Rozšířené zásobníkové automaty

Rozšíření zásobníkového automatu, které budeme probírat v tomto dokumentu, je míněno tak, že se změní způsob čtení symbolů ze zásobníku. Zatímco u obyčejného PDA jsme mohli (destruktivně) číst pouze 1 symbol, u rozšířeného PDA máme možnost „vidět“ na libovolné množství symbolů, speciálně nemusíme se na zásobník podívat vůbec. Zavedení takového typu automatu nám pomůže při tzv. **Nedeterministické syntaktické analýze zdola nahoru**, kterou budeme probírat v další kapitole.

Definice – rozšířený zásobníkový automat

Rozšířený zásobníkový automat je sedmice $R = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde všechny symboly mají tentýž význam jako u obyčejného PDA s výjimkou přechodové funkce δ, která je zobrazením

- **z konečné podmnožiny** množiny $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*$
- **do konečných podmnožin** množiny $Q \times \Gamma^*$.

Poznámka.

V definici rozšířeného PDA je velmi důležité uvést, že se jedná o konečné zobrazení. Proto je definiční obor i obor hodnot zaveden jako konečná podmnožina množiny. Pokud bychom to explicitně neuvedli, jednalo by se o zobrazení mezi dvěma nekonečnými množinami. To je však v rozporu s naším cílem: konečně reprezentovat nekonečné jazyky.

Poznámka.

Pojmy konfigurace, akceptování koncovým stavem či prázdným zásobníkem zůstávají stejné, pouze krok výpočtu se malinko změní v tom smyslu, že

$$(p, aw, \gamma_1\alpha) \mapsto (q, w, \gamma_2\alpha), \text{ právě když } \delta(p, a, \gamma_1) \text{ obsahuje dvojici } (q, \gamma_2),$$

kde $\gamma_1, \gamma_2, \alpha \in \Gamma^*$ jsou libovolné řetězce zásobníkových symbolů, $p, q \in Q,\ a \in \Sigma,\ w \in \Sigma^*$. V obyčejném PDA jsme místo řetězce $\gamma_1 \in \Gamma^*$ měli pouze $A \in \Gamma$, tedy jeden symbol.

Příklad 1.

Mějme rozšířený zásobníkový automat $R = (\{q_0, p, f\}, \{a, b, c, d\}, \{A, B, Z\}, \delta, q_0, Z, \{f\})$, kde δ je definována takto:
1. $\delta(q_0, a, \varepsilon) = \{(q_0, A)\}$
2. $\delta(q_0, b, \varepsilon) = \{(q_0, B)\}$
3. $\delta(q_0, \varepsilon, \varepsilon) = \{(p, \varepsilon)\}$
4. $\delta(p, a, A) = \{(p, \varepsilon)\}$
5. $\delta(p, b, B) = \{(p, \varepsilon)\}$
6. $\delta(p, c, AA) = \{(p, \varepsilon)\}$
7. $\delta(p, c, BBB) = \{(p, \varepsilon)\}$
8. $\delta(p, d, Z) = \{(f, \varepsilon)\}$

Všimněte si přechodů 1, 2, 3, kde nečteme ze zásobníku nic a pouze do něj zapisujeme (symbol

A nebo B), naopak u přechodů 6, 7 čteme více než jeden symbol. Naznačíme si výpočet na slově $aabbbccd$:

$$\begin{aligned}
(q_0, aabbbccd, Z) &\mapsto_1 (q_0, abbbccd, AZ) \\
&\mapsto_1 (q_0, bbbccd, AAZ) \\
&\mapsto_2 (q_0, bbccd, BAAZ) \\
&\mapsto_2 (q_0, bccd, BBAAZ) \\
&\mapsto_2 (q_0, ccd, BBBAAZ) \\
&\mapsto_3 (p, ccd, BBBAAZ) \\
&\mapsto_7 (p, cd, AAZ) \\
&\mapsto_6 (p, d, Z) \\
&\mapsto_8 (f, \varepsilon, \varepsilon)
\end{aligned}$$

Poznámka.

