Konstrukce zásobníkového automatu LALR(1)

Vilém Vychodil

5. listopadu 2001

Tento text se zabývá technickými aspekty konstrukce významné třídy zásobníkových automatů určených pro deterministickou syntaktickou analýzu zdola nahoru. LALR(1) automaty umožňují analyzovat širší třídu jazyků než automaty SLR(1). Redukce se uplatňuje pouze v případě, kdy je na vstupu symbol z prediktivní množiny symbolů (look–ahead). Počet zásobníkových symbolů je shodný jako u automatu SLR(1), což je proti počtu zásobníkových symbolů obecného automatu LR(1) značně výhodné. Text předpokládá intuitivní znalost zásobníkových automatů.

Deterministická analýza zdola nahoru

Analýza bezkontextového jazyka s sebou přináší několik úskalí. Proces analýzy je dán samotnou povahou jazyků. U regulárních jazyků je možné provést rozpoznání na základě konečné posloupnosti konečně mnoha stavů. U bezkontextových jazyků je situace jiná. Analýza vyžaduje simulaci průchodu stromem – nabízí se přirozeně rozšířit model automatu o zásobník.

Zásobníkové automaty pracují dvojím způsobem. Automaty pracující *shora dolů* simulují průchod derivačním stromem od jeho kořenu – startovního symbolu gramatiky – až po listy, což jsou terminální symboly. Dvě základní operace automatu analyzujícího shora dolů jsou *srovnání* a *expanse*. Při srovnání dojde buďto k odstranění shodných symbolů ze vstupu a vrcholu zásobníku, nebo k ukončení činnosti automatu. Při expansi je na vrcholu zásobníku nahrazen neterminální symbol A řetězcem α za předpokladu, že $A \to \alpha$ je odvozovací pravidlo gramatiky.

Nedeterminističnost automatu pracujícího shora dolů spočívá pouze ve vybrání správného expandujícího pravidla. Manipulace se zásobníkem je rovněž přímočará. Ze zásobníku je odebírán nanejvýš jeden symbol, přidáván je obecně řetězec. Zásobníkové symboly jsou tvořeny z terminálních a neterminálních symbolů gramatiky. Automat pracuje pouze s vrcholem zásobníku.

Na druhé straně stojí zásobníkové automaty pracující *zdola nahoru*. Jejich činnost simuluje průchod derivačním stromem zdola nahoru, tedy od listů k startovnímu symbolu gramatiky. Základní operace jsou opět dvě, je to *přesun* a *redukce*. Přesunem je přesunut vstupní symbol na vrchol zásobníku. Redukcí je odstraněn řetězec z vrcholu zásobníku a na jeho místo je položen neterminální symbol A odpovídající levé straně některého odvozovacího $A \to \alpha$ pravidla s odstraňovaným řetězcem α na pravé straně. Označme přechodovou funkci zásobníkového automatu

$$\delta\colon T \cup \{\varepsilon\} \times (T \cup N)^* \to 2^{(T \cup N)^*},$$

kde $\mathcal{G} = (N, T, P, S)$ je bezkontextová gramatika. Přechodová funkce definuje při aktuálním symbolu a při řetězci na zásobníku nové řetězce. Provedení přechodu je vždy podmíněno vstupem a aktuálním stavem zásobníku. Vstup a aktuální vrchol zásobníku je odstraněn, na vrchol zásobníku je při přechodu přidán jeden z definovaných řetězců. Základní operace lze chápat jako

$$\begin{aligned} \textit{přesun} \quad & \delta(t, \varepsilon) = \{t\}, \\ \textit{redukce} \quad & \delta(\varepsilon, A) = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}, \end{aligned}$$

kde $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n \in P$. Problémů, které musí tato analýza řešit je víc. Přesun je jednoznačný v tom smyslu, že pro každý $t \in T$ je definována operace přesunu, kde $\delta(t, \varepsilon)$ je jednoprvková

množina. Narozdíl od přímočaré operace srovnání u analýzy shora dolů již *nelze jednoduše rozhodnout*, zda-li v daný krok analýzy proběhne *přesun*, nebo *redukce*. Přesun se obecně neřídí stavem zásobníku.

Dalším nemalým problémem je rozhodnutí o náhradě řetězce ze zásobníku některou levou stranou. Problém technického charakteru představuje manipulace se zásobníkem. Při analýze zdola nahoru je nutné prohledávat zásobník do hloubky. I přes zdánlivě větší počet problémů je v technické praxi používána prakticky výhradně analýza zdola nahoru. Problémy těchto metod analýzy jsou shrnuty v následujícím přehledu.

- Při činnosti automatu je nutné *prohledávat zásobník do hloubky.* Tento nedostatek je odstraněn zavedením zásobníkových symbolů. Ty se v praxi používají místo množiny $T \cup N$. Zásobníkové symboly obsahují informaci o fázi přesunu – automat dále pracuje pouze s vrcholem zásobníku.
- Během analýzy je na základě znalosti vstupního symbolu a stavu zásobníku nutné rozhodnout o typu operace, která bude provedena. Zásobníkové symboly v sobě nesou informaci o tom, zda-li má při daném vstupu proběhnout *přesun*, či *redukce.* Při konstrukci automatu mohou vznikat *konflikty* mezi přesunem a redukcí, nebo mezi dvěma či více redukcemi.
- Zastavením automatu se rozpoznává přijetí či zamítnutí vstupního řetězce coby věty jazyka. Vstupní gramatiku je nutné upravit do takového tvaru, aby bylo přijetí vstupního řetězce technicky dobře zjistitelné.

