AUTOMATA WITH DEEP PUSHDOWNS

Lukáš Obrdlík, Master Degree Programme (4)
Dept. of Information Systems, FIT, BUT
E-mail: xobrdl01@stud.fit.vutbr.cz

Supervised by Dr. Alexander Meduna

ABSTRACT

This paper discusses the automata with deep pushdowns. It concentrates on the practical aspect of these automata. The implementation of these automata is described.

1 ÚVOD

Teze o automatech s hlubokými zásobníky, která ještě nebyla publikována, byla vytvořena pro možnost popisu jazyků, které jsou nad třídou jazyků bezkontextových a možnosti jejich přijetí pomocí automatů založených na zásobnících. Cílem této práce je seznámit čtenáře s automaty s hlubokými zásobníky a ukázat možnost implementace tohoto automatu. Tato implementace přijímá popis automatu ze vstupního souboru a snaží se rozhodnout, zda zadaný řetězec patří do jazyka přijímaného automatem popsaným vstupním souborem.

2 ROZBOR

Řešení rozdělíme na seznámení se s definicí automatů s hlubokými zásobníky (viz [1]) a popis programu, který simuluje přijetí vstupního řetězce a je založen na teorii o automatech s hlubokými zásobníky.

2.1 DEFINICE

Hluboký syntaktický analyzátor shora-dolů, zkráceně *DTDP*, je sedmice $M = (Q, \Sigma, \Gamma, R, s, S, F)$, kde Q je konečná množina stavů, Σ je vstupní abeceda, Γ je zásobníková abeceda, $I \in \{1, 2, 3,\ldots, k\}$, kde $k \in \mathbf{N}$, I, Q, Γ jsou navzájem disjunktní, $\Sigma \subseteq \Gamma$, množina $\Gamma - \Sigma$ obsahuje speciální vrcholový symbol reprezentovaný jako #, $R \subseteq (I \times Q \times \Gamma - (\Sigma \cup \{\#\}) \times Q \times (\Gamma - \{\#\})^+) \cup (I \times Q \times \{\#\} \times Q \times (\Gamma - \{\#\})^* \{\#\}$ je konečná množina relací, $s \in Q$ je počáteční stav, $S \in \Gamma$ je počáteční obsah zásobníku.

Místo $(m, q, A, p, v) \in R$, píšeme $mqA \rightarrow pv \in R$ a $mqA \rightarrow pv$ nazýváme *pravidlo*; tedy, R je označován jako množina pravidel popisujících M. Konfigurace automatu M je trojice $Q \times \Sigma^* \times (\Gamma - \{\#\})^* \{\#\}$.

X popisuje množinu všech konfigurací M. Nechť $x, y \in X$ jsou dvě konfigurace.