Tradičně nás bude zajímat, zda se rozšířením obyčejného PDA mění i jeho popisovací síla, tj. zda platí, že pokud dokážeme sestrojit rozšířený PDA přijímající libovolný jazyk L, umíme vytvořit i obyčejný PDA akceptující L? Následující věta potvrzuje, že je tomu tak.

Věta.

Ke každému rozšířenému PDA R existuje obyčejný PDA M takový, že $L(M) = L(R)$.

Idea důkazu.

Je patrné, že jediný problém, který je nutné řešit, je potenciál automatu R číst více než jeden symbol z vrcholu zásobníku. V důkazu si pomůžeme tím, že obyčejný PDA M zkonstruujeme s rozdílně pojatou množinou stavů.

Buď tedy $R = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ rozšířený PDA přijímající bez újmy na obecnosti koncovým stavem. Stavy nového automatu M budeme zapisovat jako uspořádané dvojice (p, α), kde $p \in Q$ a $\alpha \in \Gamma^*$ je řetězec zásobníkových symbolů, které jsou aktuálně na vrcholu zásobníku původního automatu R. 2. složku stavu (p, α) si představte jako jakousi cache paměť, jejíž stav budeme v závislosti na výpočtu měnit. Její délku stanovíme podle toho, jaký je maximální počet symbolů, které je potřeba v jednom kroku přečíst ze zásobníku. Např. v příkladu 1 by délka cache paměti byla pouze tři, protože nejdelší možný řetězec čtený ze zásobníku byl BBB v případě přechodu 7.

V důkazu si formálně vysvětlíme, jak udržovat cache paměť a zásobník aktuální po celou dobu výpočtu a zajistit, aby jejich zřetězením byl řetězec, který byl u původního rozšířeného PDA na zásobníku.

Důkaz.

Nechť $R = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je rozšířený zásobníkový automat. Položme

$$m = \max\{|\alpha|; \delta(q, a, \alpha) \neq \emptyset\},$$

tj. m je maximální počet zásobníkového řetězce, pro který je δ definována. Zkonstruujeme nový zásobníkový automat $P = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, Z_1, F_1)$, kde

- $Q_1 = \{[q, \alpha] \mid q \in Q, \alpha \in \Gamma_1^*, 0 \leq |\alpha| \leq m\}$ – stavy jsou nově definovány jako dvojice, kde 1. složka je původní stav automatu R a 2. složka představuje obsah cache paměti. Všimněte si, že nestanovujeme délku řetězce α přímo na m. V dalším si ukážeme, že může nastat situace, kdy cache paměť nemá přesně m symbolů a že ji musíme dodatečně aktualizovat a doplnit symboly na zásobníku.
- $\Gamma_1 = \Gamma \cup \{Z_1\}$, kde Z_1 je nově přidaný symbol.
- $q_1 = [q_0, Z_0 Z_1^{m-1}]$ – na začátku výpočtu vložíme do cache paměti původní počáteční symbol zásobníku Z_0 a za něj dáme „výplně“, tj. několik symbolů Z_1 tak, abychom měli měli cache paměť plnou (proto je vloženo právě $m-1$ symbolů Z_1).
- $F_1 = [q, \alpha]$, kde $q \in F$ je původní koncový stav automatu R a $\alpha \in \Gamma_1^*$ libovolný řetězec.

Nyní nám chybí pouze vysvětlení toho, jak bude pracovat přechodová funkce δ_1. Zobecněme si, jak může přechod v rozšířeném zásobníkovém automatu R vypadat:

$$\delta(q, a, X_1 X_2 \dots X_k) \text{ obsahuje dvojici } (r, Y_1 Y_2 \dots Y_l),$$

