Deterministický překlad jazyků LL(1) pomocí redukovaného zásobníkového automatu

Fakulta informačních technologií,
Vysoké učení technické v Brně,
vytvořeno jako seminární práce do předmětu Moderní teoretická informatika,
Adam Husár,
18. prosince 2007.

Abstrakt

Tato práce popisuje postup, kterým lze z libovolné bezkontextové gramatiky vytvořit takový zásobníkový automat, kterému při přechodu stačí buď na zásobní jeden symbol přidat, jeden symbol odebrat a nebo se zásobníkem nedělat nic. Dále je zde popsáno, jak lze pomocí daného zásobníkového automatu deterministicky překládat jistou podtřídu bezkontextových jazyků. Nakonec je pak uveden příklad, jak lze pomocí takovéhoto automatu pracovat s atributy.

Úvod

V tomto dokumentu nejprve naleznete definici globálně indexovanými gramatikami inspirovaných gramatik a definici redukovaného zásobníkového automatu. Dále si zavedeme takzvaný prioritně deterministický redukovaný zásobníkový automat.

V druhé kapitole je pak popsán postup transformace bezkontextové gramatiky na ekvivalentní redukovaný zásobníkový automat.

Ukázalo se, že takto získaný redukovaný zásobníkový automat má další zajímavé vlastnosti. Například je možné při jeho překladu získat levý rozklad věty pomocí původních bezkontextových pravidel a lze i jednoduše při překladu pracovat s atributy. Dále, při vytvoření stavového prostoru daného automatu, kde stav je dvojice (původní stav, stav zásobníku) můžeme získat LL(1) překladovou tabulku.

1 Definice

1.1 Globálně indexovanými gramatikami inspirované gramatiky (GIGIG)

Globálně indexované gramatiky (GIG) jsou jedním druhem řízených gramatik. Zavedl je José M. Castaño [Cas04] a volně vychází z indexovaných gramatik zavedených Alfredem V. Aho [Aho68]. Princip spočívá v tom, že k bezkontextové gramatice přidáme 1 zásobník. José M. Castaño GIG jistými způsoby omezil tak, aby nezískal sílu Turingova stroje, jak by se to jinak, po přidání zásobníku k bezkontextové gramatice, stalo. Tato omezení pro nás nejsou důležitá a proto je zanedbáme a zavedeme si globálně indexovanými gramatikami inspirované gramatiky.

GIGIG je pětice $G = (N, T, I, S, \#, P)$, kde N, T a S jsou definovány stejně, jako u bezkontextových gramatik, I je množina indexů, # je startovací zásobníkový symbol a P je množina pravidel, která mohou nabývat následujících podob:

a. $A \underset{\varepsilon}{\rightarrow} \alpha$ *(epsilon),*

b. $A \underset{x}{\rightarrow} \alpha$ *(push),*

c. $A \underset{\bar{x}}{\rightarrow} \alpha$ *(pop),*

kde $x \in I, y \in I \cup \{\#\}, a \in N, \alpha, \beta \cup (N \cup T)^*$ a $a \in T$.

1.1.1 Relace derivace u GIGIG

Pokud $A \underset{\mu}{\rightarrow} X_1 ... X_n$ je pravidlo typu (a.), kde $\mu = \varepsilon$, resp. $\mu = y$ pak:

$\delta \# \beta A \gamma \underset{\varepsilon}{\Rightarrow} \delta \# \beta X_1 ... X_2 \gamma$, resp.

Pokud $A \underset{\mu}{\rightarrow} X_1 ... X_n$ je pravidlo typu (b.), kde $\mu = z$, pak:

$\delta \# wA \gamma \underset{z}{\Rightarrow} z \delta \# X_1 ... X_2 \gamma$.

Pokud $A \underset{\mu}{\rightarrow} X_1 ... X_n$ je pravidlo typu (c.), kde $\mu = \bar{z}$, pak:

$z \delta \# wA \gamma \underset{\bar{z}}{\Rightarrow} \delta \# X_1 ... X_2 \gamma$.

Použité symboly mají tento význam: $\beta, \gamma \in (N \cup T)^*, \delta \in I^*, z \in I \cup \{\varepsilon\}, y \in I$ a $X_i \in (N \cup T), 1 \le i \le n$.

