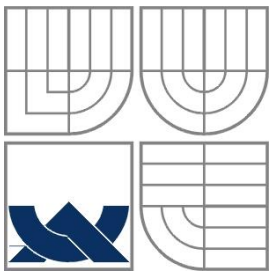
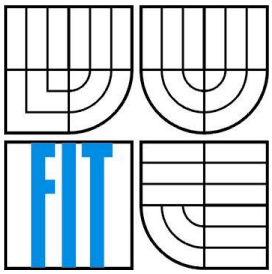


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SYSTÉMY PŘEVODNÍKŮ

TRANSDUCER SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETER HNAT
AUTHOR

VEDOUCÍ PRÁCE — Prof. RNDr. ALEXANDER MEDUNA, Csc.
SUPERVISOR

BRNO 2016

Zadání bakalářské práce/16899/2015/xhnatp00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Hnat Peter**
Obor: Informační technologie
Téma: **Systémy převodníků**
Transducer Systems
Kategorie: Teoretická informatika

Pokyny:

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

Literatura:

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

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

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 2015
Datum odevzdání: 18. května 2016

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 práce se zabývá převodníkovými systémy. Popisuje vlastnosti různých druhů překladových modelů. Na konkrétních příkladech porovnává vyjadřovací sílu těchto formálních modelů. Práce se nejvíc zaměřuje na konečné a zásobníkové převodníky. Zásobníkové převodníky tvoří navržený převodníkový systém. Výsledkem práce je program, vytvořený v programovacím jazyku Python 3.5.1, který demonstruje navržený prevodníkový systém.

Abstract

This thesis contains transducer systems. It describes properties of various translation models. It demonstrates translation power of these models on concrete examples. This thesis focuses mostly on finite and pushdown transducers. Pushdown transducers form designed transducer systems. Result of this thesis is application. Application was created in programming language Python 3.5.1 and it demostrates designed transducer system.

Klíčová slova

Překlad, systém, zásobník, zásobníkové převodníky, Python, konkatenace

Keywords

Translation, system, pushdown, pushdown transducers, Python, concatenation

Citace

HNAT, Peter. *Systémy převodníků*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Meduna Alexander.

Systémy převodníků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pána prof. Alexandra Medunu. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Peter Hnat
10.5.2016

Poděkování

Rád bych se poděkoval svému vedoucímu práce, profesoru Alexandrovi Medunovi za jeho odborné vedení, doporučení materiálů a pomoc při řešení problémů.

Obsah

1 Úvod

Preklad a prevodníky sú jedny z mnoha pojmov, ktoré presne vymedzuje teoretická informatika. Práve tieto pojmy sú definované a popísané v teórii formálnych jazykov. Hlavná myšlienka prevodníkových systémov vychádza zo známych modelov a algoritmov, ktorými sa vo svojich dielach zaoberali významní informatici. V knihe Automata and Languages [1], pán prof. Meduna formálne nadefinoval pojmy preklad, prekladové gramatiky, konečný a zásobníkový prevodník. Tieto pojmy sa stali teoretickým základom modelu prevodníkových systémov, ktoré sú obsahom tejto bakalárskej práce.

Základom celého procesu prevodu vstupného reťazca na postupnosť znakov na výstupe je preklad. Preklad môžeme vnímať v teórii formálnych jazykov z viacerých pohľadov v závislosti na použití daného teoretického modelu. Vo všeobecnosti však môžeme preklad chápať ako súbor dvojíc reťazcov. V prípade prekladačov je preklad definovaný ako dvojica zdrojový - objektový program. Zjednodušene sa prekladač skladá z troch hlavných fáz. Je to lexikálna analýza, syntaktická analýza a generovanie kódu, kde každá táto časť vníma preklad trošku inak. Lexikálna analýza má na vstupe zdrojový text programu a jej úlohou je ho preložiť na postupnosť odpovedajúcich reťazcov zvaných tokeny. Práve tokeny sú na vstupe syntaktickej analýzy. Syntaktická analýza mapuje jednotlivé reťazce tokenov na reťazce reprezentujúce syntaktický strom. Generátor kódu má na vstupe spomínaný syntaktický strom, ktorý prevedie do strojového kódu alebo asembléru. [2] Každá z týchto častí má teda na vstupe reťazce, ktoré sa preložia na postupnosť iných reťazcov podľa pravidiel vymedzených prekladom. Rovnako aj prevodníkové systémy majú na vstupe vstupný reťazec. Nami navrhnutý model sa svojou funkcionalitou a spôsobom transformácie vstupných reťazcov najviac približuje lexikálnej analýze. Výstupom však nie je postupnosť tokenov, ale reťazec. Ten vznikol konkatenáciou výsledkov prevodu jednotlivých prevodníkov.

Pre konštrukciu prevodníkov a ich zavedenie do prevodníkových systémov je nutná znalosť základných princípov uplatňujúcich sa pri preklade s využitím konečných a zásobníkových prevodníkov. Dôležitý je matematický základ z oblasti konečných a zásobníkových automatov, ktoré sa stali predpokladom pre vznik prevodníkov. Z dôvodu maximálneho rozsahu práce je u čitateľa predpokladaná dobrá teoretická znalosť práve týchto formálnych modelov.

Cieľom práce je formálne zadefinovať prevodníkové systémy založené na princípe zásobníkových konečných prevodníkov. Zásadný rozdiel medzi zásobníkovým konečným automatom a zásobníkovým prevodníkom je, že zásobníkový prevodník je schopný nielen čítať znaky na vstupe ale ich aj zapisovať - generovať výstup. Ďalej je potrebné navrhnúť a vytvoriť počítačovú aplikáciu, ktorá predvedie nadefinovaný prevodníkový systém. Implementovaný bol prevodníkový systém zložený z dvojice zásobníkových prevodníkov. Aplikácia demonštruje jednotlivé kroky prekladu. Výsledkom je konkatenácia [1] výstupných reťazcov jednotlivých prevodníkov.

V každej kapitole je čitateľovi priblížená určitá problematika. Tá je najprv rozobraná formálne prostredníctvom definícií na ktoré nadväzujú príklady pre lepšie pochopenie.

Kapitola 2 sa zaoberá prekladom, jeho definíciou a použitím v rôznych formálnych modeloch. Táto kapitola ďalej obsahuje aj jeden konkrétny prekladový model. V tejto kapitole je zadefinovaný syntaxou riadený preklad a jeho použitie.

Kapitola 3 obsahuje definíciu konečného prevodníku a jeho porovnanie so syntaxou riadeným prekladom.

Kapitola 4 okrem formálnej definície obsahuje príklady použitia zásobníkových prevodníkov a ich grafickú reprezentáciu.

Kapitola 5 sa venuje modelu prekladových gramatík. Opisuje základné princípy a postupy uplatňované pri preklade reťazca prekladovými gramatikami.

V kapitole 6 sú formálne popísané prevodníkové systémy. Rozobrané sú dva základné druhy prevodníkových systémov. Taktiež je popísaný spôsob ich využitia a prípadné nasadenie v bežnom živote.

Výsledkom celej práce je program, ktorého implementácia je popísaná v kapitole 7 . Táto kapitola bližšie popisuje použité programovacie nástroje, implementačné detaily a konkrétny príklad spolu aj s grafickou ukážkou z finálnej aplikácie.

2 Preklad

Preklad je binárna relácia medzi zdrojovým a cieľovým jazykom. [3] V teoretickej informatike existujú dve základné metódy, ktoré formálne definujú preklad. Prvou je prekladová schéma. Tá je definovaná ako gramatika obsahujúca mechanizmus k produkovaniu výstupu. [4] Táto gramatika produkuje výstup pre každý reťazec generovaný touto gramatikou. Druhou možnosťou je popis prekladu prostredníctvom prevodníkov. Pre regulárne a bezkontextové jazyky je možné preklad zadefinovať pomocou konečných a zásobníkových prevodníkov. Tie vznikli ako rozšírenie konečného a zásobníkového automatu a sú schopné prekladať celé jazyky, ktoré sú týmito prevodníkmi prijímané. Okrem už spomínaných metód je možné použiť definíciu syntaxe riadeného prekladu, ktorý úzko súvisí s prevodníkovými systémami. Táto kapitola je venovaná obecnej definícii prekladu, jeho sémantike a syntaxou riadenému prekladu. Väčšina definícií použitých v tejto kapitole je prevzatá z knihy [3].

Sémantika prekladu

Denotačná sémantika slúži ako špecifikačný prostriedok definície významu prvkov jazyka a je podkladom pre preklad - generovanie medzijazykov resp. cieľového jazyka. V tomto prípade ide o generovanie výstupného reťazca, ktorý popisuje význam reťazca na vstupe.

Definícia 2.0.1 – Formálny preklad

Je binárna relácia $Z \subseteq D \times H$, ktorá priradzuje každému prvku z množiny D (zdrojový program) množinu prvkov množiny H (jeho prekladov). Pokiaľ Z priradí pre každý prvok množiny D najviac jeden prvok množiny H, potom Z nazývame funkciou a preklad je jednoznačný. Zapisuje $(x, y) \in Z$ alebo $Z(x) = y$ (v prípade, že preklad nie je jednoznačný píšeme $Z(x)$). Definičným oborom formálneho prekladu je množina všetkých hodnôt, ktoré môže nadobúdať prvok x, teda množina D. Obor hodnôt je množina všetkých hodnôt, ktoré môže nadobúdať prvok y, teda množina H. [5]

Definícia 2.0.2 – Preklad formálneho jazyka

Formálne si teda zadefinujme preklad z jazyka L_1 do jazyka L_2. Preklad z jazyka $L_1 \subseteq \Sigma^*$ do jazyka $L_1 \subseteq \Delta^*$ je relácia T z Σ^* do Δ^* taká, že doména relácie T je jazyk L_1 a rozsah T je určený jazykom L_2. Grécke písmeno Σ označuje vstupnú abecedu a písmeno Δ výstupnú abecedu spomínaných jazykov.

V prípade, že táto relácia obsahuje konečný počet prvkov, preklad môžeme definovať vymenovaním dvojíc (x_i, x_o), kde x_i je vstupnou vetou, $x_i \in L_1$ a x_o je výstupnou vetou a , $x_o \in L_2$. Nekonečný preklad môžeme zadefinovať prostredníctvom syntaxou riadeného prekladu, konečných alebo

zásobníkových prevodníkov. V praxi nájdeme veľa rôznych príkladov prekladov. Základným typom prekladu je však preklad špecifikovaný pomocou homomorfizmu. [3]

Definícia 2.0.3 – homomorfizmus

Nech Σ_1 a Σ_2 sú abecedy. Homomorfizmom nazývame každé zobrazenie $h : \Sigma_1^* \rightarrow \Sigma_2^*$ také, že pre každé $a \in \Sigma_1$, $b \in \Sigma_1$ platia homomorfné podmienky:

1. $h(\varepsilon) = \varepsilon$
2. $h(a\,.\,b) = h(a)\,.\,h(b)$

Homomorfizmus – základný princíp

1. Rozložíme zdrojový reťazec na jednotlivé symboly
2. Tieto symboly všetky preložíme (uplatníme zobrazenie h)
3. Výsledky prekladu symbolov zreťazíme podľa pôvodného poradia

Príklad 2.0.1 – preklad reťazca

Predpokladajme, že chceme preložiť každé písmeno gréckej abecedy na vetu v Σ^* teda na korešpondujúco slovenský preklad. Použijeme pri tom homomorfizmus h, kde

1. $h(a) = a$ ak a je členom Σ mínus písmeno gréckej abecedy a
2. $h(a)$ je definované v Tabuľka 1 ak a je písmeno gréckej abecedy:

Malé písmeno gréckej abecedy	*H*
δ	*Delta*
π	*pí*

Tabuľka 1

Napríklad veta $a = \pi\delta$ sa preloží na $a = pídelta$.

2.1 Syntaxou riadený preklad

Každý rekurzívne spočítateľný jazyk môžeme zadefinovať prostredníctvom gramatiky. V tejto časti budeme brať do úvahy iba bezkontexové gramatiky. Gramatiku môžeme veľmi ľahko rozšíriť tak, aby súbežne so vstupným jazykom generovala jazyk výstupný. Takto rozšírenú gramatiku nazývame syntaxou riadený preklad. [6]

Definícia 2.1.1 – syntaxou riadený preklad

Syntaxou riadený preklad je pätica

$$G = (N, \Sigma, \Delta, R, S)$$

kde

N je súbor neterminálov

Σ je vstupná abeceda

Δ je výstupná abeceda

R je konečný súbor pravidiel v tvare $A \rightarrow x, y$ kde $A \in N, x \in (N \cup \Sigma)^*, y \in (N \cup \Delta)^*$

$S \in N$ je štartovací symbol

Neterminály v reťazci y sú permutácie neterminálov v y. Každý neterminál z x má svoj neterminál združený s y. Nadefinujme si reláciu priameho prekladu ⇒ nasledovne:

Pre $u, v \in (N \cup \Sigma)^*, v, z \in (N \cup \Delta)^*, r = A \rightarrow x, y \in R$

$$(\boldsymbol{u}, \boldsymbol{v}) \Rightarrow (\boldsymbol{w}, \boldsymbol{z}) \qquad \textbf{[r]}$$

ak $u = \alpha A \beta, w = \alpha x \beta, v = \gamma A \delta, z = \gamma y \delta$ a neterminál A v u a v je združený. $\Rightarrow^+, \Rightarrow^*$ sú tranzitívnymi a reflexivnymi uzávermi prekladu ⇒.

