolo zmienené, možné zaručiť vďaka tomu, že vstup M_2 je určený výstupom M_1. M_2 preto vždy predá riadenie späť prevodníku M_1.

Dodatok k prevodníku M_1

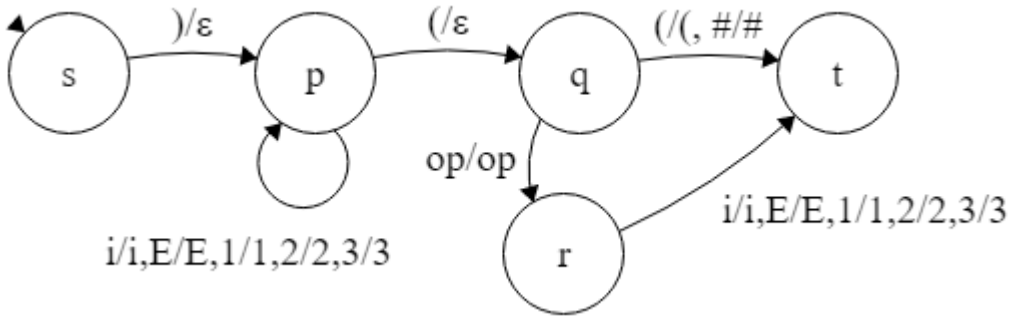
V diskusií o prevodníku M_1 bolo zmienené, že prečítanie operandu bude zložitejšie než je na obrázku grafu. M_1 musí byť rovnako ako M_2 prispôsobený na to, že operandom nemusí byť len symbol i, ale ním môže byť aj sekvencia symbolov $E, 1, 2$ alebo 3 ľubovoľnej dĺžky. Uzly o_1 a o_{2x} teda v skutočnosti majú niekoľko ďalších hrán tak, ako to je na obrázku 5.4.

5.4 Redukcia zátvoriek: Prevodník M_3

Hlavná rozhodovacia komponenta systému S je prevodník M_1. M_1 predáva riadenie prevodníku M_3 prechodom do komunikačného stavu t_1. Do stavu t_1 prechádza zo stavu zo_1 prečítaním pravej zátvorky. Význam stavu zo_1 je, že M_1 práve číta operand, ktorý je hneď za ľavou zátvorkou. Keď M_1 v tomto stave prečíta pravú zátvorku, vie, že má dôjsť k redukcii zátvoriek. M_3 plní len jednoduchú funkciu. Vracia sa smerom doľava a vymaže prvú pravú zátvorku na ktorú natrafi. Potom pokračuje a vymaže prvú ľavú zátvorku na ktorú natrafí.

Systém je zostrojený tak, že keď M_1 prečíta pravú zátvorku, neposunie sa v reťazci ďalej doprava, kým nebude zredukovaná. M_1 a M_2 si budú predávať riadenie, kým nebude prevedený celý podreťazec od danej pravej zátvorky po odpovedajúcu ľavú zátvorku, prípadne (ak ľavá zátvorka chýba) po začiatok výrazu.

Po prečítaní pravej zátvorky teda preklad vyústi do jednej z dvoch sitúácií:



Obr. 5.5: Prevodník M_3

1. Ku pravej zátvorke v reťazci existuje odpovedajúca ľavá zátvorka a do postfixovej formy je prevedený podreťazec medzi zátvorkami.

2. V reťazci neexistuje odpovedajúca ľavá zátvorka a do postfixovej formy je prevedená celá časť reťazca od pravej zátvorky až po začiatok (pokiaľ v reťazci nie je iná chyba).

Druhý prípad znamená syntaktickú chybu. Po prevedení posledného operátora, ktorý zavŕši prevod daného podreťazca sa M_1 bude nachádzať na začiatku reťazca. M_1 sa pokúsi určiť ďalšiu akciu. Začne čítať prvý operand, ktorým je prevedený podreťazec a prejde do stavu o_1. Hneď za ním natrafí na pravú zátvorku. Neexistuje pravidlo $r \in R_1$ s ľavou stranou $\text{lhs}(r) = o_1$), preto sa zasekne a reťazec odmietne. Systém teda detekuje *chýbajúcu ľavú zátvorku*, ale k detekcii dochádza dosť neskoro.