Tranzitivní a tranzitivní uzávěr, reflexivní uzávěr i jazyk generovaný GIGIG je definován jako obvykle.

Příklad

Mějme GIGIG s následujícími pravidly:

$$S \underset{i}{\rightarrow} aSd\ ,\ S \rightarrow B\ ,\ B \underset{\bar{i}}{\rightarrow} bBc\ ,\ B \rightarrow \varepsilon\ ,$$

pak derivační posloupnost vedoucí k vygenerování řetězce *aabbccdd* by byla následující:

$$\#S \underset{i}{\Rightarrow} i\#aSd \underset{i}{\Rightarrow} ii\#aaSdd \Rightarrow ii\#aaBdd \underset{\bar{i}}{\Rightarrow} i\#aabBcdd \underset{\bar{i}}{\Rightarrow} \#aabbBccdd \Rightarrow aabbccdd\ .$$

1.2 Redukovaný zásobníkový automat (RZA)

Definice redukovaného zásobníkového automatu vychází z klasického zásobníkového automatu, avšak má upravenou přechodovou funkci. Redukovaný zásobníkový automat

(RZA) je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, kde $Q, \Sigma, \Gamma, q_0, z_0$ a F jsou definovány stejně jako u zásobníkového automatu.

Přechodová funkce δ je zobrazení z množiny $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow 2^{Q \times (\Gamma \cup \{\varepsilon\})}$, kde pro libovolné $q \in Q, a \in (\Sigma \cup \{\varepsilon\})$ platí:

pokud $(q', z') \in \delta(q, a, z)$,
pak $(z = \varepsilon \wedge z' = \varepsilon) \vee (z \in \Gamma \wedge z' = \varepsilon) \vee (z = \varepsilon \wedge z' \in \Gamma)$.

Při vykonání přechodu se buď se zásobníkem nedělá nic, 1 znak se ze zásobníku odebere a nebo se 1 znak na zásobník přidá.

Definice vychází z definice zásobníkového automatu uvedené ve skriptech Gramatiky a jazyky [Ces92].

1.2.1 Konfigurace RZA je trojice $(q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$ a má stejný význam jako u zásobníkového automatu.

1.2.2 Přechod RZA budeme reprezentovat binární relací $\mapsto$ definovanou na množině konfigurací RZA. Relace

$$(q, aw, z\alpha) \mapsto (q', w, z'\alpha)$$

platí, jestliže $\delta(q, a, z)$ obsahuje prvek (q', z'), kde $q, q' \in Q, a \in (\Sigma \cup \{\varepsilon\})$, $z, z' \in (\Gamma \cup \{\varepsilon\}), w \in \Sigma^*$ a $\alpha \in \Gamma^*$.

Relace $\mapsto^i, \mapsto^+, \mapsto^*$ jsou definovány obvyklým způsobem.

1.2.3 Jazyk přijímaný RZA

Platí-li pro řetězec $w \in \Sigma^*$ relace $(q_0, w, z_0) \mapsto^* (q, \varepsilon, \varepsilon)$ pro nějaké $q \in F$, pak říkáme, že w je přijímán redukovaným zásobníkovým automatem *M*. Množinu L(*M*) všech řetězců přijímaných RZA nazveme jazykem přijímaným RZA.

1.3 Prioritně deterministický RZA

Je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, kde jednotlivé symboly mají stejný význam jako u RZA a navíc pro jehož přechodovou funkci δ platí:

$$|\delta(q, a, z)| \leq 1 \text{ pro } \forall q \in Q, \forall a \in \Sigma \cup \{\varepsilon\}, \forall z \in \Gamma \cup \{\varepsilon\},$$

tedy pro libovolnou kombinaci stavu, případně prázdného vstupního symbolu a případně prázdného zásobníkového symbolu je výsledkem nejvýše 1 dvojice nového stavu a případně prázdného zásobníkového symbolu.

RZA, pro jehož přechodovou funkci platí výše uvedená podmínka, lze převést na priotitně deterministický RZA, kde převod spočívá pouze v tom, že budeme používat jinou přechodovou funkci.

1.3.1 Přechod deterministického RZA je znovu binární relací na množině konfigurací, je však definována odlišně než u RZA.