Preklad definovaný syntaxou riadeným prekladom G:

$$T(G) = \{ (x, y) | x \in \Sigma^*, y \in \Delta^* \; a \; (S, S) \Rightarrow^* (x, y) \}$$

Príklad 2.1.1 – syntaxou riadený preklad

Uvažujme schému syntaxe riadeného prekladu $G = (\{E, T, F\}, \{x, +, -, *, /, (,)\}, \{x, +, -, *, /\}, R, E)$

kde R obsahuje tieto pravidlá:

$E \rightarrow E + T,$	$ET +$
$E \rightarrow E - T,$	$ET -$
$E \rightarrow T,$	T
$T \rightarrow T^* F,$	TF^*
$T \rightarrow T/F,$	$TF/$
$T \rightarrow F$	F
$F \rightarrow (E),$	E
$F \rightarrow x\,,$	x

G preloží výraz v infixovej notácii na opovedajúci postfixový výraz.Uvažujme výraz $x_1 + x_2 * (x_3 - x_4)$ a nasledujúce derivácie:

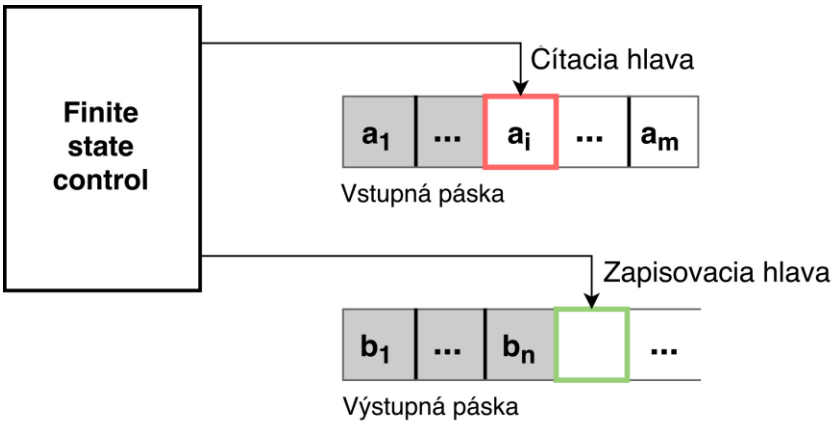
$$(\boldsymbol{E}, \boldsymbol{E}) \Rightarrow (E + T, ET +) \Rightarrow (T_1 + T, T_1T +) \Rightarrow (F + T, FT +) \Rightarrow$$
$$\Rightarrow (x_1 + T, x_1T +) \Rightarrow (x_1 + T * F, x_1TF * +) \Rightarrow$$
$$\Rightarrow (x_1 + F_1 * F, x_1F_1F * +) \Rightarrow (x_1 + x_2 * F, x_1x_2F * +) \Rightarrow$$
$$\Rightarrow (x_1 + x_2 * F, x_1x_2F * +) \Rightarrow (x_1 + x_2 * (E), x_1x_2E * +) \Rightarrow$$
$$\Rightarrow (x_1 + x_2 * (E - T), x_1x_2ET -* +) \Rightarrow$$
$$\Rightarrow (x_1 + x_2 * (T_1 - T), x_1x_2T_1T -* +) \Rightarrow$$
$$\Rightarrow (x_1 + x_2 * (F - T), x_1x_2FT -* +) \Rightarrow$$
$$\Rightarrow \big(x_1 + x_2 * (x_3 - T, x_1x_2x_3T -* +)\big) \Rightarrow$$
$$\Rightarrow (x_1 + x_2 * (x_3 - F), x_1x_2x_3F -* +) \Rightarrow$$
$$\Rightarrow (\boldsymbol{x_1 + x_2 * (x_3 - x_4), x_1x_2x_3x_4} -* +)$$

Ďalej platí, že pre každú schému syntaxe riadeného prekladu $G = (N, \Sigma, \Delta, R, S)$ sme schopní definovať vstupnú gramatiku $G_I = (N, \Sigma, R_I, S)$ kde $R_I = \{ A \to x \mid A \to x, y \in R \; pre \; každé \; y \in \Delta^* \}$. Analogicky definujeme výstupnú gramatiku $G_O = (N, \Sigma, R_O, S)$ kde $R_O = \{ A \to y \mid A \to x, y \in R \; pre \; každé \; x \in \Sigma^* \}$. Tieto gramatiky definujú vstupný a výstupný jazyk G.

3 Konečné prevodníky

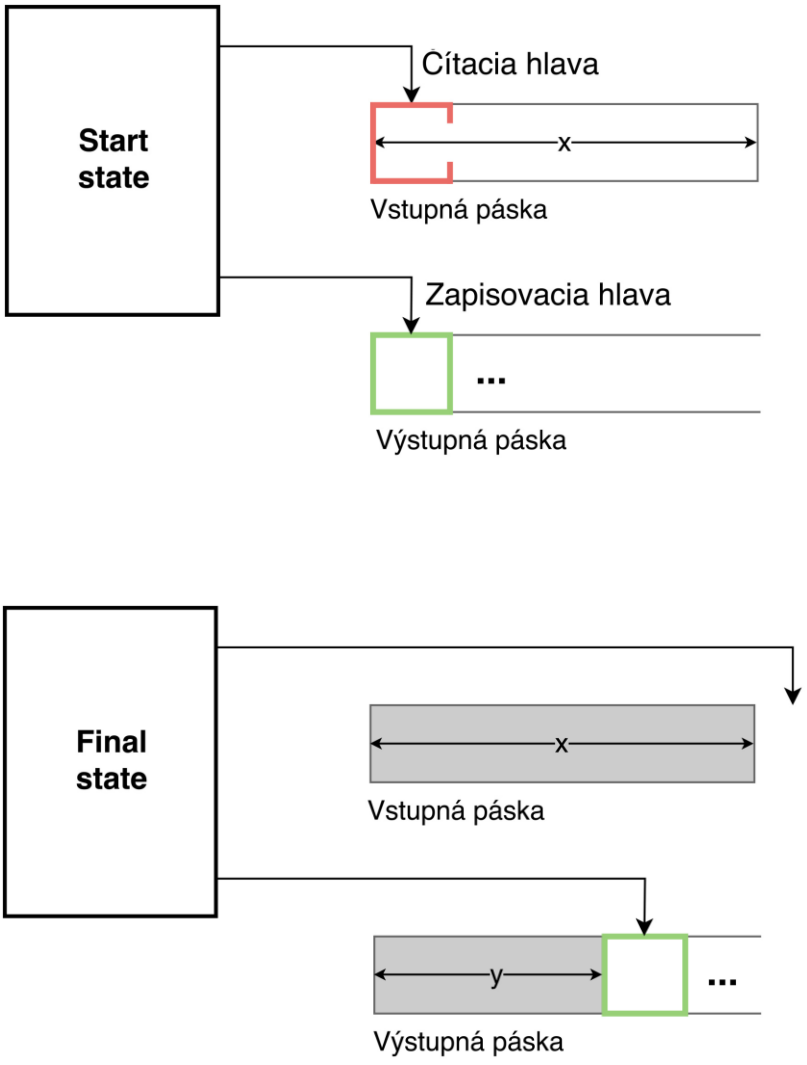
Konečné prevodníky vychádzajú z definície konečných automatov. Formálne je konečný automat výpočtový model, ktorého množina stavov je konečná, teda jasne matematicky spočítateľná. Konečné automaty sú jedným zo základných prostriedkov pre popis regulárnych jazykov. Svoje uplatnenie našli pri vyhľadávaní v texte alebo matematickom popise pamäťových obvodov. Existuje viacero druhov konečných automatov. Základné dva modely sú deterministický konečný automat (DKA) a nedeterministický konečný automat (NKA). [7] V nasledujúcej kapitole budeme uvažovať deterministické konečné prevodníky, ktoré vznikli rozšírením spomínaných konečných automatov, no principiálne sa oba modely veľmi neodlišujú.

Konečné prevodníky spolu so zásobníkovými prevodníkmi sú schopné definovať preklad nielen regulárnych ale aj bezkontextových jazykov. [8] Na rozdiel od konečných automatov, ktoré dokážu čítať symboly na vstupe, konečné prevodníky dokážu nielen čítať vstup, ale aj generovať výstup. [9] Sú teda schopné zapisovať výstupné symboly. Presnejšie povedané, konečný prevodník M, pozostáva z dvoch druhov pások. Prvá je vstupná páska z ktorej prevodník číta vstupný reťazec. Druhá páska slúži na zapisovanie výstupnej postupnosti znakov. Obidve pásky sú rozdelené na štvorce. Každý štvorec vstupnej pásky obsahuje jeden symbol zadaného vstupného slova, $a_1 \dots a_i \dots a_m$. Symbol a_i prečítaný čítacou hlavou predstavuje aktuálny symbol na vstupe. Výstupná páska je nekonečná z jednej strany. Je možné na ňu zapisovať výstupné znaky smerom zľava doprava a páska je nekonečná z pravej strany. V prípade, že výstupná páska obsahuje slovo $b_1 \dots b_n$, zapisovacia hlava sa nachádza na prvom voľnom štvorci na výstupnej páske. V tomto prípade je to hneď za symbolom b_n tak ako je to ukázané na obrázku 3.1. Definície, obrázky ako aj predchádzajúci odstavec sú prevzaté z knihy [1].



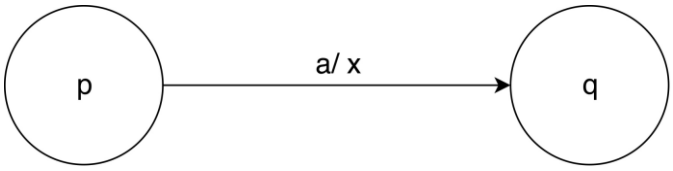
3.1 Konečný prevodník

Konečná kontrola je reprezentovaná konečným počtom stavov spolu s konečnou reláciou. Konečná relácia je špecifikovaná konečným súborom výpočtových pravidiel. Takto špecifikovaný prevodník vykonáva jednotlivé kroky na základe daných výpočtových pravidiel. V jednom kroku prevodník zmení aktuálny stav, zo vstupu neprečíta žiaden alebo prečíta jeden vstupný symbol a zapíše patričný výstup. V prípade, že prevodník M prečíta vstupný symbol a_i, posunie svoju čítaciu hlavu o jedno miesto doprava. Zapisovacia hlava sa posunie o jedno miesto za výstupné slovo. Prevodník M má jeden štartovací stav a jeden alebo viac stavov môže byť koncových. Nech vstupná páska prevodníku M obsahuje slovo x a výstupná páska je prázdna. Prevodník M začne preklad slova x zo štartovacieho stavu. Ak vykonaním sekvencie krokov prevodník M prečíta celé slovo x zo vstupu, na výstup zapíše slovo y a vstúpi do niektorého z koncových stavov môžeme povedať, že prevodník M preložil slovo x na y. Časti prekladu sú ilustrované na obrázku 3.2.



3.2 Preklad reťazca x na y konečným prevodníkom

Prevodníky môžeme takisto ako konečné automaty graficky znázorniť pomocou grafu. Uzly grafu sú pomenované stavmi z množiny Q. Štartovací stav je indikovaný šípkou. Konečné stavy sú odlíšené dvojitým kruhom. Nech je v množine pravidiel R definované pravidlo $pa \vdash qx$. Graficky môžeme toto pravidlo znázorniť hranou zo stavu p do q. Hrana musí byť pomenovaná vstupným/výstupným symbolom, tak ako to je zobrazené na obrázku 3.3. Takto sme schopní graficky popísať celý prevodník M. [10]



3.3 Grafická reprezentácia pravidla pa ⊢ qx

Definícia 3.1 – konečný prevodník

Konečný prevodník je pätica

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

kde

Q je konečná množina stavov

Σ je abeceda pre ktorú platí $\Sigma \cap Q = \Delta$ a $\Sigma = \Sigma_I \cup \Sigma_O$, kde Σ_I je vstupná abeceda a Σ_O je abeceda výstupná

$R \subseteq Q(\Sigma_I \cup \{\varepsilon\}) \times Q\Sigma_O^*$ je konečná relácia

$S \in Q$ štartovací stav

$F \subseteq Q$ množina koncových stavov

Uvažujme konečný prevodník $M = (Q, \Sigma, R, s, F)$ tak ako sme si ho zadefinovali vyššie. Všetky prvky patriace do množiny R nazývame pravidlá. Množina R teda obsahuje súbor definovaných pravidiel. Majme pravidlo $(pa, qw) \in R;\ p, q \in Q;\ a \in \Sigma_I\{\varepsilon\}$ a $z \in \Sigma_O^*$. Namiesto zápisu (pa, qw) môžeme pravidlo zapísať takto:

$$r: pa \vdash qz$$

kde pa je ľavá strana pravidla r a qz tvorí jeho pravú stranu.