Z předchozího text je patrné, že vyřešení těchto problémů spočívá především v zavedení vhodných *zásobníkových symbolů.*

Konstrukce zásobníkových symbolů

Předpokládejme bezkontextovou gramatiku $\mathcal{G} = (N, T, P, S)$, k níž máme sestrojit analyzující zásobníkový automat pracující zdola nahoru. Jak již bylo předesláno, nejprve je nutné vytvořit zásobníkové symboly. Ty v sobě ponesou informaci o fázích přesunu. Nejprve však zavedeme pojem *expandované gramatiky.*

Definice 1. Nechť $\mathcal{G} = (N, T, P, S)$ je bezkontextová gramatika. Dále označme S' neterminální symbol takový, že $S' \notin N$. Ekvivalentní bezkontextová gramatika $\mathcal{G}' = (N', T, P', S')$, kde $N' = N \cup \{S'\}$, $P' = P \cup \{S' \to S\}$ se nazývá *expandovaná gramatika.*

Expandovaná gramatika, někdy též rozšířená gramatika, je vskutku ekvivalentní. V podstatě pouze předsazuje nový startovní symbol. Tato úprava je výhodná z technického hlediska. Pokud automat při své činnosti vyčerpá vstupní symboly a na vrcholu zásobníku zůstane pravidlo symbolisující redukci podle pravidla $S' \to S$, činnost automatu lze zastavit – řetězec je přijat jako věta jazyka.

Konstrukce zásobníkových symbolů je rozdílná pro automatu LR(0), SLR(1) a LALR(1), LR(1). Konstrukce automatů LR(0), SLR(1) je výrazně jednodušší. O provedení redukce se rozhoduje buďto pouze na základě stavu zásobníku – LR(0), nebo jednoduchou metodou na základě znalosti jednoho vstupního symbolu. Je-li $A \to \alpha$ redukující pravidlo, pak musí být vstupní symbol z množiny Follow(A). Tímto způsobem fungují automaty SLR(1). Nejprve je rozebrána konstrukce zásobníkových symbolů pro automaty LR(0) a SLR(1).

Definice 2. Nechť $\mathcal{G} = (N, T, P, S)$ je expandovaná bezkontextová gramatika a nechť $A \to \alpha\beta \in P$ je libovolné odvozovací pravidlo. Symboly tvaru $[A \to \alpha_\circ \beta]$ nazveme *položky gramatiky* $\mathcal{G}$. Položky gramatiky tvaru $[A \to \alpha_\circ]$ se nazývají *redukční položky.* Neprázdná množina položek gramatiky se nazývá *zásobníkový symbol gramatiky* $\mathcal{G}$.

Poznámka. Zásobníkové symboly se skládají z položek gramatiky. Každá z nich representuje odvozovací pravidlo, na jehož pravé straně je tečkou vyznačena *fáze přesunu.* Zásobníkový symbol sestává z více polož právě proto, že může representovat několik přesunů zároveň. Je-li položka

ve tvaru $[A \to {}_\circ\alpha\beta]$, jedná se o počátek přesunu, je-li položka ve tvaru $[A \to \alpha\beta_\circ]$, přesun byl dokončen a může se začít s redukcí. Následující příklad ukazuje fáze přesunu.

$[E \to {}_\circ F + F]$	zahájení přesunu, F čeká na přesun,
$[E \to F {}_\circ+ F]$	F je přesunuto, $+$ čeká na přesun,
$[E \to F + F_\circ]$	přesun pravé strany dokončen, může se redukovat.

Je-li pravidlo ve tvaru $[A \to \alpha_\circ B\beta]$, kde B je neterminální symbol, pak representuje fakt, že daný neterminál by měl být přesunut na zásobník. To ale v praxi znamená, že nejprve je některá pravá strana odvozovacího pravidla začínajícího B celá přesunuta na zásobník a poté zredukována. To je jediný způsob, jak na zásobník přesunout neterminální symbol, který přirozeně není vstupní. Z tohoto důvodu lze ke každému zásobníkovému symbolu zavádí jeho uzávěr.

Definice 3. Nechť $\mathcal{G} = (N, T, P, S)$ je expandovaná bezkontextová gramatika, označme $Z(\mathcal{G})$ množinu všech možných zásobníkových symbolů gramatiky $\mathcal{G}$. Pro podmnožinu $Z \subseteq Z(\mathcal{G})$ zásobníkových symbolů definujeme posloupnost podmnožin $\{U_i \subseteq Z(\mathcal{G})\}_{i\in\mathbb{N}_0}$ induktivně,

$$\begin{aligned} U_0 &= Z, \\ U_{i+1} &= U_i \cup \{[B \to {}_\circ\beta]\,;\, [A \to \alpha_\circ B\gamma] \in U_i \wedge B \to \beta \in P\}. \end{aligned}$$

Dle Dirichletova principu musí existovat index $j \in \mathbb{N}_0$ tak, že $U_j = U_{j+1}$. Množinu U_j nazveme *uzávěrem množiny položek* Z a označíme $U(Z)$.

Uzávěr množiny položek již dobře representuje stav zásobníku během postupného přesunu symbolů. Zbývá dořešit technický detail, jak zkonstruovat množinu zásobníkových symbolů podle daných pravidel gramatiky. Pro expandovanou gramatiku $\mathcal{G}' = (N', T, P', S')$ je startovní symbol S' obsažen pouze v odvozovacím pravidle $S' \to S$. První zásobníkový symbol tedy vznikne uzávěrem $U(\{[S' \to {}_\circ S]\})$. Další symboly jsou konstruovány průběžně „přesunem symbolů za tečkou“.

Algoritmus 1. (Konstrukce pro LR(0), SLR(1)) *Vycházejme z expandované gramatiky $\mathcal{G}' = (N', T, P', S')$. Konstrukce zásobníkových symbolů bude representována konstrukcí konečného deterministického automatu $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S]\}), F)$ – automatu zásobníkových symbolů. Množina stavů Q označuje množinu konstruovaných zásobníkových symbolů. Vzájemný vztah zásobníkových symbolů je dán přechodovou funkcí δ.*

1. *Položme $Q = \{U(\{[S' \to {}_\circ S]\})\}$.*
2. *Pro každý dosud neuvažovaný zásobníkový symbol $Z \in Q$ a pro každý $a \in N \cup T$ označme*

   $$Z_a = U(\{[A \to \alpha a_\circ \beta]\,;\, [A \to \alpha_\circ a\beta] \in Z\}),$$

   to jest Z_a representuje zásobníkový symbol vzniklý přesunem symbolu a při stávajícím symbolu Z na vrcholu zásobníku.
3. *Pokud je $Z_a = \emptyset$, pokračujeme krokem 2.*
4. *Pokud Z_a dosud není v množině Q, pak $Q \stackrel{\text{def}}{=} Q \cup \{Z_a\}$.*
5. *Definujeme $\delta(Z, a) \stackrel{\text{def}}{=} Z_a$ a pokračujeme krokem 2.*

Po konečně mnoha krocích obdržíme automat $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S]\}), F)$, jehož stavy jsou zásobníkové symboly a přechodu definují jejich vzájemný vztah vzhledem k operaci přesunu. Za koncové stavy můžeme považovat ty symboly $Z \in Q$, které obsahují redukční položku.