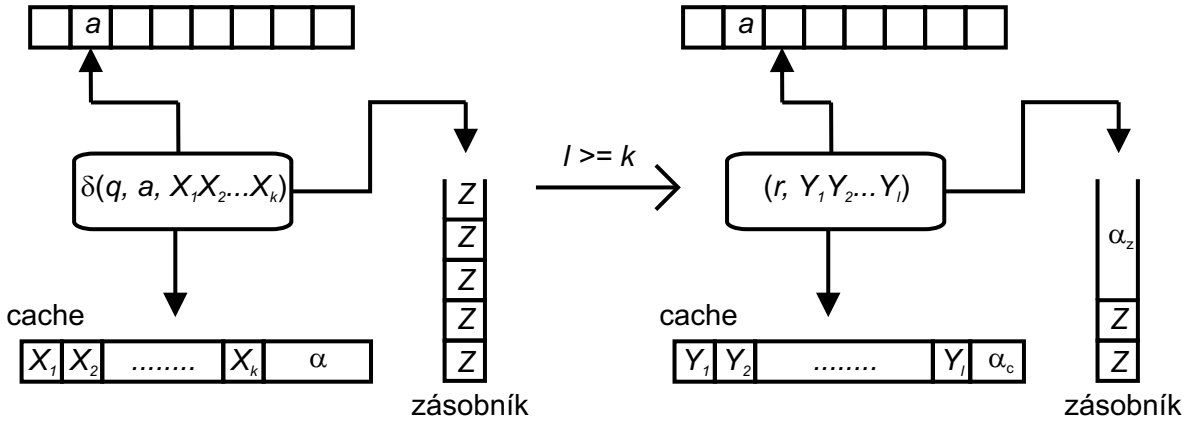
kde $q, r \in Q$ jsou stavy, $a \in \Sigma$ vstupní symbol, $X_1, \dots, X_k, Y_1, \dots, Y_l \in \Gamma_1$ zásobníkové symboly. Je zřejmé, že $k \leq m$, tj. délka řetězce $X_1 X_2 \dots X_k$ je maximálně m. Nový automat P má tento řetězec uložený v cache paměti, navíc tam může mít ještě nějaký řetězec α.

Mohou nastat dvě různé situace: buď $l \geq k$ nebo $l < k$, tj. buď je nově zapisovaný řetězec $Y_1 Y_2 \dots Y_l$ delší než $X_1 X_2 \dots X_k$ nebo kratší. Obě možnosti si rozebereme:

1. případ: $l \geq k$: přechodovou funkci δ_1 v tom případě definujeme takto:

$$\delta_1([q, X_1 X_2 \dots X_k \alpha], a, Z) \text{ obsahuje } ([r, \beta], \gamma Z),$$

kde platí, že $Z \in \Gamma_1$ je libovolný zásobníkový symbol a $\beta\gamma = Y_1 Y_2 \dots Y_l \alpha$, což si raději znázorníme na následujícím obrázku:



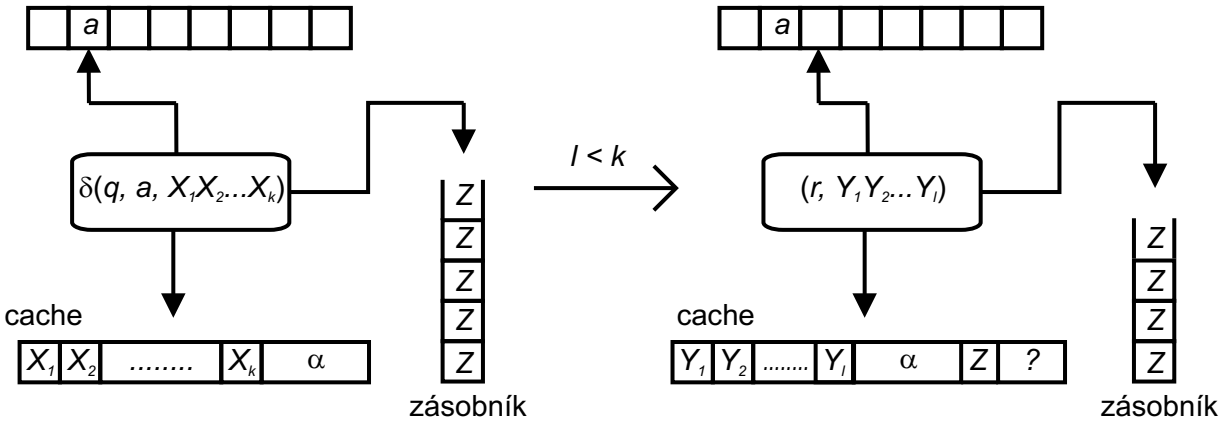
Na obrázku jste si asi všimli, že cache paměť obsahuje ve výsledku kromě řetězce $Y_1 Y_2 \dots Y_l$ ještě řetězec α_C, zásobník navíc α_Z. Platí, že $\alpha_c . \alpha_z = \alpha$, kde α byl původní řetězec, který v cache paměti následoval za $X_1 X_2 \dots X_k$. Jelikož však $l \geq k$, „vytlačil“ řetězec $Y_1 Y_2 \dots Y_l$ některé symboly α na zásobník. Snad je patrné, že řetězec α_c je to, co ještě zůstalo v cache paměti, zbytek (α_z) byl přesunut na zásobník. Navíc $|Y_1 Y_2 \dots Y_l \alpha_c| = m$.