Definícia 3.2 – konfigurácia

Konfigurácia prevodníku M je trojica (q, x, y) kde

$q \in Q$ je aktuálny stav

$x \in \Sigma_I$ je zvyšná časť (suffix) vstupného slova

$y \in \Sigma_O$ je generovaný prefix výstupného slova

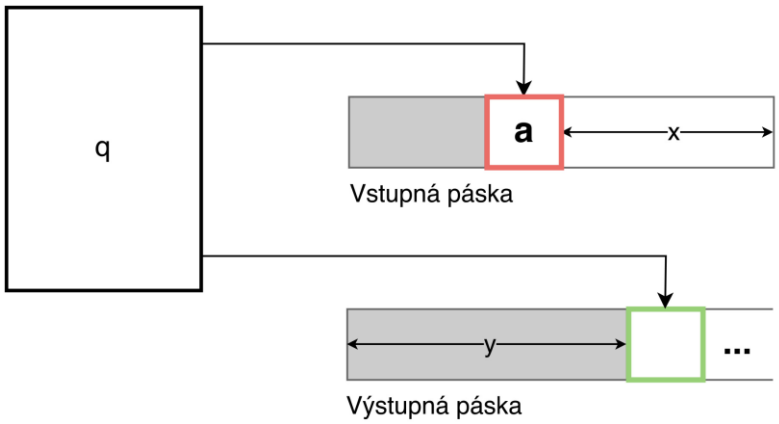
Definícia 3.3 – krok

Krok prevodníku M, označujeme symbolom $\vdash$

$$(q, ax, y) \vdash (p, x, yv)$$

kde $p, q \in Q;\ a \in (\Sigma_I \cup \{\varepsilon\}); x \in \Sigma^*; y, v \in \Delta^*$ a $((q, a), (p, v)) \in R$. $\vdash^+$ a $\vdash^*$ sú tranzitívnym a relfexívno-tranzitívnym uzáverom kroku $\vdash$. [6]

Konfigurácia prevodníku teda predstavuje aktuálny stav všetkých jeho častí (Pozri obrázok 3.4). Tak ako to opisuje formálna definícia ide o aktuálny stav v ktorom sa prevodník nachádza, zostávajúca časť reťazca na vstupe a zapísaná časť slova na výstupe. Aplikovaním pravidiel prevodník prechádza medzi jednotlivými stavmi, číta vstup a zaznamenáva výstup. Môžeme teda povedať, že v každom kroku prevodník M zmení svoju konfiguráciu. Ak sekvenciou krokov prevodník prečíta celý vstup x, zapíše výstup y a skončí v jednom z koncových stavov hovoríme, že úspešne preložil reťazec x na y.



3.4 Konfigurácia

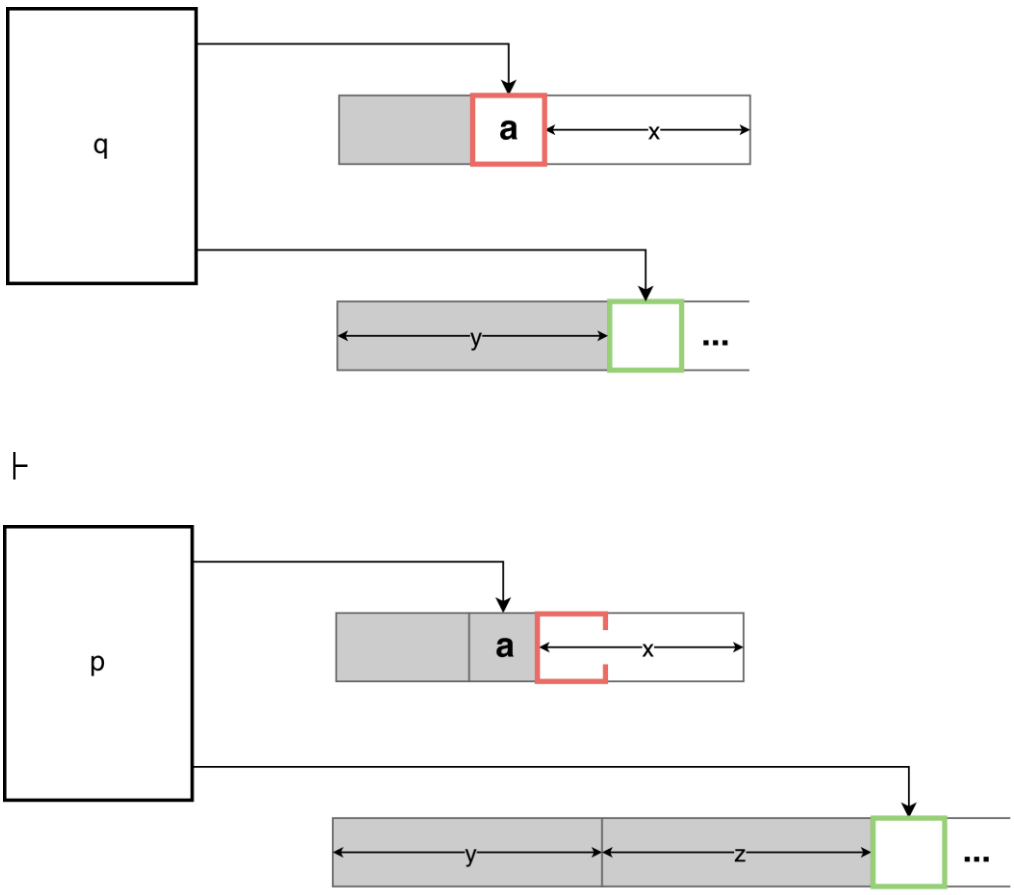
Uvažujme krok prevodníku M vzhľadom na pravidlo $r: qa \vdash pz \in R$. Pri aplikovaní uvedeného pravidla uvažujme jednu z nasledujúcich možností:

1. $a \in \Sigma_I, z \in \Sigma_O^+$
2. $a = \varepsilon, z \in \Sigma_O^+$
3. $a \in \Sigma_I, z = \varepsilon$
4. $a = \varepsilon, z = \varepsilon$

1. V prvom prípade predpokladajme, že $a \in \Sigma_I$ a $z \in \Sigma_O^+$. V tomto bode, prevodník M prejde z aktuálneho stavu q do stavu p, zo vstupu prečíta a, posunie čítaciu hlavu o jedno miesto doprava , zapíše na výstup reťazec z. Zapisovacia hlava sa posunie na prvé voľné miesto z pravej strany, teda hneď za reťazec z (Pozri obrázok 3.5).
2. V druhom prípade je $a = \varepsilon$ a $z \in \Sigma_O^+$. Pravidlo r, môžeme prepísať takto $r: q \vdash pz$. Jeho aplikovaním prevodník M znova prejde zo stavu q do stavu p, na výstup zapíše reťazec z, ale

nakoľko neprečítal zo vstupu žiaden symbol, neposunie svoju čítaciu hlavu. To znamená že v ďalšom kroku bude na vstupe rovnaký symbol.

3. Ak $a \in \Sigma_I$ a $z = \varepsilon$, pravidlo r bude mať tvar $r: qa \vdash p$. V tomto bode, prevodník M prejde zo stavu q do stavu p, zo vstupu prečíta a, posunie čítaciu hlavu o jedno miesto doprava. Na výstup sa však nezapíše žiaden symbol.
4. V poslednom prípade, kde $a = \varepsilon$ a $z = \varepsilon$, prevodník M iba prejde zo stavu q do stavu p. Zo vstupu sa neprečíta žiaden symbol ako aj na výstup sa nič nezapíše.



3.5 Krok podľa pravidla qa ⊢ pz

Definícia 3.4 – preklad definovaný konečným prevodníkom

Nech $M = (Q, , \Sigma, \mathrm{R}, \mathrm{s}, \mathrm{F})$ je konečný prevodník, $x \in \Sigma_I^*$ a $y \in \Sigma_O^*$. M preloží x na y vtedy a len vtedy ak $sx| \vdash^* f|y$ je v M, pre každý stav $f \in F$.

Preklad definovaný M, $T(M)$, je

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

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

$$L_1(M) = \{ x: (x, y) \in T(M)\ pre\ každé\ y \in \Sigma_O^* \}$$

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

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

Lemma 3.5

Rodina prekladov definovaných konečným prevodníkom je rovnaká ako rodina prekladov definovaných syntaxou riadeným prekladom s pravidlami $A \to aB, \alpha B$, kde $A \in N, B \in N \cup \{ \varepsilon \}, a \in \Sigma \cup \{ \varepsilon \}, \alpha \in \Sigma^*$. [6]

Dôkaz

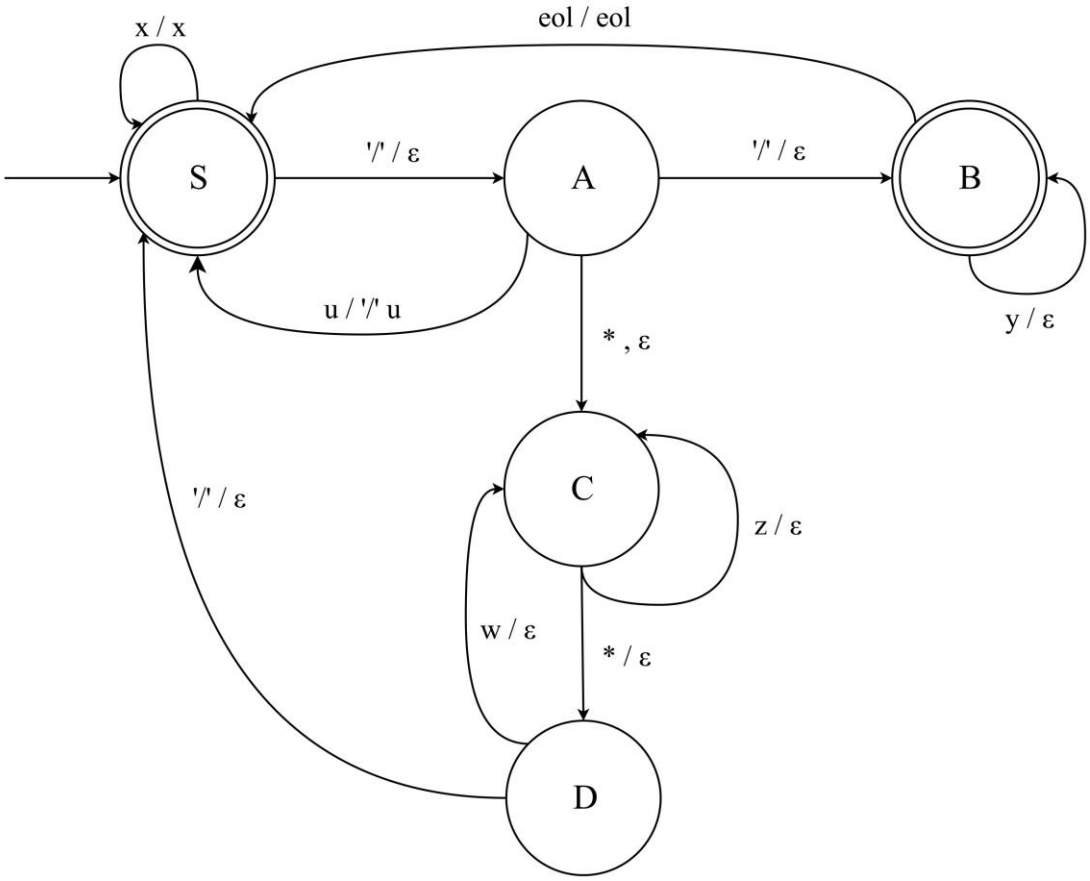
Uvažujme syntaxou riadený preklad $G = (N, \Sigma, \Delta, R, S)$ s pravidlami, ktoré spĺňajú podmienku z lemmy 3.5. Predpokladajme, že $N \cap \{ f \} = \emptyset$. Zostrojme konečný prevodník $M = (N \cup \{ f \}, \Sigma, \Delta, R, S, \{ f \})$. Ak $A \to aB, \alpha B$ je pravidlo z R, kde $A, B \in N, a \in \Sigma \cup \{ \varepsilon \}, \alpha \in \Delta^*$ potom pridaj $((A, a), (B, \alpha))$ do množiny R. Ak $A \to a, \alpha$ je pravidlo z R, kde $A \in N, a \in \Sigma \cup \{ \varepsilon \}, \alpha \in \Delta^*$ potom pridaj $((A, a), (f, \alpha))$ do R. K dokončeniu tejto časti dôkazu je potrebné dokázať, že každý preklad generovaný syntaxou riadeným prekladom G je taktiež prijímaný prevodníkom M. [6]