Poznámka. Činnost automatu analyzujícího průchodem zdola nahoru se řídí přechodovou funkcí automatu zásobníkových symbolů $\mathcal{A}$. Je-li na vrcholu zásobníku symbol Z, na vstupu terminální symbol t a existuje-li přechod $\delta(Z, t) = Z'$, potom je přesunem myšleno odstranění vstupního symbolu t a přidání zásobníkového symbolu Z' na vrchol zásobníku. Jediný problém, který zůstává vyřešit je provedení redukce. Expandovaná bezkontextová gramatika $\mathcal{G} = (N', T, P', S')$ je LR(0), právě když žádná redukční položka příslušného automatu zásobníkových symbolů nemá definovaný přechod pro žádné $a \in N \cup T$.

Definice 4. Nechť $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S]\}), F)$ je automat zásobníkových symbolů expandované bezkontextové gramatiky $\mathcal{G} = (N', T, P', S')$ a nechť koncové stavy automatu jsou zásobníkové symboly obsahující právě jednu redukční položku gramatiky. Jestliže pro každý $Z \in F$ a libovolný $t \in T$ není definován přechod $\delta(Z, t)$, pak je gramatika LR(0).

To v technické praxi znamená, že automat nepotřebuje pro rozhodnutí mezi přesunem a redukcí znát žádný symbol ze vstupu. Je-li na vrcholu zásobníku symbol s právě jednou redukční položkou, tato položka jednoznačně určuje, o kterou redukci se jedná. Je-li na vstupu symbol bez redukční položky, automat provede přesun podle přechodové funkce automatu zásobníkových symbolů.

Požadavek gramatiky LR(0) je velmi silný. I velmi jednoduché gramatiky potřebují při jednom zásobníkovém symbolu někdy přesouvat a někdy redukovat. V tomto případě se automat musí rozhodnout podle vstupního symbolu.

Definice 5. Nechť $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S]\}), F)$ je automat zásobníkových symbolů expandované bezkontextové gramatiky $\mathcal{G} = (N', T, P', S')$ a nechť koncové stavy automatu jsou zásobníkové symboly obsahující alespoň jednu redukční položku gramatiky. Gramatika $\mathcal{G}$ je jednoduchá LR(1), neboli SLR(1), pokud jsou splněny následující dvě podmínky.

1. Pro každý $Z \in F$ a každou jeho redukční položku $[A \to \alpha_\circ] \in Z$ neexistuje přechod $\delta(Z, t)$ pro žádný $t \in \mathrm{Follow}(A)$. Formálně, $\{s;\ \delta(Z, s) \text{ je definován}\} \cap \mathrm{Follow}(A) = \emptyset$.
2. Pro všechny redukční položky $[A_1 \to \alpha_{1\circ}], \ldots, [A_n \to \alpha_{n\circ}]$ zásobníkového symbolu $Z \in F$, platí $\mathrm{Follow}(A_1) \cap \mathrm{Follow}(A_2) \cap \cdots \cap \mathrm{Follow}(A_n) = \emptyset$.

Gramatiky SLR(1) se již při redukci rozhodují podle množiny symbolů $\mathrm{Follow}(A)$, kde A je levá strana redukční položky $[A \to \alpha_\circ]$. Tato motivace je zřejmá, pokud má být řetězec z vrcholu zásobníku redukován na A, potom musí být následující symbol odvoditelný z větných forem stojících za A. Množina $\mathrm{Follow}(A) \subseteq T$, tedy gramatiky SLR(1) zahrnují širší třídu jazyků. Čím menší je podmnožina symbolů T určující redukci, tím širší třídu jazyků dostáváme.

Množina $\mathrm{Follow}(A) \subseteq T$ není pro určení redukce v žádném případě minimální. V technické praxi se používají především gramatiky LALR(1) a LR(1). Gramatika LR(1) je vůbec nejobecnější, její stinnou stránkou ale je rychle vzrůstající počet zásobníkových symbolů. Gramatiky LALR(1) odstraňují tento nedostatek slučováním zásobníkových symbolů. Syntaktické analyzátory postavené na gramatikách LALR(1) tvoří základ moderních analyzátorů jazyků. Mezi nejznámější generátory syntaktických analyzátorů patří Bison, jehož tvůrcem není nikdo jiný než Richard M. Stallman. Položky gramatiky a zásobníkové symboly pro LALR(1) a LR(1) gramatiky jsou rozšířeny o množinu prediktivních symbolů.

Definice 6. Nechť $\mathcal{G} = (N, T, P, S)$ je expandovaná bezkontextová gramatika a nechť $A \to \alpha\beta \in P$ je libovolné odvozovací pravidlo. Dvojice tvaru $[A \to \alpha_\circ\beta;\ \omega]$, kde $\omega \subseteq T \cup \{\varepsilon\}$, nazveme *položky gramatiky* $\mathcal{G}$.

- První prvek dvojice je *jádro položky gramatiky,* značíme $\mathrm{Ker}\,[A \to \alpha_\circ\beta;\ \omega] = [A \to \alpha_\circ\beta]$.
- Druhý prvek je *množina prediktivních symbolů,* značíme $\mathrm{La}\,[A \to \alpha_\circ\beta;\ \omega] = \omega$.
- Položky tvaru $[A \to \alpha_\circ;\ \omega]$ se nazývají *redukční položky.*

Neprázdná množina položek gramatiky se nazývá *zásobníkový symbol gramatiky* $\mathcal{G}$.