1) Pro konfiguraci ve tvaru $(q, aw, z\alpha)$, kde $a \in \Sigma$, je přechod definován následovně:
   Pokud $(q', z') \in \delta(q, a, z)$, pak

   $$(q, aw, z\alpha) \mapsto (q', w, z'\alpha),$$

   jinak pokud $(q', z') \in \delta(q, \varepsilon, z'')$, kde $z'' \neq \varepsilon \wedge z'' = z$, pak

   $$(q, aw, z\alpha) \mapsto (q', aw, z'\alpha),$$

   jinak pokud $(q', z') \in \delta(q, \varepsilon, \varepsilon)$, pak

   $$(q, aw, z\alpha) \mapsto (q', aw, z'z\alpha),$$

2) Pro konfiguraci ve tvaru $(q, \varepsilon, z\alpha)$, je přechod definován následovně:
   Pokud $(q', z') \in \delta(q, \varepsilon, z)$, pak

   $$(q, \varepsilon, z\alpha) \mapsto (q', \varepsilon, z'\alpha).$$

   Použité symboly mají tento význam: $q, q' \in Q$, $z, z' \in (\Gamma \cup \{\varepsilon\})$, $w \in \Sigma^*$ a $\alpha \in \Gamma^*$.

U prioritně deterministického RZA vybíráme prioritně přechody, které vyžadují znak ze vstupu a nebo odebírají znak ze zásobníku. Pouze v případě, že žádný takový přechod neexistuje, je možné použít přechod, který má prázdný vstupní symbol (ε-přechod).

Pojmy konfigurace, přechod, jazyk přijímaný deterministickým RZA a kompozice RZA jsou pro deterministický RZA definovány stejně jako pro RZA.

2 Algoritmus vytváření redukovaného zásobníkového automatu

2.1 Krok 1 - Transformace bezkontextové gramatiky na GIGIG s pravě lineárními pravidly

Mějme bezkontextovou gramatiku *G = (N, T, P, S)*. Ekvivalentní GIGIG $G' = (N \cup I \cup \{X\}, T, I, S, \#, P')$ vytvoříme následujícím způsobem:

1) $P' := \varnothing$
2) Pro každé pravidlo *p*, $p \in P$ a je ve tvaru

$$A \to w_1 A_1 w_2 A_2 ... A_{n-2} w_{n-1} A_{n-1} w_n ,$$

kde $A_i \in N, n \geq 1, 1 \leq i \leq n-1,\ \alpha_j \in T^*, 1 \leq j \leq n$.

Do *P'* přidej následující pravidla:

$$A \underset{A_1^p}{\to} w_1 A_1 , \qquad X \underset{\overline{A_1^p}}{\to} A_1^p ,$$
$$A_1^p \underset{A_2^p}{\to} w_2 A_2 , \qquad X \underset{\overline{A_2^p}}{\to} A_2^p ,$$
$$\ldots \qquad\qquad \ldots$$
$$A_{n-2}^p \underset{A_{n-1}^p}{\to} w_{n-1} A_{n-1}^p , \qquad X \underset{\overline{A_{n-1}^p}}{\to} A_{n-1}^p ,$$
$$A_{n-1}^p \to w_n ,$$
$$A_{n-1}^p \to w_n X \text{ a}$$

do *I* přidej $A_1^p, A_2^p, ..., A_{n-1}^p$.

Každé pravidlo z původní gramatiky si rozdělíme na několik dvojic $w_i A_i$. Pro tyto dvojice vytvoříme pravidla $A_{i-1}^p \underset{A_i^p}{\to} w_i A_i$, kde si vždy při vykonávání pravidla uložíme na zásobník nějaký nonterminál A_i^p. S jeho pomocí si naplánujeme vygenerování další dvojice z původního pravidla.

Pak je zde speciální nonterminál *X*. Ten se ve větné formě objeví vždy, když byl dokončen rozvoj fráze větné formy vzhledem k nějakému nonterminálu. Výskyt nonterminálu *X* nám tedy říká, že se máme podívat na zásobník, tam máme uložen nonterminál určující, kterou další dvojici $\alpha_i A_i$ máme začít rozgenerovávat.

Výsledkem kroku 1 je dvojice gramatika G' a speciální nonterminál X: (G', X).