Príklad 3.1 – porovnanie syn. riadeného prekladu a kon. prevodníku

Uvažujme syntaxou riadený preklad $G = (N, \Sigma, \Delta, R, S)$ s pravidlami:

$S \to x\, S,$	$x\, S$	$\forall\, x \in \Sigma - \{ '/' \}$
$S \to / A,$	A	*možný začiatok komentáru*
$A \to / B,$	B	*začiatok riadkového komentáru* //
$A \to * C,$	C	*začiatok blokového komentáru* /*
$A \to u\, S,$	$/\, u\, S$	$\forall\, u \in \Sigma - \{ '*', '/' \}$ *nie je komentár*
$B \to y\, B,$	B	$\forall\, y \in \Sigma \{ eol \}$
$B \to eol\, S,$	$eol\, S$	*koniec riadkového komentáru*
$C \to * D,$	D	*možný koniec blokového komentáru*
$C \to z\, C,$	C	$\forall\, z \in \Sigma - \{ '*' \}$
$D \to / S,$	S	*koniec blokového komentáru* */
$D \to w\, C,$	C	$\forall\, w \in \Sigma - \{ '/' \}$

Takto definovaný syntaxou riadený preklad slúži na odstránenie riadkových a blokových komentárov v jazyku C zo vstupného reťazca. Stavový diagram prevodníku M prijímajúci tento preklad je znázornený na obrázku 3.6. [6]



3.6 Stavový diagram prevodníku M

Vzhľadom na definíciu konečného prevodníku, ktorú sme použili v úvodnej časti tejto kapitoly, takto definovaný prevodník pracuje nedeterministicky. To znamená, že môže spraviť niekoľko rôznych krokov z rovnakej konfigurácie. Preto je vhodné upresniť túto definíciu a upraviť ju tak, aby definovaný konečný prevodník pracoval deterministicky. [1] [11]

Definícia 3.6 – deterministický konečný prevodník

Nech $M = (Q, \Sigma, R, s, F)$ je konečný prevodník. M je deterministický ak každé pravidlo $r \in R$, kde ľavá strana pravidla $lhs(r) = pa$ a $a \in \Sigma_I \cup \{ \varepsilon \}$, spĺňa

$$\{r\} = \{ r' : r' \in R, pa = lhs(r') \; alebo \; p = lhs(r') \}$$

Nech $M = (Q, \Sigma, R, s, F)$ je deterministický konečný prevodník. Poznamenajme, že pre každý stav $q \in Q$ a $a \in \Sigma_I \cup \{ \varepsilon \}$ platí že, ak sa qa vyskytuje na ľavej strane pravidla r, tak žiadne ďalšie pravidlo v $R - \{r\}$ nemá ľavú stranu rovnú qa alebo q. Táto vlastnosť zaručuje, že prevodník M nemôže spraviť viac rôznych krokov z tej istej konfigurácie. Aj napriek tomu, že M je deterministický, môže preložiť ten istý vstup na viacero rôznych výstupných slov.

Príklad 3.2 – prázdny reťazec preložený na nekonečne veľa reťazcov

Uvažujme nasledujúce pravidlo deterministického prevodníku M definované nasledovne:

$$f \vdash f0$$

kde f je štartovacím a zároveň koncovým stavom. Prevodník M prevedie ε aplikovaním tohto pravidla na nekonečne mnoho slov

$$\varepsilon, 00, 000, \ldots$$

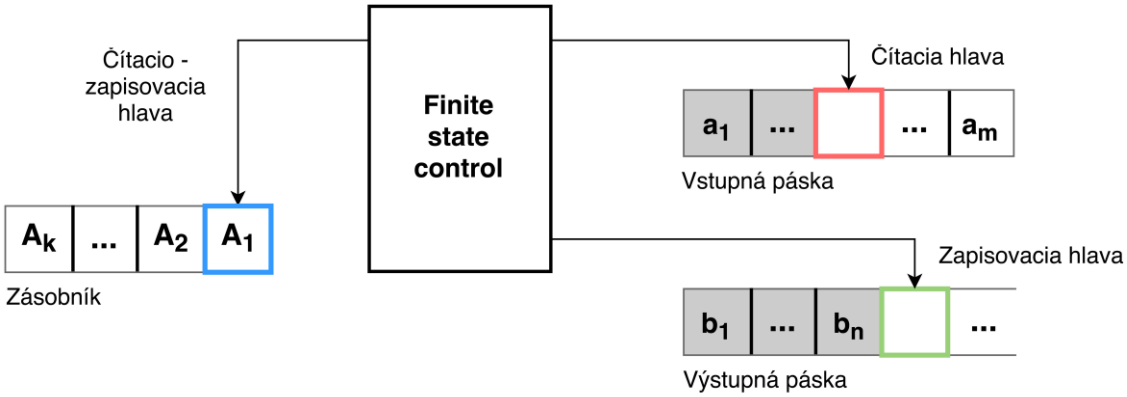
Formálne môžeme takýto preklad zapísať ako

$$T(M) = \{ (\varepsilon, 0^i) : i \geq 0 \}$$

4 Zásobníkové prevodníky

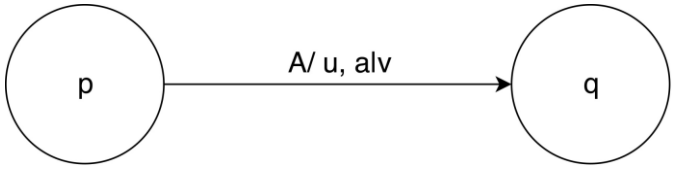
Zásobníkové prevodníky sú založené na zásobníkových automatoch. Rozdiel medzi zásobníkovými a konečnými prevodníkmi spočíva v tom , že obsahujú zásobník (Pozri obrázok 4.1). [8] Nasledujúci odstavec, definície a obrázky sú prevzaté z knihy [1].

Zásobník má rovnaký význam ako u zásobníkových automatov. Rolu teda hrá zásobník pri výpočtovom kroku. V jednom kroku zásobníkový prevodník zmení aktuálny stav, nahradí symbol na vrchole zásobníku, zo vstupu neprečíta žiaden alebo prečíta jeden vstupný symbol a zapíše výstupné slovo. Preklad končí ak sekvenciou krokov prevodník prečíta celý vstup, vyprázdni zásobník (po počiatočný symbol), skončí v jednom z koncových stavov a zapíše odpovedajúci výstup.



4.1 Zásobníkový prevodník

Grafické znázornenie zásobníkových prevodníkov je podobné ako u konečných prevodníkov. Jediným rozdielom je, že pri označení hrany je uvedený aj stav zásobníka, konkrétne je uvedený symbol ktorým sa má nahradiť znak na vrchole zásobníka (Pozri obrázok 4.2).



4.2 Grafická reprezentácia pravidla Apa ⊢ uqv

Definícia 4.1 – zásobníkový prevodník

Zásobníkový prevodník je pätica

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

kde

Q je konečná množina stavov

Σ je abeceda pre ktorú platí že, $\Sigma \cap \mathrm{Q} = \emptyset$ a $\Sigma = \Sigma_\mathrm{I} \cup \Sigma_\mathrm{O} \cup \Sigma_\mathrm{PD}$, kde Σ_I je vstupná abeceda, Σ_O je výstupná abeceda a Σ_PD je zásobníková abeceda, ktorá obsahuje počiatočný symbol S, nazývaný taktiež dno zásobníka

$R \subseteq \Sigma_\mathrm{PD} Q(\Sigma_\mathrm{I} \cup \{\, \varepsilon \,\} \times \Sigma_{PD}^* Q \Sigma_O^*$ je konečná relácia

$S \in Q$ je štartovací stav

$F \subseteq Q$ je množina koncových stavov

Prvky z množiny R nazývame pravidlá. Nech $(Apa, wqv) \in R$, kde $A \in \Sigma_{PD}, p, q \in Q, a \in \Sigma_I \cup \{\varepsilon\}, w \in \Sigma_{PD}^*$ a $v \in \Sigma_O^*$. Namiesto tohto zápisu (Apa, wqv) je vhodnejšie pravidlo zapisovať v tvare

$$r\colon Apa \vdash wqv$$

Definícia 4.2 – vstup zásobníkového prevodníku

Nech $M = (Q, \Sigma, R, s, F)$ je zásobníkový prevodník. Vstup prijímaný zásobníkovým automatom M_I je definovaný ako

$$M_I = (Q, \Sigma_I \cup \Sigma_\mathrm{PD}, R_I, s, F)$$

kde Σ_I je vstupná abeceda M, Σ_PD je zásobníková abeceda M a

$$\mathrm{R_I} = \{\, Aqa\ up\colon Aqa \Vdash upv \in R\ pre\ každé\ v \in \Sigma_O^* \,\}$$

Definícia 4.3 – konfigurácia

Konfigurácia zásobníkového prevodníku M je štvorica (q, x, α, y) kde

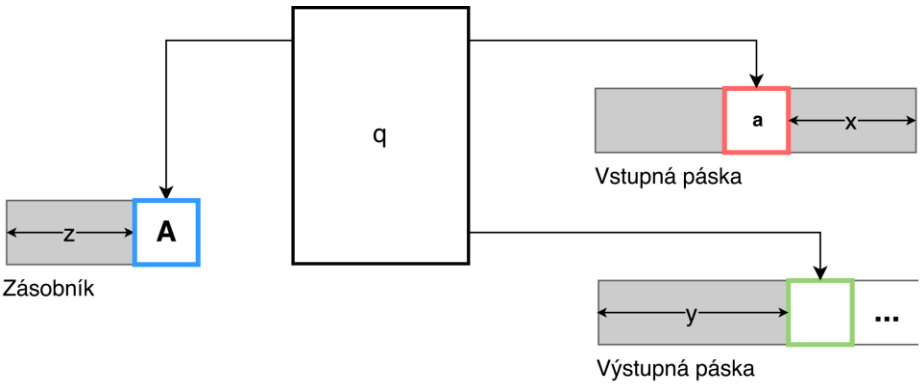
$q \in Q$ je aktuálny stav

$x \in \Sigma_I$ je zvyšná časť (suffix) vstupného slova

$\alpha \in \Sigma_{PD}$ je obsah zásobníka

$y \in \Sigma_O$ je generovaný prefix výstupného slova

Konfigurácia zásobníkového prevodníku teda predstavuje stav všetkých jeho častí (Pozri obrázok 4.3). Tak ako je to vo formálnej definícii ide o aktuálny stav v ktorom sa zásobníkový prevodník nachádza, zostávajúca časť reťazca na vstupe, aktuálny obsah zásobníka a zapísaná časť slova na výstupnej páske. Aplikovaním pravidiel prevodník prechádza medzi jednotlivými stavmi, číta vstup, mení obsah zásobníka a zaznamenáva výstup. V každom kroku teda prejde do novej konfigurácie. Ak zásobníkový prevodník M sekvenciou krokov prečíta celý vstup x, zapíše výstup y, vyprázdni zásobník a skončí v jednom z koncových stavov hovoríme, že úspešne preložil reťazec x na y. Formálne preklad popisuje definícia 2.0.1.



4.3 Konfigurácia

Definícia 4.4 – krok

Krok zásobníkového prevodníku M, označujeme symbolom $\vdash$

$$(q, ax, Z\gamma, y) \vdash (p, x, \alpha\gamma, yv)$$

kde $p, q \in Q; a \in (\Sigma_I \cup \{\varepsilon\}); x \in \Sigma_I^*; y, v \in \Sigma_O^*; Z \in (\Sigma_{PD} \cup \{\varepsilon\}); \alpha, \gamma \in \Sigma_{PD}^*$ a $(q, a, Z), (p, \alpha, v) \in R$. $\vdash^+$ a $\vdash^*$ sú definované ako tranzitívny a reflexívno-tranzitívny uzáver $\vdash$. [6]

Uvažujme krok podľa pravidla $r: Aqa \Vdash upv \in R$ kde $\{a, u, v\} \cap \{\varepsilon\} \neq \emptyset$. Ak $a = \varepsilon$, M neposunie svoju čítaciu hlavu. Ak $u = \varepsilon$, M odstráni symbol A zo zásobníka. Ak $y = \varepsilon$, M neposunie svoju zapisovaciu hlavu. Nasledujúci príklad lepšie popisuje krok zásobníkového prevodníku. Ide teda o prechod z jednej konfigurácie do inej.