V prípade, že v reťazci chýba pravá zátvorka, systém prevedie celý podreťazec od ľavej zátvorky, ktorá nemá pár až po koniec reťazca. Systém sa vždy dostane najprv k najzanorenejšiemu páru zátvoriek. Ak tento podreťazec obsahuje páry zátvoriek, sú tiež normálne zredukované.

Po prevedení podreťazca sa M_1 nachádza na pozícii ľavej zátvorky. Prečíta ju a prejde do stavu z. Začne čítať prvý operand, ktorým je celý zvyšok reťazca až po zarážku na konci: prejde do stavu zo_1. Keď sa potom dostane k zarážke na konci, zasekne sa, pretože neexistuje pravidlo $r \in R_1$ s ľavou stranou $\text{lhs}(r) = zo_1\#$. Systém detekuje aj *chýbajúcu pravú zátvorku* ale v niektorých prípadoch opäť dosť neskoro.

Formálna definícia

Nech $M_3 \in S$ je konečný prevodník

$$M_3 = (\{s,\ p,\ q,\ r,\ t\}, \Sigma, R_3, s, \{\}, \{t\}, \{(t, M_1)\}, \{\emptyset\})$$

kde

$$\begin{aligned} R_3 = \{&s) \to p\epsilon,\ pi \to pi,\ pE \to pE,\ p1 \to p1,\ p2 \to p2,\ p3 \to p3,\ p(\to q\epsilon,\\ &q(\to t(,\ q\# \to t\#,\ q <\!\!\to r <,\ q+ \to r+,\ q* \to r*,\\ &ri \to ti,\ rE \to tE,\ r1 \to t1,\ r2 \to t2,\ r3 \to t3\} \end{aligned}$$

Platí, že $d_1 \neq d_3$, takže M_3 číta reťazec opačne ako M_1 – sprava doľava. Po komunikačnom prechode (a s ním súvisiacom prevrátení výstupu) má M_3 vždy na vstupe pravú

zátvorku a za ňou operand, ktorým je buď jeden symbol i alebo ľubovolne dlhá sekvencia symbolov $E, 1, 2$ alebo 3. Pokračujme v príklade zo sekcie o prevodníku M_2. Riadenie bolo predané späť prevodníku M_1 potom, ako M_2 spracoval operátor v zátvorkách:

$$M_1 : s(EE2)\# \mid\#i + i* \quad \vdash^* \quad \mathbf{zo_1})\# \mid\#i + i * (EE2 \quad \vdash \quad \mathbf{t_1}\# \mid\#i + i * (EE2) \quad \vdash$$

M_1 rozhodne, že má dôjsť k redukcii zátvoriek a predáva riadenie M_3.

$$\vdash M_3 : s)2EE(*i + i\# \mid\# \quad \vdash \quad p2EE(*i + i\# \mid\# \quad \vdash^* p(*i + i\# \mid\#2EE \quad \vdash$$
$$\vdash \quad q * i + i\# \mid\#2EE$$

M_3 vymazal pravú zátvorku a prešiel do stavu p. V stave p prečítal operand. V tomto prípade ním bola sekvencia $EE2$ (pri čítaní sprava doľava $2EE$). Ďalej zmazal ľavú zátvorku a prešiel do stavu q. Tým je pár zátvoriek zredukovaný.

Predtým než M_3 odovzdá riadenie späť prevodníku M_1, musí sa presunúť na správnu pozíciu, aby mohol M_1 správne nadviazať. Ak je pred ľavou zátvorkou zarážka na začiatku reťazca # alebo ďalšia ľavá zátvorka, prejde priamo do komunikačného stavu t. Ak je pred ľavou zátvorkou operátor, musí sa vrátiť ešte o jeden symbol, aby sa nastavil na operand pred ním. Tento krok je vlastne rovnaký ako u prevodníku M_2, ktorý tiež po vykonaní potrebných krokov predáva riadenie M_1.

$$M_3 : q * i + i\# \mid\#2EE \quad \vdash \quad ri + i\# \mid\#2EE* \quad \vdash \quad t + i\# \mid\#2EE * i$$

V tomto prípade M_3 prečítal operátor a prešiel do stavu r. Potom sa nastavil na operand pred operátorom a prešiel do komunikačného stavu t.