Poznámka. Definici jádra a množiny prediktivních symbolů lze přirozeně rozšířit z položek gramatiky na celé zásobníkové symboly prostým sjednocením přes jednotlivé položky gramatiky. Označme Z zásobníkový symbol. Pak

$$\mathrm{Ker}\, Z = \bigcup_{\forall p \in Z} \mathrm{Ker}\, p, \qquad \mathrm{La}\, Z = \bigcup_{\forall p \in Z} \mathrm{La}\, p,$$

a nazveme je *jádrem zásobníkového symbolu* a *prediktivní množinou zásobníkového symbolu.* Jelikož již zásobníkové symboly nerepresentují pouze stav přesunu, ale i množiny symbolů, které

mohou být na vstupu po dokončení redukce, musí se adekvátně upravit i uzávěr množiny položek gramatiky.

Definice 7. Nechť $\mathcal{G} = (N, T, P, S)$ je expandovaná bezkontextová gramatika, označme $Z(\mathcal{G})$ množinu všech možných zásobníkových symbolů gramatiky $\mathcal{G}$. Pro podmnožinu $Z \subseteq Z(\mathcal{G})$ zásobníkových symbolů definujeme posloupnost podmnožin $\{U_i \subseteq Z(\mathcal{G})\}_{i\in\mathbb{N}_0}$ induktivně,

$$\begin{aligned} U_0 &= Z, \\ U_{i+1} &= U_i \cup \{[B \to {}_\circ\beta;\, \kappa(\gamma,\omega)]\,;\, [A \to \alpha_\circ B\gamma;\, \omega] \in U_i \wedge B \to \beta \in P\}, \end{aligned}$$

kde nová množina prediktivních symbolů $\kappa(\gamma, \omega)$ je

$$\kappa(\gamma,\omega) = \bigcup_{\forall t\in\omega} \operatorname{First}\gamma t.$$

Dle Dirichletova principu musí existovat index $j \in \mathbb{N}_0$ tak, že $U_j = U_{j+1}$. Množinu U_j nazveme *uzávěrem množiny položek* Z a označíme $U(Z)$.

Definice uzávěru je v souladu s intuicí. Je-li ω množina symbolů, které mohou být na vstupu pro provedení redukce, potom musí být každá položka gramatiky vybavena novou množinou prediktivních symbolů – ta musí obsahovat symboly, které mohou být na vstupu po její redukci. Jelikož je přesunován neterminál z $A \to \alpha_\circ B\gamma$ na $B \to {}_\circ\beta$, za předpokladu $\gamma \neq \varepsilon$ stačí vzít $\kappa(\gamma, \omega) = \operatorname{First}\gamma$. Pokud by bylo $\gamma = \varepsilon$, pak zůstane prediktivní množina $\kappa(\gamma, \omega) = \omega$.

Konstrukce automatu LR(1) již je v tuto chvíli možná. Při konstrukci zásobníkových symbolů se pouze uplatňuje nový uzávěr. Počátečním zásobníkovým symbolem je zřejmě $U(\{[S' \to {}_\circ S;\, \varepsilon]\})$. Prediktivní množina položky $[S' \to {}_\circ S;\, \varepsilon]$ je prázdná, položka representuje ukončení činnosti automatu – vstup je očekáván prázdný.

Algoritmus 2. (Konstrukce pro gramatiky LR(1)) *Vycházejme z expandované gramatiky $\mathcal{G}' = (N', T, P', S')$. Konstrukce zásobníkových symbolů bude representována konstrukcí konečného deterministického automatu $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S;\, \varepsilon]\}), F)$ – automatu zásobníkových symbolů. Množina stavů Q označuje množinu konstruovaných zásobníkových symbolů. Vzájemný vztah zásobníkových symbolů je dán přechodovou funkcí δ.*

1. *Položme $Q = \{U(\{[S' \to {}_\circ S;\, \varepsilon]\})\}$.*
2. *Pro každý dosud neuvažovaný zásobníkový symbol $Z \in Q$ a pro každý $a \in N \cup T$ označme*

   $$Z_a = U(\{[A \to \alpha a_\circ \beta;\, \omega]\,;\, [A \to \alpha_\circ a\beta;\, \omega] \in Z\}),$$

   to jest Z_a representuje zásobníkový symbol vzniklý přesunem symbolu a při stávajícím symbolu Z, prediktivní množina symbolů zůstává zachována.
3. *Pokud je $Z_a = \emptyset$, pokračujeme krokem 2.*
4. *Pokud Z_a dosud není v množině Q, pak $Q \stackrel{\text{def}}{=} Q \cup \{Z_a\}$.*
5. *Definujeme $\delta(Z, a) \stackrel{\text{def}}{=} Z_a$ a pokračujeme krokem 2.*

Po konečně mnoha krocích obdržíme automat $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S;\, \varepsilon]\}), F)$, jehož stavy jsou zásobníkové symboly a přechodu definují jejich vzájemný vztah vzhledem k operaci přesunu. Za koncové stavy můžeme považovat ty symboly $Z \in Q$, které obsahují redukční položku.

Poznámka. Dvě *položky gramatiky* jsou *ekvivalentní,* právě když jsou si rovna jejich jádra i prediktivní množiny. Položky $[A \to \alpha_\circ\beta;\, \omega_1]$, $[A \to \alpha_\circ\beta;\, \omega_2]$ jsou tedy ekvivalentní, právě když platí $\omega_1 = \omega_2$. Dva *zásobníkové symboly* jsou *ekvivalentní,* právě když lze mezi jejich položkami gramatiky zavést bijektivní zobrazení tak, že vzájemně přiřazené položky jsou ekvivalentní.

Definice 8. Nechť $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S;\, \varepsilon]\}), F)$ je automat zásobníkových symbolů expandované bezkontextové gramatiky $\mathcal{G} = (N', T, P', S')$ a nechť koncové stavy automatu jsou zásobníkové symboly obsahující alespoň jednu redukční položku gramatiky. Gramatika $\mathcal{G}$ je LR(1), pokud jsou splněny následující dvě podmínky.