Príklad 4.1 – zmena konfigurácie

Uvažujme zásobníkový prevodník M a jeho konfiguráciu

$$S^*A + qa|aa$$

a pravidlo

$$r: +q \vdash q+$$

Aplikovaním tohto pravidla, M spraví krok

$$S^*A + qa|aa \vdash S^*Aqa|aa +$$

Definícia 4.5 – preklad definovaný zásobníkovým prevodníkom

Nech $M = (Q, \Sigma, R, s, F)$ je zásobníkový prevodník, $x \in \Sigma_I^*$ a $y \in \Sigma_O^*$. M preloží x na y vtedy a len vtedy ak

$$Ssx| \vdash^* zf|y$$

je v M, kde $y \in \Sigma_{PD}^*$ a $f \in F$. Preklad definovaný prevodníkom M, $T(M)$, je

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

Vstupný jazyk $L_I(M)$ odpovedajúci $T(M)$ je definovaný ako

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

Výstupný jazyk $L_O(M)$ odpovedajúci $T(M)$ je definovaný ako

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

Zásobníkový prevodník, ktorý sme si formálne zadefinovali na začiatku tejto kapitoly však pracuje nedeterministicky. To znamená, že môže spraviť viacero rôznych krokov z tej istej konfigurácie. Pre praktické účely je lepšie pracovať s jeho deterministickou verziou. Preto je vhodné si definíciu č. 4.1 rozšíriť a zadefinovať si deterministický zásobníkový prevodník.

Definícia 4.6 – deterministický zásobníkový prevodník

Nech $M = (Q, \Sigma, R, s, F)$ je zásobníkový prevodník. M pracuje deterministicky ak každé pravidlo $r \in R$, s ľavou stranou $lhs(r) = Apa$ kde $a \in \Sigma_I \cup \{ \varepsilon \}$, spĺňa túto vlastnosť

$$\{r\} = \{ r': r' \in R, Apa = lhs(r') \; alebo \; Ap = lhs(r') \}$$

Pre každý takýto stav $q \in Q$, $A \in \Sigma_{PD}$, $a \in \Sigma_I \cup \{ \varepsilon \}$, ak existuje pravidlo r s ľavou stranou Aqa, žiadne ďalšie pravidlo nemá svoju ľavú stranu rovnú Aqa alebo Aq. V praxi to znamená, že pri preklade akéhokoľvek vstupu, deterministický prevodník M spraví jedinečnú postupnosť krokov. Práve táto vlastnosť zjednodušuje implementáciu zásobníkového prevodníku M.

5 Prekladové gramatiky

Bezkontextová gramatika a zásobníkové prevodníky definujú preklad lepšie ako formálne jazyky. Základom celého prekladu sú prekladové gramatiky. Jedná sa o modifikáciu bezkontextovej gramatiky. Rozdiel spočíva v tom, že každá produkcia (tvorba) v prekladovej gramatike G sa skladá z dvoch slov na jej pravej strane. Gramatika G teda generuje dvojicu slov, ktoré patria prekladu definovaným touto gramatikou G. Táto kapitola obsahuje základné definície zaoberajúce sa problematikou prekladových gramatík. Na konkrétnych príkladoch je vysvetlený preklad definovaný práve týmito gramatikami. Definície a príklady sú prevzaté z knihy [1].

Definícia 5.1 – prekladová gramatika

Prekladová gramatika je štvorica

$$G = (N, T, P, S)$$

kde

N je abeceda neterminálov

T je abeceda taká, že $T \cap N = \emptyset$ a $T = T_I \cup T_O$, kde T_I je vstupná abeceda a T_O je výstupná abeceda

P je konečný súbor prepisovacích pravidiel v tvare

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

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

$S \in N$ je počiatočný (štartovací) symbol

Príklad 5.1 – prekladová gramatika

Uvažujme prekladovú gramatiku G definovanú nasledujúcimi šiestimi pravidlami:

1. $\langle \text{expression} \rangle \twoheadrightarrow \langle \text{expression} \rangle + \langle \text{term} \rangle | \langle \text{expression} \rangle \langle \text{term} \rangle +$
2. $\langle \text{expression} \rangle \twoheadrightarrow \langle \text{term} \rangle | \langle \text{term} \rangle$
3. $\langle \text{term} \rangle \twoheadrightarrow \langle \text{term} \rangle^* \langle \text{factor} \rangle | \langle \text{term} \rangle \langle \text{factor} \rangle^*$
4. $\langle \text{term} \rangle \twoheadrightarrow \langle \text{factor} \rangle | \langle \text{factor} \rangle$
5. $\langle \text{factor} \rangle \twoheadrightarrow (\langle \text{expression} \rangle) | \langle \text{expression} \rangle$
6. $\langle \text{factor} \rangle \twoheadrightarrow a | a$

Prekladová gramatika G má tri neterminály. Konkrétne ide o neterminály $\langle \text{expression} \rangle$, $\langle \text{term} \rangle$ a $\langle \text{factor} \rangle$. Štartovacím symbolom gramatiky G je neterminál $\langle \text{expression} \rangle$. Vstupná abeceda gramatiky pozostáva z piatich symbolov - $+, *, (,), a$. Výstupná abeceda obsahuje tri symboly - $+, *$ a symbol a. Nech $G = (N, T, P, S)$ je prekladová gramatika. Použitím pravidiel z P derivuje dvojicu slov na inú dvojicu slov. Presnejšie, uvažujme dve slová, $u_1 A u_2$ a $v_1 A v_2$. Obidve slová majú pre jednoduchosť rovnaký počet neterminálov. Nech $p \in P$ s $lhs(p) = A$. Použitím pravidla p gramatika G prepíše $u_1 A u_2 | v_1 A v_2$ na $u_1 irhs(p) u_2 | v_1 orhs(p) v_2$.

Definícia 5.2 – priama derivácia

Nech $G = (N, T, P, S)$ je prekladová gramatika, $p \in P$ a $x, y, u, v \in (N \cup T)^*$. Počet neterminálov nachádzajúcich sa v x a u je rovnaký. V takomto prípade môžeme povedať, že $xlhs(p)y | ulhs(p)v$ priamo derivuje $xirhs(p)y | uorhs(p)v$ podľa pravidla p z gramatiky G, symbolicky zapísané takto:

$$xlhs(p)y | ulhs(p)v \Rightarrow xirhs(p)y | uorhs(p)v \qquad [p]$$

alebo v skrátenom tvare

$$xlhs(p)y | ulhs(p)v \Rightarrow xirhs(p)y | uorhs(p)v$$

Príklad 5.2 – priama derivácia

Uvažujme prekladovú gramatiku popísanú v predchádzajúcom príklade. Gramatika G spraví jeden derivačný krok pomocou priamej derivácie nasledovne:

$$\langle \text{factor} \rangle^* \langle \text{term} \rangle | \langle \text{factor} \rangle \langle \text{term} \rangle^* \Rightarrow (\langle \text{expression} \rangle)^* \langle \text{term} \rangle | \langle \text{expression} \rangle \langle \text{term} \rangle^* \quad [5]$$

kde

$$5: \langle \text{factor} \rangle \twoheadrightarrow (\langle \text{expression} \rangle) | \langle \text{expression} \rangle$$

Aplikovaním uvedeného pravidla č.5, sa výraz zapísaný na ľavej strane prevedie na odpovedajúci výraz vpravo. Pre lepšie pochopenie uvedieme ešte jeden príklad. Tentokrát bude, na vzniknutú pravú stranu z predchádzajúceho príkladu, aplikované iné pravidlo. Konkrétne pôjde o pravidlo č.2, ktoré má tvar:

$$2: \langle \text{expression} \rangle \twoheadrightarrow \langle \text{term} \rangle | \langle \text{term} \rangle$$

Jeho použitím dostávame

$$(\langle \text{expression} \rangle)^* \langle \text{term} \rangle | \langle \text{expression} \rangle \langle \text{term} \rangle^* \Rightarrow (\langle \text{term} \rangle)^* \langle \text{term} \rangle | \langle \text{term} \rangle \langle \text{term} \rangle^* \quad [2]$$

Tento príklad však popisuje stav, keď je konkrétne pravidlo použité iba raz, to znamená pre prípad $\Rightarrow^n$, kde $n = 1$. Pri preklade je však nutné používať prepisovacie pravidlá viac krát, preto je nutné rozšíriť definíciu priamej derivácie pre prípad, kde $n \geq 0$, tak ako je to popísané v ďalšej definícii.

Definícia 5.3 – derivácia

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

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

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

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

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

kde $p_i \in P$. Potom gramatika G spraví n derivačných krokov z $u_0|v_0$ do $u_n|v_n$ podľa pravidiel $p_1 \ldots p_n$, formálne zapísané ako

$$u_0|v_0 \Rightarrow^0 u_n|v_n \qquad [p_1 \ldots p_n]$$

Uvažujme prekladovú gramatiku G. Nech $u_0|v_0 \Rightarrow^n u_n|v_n$ $[\pi]$ v G, kde π označuje postupnosť n pravidiel z P ($\pi = \varepsilon$ ak $n = 0$). Treba si všimnúť, že π korešpondujúce k $u_0|v_0 \Rightarrow^n u_n|v_n$, reprezentuje postupnosť pravidiel podľa ktorej G spraví $0 \ldots n$ krokov. V prípade, že nezáleží na reprezentácií je možné vo formálnom zápise symbol π vynechať.

Definícia 5.4 – preklad definovaný prekladovou gramatikou

Nech $G = (N, T, P, S)$ je prekladová gramatika. Ak gramatika G obsahuje pravidlo $S|S \Rightarrow^* u|v$, kde $u \in T_I^*$ a $v \in T_O^*$, hovoríme, že G preloží u na v. Preklad definovaný gramatikou G, $T(G)$, je

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

Príklad 5.3 – preklad

Uvažujme gramatiku G, ktorú sme si zadefinovali v príklade č. 5.1. Gramatika teda bude obsahovať spomínaných šesť pravidiel. Ich postupným aplikovaním gramatika G preloží reťazec $(a + a)^*$ na $aa + a^*$. V nasledujúcom príklade si teda ukážeme preklad reťazca zapísaného v infixovej aritmetickej notácii na odpovedajúci postfixový výraz.

$\langle \text{expression} \rangle | \langle \text{expression} \rangle$
$\Rightarrow \langle \text{term} \rangle | \langle \text{term} \rangle$
$\Rightarrow \langle \text{term} \rangle^* \langle \text{factor} \rangle | \langle \text{term} \rangle \langle \text{factor} \rangle^*$
$\Rightarrow \langle \text{factor} \rangle^* \langle \text{factor} \rangle | \langle \text{factor} \rangle \langle \text{factor} \rangle^*$
$\Rightarrow (\langle \text{expression} \rangle)^* \langle \text{term} \rangle | \langle \text{expression} \rangle \langle \text{term} \rangle^*$
$\Rightarrow (\langle \text{expression} \rangle + \langle \text{term} \rangle)^* \langle \text{term} \rangle | \langle \text{expression} \rangle \langle \text{term} \rangle + \langle \text{term} \rangle^*$
$\Rightarrow (\langle \text{term} \rangle + \langle \text{term} \rangle)^* \langle \text{term} \rangle | \langle \text{term} \rangle \langle \text{term} \rangle + \langle \text{term} \rangle^*$

$\Rightarrow$ $(\langle \text{factor} \rangle + \langle \text{term} \rangle)^* \langle \text{term} \rangle | \langle \text{factor} \rangle \langle \text{term} \rangle + \langle \text{term} \rangle^*$

$\Rightarrow$ $(a + \langle \text{term} \rangle)^* \langle \text{term} \rangle | a \langle \text{term} \rangle + \langle \text{term} \rangle^*$

$\Rightarrow$ $(a + \langle \text{factor} \rangle)^* \langle \text{term} \rangle | a \langle \text{factor} \rangle + \langle \text{term} \rangle^*$

$\Rightarrow$ $(a + a)^* \langle \text{term} \rangle | aa + \langle \text{term} \rangle^*$

$\Rightarrow$ $(a + a)^* \langle \text{factor} \rangle | aa + \langle \text{factor} \rangle^*$

$\Rightarrow$ $(a + a)^* a | aa + a^*$

6 Prevodníkové systémy

Koncept prevodníkových systémov je založený na princípoch jednoduchých zásobníkových prevodníkov. Práve spojením viacerých prevodníkov vzniká prevodníkový systém. Výhodou prevodníkových systémov je, že môžu pracovať paralelne a zrýchliť tak proces prekladu. Samozrejme paralelné spracovanie vstupu nie je podmienkou. Pre jednoduchosť uvažujeme prevodníkové systémy zložené z dvoch prevodníkov. Počet použitých prevodníkov v danom systéme však môže byť aj omnoho väčší. V našom prípade sme uvažovali dva druhy prevodníkových systémov, ktorých princípy, formálne definície ako aj ich možnosti využitia sú popísané v nasledujúcej časti.