5.5 Príklad – prevod do postfixovej notácie

Táto sekcia pokračuje v príklade z prechádzajúcich sekcii a prezentuje kroky, ktoré systém urobí, aby dokončil prevod reťazca $\#i+i*(i+i)\#$ do postfixovej notácie. Preklad pokračuje po zredukovaní zátvoriek:

$$M_3 : t + i\# \mid\#2EE * i \quad \vdash \quad M_1 : si * EE2\# \mid\#i+ \quad \vdash^* \quad t \mid\#i + i * EE23\# \quad \vdash$$

M_1 znovu prebral riadenie a určil operátor, ktorého spracovanie má nasledovať. Je to operátor $*$. Za druhý operand, ktorý sa k nemu viaže ($EE2$) poznačil symbol 3 a predáva riadenie M_2.

$$\vdash \quad M_2 : s\#32EE * i + i\# \mid \quad \vdash^* \quad t\# \mid\#32EEE + i \quad \vdash$$

M_2 spracoval operátor $*$ a nastavil sa na pozíciu pred prechádzajúcim operátorom. Predáva riadenie späť prevodníku M_1.

$$\vdash \quad M_1 : si + EEE23\# \mid\# \quad \vdash^* \quad t \mid\#i + EEE232\# \quad \vdash$$

M_1 vybral ďalší operátor (+). Za druhý operand, ktorý sa k nemu viaže (EEE23), poznačil symbol 2 a predáva riadenie M_2.

$$\vdash \quad M_2 : s\#232EEE + i\# \mid \quad \vdash^* \quad t \mid\#232EEEE\# \quad \vdash$$
$$\vdash \quad M_1 : s\#EEEE232\# \quad \vdash^* \quad f \mid\#EEEE232\#$$

Prevodník	**Akcia**	**Reťazec**
M_1	Určenie operátora	$\#i + i * (i + i2)\#$
M_2	Prepis	$\#i + i * (EE2)\#$
M_1	Určenie operátora	$\#i + i * (EE2)\#$
M_3	Redukcia zátvoriek	$\#i + i * EE2\#$
M_1	Určenie operátora	$\#i + i * EE23\#$
M_2	Prepis	$\#i + EEE23\#$
M_1	Určenie operátora	$\#i + EEE232\#$
M_2	Prepis	$\#EEEE232\#$
M_1	Určenie operátora	

Tabuľka 5.2: Prevod výrazu $i + i * (i + i)$

M_2 spracoval posledný operátor. Predal riadenie späť prevodníku M_1. Medzi zarážkou na začiatku a na konci je sekvencia symbolov, ktorú systém interpretuje ako jeden operand. M_1 prešiel do koncového stavu a preklad úspešne skončí. Výsledok prekladu je reťazec $\#EEEE232\#$, ktorý odpovedá výrazu $iiii + *+$. Postup pri prevode je prehľadne znázornený v tabuľke 5.2.

Tento príklad má len demonštračný účel. Kapitola neobsahuje viac príkladov, pretože sú príliš rozsiahle. Systém bude dôkladnejšie otestovaný pomocou programu.

Kapitola 6

Implementácia a testovanie

Táto kapitola popisuje programovú implemetáciu systému prevodníkov S (parseru aritmetických výrazov) z kapitoly 5.1. Program je napísaný v objektovo orientovanom jazyku Java. Je k nemu vytvorené jednoduché grafické užívateľské rozhranie, ktorým sa ale táto práca nezaoberá. Na obrázku 6.1 je diagram tried, ktorý zobrazuje všetky triedy objektov, vzťahy medzi nimi a verejné metódy, ktoré triedy implementujú. Pomocou programu je testovaná správnosť parseru. V druhej časti kapitoly je popísaný prístup k testovaniu a konkrétne testovacie prípady.

