DEEP PUSHDOWN AUTOMATA OF FINITE INDEX

Vendula Poncová
Bachelor Degree Programme (3), FIT BUT
E-mail: xponco00@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: In this paper I discuss the power of finitely expandable deep pushdown automatas with a limited number of non-input symbols. Then I demonstrate that these automatas characterize an infinite hierarchy of language families resulting from programmed grammars of finite index.

Keywords: pushdown automata, programmed grammar, infinite hierarchy

1 ÚVOD

V této práci se věnuji hlubokým zásobníkovým automatům konečného indexu. Snažím se omezit počet nevstupních symbolů na zásobníku a zkoumám sílu tohoto typu automatu. Dále se zabývám ekvivalencí s programovými gramatikami, což vede k otázce, zda rodina jazyků přijímaná hlubokými zásobníkovými automaty konečného indexu s jedním nevstupním symbolem tvoří nekonečnou hierarchii.

2 DEFINICE

V tomto článku používám značení a pojmy z teorie formálních jazyků zavedené například v [1]. Dále vycházím z definice maticové a programové gramatiky v [4].

Hluboký zásobníkový automat konečného indexu je dle [2] definován jako uspořádaná osmice $M = (Q, \Sigma, \Gamma, R, s, S, F, n)$, kde Q je konečná množina stavů, Σ vstupní abeceda, Γ zásobníková abeceda, přičemž $\Sigma \subset \Gamma$, R je konečná množina pravidel, $s \in Q$ je počáteční stav, $S \in \Gamma$ počáteční symbol na zásobníku a $F \subset Q$ je množina koncových stavů. R je konečná množina pravidel typu $(m, q, A, p, v) \in \{1, 2, 3, \ldots, n\} \times Q \times \Gamma \times Q \times \Gamma^*$. Zapisují se ve tvaru $mqA \to pv$. Symbol $n \in \{1, 2, 3, \ldots\}$ označuje maximální počet nevstupních symbolů, které mohou být uložené na zásobníku. Hloubka tohoto typu zásobníku není definovaná. Předpokládá se, že je rozšiřitelná tak, aby zásobník mohl přijmout až n nevstupních symbolů.

3 OMEZENÍ POČTU NEVSTUPNÍCH SYMBOLŮ

Omezím-li počet nevstupních symbolů zásobníkového automatu M na jeden, mohu zavést nový automat $M_\#$ a porovnat jejich sílu. Převod automatu M na $M_\#$ popisuji algoritmem 1. Simulaci provádím pomocí stavů, do kterých ukládám informace o původním stavu a aktuálním obsahu zásobníku. To je možné díky konečnému počtu nevstupních symbolů na zásobníku, neboť jinak by takové řešení vedlo k nekonečnému počtu stavů.

Algoritmus 1. Převod hlubokého zásobníkového automatu konečného indexu na ekvivalentní s jedním nevstupním symbolem.

Vstup: $M = (Q, \Sigma, \Gamma, R, s, S, F, n)$

Výstup: $M_{\#} = (Q_{\#}, \Sigma_{\#}, \Gamma_{\#}, R_{\#}, s_{\#}, S_{\#}, F_{\#}, n)$

$\Sigma_{\#} := \Sigma$
$\Gamma_{\#} := \{\#\} \cup \Sigma$
$s_{\#} :=< s, \# >$
$S_{\#} := \#$

Pro každé $mqA \to pv \in R$, kde $v = b_0B_1b_1B_2b_2 \ldots b_{j-1}B_jb_j$, $j \in \{0,1,2,3,\ldots,n\}$, $b_0, b_i \in \Sigma^*$ a $B_i \in (\Gamma - \Sigma)$ pro všechna $i \in \{1,2,\ldots,j\}$:

A pro každé $(u,z) \in (\Gamma - \Sigma)^* \times (\Gamma - \Sigma)^*$, kde $|u| = m-1$, $|z| \leq n-m$:

přidej do $Q_{\#}$ stavy $< q, uAz >$, $< p, uB_1B_2 \ldots B_{j-1}B_jz >$,
přidej do $R_{\#}$ pravidlo $m < q, uAz > \# \to < p, uB_1B_2 \ldots B_{j-1}B_jz > b_0\#b_1\#b_2 \ldots b_{j-1}\#b_j$,
pokud $q \in F$, přidej do $F_{\#}$ stav $< q, uAz >$,
pokud $p \in F$, přidej do $F_{\#}$ stav $< p, uB_1B_2 \ldots B_{j-1}B_jz >$.

Je zřejmé, že každý hluboký zásobníkový automat konečného indexu s jedním nevstupním symbolem splňuje definici pro obecný hluboký zásobníkový automat konečného indexu. Tudíž spolu s algoritmem 1 je prokázána ekvivalence těchto automatů.

4 NEKONEČNÁ HIERARCHIE TŘÍD JAZYKŮ

V [2] je dokázáno, že hluboké zásobníkové automaty konečného indexu jsou ekvivalentní s maticovými gramatikami konečného indexu, tudíž třídy jazyků, které zásobníkový automat přijímá, tvoří nekonečnou hierarchii. Vzhledem k výsledkům v kapitole 3 je zřejmé, že tato vlastnost automatu zůstane i při omezení počtu nevstupních symbolů. Konstrukci důkazu tohoto tvrzení ukážu na ekvivalenci s programovými gramatikami, přičemž je očividné, že maticové a programové gramatiky konečného indexu generují stejný jazyk (viz [4]).