Definícia 6.0.1 – prevodníkový systém

Prevodníkový systém je množina prevodníkov:

$$\Gamma = (M_i \dots M_n)$$

Kde $M_i \dots M_n = (Q_i, \Sigma_i, R_i, s_i, F_i)$, kde $1 \leq i \leq n$, pre nejaké $n \geq 1$

Ďalej pre prevodníky $M_i \dots M_n$ platí, že:

$$\text{Vstupná abeceda } \Sigma_{IMi} = \Sigma_{IMn}$$

$$\text{Výstupná abeceda } \Sigma_{OMi} \neq \Sigma_{OMn}$$

Definícia 6.0.2 – preklad

Preklad definovaný prevodníkovým systémom Γ:

$$T(\Gamma) = \{ (x, y) \mid x \in \Sigma_{I\Gamma}, y \in \Sigma_{O\Gamma}, (x, y) \in T(\Gamma) \}$$

Vstupný jazyk:

$$L_I\big(T(\Gamma)\big) = \{ x \mid (x, y) \in T(\Gamma) \}$$

Výstupný jazyk:

$$L_O\big(T(\Gamma)\big) = \{ y \mid (x, y) \in T(\Gamma) \}$$

Definícia 6.0.3 – vstupy systému

Vstupom prevodníkového systému Γ_1 a Γ_2, je reťazec z. Formálne môžeme vstup definovať takto:

$$I = \{ z \mid (x, z) \in T(\Gamma_1), (y, z) \in T(\Gamma_2), z \in I^* \}$$

6.1 Prevodníkový systém č.1

Definícia 6.1.1 – výstup systému

Prevodníkový systém č.1, označme ho Γ_1, pracuje nad vstupným reťazcom z. Prevodník M_1 vytvára z reťazca z výstupný reťazec x. Druhý prevodník M_2 generuje výstupný reťazec y. Výstupom prevodníkového systému Γ_1 je konkatenácia reťazcov x a y. Formálne zapísané nasledovne:

$$O_1 = \{ x.y \mid (z,x) \in T(\mathrm{M}_1), (z,y) \in T(\mathrm{M}_2), z \in I^* \}$$

6.2 Prevodníkový systém č.2

Definícia 6.1.2 – výstup systému

Prevodníkový systém č.2, označme ho Γ_2, pracuje nad vstupným reťazcom z. Prevodník M_1 vytvára z reťazca z výstupný reťazec x. Druhý prevodník M_2 generuje výstupný reťazec y. Výstupom prevodníkového systému Γ_2 je iba reťazec x. Formálny zápis prevodníkového systému Γ_2:

$$O_2 = \{ x \mid (z,x) \in T(\mathrm{M}_1), (z,y) \in T(\mathrm{M}_2), z \in I^* \}$$

6.3 Možnosti využitia prevodníkových systémov

Príklad 6.1 – prevod postfixovej notácie na infixovú

Uvažujme prevodníkový systém $\Gamma_1 = (M_1, M_2)$. Obidva prevodníky M_1 aj M_2 majú dva stavy q, f, kde q je štartovací stav a f je koncový. Vstupná abeceda Σ_I oboch prevodníkov pozostáva zo symbolov a, $+$, $*$. Výstupná abeceda prevodníku M_1, Σ_{OM1} obsahuje symboly a, $+$, $*$ a výstupná abeceda prevodníku M_2 obsahuje iba symbol ε. Zásobníková abeceda prvého prevodníku pracuje so symbolmi E, $+$, $*$, kde E je počiatočný symbol na zásobníku. Zásobníková abeceda druhého prevodníku M_2 pracuje iba so spomínaným počiatočným symbolom E.

Množina pravidiel pre prevodník M_1:

1. $Eqa \vdash \varepsilon qa$
2. $Eq+ \vdash EE+q\varepsilon$
3. $Eq* \vdash EE*q\varepsilon$
4. $+q\varepsilon \vdash \varepsilon q+$
5. $*q\varepsilon \vdash \varepsilon q*$
6. $\varepsilon q\varepsilon + \vdash \varepsilon f\varepsilon$

Množina pravidiel pre prevodník M_2:

1. $Eqa \vdash Eq\varepsilon$
2. $Eq+ \vdash Eq\varepsilon$
3. $Eq* \vdash Eq\varepsilon$
4. $Eq\varepsilon \vdash Eq\varepsilon$
5. $\varepsilon q\varepsilon \vdash \varepsilon f\varepsilon$

Na vstupe obidvoch prevodníkov je reťazec v prefixovej poľskej aritmetickej notácii $+*aaa$. Aplikáciou vyššie uvedených pravidiel prevodník M_1 prevedie výraz z prefixovej notácie na korešpondujúci postfixový výraz $aa*a+$. Výsledkom prevodu prevodníkom M_2 bude prázdny reťazec ε. Čo v konečnom dôsledku demonštruje prevodníkový systém č. 1, ktorého výsledkom je konkatenácia výstupných reťazcov prevodníkov M_1 a M_2.

Príklad 6.2 – preklad reťazca

Ďalšou možnosťou využitia prevodníkových systémov je prevod reťazca zapísaného v cudzom jazyku na odpovedajúci reťazec v inom jazyku viď. nasledujúci príklad. Majme prevodníkový systém $\Gamma_1 = (M_1,\ M_2)$. Spoločná vstupná abeceda tentokrát nech obsahuje symboly $T, R, A, N, S, D, U, C, E, R$. Výstupné abecedy sú definované nasledovne: $\Sigma_{OM1} = P, R, E, V, O, D, N, I, K$ a $\Sigma_{OM2} = A$. Zásobníková abeceda pre oba prevodníky $\Sigma_{PD} = S, E$ kde S je štartovací symbol na zásobníku. Množina stavov obsahuje stavy s, q, f kde s je štartovacím stavom a stav f je koncovým.
Množina pravidiel pre prevodník M_1:

1. $SsT \vdash SEqP$
2. $EqR \vdash EqR$
3. $EqA \vdash EqE$
4. $EqN \vdash EqV$
5. $EqS \vdash EqO$
6. $EqD \vdash EqD$
7. $EqU \vdash EqN$
8. $EqC \vdash EqI$
9. $EqE \vdash \varepsilon qK$
10. $SqR \vdash \varepsilon f\varepsilon$

Množina pravidiel pre prevodník M_2:

1. $SsT \vdash SEqA$
2. $EqR \vdash EqA$
3. $EqA \vdash EqA$

4. $EqN \vdash EqA$
5. $EqS \vdash EqA$
6. $EqD \vdash EqA$
7. $EqU \vdash EqA$
8. $EqC \vdash EqA$
9. $EqE \vdash \varepsilon qA$
10. $SqR \vdash \varepsilon f$A

V tomto prípade majú obidva prevodníky na vstupe opäť rovnaký reťazec. Výsledkom prekladu prevodníku M_1 je reťazec $PREVODNIK$. Prevodník M_2 prevedie vstupný reťazec na postupnosť symbolov A. Nakoľko sa jedná o výstup prevodníkového systému č. 2, kde sa za výsledok prekladu považuje iba výstup prevodníku M_1, výstupom bude iba reťazec $PREVODNIK$.

Príklad 6.3 – prevod ASCII hodnoty

Nasledujúci príklad demonštruje ďalšiu z možností využitia prevodníkových systémov, konkrétne sa jedná o prevodníkový systém č. 1. Tento systém tentokrát bude prevádzať znak na jeho odpovedajúcu ASCII hodnotu v hexadecimálnej podobe. Vstupný reťazec bude pre zjednodušenie obsahovať iba jeden znak @. Výsledkom prekladu by teda mal byť reťazec $0x40$. Vstupná abeceda obsahuje symbol @. Výstupná abeceda prevodníku M_1 bude obsahovať symboly $0, x$. Výstupná abeceda prevodníku M_2 bude zložená zo znakov $0, 4$. Obidva prevodníky budú mať iba dva stavy, stav s ako počiatočný, stav f ako koncový, jedno pravidlo a zásobníkovú abecedu zloženú z počiatočného symbolu S.
Pre prevodník M_1 bude pravidlo v tvare:

1. $Ss@ \vdash \varepsilon f0x$

Podobne aj pre prevodník M_2 bude pravidlo definované nasledovne:

1. $Ss@ \vdash \varepsilon f40$

Aplikáciou pravidiel a následnou konkatenáciou vzniknutých reťazcov dostávame na výstupe prevodníkového systému reťazec $0x40$ presne tak, ako sme na začiatku očakávali.

Príklad 6.4 – šifrovanie

Praktické využitie by prevodníkové systémy mohli nájsť aj šifrovaním rôznych vstupných reťazcov. V takom prípade, by jednotlivé výstupy boli zapisované na základe náhodne vygenerovaných pravidiel pre daný vstupný symbol v závislosti aktuálneho symbolu na zásobníku. Postupnosť symbolov na zásobníku by bola zároveň aj šifrovacím kľúčom, ktorý by bol použitý pri spätnom odšifrovaní reťazca.

7 Programová časť

Pre tvorbu programu bol zvolený programovací jazyk Python 3.5.1. Vývoj aplikácie prebiehal vo vývojovom prostredí PyCharm od spoločnosti IntelliJ. To bolo zvolené na základe predchádzajúcich skúseností a komplexnej podpore nástrojov. Pre tvorbu grafického rozhrania a vizualizácie celej aplikácie bola použitá voľne dostupná knižnica TkInter. [12] Táto knižnica poskytuje jednoduchý a interaktívny vývoj GUI pre aplikácie naprogramované v spomínanom jazyku Python. Grafy konečných automatov, aplikované pravidlá ako aj priebežný stav jednotlivých prevodníkov boli vykresľované s použitím knižnice Graphviz. [13] Pre korektné spustenie aplikácie je potrebné mať nainštalované všetky spomenuté knižnice.

7.1 Návrh aplikácie

Celá aplikácia je rozdelená do dvoch základných celkov. Back-end, ktorý má na starosti načítanie a spracovanie vstupných pravidiel a realizáciu prekladu vstupného reťazca. Rieši tak celú logiku aplikácie. Druhou časťou je Front-end, ktorý sa stará o vizualizáciu výsledkov a interakciu užívateľa s celou aplikáciou. Tieto celky sú ďalej rozdelené na moduly a submoduly, ktoré zabezpečujú jednoduchú údržbu, vývoj a testovanie aplikácie. Jednotlivé moduly a ich funkcionalita sú bližšie popísané v nasledujúcej časti.

7.2 Back-end

Back-end pozostáva z troch základných častí:

- získanie a uloženie vstupného reťazca, ktorý bol užívateľom zadaný
- spracovania pravidiel potrebných pre prevod reťazca
- realizáciu prekladu vstupného na výstupný reťazec na základe vopred definovaných pravidiel

7.2.1 Vstupy programu

Pre uskutočnenie prekladu je užívateľ nútený zadať vstupný reťazec, ktorý chce preložiť. Ďalej je potrebné otvoriť v aplikácii súbor, kde sú presne definované pravidlá pre prevod vstupného reťazca na odpovedajúci výstupný reťazec. Užívateľ musí v súbore definovať pravidlá pre oba prevodníky, ktoré dokopy vytvárajú prevodníkový systém. Pre jednoduchú a prehľadnú definíciu pravidiel prekladu bol zvolený značkovací jazyk XML. Jednotlivé značky však boli upravené pre potreby programu a ich význam je popísaný v prehľade nižšie.

Prehľad značiek a ich význam:

<system> - definuje prevodníkový systém ako celok, tvorí koreňový uzol dokumentu

<transducer1> - definuje stavy a pravidlá pre prevodník č. 1

<transducer2> - definuje stavy a pravidlá pre prevodník č. 2

<state> - definuje konkrétny stav

<rule> - definuje pravidlo

<lside> - definuje ľavú stranu pravidla

<rside> - definuje pravú stranu pravidla

7.2.2 Atribúty

Všetky z uvedených značiek nemajú textový obsah. Pravidlá sú definované len pomocou atribútov, ktoré sú uvedené pri konkrétnych značkách. Táto technika bola použitá z dôvodu jednoduchého získania hodnôt atribútov. Na ich získanie bola použitá voľne dostupná knižnica `xml.etree.ElementTree`. [14] Všetky atribúty uvedené v prehľade nižšie sú povinné. Nemusia však obsahovať žiaden symbol. V takomto prípade je ich hodnota reprezentovaná malým písmenom gréckej abecedy ε, ktorý popisuje v terminológií formálnych jazykov prázdny reťazec (reťazec nulovej dĺžky). [15]

Značka `<state>` má povinné atribúty `name`, `start` a `end`. Kde atribút `name` vyjadruje meno stavu. Atribúty `start` a `end` určujú typ stavu a môžu nadobúdať hodnoty `T` alebo `F`, kde `T` značí booleovskú hodnotu `True` a `F` vyjadruje hodnotu `False`. V prípade, že sa jedná o štartovací stav, je atribút `start` nastavený na `T` a atribút `end` nastavený na `F`. Komplementárne môžeme označiť koncový stav. V prípade, že stav nie je ani štartovací ale ani koncový, obidva atribúty sú nastavené na hodnotu `F`, teda `False`. Nasledujúci zápis popisuje stav *Q*, ktorý je štartovacím stavom:

```
<state name="Q" start="T" end="F">
```

Ďalšími značkami s atribútmi sú značky `<lside>`, `<rside>` popisujúce ľavú alebo pravú stranu pravidla `<rule>`. Ľavá strana pravidla má dva atribúty a to atribút `in` a `stack`, kde `in` vyjadruje znak na vstupe a `stack` vyjadruje symbol na zásobníku, ktorý sa bude aplikovaním pravidla čítať a nahrádzať pravou stranou pravidla. Pravá strana pravidla má atribúty `state`, `out` a `stack`. `Out` obsahuje znak ktorý sa zapíše na výstup. `State` obsahuje meno stavu do ktorého prejdeme a `stack` ako v predchádzajúcom prípade označuje zásobník. Tentokrát však obsahuje znak, ktorým sa nahradí symbol na vrchole zásobníka. Vo vstupnom súbore by ich zápis mohol vyzerať nasledovne:

```
<lside in="a" stack="">
```

Na vstupe je symbol a, zásobník je prázdny

```
<rside state="X" out="r" stack="">
```

Prejdeme do stavu *X*, na výstup sa zapíše symbol *r* a symbol na vrchole zásobníku sa odstráni.