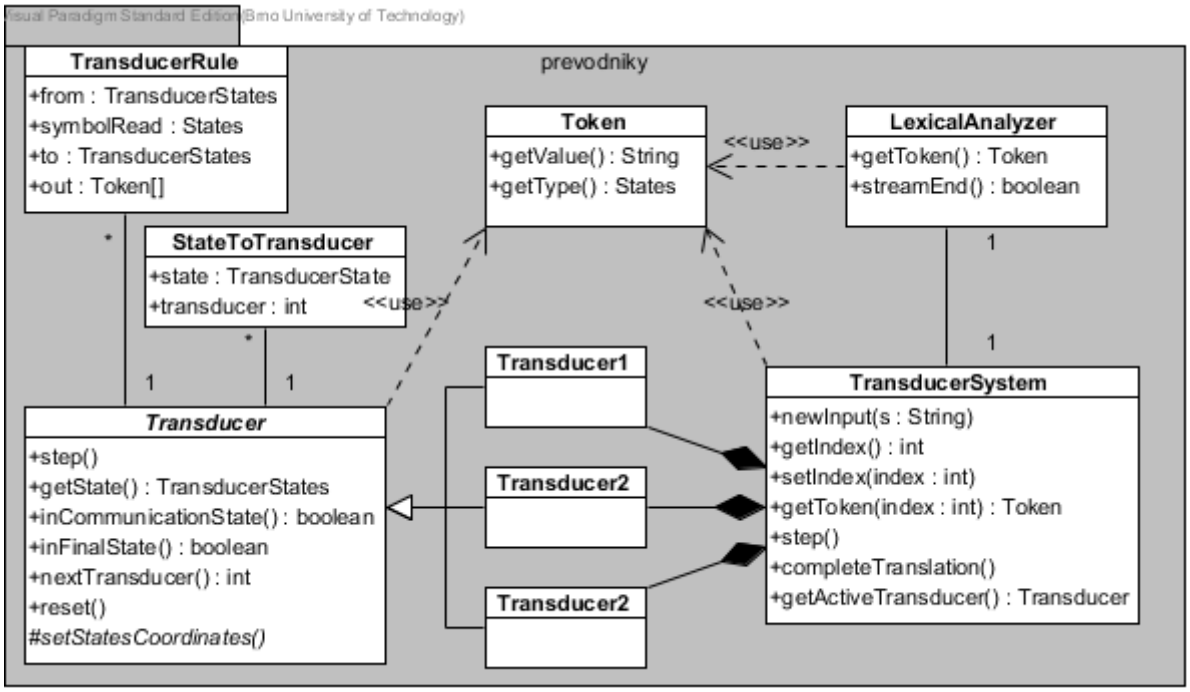
Implementácia parseru

Centrálna komponenta návrhu je samozrejme trieda *TransducerSystem*, ktorá reprezentuje implementovaný systém prevodníkov. Ako už je známe, jeho funkciou preklad aritmetických výrazov v infixovej notácii do notácie postfixovej. Systém prevodníkov získava vstup z lexikálneho analyzátoru, ktorý identifikuje v „surovom“ reťazci znakov jednotlivé symboly.

Trieda *TransducerSystem* teda používa instanciu triedy *LexicalAnalyzer*, ktorá reprezentuje lexikálny analyzátor. Vstupom programu je obyčajný reťazec znakov *String*. Vstup je možné predať ako argument konštruktoru triedy *LexicalAnalyzer* alebo volaním metódy newInput() triedy *TransducerSystem*. Trieda *LexicalAnalyzer* poskytuje metódy getToken() a streamEnd(). Metódou getToken() systém získava z lexikálneho analyzátoru vstupné symboly. Metódou streamEnd() systém zisťuje, či už analyzátor nedostal na koniec vstupného reťazca. Vstupné symboly sú instancie triedy *Token*.

Trieda *TransducerSystem* implementuje konkrétny systém prevodníkov. Tento systém prevodníkov je kompozíciou troch prevodníkov tried *Transducer1, Transducer2* a *Transducer3*, ktoré sú podtriedami abstraktnej triedy *Transducer*. Všetka všeobecná funkcionalita prevodníku je implementovaná v triede *Transducer*. Podtriedy musia implementovať len metódu setStatesCoordinates(), ktorá nastaví množinu stavov a každému stavu priradí súradnice pre zobrazenie v grafickom užívateľskom rozhraní. Inak triedy *Transducer1, Transducer2* a *Transducer3* obsahujú len statické data na zostrojenie konkrétnych prevodníkov.

Hlavné metódy triedy *TransducerSystem* sú step() a completeTranslation(). Metoda step() urobí jeden prechod (v rámci prevodníku alebo komunikačný) a vracia hodnotu *true*, ak je možné urobiť ďalší prechod alebo *false*, ak nastala chyba, alebo preklad skončil. Metoda completeTranslation() dokončí celý preklad. To znamená, že systém robí prechody, kým sa aktívny prevodník nedostane do koncového stavu alebo do konfigurácie z ktorej nemôže urobiť žiadny prechod.