Pro hluboký zásobníkový automat dle [2] platí, že jestliže přijme slovo w, pak existuje taková komputace tohoto slova, která provádí pouze pop operace po první pop operaci. Toto tvrzení využiji v následujícím důkazu, neboť zásobníkový automat, na kterém probíhají jen expanze funguje jako gramatika. Stačí proto ukázat, že automat je schopen na svém zásobníku simulovat všechny derivace gramatiky a opačně.

Algoritmus 2 popisuje způsob, jak převést programovou gramatiku konečného indexu na hluboký zásobníkový automat konečného indexu s jedním nevstupním symbolem. Stav automatu se v tomto případě skládá ze dvou položek: označení pravidla, které se bude simulovat v dalším kroku, a řetězce neterminálů, které jsou na zásobníku nahrazené symbolem #. Automat přejde do koncového stavu, pokud jeho zásobník neobsahuje žádné neterminály.

Algoritmus 2. Převod programové gramatiky konečného indexu na ekvivalentní hluboký zásobníkový automat konečného indexu s jedním nevstupním symbolem.

Vstup: $G = (T \cup N, T, P, S)$ konečného indexu n
Výstup: $M = (Q, \Sigma, \Gamma, R, s, S, F, n)$

$\Sigma := T$
$\Gamma := T \cup \{\#\}$
$s_{\#} :=< \sigma >$
$S_{\#} := \#$

Pro každé $p : S \to v, g(p) \in P$:
přidej do R pravidlo $< \sigma >_1 \# \to < p, S > \#$ a do Q stav $< p, S >$.

Pro každé $q \in (Q \cup \{\varepsilon\})$:
přidej do F stav $< q, \varepsilon >$.

Pro každé $p : A \to v, g(p) \in P$, kde $v = b_0 B_1 b_1 B_2 b_2 \dots b_{j-1} B_j b_j$, $j \in \{0,1,2,3,\dots,n\}$, $b_0, b_i \in T^*$ a $B_i \in N$ pro všechna $i \in \{1,2,\dots,j\}$:

Pro každé $(k,u,z) \in \{1,2,3,\dots,n-j+1\} \times N^* \times N^*$, kde $|u| = k-1$, $|z| \leq n-k$:

Pokud $g(p) = \emptyset$:
přidej do Q stavy $< p, uAz >$, $< \varepsilon, uB_1B_2 \dots B_{j-1}B_jz >$,
přidej do R pravidlo $< p, uAz >_k \# \to < \varepsilon, uB_1B_2 \dots B_{j-1}B_jz > b_0\#b_1\#b_2 \dots b_{j-1}\#b_j$.

Jinak pro každé $q \in g(p)$:
přidej do Q stavy $< p, uAz >$, $< q, uB_1B_2 \dots B_{j-1}B_jz >$,
přidej do R pravidlo $< p, uAz >_k \# \to < q, uB_1B_2 \dots B_{j-1}B_jz > b_0\#b_1\#b_2 \dots b_{j-1}\#b_j$.

Konstrukce důkazu o převodu hlubokého zásobníkového automatu konečného indexu s jedním nevstupním symbolem na programovou gramatiku konečného indexu je dohledatelná ve článku [3], který srovnává programové gramatiky s #-Rewriting Systems. Programová gramatika simuluje každý krok zásobníkového automatu sekvencí několika derivací. Neterminály obsahují informace o stavu zásobníku, aktuální pozici výskytu symbolu # a celkovém počtu všech # v aktuální konfiguraci. Vlastní simulace probíhá následovně:

1. Aktualizace pozice a celkového počtu nevstupních symbolů u všech neterminálů.
2. Expanze neterminálu na příslušné pozici.
3. Odstranění pomocných stavů.

Tím jsem dokázala, že výše zavedený typ zásobníkového automatu je ekvivalentní s programovými gramatikami konečného indexu. Tudíž rodina jazyků přijímaná tímto automatem tvoří nekonečnou hierarchii vycházející z programových gramatik konečného indexu.

5 ZÁVĚR

V tomto příspěvku jsem dokázala, že hluboký zásobníkový automat konečného indexu omezením počtu nevstupních symbolů neztrácí na své síle. Následně jsem ukázala ekvivalenci tohoto typu automatu s programovými gramatikami konečného indexu. Z toho vyplývá, že rodiny jazyků, které tento automat přijímá, tvoří nekonečnou hierarchii.

REFERENCE

[1] Meduna, A.: *Automata and Languages: Theory and Applications*. London: Springer Verlag, 2005, ISBN 1-85233-074-0, 892 s.

[2] Meduna, A.: Finitely Expandable Deep PDAs. In *Automata, Formal Languages and Algebraic Systems*, Hong Kong: Hong Kong University of Science and Technology, 2010, ISBN 981-4317-60-8, s. 113–123.

[3] Křivka, Z.; Meduna, A.; Schönecker, R.: Generation of Languages by Rewriting Systems that Resemble Automata. *International Journal of Foundations of Computer Science*, ročník 17, č. 5, 2006: s. 1223–1229, ISSN 0129-0541.

[4] Dassow, J.; Păun, G.: *Regulated Rewriting in Formal Language Theory*. London: Springer, 1989, ISBN 978-3-642-74934-6, 308 s.