M odebírá symbol ze svého zásobníku při přechodu mezi konfiguracemi x a y, symbolicky zapsáno jako $x \ {}_o\Rightarrow y$, pokud $x = (q, az, au)$, $y = (q, z, u)$, kde $a \in \Sigma$, $z \in \Sigma^*$, $u \in \Gamma^*$. M zapisuje na svůj zásobník při přechodu mezi konfiguracemi x a y, symbolicky zapsáno jako $x \ {}_e\Rightarrow y$, pokud $x = (q, w, uAz)$, $y = (p, w, uvz)$, $mqA \rightarrow pv \in R$, kde $A \in \Gamma$, $u, v, z \in \Gamma^*$, $w \in \Sigma^*$, $q, p \in Q$, a $occur(u, \Gamma - \Sigma) = m - 1$. Pro vyjádření, že M provede přechod $x \ {}_e\Rightarrow y$ podle $mqA \rightarrow pv$, píšeme $x \ {}_e\Rightarrow y \ [mqA \rightarrow pv]$. Říkáme, že $mqA \rightarrow pv$ je pravidlo hloubky m; tedy, $x \ {}_e\Rightarrow y \ [mqA \rightarrow pv]$ je expanze hloubky m. Automat M provede přechod z x do y, symbolicky zapsáno jako $x \Rightarrow y$, pokud M může provést buď $x \ {}_e\Rightarrow y$ nebo $x \ {}_o\Rightarrow y$. Pokud $n \in I$ je nejmenší kladné celé číslo takové, že pro každé pravidlo z R je jeho hloubka menší nebo rovna n, říkáme, že M je hloubky n, symbolicky zapsáno jako ${}_nM$. M je deterministický vzhledem k hloubce jeho expanzí, pokud pro každé $q \in Q$, $card(\{m \mid mqA \rightarrow pv \in R, A \in \Gamma, v \in \Gamma^+, p \in Q\}) \leq 1$, protože v tomto bodě ze stejného stavu všechny expanze, které může M provést jsou stejné hloubky. Standardním způsobem rozšiřujeme ${}_o\Rightarrow$, ${}_e\Rightarrow$ a $\Rightarrow$ na ${}_o\Rightarrow^l$, ${}_e\Rightarrow^l$ a $\Rightarrow^l$ pro $l \geq 0$. Potom, na základě ${}_o\Rightarrow^l$, ${}_e\Rightarrow^l$ a $\Rightarrow^l$, definujeme ${}_o\Rightarrow^+$, ${}_o\Rightarrow^*$, ${}_e\Rightarrow^+$, ${}_e\Rightarrow^*$, $\Rightarrow^+$ a $\Rightarrow^*$. Definujeme jazyk přijímaný ${}_nM$, $L({}_nM)$, jako $L({}_nM) = \{ w \in \Sigma^* \mid (s, w, S) \Rightarrow^* (f, \varepsilon, \#), \text{ kde } f \in F\}$. Dále definujeme jazyk, který ${}_nM$ *přijímá prázdným zásobníkem*, $E({}_nM)$, jako $E({}_nM) = \{ w \in \Sigma^* \mid (s, w, S) \Rightarrow^* (q, \varepsilon, \#), \text{ kde } q \in Q\}$.

Pro všechny $k \geq 1$, označme $\mathbf{TD}_k = \{L({}_iM) \mid {}_iM \text{ je } DTDP, 1 \leq i \leq k\}$ a $\mathbf{ETD_k} = \{E({}_iM) \mid {}_iM \text{ je } DTDP, 1 \leq i \leq k\}$. **CF** a **CS** popisují třídy bezkontextových a kontextových jazyků.

Vztah jazyků přijímaných automaty s hlubokými zásobníky a třídami jazyků **CF** a **CS** je následující: $\mathbf{TD}_1 = \mathbf{ETD}_1 = \mathbf{CF}$ a pro každé $n \geq 1$, $\mathbf{ETD}_n = \mathbf{TD}_n \subset \mathbf{ETD}_{n+1} = \mathbf{TD}_{n+1} \subset \mathbf{CS}$ [1].

2.2 PŘÍKLAD

Mějme $DTDP$, ${}_2M = (\{s, q, p\}, \{a, b, c\}, \{A, S, \#\}, R, s, S\#, \{f\})$ s R obsahujícím tyto pravidla $1sS \rightarrow qAA$, $1qA \rightarrow paAb$, $1qA \rightarrow oab$, $2pA \rightarrow qAc$, $1oA \rightarrow fc$. Upozorňuji, že M je deterministický vzhledem k hloubce jeho expanzí.

Pro vstup $aabbcc$, ${}_2M$ provede tyto přechody:
$(s, aabbcc, S\#) \ {}_e\Rightarrow (q, aabbcc, AA\#) \ [1sS \rightarrow qAA]$
${}_e\Rightarrow (p, aabbcc, aAbA\#) \ [1qA \rightarrow paAb]$
${}_o\Rightarrow (p, abbcc, AbA\#)$
${}_e\Rightarrow (q, abbcc, AbAc\#) \ [2pA \rightarrow qAc]$
${}_e\Rightarrow (q, abbcc, abbAc\#) \ [1qA \rightarrow oab]$
${}_o\Rightarrow^3 (o, cc, Ac\#)$
${}_e\Rightarrow (o, cc, cc\#) \ [1oA \rightarrow fc]$
${}_o\Rightarrow^2 (f, \varepsilon, \#)$

Stručně, $(s, aabbcc, S\#) \Rightarrow^* (f, \varepsilon, \#)$. Poznamenejme, že pro každém $m \geq 2$, $L({}_2M) = E({}_2M) = \{a^n b^n c^n \mid n \geq 1\}$, kde m $\in I$, což je jazyk, který patří do množiny **CS** – **CF**.