1. Pro každý $Z \in F$ a každou jeho redukční položku $[A \to \alpha_\circ;\, \omega] \in Z$ neexistuje přechod $\delta(Z, t)$ pro žádný $t \in \omega$. Formálně, $\{s;\, \delta(Z, s) \text{ je definován}\} \cap \omega = \emptyset$.
2. Pro všechny redukční položky $[A_1 \to \alpha_{1\circ};\, \omega_1], \ldots, [A_n \to \alpha_{n\circ};\, \omega_n]$ zásobníkového symbolu $Z \in F$, platí $\omega_1 \cap \omega_2 \cap \cdots \cap \omega_n = \emptyset$.

Vlastní činnost automatu během analýzy se nemění, při rozhodování o typu operace se automat rozhoduje podle prediktivní množiny. To jest, je-li na vrcholu zásobníku symbol s redukční položkou, lze se orientovat podle množiny prediktivních symbolů. Obecně lze říct, že úspěšná analýza je podmíněna vzájemnou neincidencí množin prediktivních symbolů redukčních položek automatu a jejich neincidencí vůči množině symbolů, které lze přesouvat.

Tvrzení 1. *Pro každou gramatiku LR(k), $k > 1$ existuje ekvivalentní gramatika LR(1).*

Důkaz. $\cdots$ NAPÍŠU ČASEM $\cdots$ □

Rychle rostoucí počet zásobníkových symbolů je hlavním důvodem, proč se LR(1) gramatiky v praxi nepoužívají. Gramatiky LALR(1) jsou rovněž založeny na identifikaci redukce podle prediktivní množiny, počet symbolů je ale shodný jako v případě gramatik SLR(1). Během konstrukce zásobníkových symbolů jsou slučovány symboly se stejným jádrem. Sloučení symbolů formálně popisujeme *operací* Merge.

Definice 9. Nechť $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S;\, \varepsilon]\}), F)$ je automat zásobníkových symbolů expandované bezkontextové gramatiky. Pro zásobníkové symboly $Z_1, Z_2, \ldots, Z_n \in Q$, které mají shodné jádro, to jest $\operatorname{Ker} Z_1 = \operatorname{Ker} Z_2 = \cdots = \operatorname{Ker} Z_n$ definujeme *sloučený zásobníkový symbol* $\operatorname{Merge}(Z_1, \ldots, Z_n)$ předpisem

$$\operatorname{Merge}(Z_1, \ldots, Z_n) = \{[A \to \alpha_\circ \beta;\, \omega]\,;\, [A \to \alpha_\circ \beta] \in \operatorname{Ker} Z_1\},$$
$$\text{kde } \omega \text{ značí } \bigcup_{i=1}^{n} \Big\{ \operatorname{La} p;\, p \in Z_i \wedge \operatorname{Ker} p = [A \to \alpha_\circ \beta] \Big\}.$$

To jest při sloučení dochází pouze u zásobníkových symbolů se stejným jádrem, přitom množiny prediktivních symbolů jsou sjednoceny. Konstrukce zásobníkových symbolů LALR(1) automatu je v zásadě dvojí. Buďto je nejprve vytvořen automat LR(1) a nakonec je zredukován počet zásobníkových symbolů. Mnohem častěji jsou však symboly slučovány již během konstrukce automatu. Při sloučení nového symbolu s již existujícím je nutné znovu projít množiny prediktivních symbolů těch zásobníkových symbolů, které za slučovaným symbolem následují ve smyslu přechodové funkce automatu zásobníkových symbolů. Konstrukci zásobníkových symbolů shrnuje následující algoritmus.

Algoritmus 3. (Konstrukce pro gramatiky LALR(1)) *Vycházejme z expandované gramatiky $\mathcal{G}' = (N', T, P', S')$. Konstrukce zásobníkových symbolů bude representována konstrukcí konečného deterministického automatu $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S;\, \varepsilon]\}), F)$ – automatu zásobníkových symbolů. Množina stavů Q označuje množinu konstruovaných zásobníkových symbolů. Vzájemný vztah zásobníkových symbolů je dán přechodovou funkcí δ.*

1. *Položme $Q = \{U(\{[S' \to {}_\circ S;\, \varepsilon]\})\}$.*
2. *Pro každý dosud neuvažovaný zásobníkový symbol $Z \in Q$ a pro každý $a \in N \cup T$ označme*

$$Z_a = U(\{[A \to \alpha a_\circ \beta;\, \omega]\,;\, [A \to \alpha_\circ a \beta;\, \omega] \in Z\}),$$

to jest Z_a representuje zásobníkový symbol vzniklý přesunem vstupního symbolu a při stávajícím symbolu Z, prediktivní množina symbolů zůstává zachována.

3. *Pokud je* $Z_a = \emptyset$*, pokračujeme krokem 2.*

4. *Pokud* Z_a *dosud není v množině* Q *a* $\operatorname{Ker} Z_a$ *je různé od jader všech ostatních zásobníkových symbolů, pak* $Q \stackrel{\text{def}}{=} Q \cup \{Z_a\}$ *a pokračujeme bodem 6.*

5. *Pokud* Z_a *dosud není v množině* Q *a jádro* $\operatorname{Ker} Z_a$ *se shoduje s jádrem zásobníkového symbolu* Z'*, to jest* $\operatorname{Ker} Z_a = \operatorname{Ker} Z'$*, pak zavedeme sloučený zásobníkový symbol* $Z'' = \operatorname{Merge}(Z_a, Z')$*. Dále upravíme množiny prediktivních symbolů.*

   (i) *Zásobníkový symbol* Z' *nahradíme symbolem* Z''*. To jest* $Q \stackrel{\text{def}}{=} (Q - \{Z'\}) \cup \{Z''\}$*. Adekvátně upravíme přechodovou funkci,* $\delta(S,t) = Z'$ *nahradíme* $\delta(S,t) = Z''$*.*

   (ii) *Každému symbolu odvozenému přesunem z* Z' *upravíme množinu prediktivních symbolů podle sloučeného symbolu* Z'' *– v souladu s předchozími pravidly. Tuto úpravu provádíme rekursivně i pro každý upravený symbol, dokud nejsou upraveny všechny symboly, jejichž předkem byl* Z'*.*