Obr. 6.1: Diagram tried

Trieda *Transducer* obsahuje verejné metódy step(), inCommunicationState(), inFinalState(), nextTransducer() a reset(). Metóda step() urobí jeden prechod v rámci prevodníku. Ak množina pravidiel prevodníku obsahuje pravidlo, ktoré sa dá v aktuálnej konfigurácii aplikovať, prevodník urobí prechod do novej konfigurácie. Inak prevodník prejde do stavu CHYBA. Metódami inCommunicationState() a inFinalState() sa možno dotazovať, či je prevodník v komunikačnom, respektíve koncovom stave. Ak je prevodník v komunikačnom stave, metódou nextTransducer() sa dá zistiť na ktorý prevodník sa daný komunikačný stav mapuje. Návratová hodnota je typu int.

Trieda *Transducer* obsahuje niekoľko štruktúr typu (instancií triedy) *StateToTransducer*. Táto trieda reprezentuje dvojice v zobrazení komunikačných stavov na prevodníky. Prevodníkom v systéme sú priradené celočíselné konštanty. Komunikačné stavy sa nedajú mapovať priamo na objeky prevodníkov, pretože zobrazenie sú statické data, zatiaľ čo prevodníky sú objekty vznikajúce až počas behu programu. Trieda *Transducer* ďalej obsahuje zoznam štruktúr typu *TransducerRule*, reprezentujúci množinu pravidiel.

Instancia triedy *TransducerSystem* riadi preklad, spravuje symboly a uchováva informáciu o tom, z ktorej pozície pásky čítať ďalší symbol. Je potrebné uchovávať zoznam všetkých doposiaľ prečítaných symbolov, pretože prevodníky M_2 a M_3 sa potrebujú vracať späť smerom doľava rôzne ďaleko. Metódy getIndex() a setIndex() triedy *TransducerSystem* používajú prevodníky, aby získali, respektíve zmenili aktuálnu pozíciu na vstupnej páske. Volaním metódy getToken() získavajú od systému vstupné symboly.

Lexikálny analyzátor a typy symbolov

Vo formálnej definícii systému bola použitá množina symbolov

$$\Sigma = \{i, <, +, *, (,), E, 1, 2, 3, \#\}$$

V programovej implementácii sú symboly reprezentované triedou *Token*. Instancie triedy majú dva atribúty value (hodnota) a type (typ symbolu) a vytvára ich lexikálny analyzátor.

Symbol i vo formálnom modeli systému zastupoval ľubovoľný nespracovaný operand. Program rozlišuje tri typy operandov: IDENTIFIER (identifikátor premennej), NUMBER (celočíselná konštanta) a FLOAT (desatinná konštanta). Lexikálny analyzátor je veľmi jednoduchý. Identifikátory premennej sú reťazce alfanumerických znakov a môžu obsahovať znak podčiarkovník „_“. Musia začínať písmenom. Celé číslo je iba reťazec číslic 0 až 9. Desatinné číslo sa skladá z celočíselnej časti a desatinnej časti. Celočíselná a desatinná časť sú reťazce číslic 0 až 9 a sú oddelené bodkou.

Symbolu $<$ (operátor prvej úrovne) z množiny Σ odpovedá v programe typ OPERATOR1, symbolu $+$ (operátor druhej úrovne) typ OPERATOR2 a symbolu $*$ typ OPERATOR3. Lexikálny analyzátor rozpoznáva operátory prvej úrovne $<, >, =, >=, <=$ a $!=$, operátory druhej úrovne $+$ a $-$, a operátory tretej úrovne $*$ a $/$. Instancia triedy *Token*, ktorú lexikálny analyzátor vytvorí, nesie v atribúte value vlastný operátor vo forme reťazca znakov a atribútu type priradí hodnotu OPERATOR1, OPERATOR2, alebo OPERATOR3 podľa úrovne priority.

Symboly $E, 1, 2$ a 3 sú v programe reprezentované symbolom typu PROCESSED. Keď je symbol označený za spracovaný, nastaví sa do atribútu value táto hodnota.

Testovanie

Táto sekcia popisuje, aký prístup bol zvolený k testovaniu správnosti navrhnutého modelu. Nebude rozoberať testovanie lexikálneho analyzátoru, pretože táto súčasť programu tak úzko nesúvisí s riešeným problémom.