2.3 POPIS IMPLEMENTACE

Program je napsán pro přijímání jazyka popsaného nedeterministickým automatem s hlubokými zásobníky. Proto se všechna použitá pravidla zapisují do historie stavů automatu. Při zjištění, že daný řetězec nelze generovat danou posloupností pravidel, se automat vrací zpět historií. Snaží se nalézt další pravidlo v daném stavu automatu, které se dá pro vstupní řetězec použít. Jestliže se takové pravidlo nenalezne, vrací se k další konfiguraci. Tzn. použí-

vá se metoda Backtracking. Výběr pravidel je řízen jejich pořadím, ve kterém byly zadány.

Rozgenerování pravidel končí, pokud automat zastaví nebo pokud se vrátí do počátečního stavu automatu a již nelze použít žádné pravidlo. O výsledku syntaktické analýzy je uživatel informován. Samotný program byl napsán v jazyce ANSI C++.

2.4 POPIS VSTUPNÍCH DAT

Vstupní data zadávaná pomocí souboru či standardního vstupu musí splňovat následující omezení. Pravidla se zadávají na jednotlivých řádcích ve formátu, který je shodný se zápisem uvedeným v definici, tzn. $mqA \rightarrow pv$.

Chceme-li zapsat pravidlo, které přepisuje daný nonterminál na prázdný řetězec, tedy např. $1sS \rightarrow f\varepsilon$, píšeme `1sS->f`. Není-li toto pravidlo jediným pravidlem stavu, doporučuji ho psát až jako poslední možnost pro daný stav, kromě použití u počátečního stavu.

Pro nastavení počátečního stavu a počátečního obsahu zásobníku se používá první zadané pravidlo.

Ve vstupních datech můžeme používat jednořádkové komentáře. Řádky, které obsahují komentář, vždy začínají dvojznakem //. Dalším vstupem je řetězec, u kterého chceme zjistit, zda patří do jazyka popsaného pravidly. Tento řetězec se zadává až při běhu programu.

2.5 POPIS OVLÁDÁNÍ

Program byl napsán pro spouštění v příkazovém řádku. Z toho plyne, že jeho ovládání se řídí přes data zadaná jako parametry programu. Program přijímá tři volitelné parametry.

Prvním možným parametrem je vstupní soubor. A není-li uveden, je očekáváno zadání pravidel ze standardního vstupu, které musí být ukončeno prázdným řádkem.

Další parametr, který je uvozen přepínačem **–o**, označuje soubor, do kterého se vypisují všechny stavy automatu provedené k přijetí (nepřijetí) zadaného řetězce. Poslední parametr, který předchází přepínač **–m**, určuje číslo, kterým se násobí délka testovaného řetězce k získání maximální možné délky zásobníku.

3 ZÁVĚR

Automaty s hlubokými zásobníky rozšiřují množinu jazyků, které lze přijmout zásobníkovými automaty, protože představují jejich modifikaci. Jazyky, které takto získáme, můžeme použít pro další popis programovacích jazyků.

Použití uvedeného programu by mělo pomoci k pochopení automatů s hlubokými zásobníky.

Ke zlepšení efektivity programu by bylo vhodné provést optimalizaci funkcí starajících se o vyhledávání pravidel, které se mají použít.

LITERATURA

[1] Meduna, A.: Deep Pushdown Expansions in Top-Down Parsing, Brno, 2005, manuscript

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