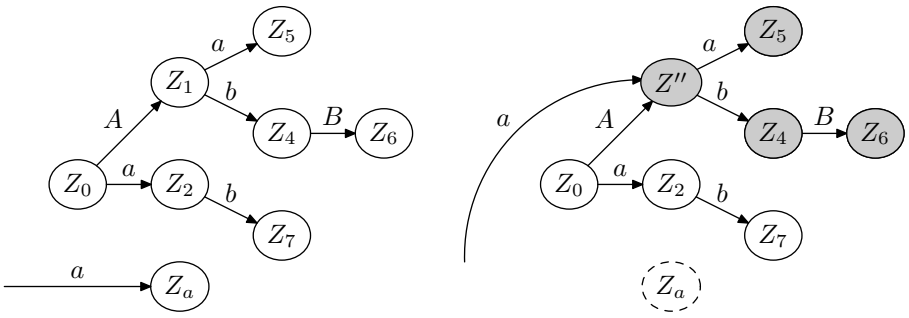
   (iii) *Původní zásobníkový symbol* Z' *zanikl.*

   Definujeme $\delta(Z,a) \stackrel{\text{def}}{=} Z''$ *a pokračujeme krokem 2.*

6. *Definujeme* $\delta(Z,a) \stackrel{\text{def}}{=} Z_a$ *a pokračujeme krokem 2.*

Po konečně mnoha krocích obdržíme automat $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S; \varepsilon]\}), F)$*, jehož stavy jsou zásobníkové symboly a přechodu definují jejich vzájemný vztah vzhledem k operaci přesunu. Za koncové stavy můžeme považovat ty symboly* $Z \in Q$*, které obsahují redukční položku.*

Pátý bod předchozího algoritmu je stěžejní. Díky němu nedochází k nárůstu zásobníkových symbolů. Graficky lze slučování znázornit následujícím obrázkem. Předpokládejme, že symboly Z_a a Z_1 mají stejné jádro. Levá část vystihuje konstrukci zásobníkových symbolů LR(1) gramatiky a vznik nového symbolu Z_a. V pravé části obrázku je zachycena konstrukce zásobníkových symbolů LALR(1) gramatiky.



Nový symbol není zaveden, místo toho je slit s původním symbolem Z_1. Šedou barvou jsou vyznačeny ty symboly, jejichž množiny prediktivních symbolů musejí být upraveny. Definice LALR(1) gramatiky je v tomto kontextu stejná, jako u gramatiky LR(1), při konstrukci zásobníkových symbolů pouze dochází k jejich slučování podle společného jádra.

Činnost automatu

Tento odstavec je věnován popisu činnosti automatu. Samotná činnost všech automatů vychází z obecného modelu automatu analyzujícího průchodem zdola nahoru. Je-li k dané gramatice sestaven adekvátní automatu zásobníkových symbolů $\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S; \varepsilon]\}), F)$, jsou všechny operace přímočaré. Operace přesunu se řídí přechodovou funkcí automatu $\mathcal{A}$, je-li na vstupu t a na vrcholu zásobníku Z, pak se t odebere ze vstupu a na vrchol zásobníku se vloží zásobníkový symbol $\delta(Z,t)$. Pokud by přechod neexistoval a Z neobsahuje redukční položku, to jest $Z \notin F$, pak automat svou činnost ukončí a řetězec nebyl přijat jako věta jazyka.

Pokud pro vstupní t a Z na vrcholu zásobníku neexistuje přechod, ale Z obsahuje redukční položku (položky), $Z \in F$, pak se automat rozhoduje o redukci. Podle vstupního symbolu je vybráno redukující pravidlo $A \to \alpha$, odpovídající redukční položce $A \to \alpha_\circ$. Počet symbolů $|\alpha|$ je odebrán z vrcholu zásobníku, nový vrchol zásobníku označme Z'. V této fázi byla odebrána pravá strana pravidla a na zásobník je nutné přidat symbol representující neterminál na levé straně. Na vrchol je položen symbol $\delta(Z', A)$.

Celou činnost automatu shrnuje následující algoritmus.

Algoritmus 4. *Předpokládejme, že máme zkonstruovány automaty zásobníkových symbolů boď pro LR(0), SLR(1), nebo pro LALR(1), LR(1) gramatiku, to jest automat*

$$\mathcal{A} = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S]\}), F),$$
$$\textit{nebo automat } \mathcal{A}' = (T \cup N, Q, \delta, U(\{[S' \to {}_\circ S;\, \varepsilon]\}), F).$$

1. *Na počátku analýzy je na vrcholu zásobníku počáteční zásobníkový symbol, to je buďto symbol $U(\{[S' \to {}_\circ S]\})$, nebo $U(\{[S' \to {}_\circ S;\, \varepsilon]\})$. Počáteční zásobníkový symbol representuje počátek přesunů, na vrchol zásobníku je třeba přesunout S. Počáteční zásobníkový symbol rovněž indikuje konec analýzy.*

2. *V průběžném kroku je proveden buďto přesun, nebo redukce. Na vrcholu zásobníku je symbol Z, na vstupu je terminální symbol t.*

   (i) *Pokud je definován přechod $\delta(Z, t) = S$, je provedena operace přesunu. Vstupní symbol t je odebrán ze vstupu a na vrchol zásobníku je položen symbol S. Dále proběhne krok 3.*

   (ii) *Pokud není definován přechod $\delta(Z, t)$ a zásobníkový symbol neobsahuje redukční položku, $Z \notin F$, pak se činnost automatu* zastaví s neúspěchem.

   (iii) *Pokud není definován přechod $\delta(Z, t)$ a zásobníkový symbol obsahuje redukční položky $p_1, \ldots, p_n$, pak je rozhodnuto o redukci podle následujících pravidel.*

   - *Je-li gramatika LR(0), pak je redukční položka právě jedna. Redukční pravidlo je vybráno podle ní, to jest pro jedinou redukční položku tvaru $[A \to \alpha_\circ]$ je vybráno redukční pravidlo $A \to \alpha$.*
   - *Je-li gramatika SLR(1), pak je vybráno redukční pravidlo $A \to \alpha$ příslušné redukční položce $[A \to \alpha_\circ]$, kde $t \in \operatorname{Follow} A$. Pokud neexistuje ani jedna redukční položka této vlastnosti, činnost automatu se* zastaví s neúspěchem.
   - *Je-li gramatika LALR(1), nebo LR(1), pak je vybráno redukční pravidlo $A \to \alpha$ příslušné redukční položce $[A \to \alpha_\circ;\, \omega]$, kde $t \in \omega$. Pokud neexistuje ani jedna redukční položka této vlastnosti, činnost automatu se* zastaví s neúspěchem.