Je potrebné, aby testovacia sada otestovala aspoň tieto vlastnosti systému:

- Systém prijme všetky povolené kombinácie symbolov.
- Systém odmietne všetky chybné kombinácie symbolov.
- Systém dokáže overiť, že každá zátvorka má svoj pár, aj keď sú zátvorky ľubovolne zanorené.
- Systém správne vyhodnocuje precedenciu operátorov

Ako prvé pridáme do testovacej sady zložitejšie prípady, pretože pokryjú zároveň aj niekoľko ďalších jednoduhších požiadavkov na test. Ako prvé budú testované zátvorky. Testovacia sada bude obsahovať jeden zožitejší, syntakticky správny aritmetický výraz s viacnásobne zanorenými zátvorkami, a potom výrazy takmer totožné, iba s jednou chýbajúcou zátvorkou. Testovaná bude absencia rôzne zanorenej ľavej aj pravej zátvorky na rôznych pozíciách. Otestovaný musí byť aj prípad, keď je v reťazci rovnaký počet ľavých a pravých zátvoriek, ale výraz je syntakticky nesprávny.

Správnosť určenia precendencie bude otestovaná pre všetky dvojice operátorov: úroveň 1 a 2, úroveň 2 a 3, úroveň 1 a 3 a aj ich obrátené varianty. Okrem toho treba zahrnúť i prípad, ktorý overí, že systém uprednostní výraz v zárvorkách pred operátorom, ktorý je

Testovací prípad	Očakávaný výsledok	Výsledok testu
$(((a+(b))*42)/(e*(42.42+f)))-2$	$ab+42*e42.42f+*/2-$	OK
$((a+(b))*42)/(e*(42.42+f)))-2$	Chyba (chýba ľavá)	OK
$(((a+(b))*42)/e*(42.42+f)))-2$	Chyba (chýba ľavá)	OK
$(((a+(b))*42)/(e*42.42+f)))-2$	Chyba (chýba ľavá)	OK
$(((a+(b)*42)/(e*(42.42+f)))$	Chyba (chýba pravá)	OK
$(((a+(b))*42/(e*(42.42+f)))$	Chyba (chýba pravá)	OK
$(((a+(b))*42)/(e*(42.42+f))$	Chyba (chýba pravá)	OK
$(a+(b)*42)/(e*(42.42+f))$	$ab42*+e42.42f+*/$	OK
$a+(b)*42)/(e*(42.42+f)$	Chyba	OK

Tabuľka 6.1: Testovacie prípady #1: zátvorky

Testovací prípad	Očakávaný výsledok	Výsledok testu
$a<=b-c<d$	$abc-<=d<$	OK
$a+b*c+d$	$abc*+d+$	OK
$a>b*c>d$	$abc*>d>$	OK
$(a+b)*d=e*(f+g)$	$ab+d*efg+*=$	OK

Tabuľka 6.2: Testovacie prípady #2: precedencia

pred zátvorkami aj pred operátorom, ktorý je za zátvorkami a to aj keď má operátor v zátvorkách nižšiu prioritu. V tabuľke 6.2 sú testovacie prípady, ktoré pokryjú všetky tieto požiadaky.

To, že systém prijíma všetky správne a odmieta všetky nesprávne kombinácie symbolov, možno overiť tak, že identifikujeme všetky dvojice symbolov, pre ktoré platí, že jeden za druhým môžu nasledovať a všetky dvojice symbolov, pre ktoré platí, že jeden za druhým nasledovať nemôžu. Vďaka tomu, že bolo párovanie zátvoriek otestované zvlášť, môžeme odhliadnuť od faktu, že zátvorky sú bezkontextové štruktúry a môžeme uvažovať o aritmetických výrazoch ako o regulárnom jazyku.

Pre účel identifikovania dvojic využijeme regulárnu gramatiku $G_1 = \{N, T, P, S\}$ zo sekcie 5.2, ktorá zjednodušene popisuje syntax aritmetických výrazov. Pre každý terminálny symbol $t \in T$, určíme množinu terminálnych symbolov $next()$, ktoré za ním môžu nasledovať. Z množiny $next()$ odvodíme povolené kombinácie. Potom pre každý terminálny symbol t určíme množinu terminálnych symbolov $\Sigma - next()$, ktoré za symbolom t nemôžu nasledovať a z tejto množiny odvodíme chybné kombinácie. Množiny sú v tabuľke 6.3. Kombinácie sú usporiadané dvojice, pre ktoré platí, že druhý symbol môže, respektíve nemôže nasledovať za prvým. Dvojice nie sú zapísané v guľatých zátvorkách, ale v hranatých, kvôli lepšej prehľadnosti.