7.2.3 Štruktúra

Zanorenie a štruktúru celého XML dokumentu je možné popísať jazykom DTD. Ide o tzv. Document Type Definition, teda jazyk, ktorý presne popisuje zanorenie jednotlivých xml značiek (angl. tagov). Ďalej udáva povinnosť zatvárať značky koncovou značkou. V našom prípade sa vo vstupnom dokumente používajú iba tagy s povinnou počiatočnou aj koncovou značkou. DTD nám zaručuje správny formát XML súboru, ktorého validitu môžeme na základe takto definovanej štruktúry dokumentu, druhov použitých značiek a ich atribútov ľahko overiť. DTD pre XML súbor s použitím nami definovaných značiek teda vyzerá nasledovne:

```
<!DOCTYPE system
[
<!ELEMENT system (transducer1, transducer2)>
<!ELEMENT transducer1 (state)>
<!ELEMENT transducer2 (state)>
<!ELEMENT state (rule)>
<!ELEMENT rule (lside, rside)>
]>
```

7.3 Spracovanie vstupného súboru

7.3.1 Trieda XMLParser

Spracovanie vstupného súboru má na starosti trieda `XMLParser`. Na začiatku sa vstupný súbor analyzuje volaním statickej metódy `parse_xml`, ktorá má jeden povinný argument a to súbor vo formáte XML. Metóda vytvorí pomocou funkcií z knižnice `xml.etree.ElementTree` strom zo vstupného súboru a vráti koreňový uzol (`root`), ktorý je predpokladom pre ďalšie kroky analýzy. Po získaní koreňového prvku dokumentu je potrebné vytvoriť tabuľku. Tabuľka je inštanciou triedy `MachineTable`. Volaním metódy `createTable` sa vytvorí tabuľka, zistia sa všetky stavy pomocou metódy `get_states`, ktoré sú uložené v poli. Tabuľky sa vytvoria pre obidva prevodníky. Následne sa pole stavov pre každý prevodník prejde v cykle a zistia sa pre každý stav všetky pravidlá patriace danému stavu. Stav spolu s pravidlami sú uložené do jedného riadku tabuľky. V triede sú implementované ďalšie pomocné metódy. Ide o metódy `get_state_rules`, `get_left_side` a `get_right_side`. Metóda `get_state_rules` vracia pole pravidiel patriace danému stavu. `Get_left_side` a `get_right side` slúžia na jednoduchú prácu s pravidlami. Rozdeľujú pravidlo na ľavú a pravú stranu, ktorú vracajú.

7.3.2 Trieda MachineTable

Výsledkom analýzy súboru s pravidlami je dátová štruktúra vo forme tabuľky. Tabuľka pozostáva z riadkov, kde každý riadok tabuľky obsahuje stav a zoznam pravidiel patriacich danému stavu. Riadky je možné do tabuľky jednoducho pridávať volaním metódy `add_row`. Trieda ďalej obsahuje metódu pre zistenie štartovacieho stavu analýzy. Tento stav je kľúčovým pre úspešné zahájenie celého procesu prekladu. V prípade, že štartovací stav v súbore so vstupnými pravidlami prekladu neexistuje, preklad nebude uskutočnený. Užívateľ bude upozornený chybovým hlásením a chýbajúcom štartovacom stave. Štartovací stav musia mať uvedené obidva prevodníky, ktoré majú preklad reťazca realizovať.

7.3.3 Trieda MachineState

Reprezentuje stav zásobníkového prevodníku. Každý stav získaný rozparsrovaním XML súboru a uložením do tabuľky má svoje meno, typ a pravidlá. Typ určuje či ide o stav štartovací, koncový alebo normálny. V prípade, že štartovací stav je zároveň aj koncovým bola zavedená špecifická identifikácia stavu typom `both`. Táto trieda má pomocný charakter a obsahuje iba metódy na získanie popísaných atribútov objektu, ktorý je inštanciou tejto triedy.

7.3.4 Trieda MachineLogic

Táto trieda obsahuje metódy, ktoré riešia celú logiku prevodu. Prevod sa realizuje postupným čítaním vstupného reťazca, prechádzaním medzi stavmi na základe získanej množiny pravidiel a generovaním príslušného výstupu. Celý prevod končí v prípade, že dočítame celý vstup, vyprázdnime zásobník a skončime v jednom z definovaných koncových stavov.

Pre úspešnú inicializáciu je potrebná tabuľka stavov s pravidlami a vstupný reťazec, ktorý užívateľ zadal pred začiatkom prekladu. Následne sa z tabuľky zistia štartovacie stavy prevodníkov. Tie sa stanú štartovacími stavmi celého prekladu. Inicializuje sa aj obsah zásobníka a výstup, ktorý je na začiatku prevodu prázdny. Taktiež sa vytvorí prázdny graf, ktorý bude priebeh prekladu vizualizovať. Celý priebeh vizualizácie a jednotlivé zobrazované prvky sú bližšie popísané v kapitole 7.5.

7.4 Preklad

Preklad má na starosti metóda `logic_step`. Táto metóda je volaná pre obidva prevodníky. Prevádza vstupný reťazec na výstup po jednotlivých krokoch. Celý preklad je tak možné postupne krok po kroku odkrokovať. V každom kroku sa načíta počiatočný znak vstupného reťazca. Zistia sa všetky pravidlá pre aktuálny stav. Na začiatku je aktuálny stav nastavený na štartovací stav, ktorý bol nájdený a nastavený pri inicializácií. V prípade, že pre daný stav žiadne pravidlo neexistuje, preklad zahlási chybu a končí. V opačnom prípade sa zo získaných pravidiel vyberie to, ktoré vyhovuje a bude aplikované. Po nájdení správneho pravidla dochádza k aplikovaniu pravidla presne podľa formálnej definície.

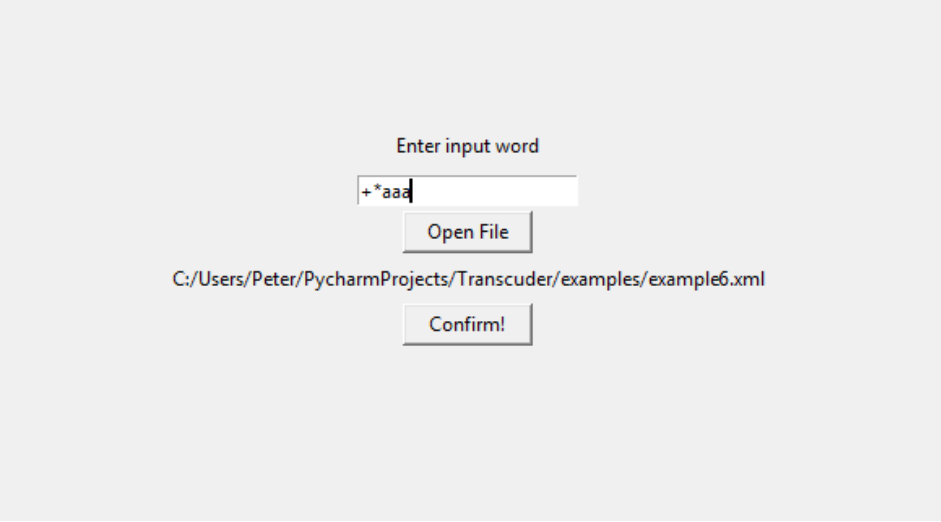
Najprv sa aplikuje jeho ľavá časť, kde sa prečíta znak zo vstupu a z vrcholu zásobníka sa vyberie uvedený symbol. Potom sa aplikuje pravá strana pravidla. Na výstup sa zapíše znak, obdobne na vrchol zásobníka sa pridá uvedený obsah a prevodník prejde do ďalšieho stavu. Prechod do ďalšieho stavu sa realizuje nastavením aktuálneho stavu na stav uvedený v pravej strane pravidla. V ďalšom kroku sa opäť bude volať metóda `logic_step` a pokračovať sa bude už od zmeneného aktuálneho stavu. Na konci každého kroku sa volajú metódy pre vizualizáciu prekladu, ktoré majú reflektovať uvedené zmeny.

7.5 Front – end

Program obsahuje jednoduché grafické užívateľské rozhranie. Celé GUI je zložené z dvoch obrazoviek. Úvodná obrazovka slúži k zadaniu príslušných vstupov potrebných pre realizovanie prekladu. V druhom okne, ktoré je užívateľovi zobrazené po potvrdení vstupov, prebieha vizualizácia prekladu. Užívateľovi je zobrazený celkový priebeh prekladu vstupného reťazca na odpovedajúci výstup. Rozloženie prvkov, ich popis a funkcionalita je popísaná v nasledujúcej podkapitole.

7.5.1 Prvky GUI a ich rozloženie

Po spustení programu je užívateľovi zobrazená úvodná obrazovka. Tá obsahuje vstupné pole pre zadanie vstupného reťazca. Ďalej sa tam nachádza tlačidlo „Open File“ pre otvorenie a nahranie súboru s definovanými pravidlami pre jednotlivé prevodníky. Po nahratí XML súboru a zadaní textového reťazca do vstupného poľa, užívateľ potvrdí vstupy tlačidlom „Confirm“. Názorná ukážka úvodnej obrazovky aplikácie je uvedená na obrázku 7.1.



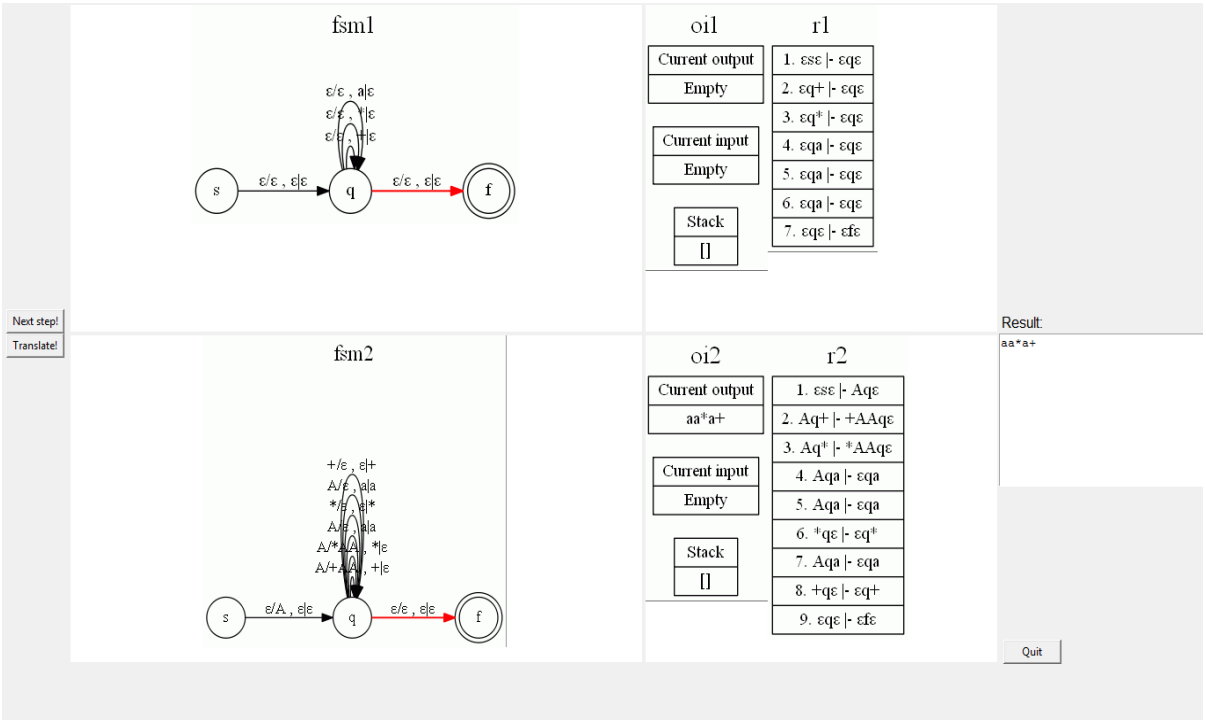
7.1 Úvodná obrazovka aplikácie

V prípade, že všetky vstupy boli zadané správne a neobsahujú chyby, užívateľovi sa zobrazí nová obrazovka. Tá obsahuje tlačidlá „Next Step“, „Translate“, „Quit“ a zobrazovacie plochy. Tlačidlo „Next Step“ slúži na krokovanie prekladu. V každom kroku je v okne zobrazený graf konečného automatu, ktorý znázorňuje prechody medzi jednotlivými stavmi. Hrany ako aj stav do ktorého sa prechádza sú graficky odlíšené od ostatných hrán/stavov. V ďalšej oblasti sa zobrazuje obsah zásobníku a aplikované pravidlá. Pravidlá sú zoraďované do tabuľky v poradí v akom boli aplikované pri preklade. Oddelená je aj oblasť aktuálneho vstupu a výstupného reťazca. Tieto prvky sú taktiež súčasťou hlavného okna aplikácie. Používateľ tak má prehľad o všetkom čo sa v danom kroku prekladu udialo. Spomínané informácie sú zobrazované súčasne pre obidva prevodníky uskutočňujúce preklad. Graf konečného automatu, zoznam aplikovaných pravidiel a aktuálny stav prevodníkov sú reprezentované ako obrázky. Pri väčších vstupoch je zobrazovacia oblasť týchto elementov nedostatočná a obrázky nebudú zobrazené celé. Pre takýto prípad sú všetky obrázky dostupné v adresári `.resources`, kde si ich užívateľ môže prehliadať v plnej veľkosti samostatne.