   Pro vybrané redukční pravidlo $A \to \alpha$ je nejprve z vrcholu zásobníku odebráno $|\alpha|$ symbolů. Zásobníkový symbol, který zůstane po odebrání na vrcholu zásobníku označme Z'. Provedeme přesun symbolu $\delta(Z', A)$ na vrchol zásobníku. Pokud by při odstraňování „podtekl zásobník", nebo nebyl definován přechod $\delta(Z', A)$, činnost automatu se zastaví s neúspěchem. *V opačném případě je pokračováno krokem 3.*

3. *Pokud je na vstupu ε, to jest vstup je kompletně zpracován a na vrcholu zásobníku je symbol obsahující redukční položku $[S' \to S_\circ]$, případně $[S' \to S_\circ, \varepsilon]$, pak je činnost automatu zastavena a* automat přijal vstupní řetězec *jako větu jazyka. V opačném případě proběhne další průběžný krok.*

Technická realisace algoritmu je přímočará. V podstatě lze pouze udržovat *dvojrozměrnou tabulku* o $|Q|$ řádcích představujících zásobníkové symboly a o $|N \cup T \cup \{\varepsilon\}|$ sloupcích představující terminální a neterminální symboly spolu s prázdným řetězcem ε. Přechody $\delta(Z, t)$ se doplní do celé tabulky. Redukce se doplní pouze do části $T \cup \{\varepsilon\}$. Do tabulky je rovněž vhodné doplnit symbol „halt". Sloupec symbolu „halt" odpovídá prázdnému vstupu. Řádek symbolu „halt" representuje zásobníkový symbol obsahující redukční položku $[S' \to S_\circ]$, případně $[S' \to S_\circ;\, \varepsilon]$.

Jednoznačnost přesunu a redukce

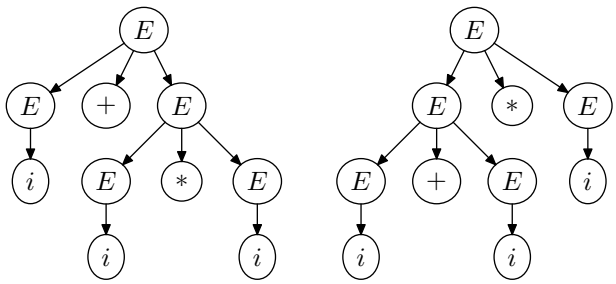
Z předchozí uvedené technické realisace algoritmu analýzy je dobře vidět, v jakých případech není možné prohlásit danou gramatiku za LR(0), SLR(1), LALR(1), či LR(1). Z tohoto pohledu jde v zásadě jen o jednu věc, na témže poli dvojrozměrné tabulky nemůže být uveden zároveň přesun a redukce, nebo dvě či více redukcí zároveň. Uvědomte si, že na jednom poli se nikdy nemohou sejít dva přesuny. Konstruovaný automat zásobníkových symbolů je totiž vždy deterministický – odtud toto tvrzení.

Situace, kdy na jednom poli přechodové tabulky dochází ke střetu víc akcí, nazýváme *konflikty.* Nejčastěji vznikajícím konfliktem je *konflikt přesun/redukce.* Konflikt přesun/redukce vzniká, pokud při konstrukci zásobníkových symbolů při Z na vrcholu zásobníku a při t na vstupu existují dvě alternativy, buďto přesun, nebo redukce. Dobrou zprávou budiž to, že konflikty typu přesun/redukce jsou dobře odstranitelné a zpravidla nevedou ke zhroucení analýzy.

▷ Příklad 1. Uvažujme expandovanou bezkontextovou gramatiku $\mathcal{G} = (N, T, P, E')$, kde $N = \{E', E\}$, $T = \{\mathtt{i}, *, +, (,)\}$ a odvozovací pravidla jsou tvaru,

$$P = \{E' \to E,\\ E \to E{+}E \mid E * E \mid (E) \mid \mathtt{i}\}.$$

Tato gramatika odpovídá uzávorkovaným aritmetickým výrazům, obsahujícím operace sčítání a násobení, terminální symbol `i` representuje sčítané hodnoty. Výše uvedená *gramatika je nejednoznačná,* například ke větám $\mathtt{i} + \mathtt{i} + \mathtt{i}$, či $\mathtt{i} + \mathtt{i} * \mathtt{i}$ existuje víc derivačních stromů. Například pro $\mathtt{i} + \mathtt{i} * \mathtt{i}$ existují následující dva stromy.



Obě části derivačního stromu representují tutéž větu. Logický význam obou derivací je však jiný, v prvním případě je nejprve zkompletována operace násobení, výraz má tedy význam $\mathtt{i} + (\mathtt{i} * \mathtt{i})$, v druhém případě je nejprve dokončeno sčítání, výraz má význam $(\mathtt{i} + \mathtt{i}) * \mathtt{i}$. Jelikož je gramatika nejednoznačná, k jedné větě existuje víc derivačních stromů, nelze k ní sestrojit deterministický analyzátor. Jakýkoliv analyzátor, třeba LR(1), bude mít v předchozím případě konflikt přesun/redukce. □

Většina generátorů analyzátorů si dokáže s konflikty přesun/redukce poradit zavedením *asociativity* a *priority* operací. V předchozím případě chceme, aby analyzátor provedl derivaci ve smyslu prvního obrázku. To znamená, že je-li na vstupu $\mathtt{i} + (\mathtt{i} * \mathtt{i})$, nejprve je celý výraz přesunut na zásobník a až poté je redukován. Kdyby byl přesunut pouze začátek výrazu, to jest $\mathtt{i} + \mathtt{i}$ a ten redukován, derivace by vedla na druhý případ.