$t \in T$	$next(t)$	povolené	$\Sigma - next(t)$	chybné
$\#$	$\{(, i\}$	$[\#, "("], [\#, i]$	$\{), op\}$	$[\#, ")"], [\#, op]$
i	$\{), op, \#\}$	$[i, ")"], [i, op], [i, \#]$	$\{(, i\}$	[i,"(", [i,i]
op	$\{(, i\}$	$[op, "("], [op, i]$	$\{), op, \#\}$	$[op, ")"], [op, op], [op, \#]$
(	$\{(, i\}$	$["(", "("], ["(", i]$	$\{), op, \#\}$	$["(", ")"], ["(", op], ["(", \#]$
)	$\{), op, \#\}$	$[")", ")"], [")", op], [")", \#]$	$\{(, i\}$	$[")", "("], [")", i]$

Tabuľka 6.3: Množiny $next()$ a $\Sigma - next()$

Testovací prípad	**Očakávaný výsledok**	**Výsledok testu**
$a * (b + c)$	$abc + *$	OK
$)a * (b + c)$	Chyba (pravá zátvorka na začiatku výrazu)	OK
$<= a * (b + c)$	Chyba (operátor na začiatku výrazu)	OK
$a(b + c)$	Chyba (ľavá zátvorka za identifikátorom)	OK
$a * (b\ c)$	Chyba (dva identifikátory za sebou)	OK
$a * (b+)$	Chyba (pravá zátvorka za operátorom)	OK
$a * /(b + c)$	Chyba (dva operátory za sebou)	OK
$a * (b + c)-$	Chyba (operátor na konci reťazca)	OK
$a * ()$	Chyba	OK
$a * (+b + c)$	Chyba (operátor za ľavou zátvorkou)	OK
$a * ($	Chyba (ľavá zátvorka na konci reťazca)	OK
$(b + c) * (d + e)$	$bc + de + *$	OK
$(b + c)(d + e)$	Chyba	OK
$(b + c) * a$	$bc + a*$	OK
$(b + c)a$	Chyba (identifikátor za pravou zátvorkou)	OK

Tabuľka 6.4: Testovacie prípady #3: chybná syntax

Množina všetkých dvojic v stĺpci $next()$ tabuľky 6.3 sú všetky povolené kombinácie. Množina všetkých dvojic v stĺpci $\Sigma - next()$ sú všetky chybné kombinácie. Všetky povolené kombinácie už testovacia sada pokrýva. Zostáva ešte pridať prípady, ktoré overia, že systém rozoznáva chybné kombinácie. Budeme pri tom vychádzať zo správneho výrazu, do ktorého postupne zavedieme všetky chyby tak, aby každý testovací prípad obsahoval len jednu chybu.

Kapitola 7

Záver

V práci sa úspešne podarilo definovať formálny model systém prevodníkov. Ukázalo sa, že sila systému prevodníkov, ako modelu pre príjímanie jazykov, oproti samotnému konečnému prevodníku vzrástla. Systém prevodníkov je schopný prijímať (minimálne) niektoré bezkontextové a niektoré kontextové jazyky.

Ďalej sa ukázalo, že systém prevodníkov je možné použiť aj ako výpočetný model. Pomocou systému prevodníkov možno modelovať riadiacu štruktúru vetvenie a vďaka možnosti posunút čítaciu hlavu opačným smerom aj štruktúru cyklus.

Pomocou modelu systém prevodníkov sa podarilo namodelovať jednoduchý parser aritmetických výrazov, ktorý výrazy zároveň prekladá do postfixovej notácie. Parser bol otestovaný a funguje správne. Pokiaľ ide o efektivitu, zaostáva v tom, že chybájúce, respektíve prebytočné zátvorky identifikuje v pokročilých fázach prekladu.

V rámci tejto témy by bolo ďalej možné venovať sa otázke systému zásobníkových prevodníkov, či hľadať spôsob, ako prevádzať systémy prevodníkov na bezkontextové gramatiky. Ďalšia výzva do budúcna je zaviesť do systému prevodníkov paralelizmus.

Literatúra

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