Užívateľ má možnosť previesť preklad vstupného reťazca aj v jednom kroku prostredníctvom tlačidla „Translate“. Po úspešnom preklade reťazca obidvoma prevodníkmi je výsledok zobrazený vo vstupnom poli „Result“. Toto riešenie bolo zvolené z dôvodu jednoduchého skopírovania výsledku a jeho prípadné ďalšie použitie. Aplikáciu je možné ukončiť tlačidlom „Quit“. Jednotlivé chybové hlásenia a hlásenie o úspešnom ukončení prekladu sú zobrazované prostredníctvom dialógových okien.

7.6 Príklad – prevod prefixového reťazca

V nasledujúcom príklade je popísaný prevod vstupného reťazca $+*aaa$ zapísaného v poľskej prefixovej notácii na odpovedajúci reťazec v postfixovej notácii. Prvý prevodník prevádza jednotlivé znaky vstupného reťazca na znaky ε, teda výsledkom jeho prevodu je prázdny reťazec. Druhý prevodník prevedie reťazec do post fixovej notácie. Výsledkom je konkatenácia prázdneho reťazca s reťazcom zapísaným v post fixovej notácii. Časť príkladu bola prevzatá z knihy [1].



7.2 Prevodníkový systém - ukážka z aplikácie

Na obrázku 7.2 je možné vidieť zvýraznenie prechodov do jednotlivých stavov červenou farbou. V grafe je ďalej označený koncový stav. V oboch prevodníkoch je koncovým stavom stav f. Vľavo sa nachádza tabuľka pravidiel a stav prevodu. Konkrétne sa jedná o aktuálny obsah vstupu, výstupu a zásobníku v danom kroku. V poli Result sa po dokončení prekladu zobrazí spomínaný reťazec $+*aaa$ v postfixovej notácii. Výsledkom prevodu je teda zobrazený výstupný reťazec $aa*a+$

8 Záver

V teoretickej informatike môžeme termín preklad chápať z viacerých hľadísk. Preklad, jeho sémantika a realizácia prekladu prostredníctvom rôznych formálnych modelov bol prezentovaný v tejto práci. Okrem definície formálneho prekladu, práca obsahuje aj popis prekladu formálneho jazyka. Od najjednoduchšej formy prekladu prostredníctvom homomorfizmu boli predstavené aj ďalšie typy prekladových modelov. Predovšetkým sa jednalo o modely pre regulárne a bezkontextové jazyky, ktoré je možné zadefinovať prostredníctvom konečných a zásobníkových prevodníkov. Taktiež bol vysvetlený koncept prevodníkových systémov, ich definícia a spôsob použitia takto zadefinovaných systémov.

V práci boli ďalej uvedené základné pojmy a definície, ktoré sú potrebné pre hlbšie preniknutie do tejto problematiky. Samozrejme aj tieto základy sa neobídu bez predchádzajúcich znalostí z oblasti formálnych jazykov a prekladačov.

V tejto práci boli prezentované konkrétne prekladové modely. Popísaný bol syntaxou riadený preklad, konečné a zásobníkové prevodníky a prekladové gramatiky. U každého zo spomínaných modelov bol uvedený základný popis a formálne definície prislúchajúce danému prekladovému modelu. Na praktických príkladoch boli prezentované základné princípy uvedených formálnych modelov. Porovnaná bola aj vyjadrovacia sila syntaxou riadeného prekladu oproti konečným prevodníkom. V práci bol uvedený formálny dôkaz, v ktorom bolo dokázané, že každý preklad generovaný syntaxou riadeným prekladom je taktiež prijímaný konečným prevodníkom. Opísaný bol vznik konečných prevodníkov a ich odvodenie z konečných automatov. Prezentovaná bola základná obrazová schéma modelu, preklad realizovaný konečnými prevodníkmi, jednotlivé fázy a kroky prekladu, grafická reprezentácia konečným prevodníkov prostredníctvom grafov. Rozobrané boli jednotlivé možnosti, ktoré môžu nastať pri prechode z jednej konfigurácie do inej spolu s grafickou ukážkou pre lepšie pochopenie a názornosť. Upresnená bola obecná definícia a popísaná deterministická verzia konečných prevodníkov.

Rozšírením konečných prevodníkov vznikli zásobníkové prevodníky. Opísaná bola grafická reprezentácia zásobníkových prevodníkov a základné rozdiely medzi zásobníkovými a konečnými prevodníkmi. Uvedený bol aj význam zásobníku pri procese prekladu vstupného reťazca a aplikovanie pravidiel. Nakoľko navrhnuté prevodníkové systémy sú postavené na zásobníkových prevodníkoch, aj tu bolo nutné rozšírenie ich formálnej definície. Tá bola rozšírená tak ako v prípade konečných prevodníkov o determinizmus. Práve deterministická verzia zásobníkových prevodníkov zjednodušila implementáciu nami navrhnutého prevodníkového systému.

Okrem prevodníkov práca obsahuje kapitolu venovanú prekladovým gramatikám. Prekladové gramatiky sú veľmi silným prekladovým modelom. Jedná sa o modifikáciu bezkontextovej gramatiky. Rozdiel spočíva v tom, že každá produkcia v prekladovej gramatike sa skladá z dvoch slov na jej pravej

strane. Prekladová gramatika teda generuje dvojicu slov, ktoré patria prekladu definovaným touto gramatikou. V tejto časti boli popísané základné definície zaoberajúce sa problematikou prekladových gramatík. Na konkrétnych príkladoch bol vysvetlený preklad definovaný práve týmito gramatikami.

Predpokladom pre vznik prevodníkových systémov, ktoré sú obsahom tejto práce sa stali zásobníkové prevodníky. Tie vznikli rozšírením konečných prevodníkov. Zásobníkové prevodníky boli zavedené do dvoch navrhnutých prevodníkových systémov. Práve spojením viacerých zásobníkových prevodníkov vznikol prevodníkový systém. Výhodou prevodníkových systémov je, že môžu pracovať paralelne a zrýchliť tak proces prekladu. Samozrejme paralelné spracovanie vstupu nie je podmienkou. Jeden z uvedených systémov bol implementovaný v počítačovej aplikácii. Implementovaný bol prevodníkový systém zložený z dvoch zásobníkových prevodníkov. Konkrétne išlo o prevodníkový systém, kde obidva prevodníky čítali rovnaký vstupný reťazec. Každý prevodník mal definovanú vlastnú množinu pravidiel. Výstupom celého systému bol reťazec, ktorý vznikol konkatenáciou výstupných reťazcov zásobníkových prevodníkov tvoriacich prevodníkový systém. Rozobraté boli aj ďalšie možnosti využitia prevodníkových systémov a ich možné nasadenie v praktickom živote. Opísané boli len základné možnosti využitia, ktoré by mohli byť neskôr doplnené o radu ďalších a komplexnejších príkladov.

Výsledkom celej práce je počítačová aplikácia demonštrujúca prevodníkový systém zložený z dvojice zásobníkových prevodníkov. Aplikácia poskytuje užívateľovi grafické rozhranie prostredníctvom ktorého užívateľ môže zadať vstupný reťazec a xml súbor s definovanými pravidlami pre obidva prevodníky. Tie realizujú preklad. Aplikácia graficky demonštruje jednotlivé fázy prekladu. V aplikácii sú zobrazené grafy obidvoch zásobníkových automatov, zostávajúca časť reťazca na vstupe, zapísaná časť reťazca na výstupe a obsah zásobníku. Taktiež je zobrazená tabuľka aplikovaných pravidiel. Použité pravidlá sú v tabuľke očíslované a zoradené podľa poradia v akom boli aplikované pri prevode vstupného reťazca. Pri vývoji boli najprv špecifikované a analyzované požiadavky aplikácie. V implementácii boli bližšie špecifikované požiadavky na vstupy programu. Popísané boli najhlavnejšie triedy a ich účel ako aj grafické prvky užívateľského rozhrania, ich rozloženie a význam. Na konkrétnom príklade bola demonštrovaná názorná ukážka z implementovaného programu. Zvolený bol príklad, kde sa výraz zapísaný v prefixovej (poľskej) aritmetickej notácii previedol na odpovedajúci postfixový výraz. Ten vznikol konkatenáciou výstupov prevodníkov, ktoré tvorili prevodníkový systém.

Tento text je možné využiť a ďalej rozšíriť o ďalšie prekladové modely. Napríklad o podrobnejšiu analýzu prekladových gramatík, ktoré by sa taktiež mohli zaviesť do systému. Taktiež by sa mohlo použiť paralélne spracovanie vstupu prevodníkmi tvoriacich prevodníkový systém. Následne by sa skúmal vplyv paralelizmu na rýchlosť prevodu pri rôznych špecifikáciách pravidiel a vstupov.

Zoznam obrázkov

Literatúra

1. MEDUNA, A. *Automata and Languages: Theory and Applications*. London: Springer, 2000. ISBN 81-8128-333-3.
2. AHO, A. V. et al. *Compilers: Principles, Techniques, and Tools*. 2nd Edition. Harlow: Pearson Education, 2006. ISBN 0-321-48681-1.
3. AHO, A. V. a J. D. ULLMAN. *The Theory of Parsing, Translation, and Compiling: Parsing*. Englewood Cliffs: Prentice-Hall, 1972. ISBN 0139145567.
4. JOHNSON, M. In: *Syntax Directed Translation* [online]. 2008 [cit. 2016-04-10]. Dostupné z: http://dragonbook.stanford.edu/lecture-notes/Stanford-CS143/13-Syntax-Directed-Translation.pdf
5. VAVREČKOVÁ, Š. In: *Syntaxí řízený překlad* [online]. 2008 [cit. 2016-04-10]. Dostupné z: http://vavreckova.zam.slu.cz/obsahy/prekl/prezentace/prekl_09b_prekl_gr.pdf
6. ELBL, S. In: *Finite and Pushdown Transducers* [online]. [cit. 2016-03-20]. Dostupné z: https://www.google.cz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjz6LL2-__LAhXBCpoKHdJsBdgQFggcMAA&url=http%3A%2F%2Fwww.fit.vutbr.cz%2F~meduna%2Fmti%2F2001_2002%2Felbl.ps&usg=AFQjCNF3Lf-0LGKkwPwE6HAQBjFaiMlrpw&sig2=civPtfzw4iyb5wKj92AKRQ&cad=rja
7. Wikipedia, the free encyclopedia. *Finite state transducer* [online]. 2016 [cit. 2016-03-15]. Dostupné z: https://en.wikipedia.org/wiki/Finite_state_transducer
8. ROZENBERG, G. a A. SALOMAA. *Handbook of Formal Languages / vol. III, Beyond words*. Berlin: Springer Verlag, 1997. ISBN 3-540-60649-1.
9. CHOFFRUT, C. a K. CULIK. Properties of Finite and Pushdown Transducers. SIAM Journal on Computing, č. 12, s. 300-15 [cit. 2016-04-03].
10. Wikipedia, the free encyclopedia. *State diagram* [online]. 2016 [cit. 2016-03-30]. Dostupné z: https://en.wikipedia.org/wiki/State_diagram
11. Wikipedia, the free encyclopedia. *Deterministic pushdown automaton* [online]. 2016 [cit. 2016-03-15]. Dostupné z: https://en.wikipedia.org/wiki/Deterministic_pushdown_automaton
12. *Tkinter — Python interface to Tcl/Tk* [online]. [cit. 2016-03-10]. Dostupné z: https://docs.python.org/2/library/tkinter.html
13. *Graphviz - Graph Visualization Software* [online]. [cit. 2016-03-25]. Dostupné z: http://www.graphviz.org/

14. Python Documentation. *xml.etree.ElementTree — The ElementTree XML API* [online]. [cit. 2016-04-05]. Dostupné z: https://docs.python.org/2/library/xml.etree.elementtree.html
15. CARROL, J. a D. LONG. *Theory of Finite Automata*. New Yersey: Prentice Hall, 1989. ISBN 978-0139137082.

Register