Pozn.: Pravidla ve tvaru $A \to w_1 A_1 ... w_{n-2} A_{n-2} \varepsilon A_{n-1} \varepsilon$ lze převádět tak, aby byl počet výsledných pravidel i menší.

2.2 Krok 2 - Transformace GIGIG s pravě lineárními pravidly na globálně indexovanou gramatiku s regulárními pravidly

Použil by se klasický algoritmus konverze pravé regulární gramatiky na gramatiku regulární s tím, že pokud pravidlo nějak manipulovalo se zásobníkem, tak tato akce by zůstala pouze u prvního pravidla z něj vytvořeného.

Výsledkem by byla dvojice (G', X), obdobně jako v kroku 1.

2.3 Krok 3 - Vytvoření redukovaného zásobníkového automatu na základě GIGIG gramatiky s regulárními pravidly

Mějme dvojici (G, X) získanou z kroku 2, kde G je globálně indexovaná gramatika a X je speciálním nonterminálem:

$G = (N, T, I, S, \#, P)$,

kde v P jsou pravidla pouze v podobě $A \underset{x}{\rightarrow} aB$ nebo $A \underset{x}{\rightarrow} a$, kde $A, B \in N$, $a \in T \cup \{\varepsilon\}$ a $x \in \{z, \bar{z}, \varepsilon\}, z \in I$. Výsledkem kroku 3 bude redukovaný zásobníkový automat

$$M = (N \cup \{f\}, T, I, \delta, S, \#, \{f\}),$$

kde přechodovou funkci $\delta : N \times (T \cup \{\varepsilon\}) \times (I \cup \{\varepsilon\}) \rightarrow 2^{(N \cup \{f\}) \times (I \cup \{\varepsilon\})}$ získáme následujícím způsobem:

Pro každé pravidlo p, $p \in P$, které je ve tvaru:

a) $A \rightarrow aB$, nechť $(B, \varepsilon) \in \delta(A, a, \varepsilon)$,
b) $A \underset{z}{\rightarrow} aB$, nechť $(B, z) \in \delta(A, a, \varepsilon)$,
c) $A \underset{\bar{z}}{\rightarrow} B$, nechť $(B, \varepsilon) \in \delta(A, \varepsilon, z)$ a
d) $A \rightarrow a$, nechť $(X, \varepsilon) \in \delta(A, a, \varepsilon)$.

A dále nechť $(f, \varepsilon) \in \delta(X, \varepsilon, \#)$.

Princip je obdobný jako při převodu regulárních pravidel na konečný automat. Zde pouze navíc pracujeme i s akcí se zásobníkem.

Pozn.: Obecně může GIGIG ještě obsahovat pravidla ve tvaru $A \underset{\bar{z}}{\rightarrow} aB$, $A \underset{z}{\rightarrow} a$ a $A \underset{\bar{z}}{\rightarrow} a$, avšak takováto pravidla se pomocí kroků 1 ani 2 nevytvoří.

Věta 2.1:

Pro libovolnou bezkontextovou gramatiku G_1, na jejímž základě jsme pomocí kroků 1, 2 a 3 redukovaný zásobníkový automat M_1 vytvořili, platí $L(G_1) = L(M_1)$.

Princip důkazu

Důkaz by se provedl tak, že by se nejprve ověřila ekvivalence bezkontextové gramatiky sloužící jako vstup kroku 1 a výsledné GIGIG. Dále by se podobně ověřil krok 2 i krok 3.

Věta 2.2

Pokud bezkontextová gramatika G_2, na jejímž základě jsme pomocí kroků 1, 2 a 3 redukovaný zásobníkový automat M_2 vytvořili, je gramatikou LL(1) a zároveň platí, že RZA M_2 lze převést na prioritně deterministický RZA M_{2D}, pak $L(G_2) = L(M_{2D})$.

Princip důkazu

Vyšlo by se z toho, že lze z vytvořeného RZA získat LL(1) překladovou tabulku stejnou, jaká se vytvoří pomocí výpočtu množin FIRST a FOLLOW. Dále by se ukázalo, že při deterministickém přijímání libovolného řetězce klasickým zásobníkovým automatem s využitím LL(1) tabulky a přijímáním stejného řetězce prioritně deterministickým RZA se simuluje vykonávání těch samých pravidel ve stejném pořadí. Příklad vytváření LL(1) překladové tabulky naleznete v příkladě 3.4.