Je samozřejmě možné, že délka řetězce $Y_1Y_2 \dots Y_l$ je větší než kapacita cache paměti, tj. $|Y_1Y_2 \dots Y_l| \geq m$. V tom případě je celý řetězec α přesunut na zásobník ($\alpha_c = \varepsilon$), přičemž nad ním můžou být i některé symboly řetězce $Y_1Y_2 \dots Y_l$. Každopádně musí platit, že zřetězíme-li obsah cache paměti a zásobníku, dostaneme $Y_1Y_2 \dots Y_l\alpha Z$.

2. případ: $l < k$: přechodovou funkci δ_1 v tom případě definujeme takto:

$$\delta_1([q, X_1X_2 \dots X_k\alpha], a, Z) \text{ obsahuje } ([r, Y_1Y_2 \dots Y_l\alpha Z], \varepsilon),$$

kde opět platí, že $Z \in \Gamma_1$ je libovolný zásobníkový symbol. Krok výpočtu ve 2. případě si opět znázorníme na obrázku:



Vzhledem k tomu, že jsme do cache paměti umístili méně symbolů, než kolik tam původně bylo, může se stát, že nebude zcela zaplněná. Je to naznačeno i na předchozím obrázku, kde po výpočetním kroku je v cache paměti řetězec $Y_1Y_2 \dots Y_l$, dále je zde i Z, který jsme z vrcholu zásobníku přesunuli. Řešením je zařazení následujících přechodů:

$$\delta_1([q, \alpha], \varepsilon, Z) = ([q, \alpha Z], \varepsilon),$$

kde $q \in Q, Z \in \Gamma_1$ jsou libovolné a pro $\alpha \in \Gamma_1^*$ platí, že $|\alpha| < m$. Bez čtení vstupu dostáváme ze zásobníku do cache paměti potřebné množství symbolů.

Korektnost: k ověření korektnosti je potřeba ukázat, že platí:

$$(q, aw, X_1X_2 \dots X_kX_{k+1} \dots X_n) \mapsto_R (r, w, Y_1Y_2 \dots Y_lX_{k+1} \dots X_n) \iff ([q, \alpha], aw, \beta) \mapsto_P ([r, \alpha'], \beta'),$$

kde:

1. $\alpha\beta = X_1X_2 \dots X_nZ_1^{m-n}$,
2. $\alpha'\beta' = Y_1Y_2 \dots Y_lX_{k+1} \dots X_nZ_1^{m-n}$,
3. $|\alpha| = |\alpha'| = m$,
4. mezi těmito dvěma konfiguracemi automatu P neexistuje žádná jiná konfigurace taková, že by 2. složka stavu (tj. cache paměť) měla délku rovnou konstantě m.

Poznámka.

Je potřeba uvést, že uvedený důkaz je těžko aplikovatelný do praxe, neboť s rostoucí délkou cache paměti roste velmi výrazně i počet přechodů.