▷ Příklad 2. Předpokládejme, že máme sestaven SLR(1) automat a snažíme se vyřešit předchozí konflikt. Na vrcholu zásobníku je symbol $Z = \{[E{+}E_\circ], [E_\circ{+}E], [E_\circ{*}E]\}$. Pokud je na vstupu $*\mathtt{i}\alpha$, což je zbývající vstup pro zpracování první části výrazu $\mathtt{i} + \mathtt{i} * \mathtt{i}$, pak automat se musí rozhodnout mezi přesunem symbolu $*$, nebo redukcí podle pravidla $E \to E{+}E$. Jelikož má $*$ vyšší prioritu než $+$, volíme operaci přesunu. Konflikt je vyřešen. □

Analogicky se řeší problémy asociativity zprava a zleva, to jest u výrazů $\mathtt{i} + \mathtt{i} + \mathtt{i}$ se upřednostňuje redukce – sčítání je zleva asociativní. Naopak u mocninných výrazů, třeba `i^i^i`, které jsou zpravidla zprava asociativní, se volí přesun. Na závěr k problematice konfliktů přesun/redukce

uveďme, že výše uvedená gramatika pro popis aritmetických výrazů lze nahradit ekvivalentní, která již v sobě obsahuje informaci o asociativitě a prioritě operací. Viz následující příklad.

▷ Příklad 3. Uvažujme expandovanou bezkontextovou gramatiku $\mathcal{G} = (N, T, P, E')$, kde $N = \{E', E, T, F\}$, $T = \{\mathtt{i}, *, +, (,)\}$ a odvozovací pravidla jsou tvaru,

$$\begin{aligned} P = \{&E' \to E,\ E \to E{+}T \mid T,\\ &T \to T * F \mid F,\ F \to (E) \mid \mathtt{i}\}. \end{aligned}$$

V této modelové gramatice pro $\mathtt{i} + \mathtt{i} * \mathtt{i}$ existuje pro $E' \Rightarrow^* \mathtt{i} + \mathtt{i} + \mathtt{i}$ pouze jeden derivační strom. Podrobně,

$$\begin{aligned} E' \Rightarrow E \Rightarrow E{+}T \Rightarrow T{+}T \Rightarrow F{+}T \Rightarrow \mathtt{i}{+}T \Rightarrow \mathtt{i}{+}T{*}F \Rightarrow\\ \mathtt{i}{+}F{*}F \Rightarrow \mathtt{i}{+}\mathtt{i}{*}F \Rightarrow \mathtt{i} + \mathtt{i} * \mathtt{i}. \end{aligned}$$

□

Druhým významným typem konfliktů jsou konflikty *redukce/redukce*. Tyto konflikty vznikají v případě, kdy má jeden vstupní symbol víc redukčních položek a při daném t na vstupu není možné jednoznačně rozhodnout o tom, která redukce bude provedena. Konflikty mezi dvěma či více redukcemi by v dobře napsaných gramatikách neměly nastávat. Většina generátorů analyzátorů tyto problémy řeší bez větší vnitřní logiky. Například generátor analyzátorů Bison použije první vyhovující redukční pravidlo. V každém případě konflikty redukce/redukce je nejlepší řešit ručním zásahem do gramatiky.

▷ Příklad 4. Uvažujme expandovanou bezkontextovou gramatiku $\mathcal{G}_1 = (N, T, P, S')$, kde $N = \{S', S, T\}$, $T = \{\mathtt{i}\}$ a odvozovací pravidla jsou tvaru,

$$\begin{aligned} P = \{&S' \to S,\\ &S \to T \mid S\,\mathtt{i} \mid \varepsilon,\\ &T \to \mathtt{i} \mid \varepsilon\}. \end{aligned}$$

V této gramatice vzniknou dva konflikty redukce/redukce a jeden konflikt přesun/redukce. Je-li na vstupu $\mathtt{i}$, nebo ε, pak existují dvě možné redukce. Navíc vyvstane konflikt, zda-li při vstupu $\mathtt{i}$ redukovat, či přesouvat. Uvažme zásobníkový symbol

$$Z = U(\{[S' \to {}_\circ S]\}) = \{[S' \to {}_\circ S], [S \to {}_\circ T], [S \to {}_\circ S\,\mathtt{i}], [S \to \varepsilon_\circ], [T \to {}_\circ \mathtt{i}], [T \to \varepsilon_\circ]\}.$$

Tento symbol obsahuje dvě redukční položky $[S \to \varepsilon_\circ]$ a $[T \to \varepsilon_\circ]$. Při $\mathtt{i}$, či ε na vstupu není možné rozhodnout, která redukce se má použít. Navíc vzniká konflikt přesun/redukce mezi přesunem terminálu $\mathtt{i}$ v $[T \to {}_\circ \mathtt{i}]$ a mezi oběma redukcemi.

Ještě jednodušším případem konfliktu redukce/redukce je gramatika $\mathcal{G}_2 = (N, T, P, S')$, kde $N = \{S', S\}$, $T = \{\mathtt{i}\}$ a množina odvozovacích pravidel je $P = \{S' \to S, S \to \mathtt{i} \mid \mathtt{i}S \mid \varepsilon\}$. V této gramatice dojde pouze k jednomu konfliktu redukce/redukce, konkrétně v zásobníkovém symbolu $Z = \{[S \to \mathtt{i}_\circ], [S \to \varepsilon_\circ], [S \to \mathtt{i}_\circ S]\}$. □

Obě konfliktní gramatiky z předchozího příkladu lze upravit do bezkonfliktního tvaru, například gramatiku $\mathcal{G}_1$ lze nahradit gramatikou $\mathcal{G} = (N, T, P, S')$, kde $N = \{S', S, F\}$, $T = \{\mathtt{i}\}$ a odvozovací pravidla jsou tvaru,

$$\begin{aligned} P = \{&S' \to S,\\ &S \to F \mid \varepsilon,\\ &F \to \mathtt{i} \mid S\mathtt{i}\}. \end{aligned}$$

Analogicky i pro druhý případ.