Poznámky

Samotná podmínka, kterou si klademe na prioritně deterministický RZA nám však nezaručuje, že vytvořený zásobníkový automat bude přijímat stejný jazyk, který generuje gramatika na jejímž základě byl automat vytvořen. Jako příklad může sloužit gramatika $S \to a, S \to B, B \to a$. Ta není LL(1), ale to pomocí výše uvedené konstrukce automatu nezjistíme.

Dále můžeme mít například gramatiku s pravidly $S \to A \mid B, A \to a, B \to b$. Ta je sice LL(1), ale získaný automat bude nedeterministický.

Obě podmínky uvedené ve větě 2.2 tedy musí pro její platnost platit.

3 Příklady

3.1 Příklad 1

Mějme gramatiku G_1, která má následující pravidla:

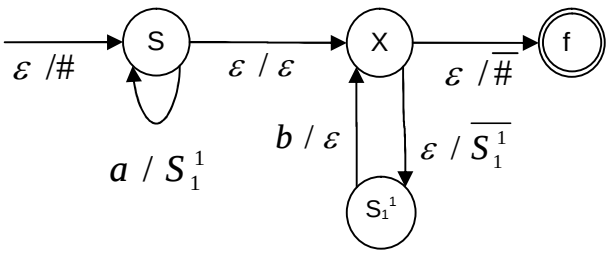
1: $S \to aSb$
2: $S \to \varepsilon$

Nyní tyto dvě pravidla převedeme podle kroku 1 a získáme GIGIG G_1' s následujícími pravidly:

1: $S \underset{S_1^1}{\to} aS$, $\quad X \underset{\overline{S_1^1}}{\to} S_1^1$,
$\quad S_1^1 \to b$, $\quad S_1^1 \to bX$,
2: $S \to \varepsilon$, $\quad S \to X$.

Nyní by následoval krok 2, avšak ten tuto gramatiku nijak nezmění, protože G' obsahuje pouze pravě regulární pravidla.

Vykonáním kroku 3 získáme zásobníkový automat M_1, ten můžete vidět na následujícím obrázku. Jeho hrany jsou ohodnoceny dvojicí (vstup/akce se zásobníkem).



Obrázek 3.1: Automat přijímající jazyk $a^n b^n,\ n \geq 0$.

Uvedený automat přijímá stejný jazyk, který je generovaný gramatikou G_1, tedy $\{a^n b^n \mid n \geq 0\}$. Dále pro něj platí podmínka determinismu:

$$\forall q \in Q, \forall a \in \Sigma \cup \{\varepsilon\}, \forall z \in \Gamma \cup \{\varepsilon\} : |\delta(q,a,z)| \leq 1.$$

Nyní, pokud budeme prioritně vybírat přechody, které buď vyžadují symbol ze vstupu a nebo symbol ze zásobníku (viz přechodová funkce deterministického RZA), pak můžeme vstupní řetězce rozpoznávat deterministicky.

Zkusíme rozpoznat řetězec *aabb*. Pro tento řetězec by automat M_1 procházel následující posloupností konfigurací:

$(S, aabb, \#) \mapsto (S, abb, S_1^1\#) \mapsto (S, bb, S_1^1 S_1^1\#) \mapsto (X, bb, S_1^1 S_1^1\#) \mapsto (S_1^1, bb, S_1^1\#) \mapsto (X, b, S_1^1\#) \mapsto (S_1^1, b, \#) \mapsto (X, \varepsilon, \#) \mapsto (f, \varepsilon, \varepsilon)$

Konec příkladu

3.2 Příklad 2

Mějme gramatiku G_2, která má následující pravidla:

$1: E \to FD$
$2: D \to +FD$
$3: D \to \varepsilon$
$4: F \to (E)$
$5: F \to i$

Pokud každé pravidlo transformujeme podle kroku 1, získáme GIGIG G_2' s následujícími pravidly:

1: $E \underset{E_1^1}{\to} F$, $X \underset{E_1^1}{\Rightarrow} E_1^1$,
$E_1^1 \underset{E_2^1}{\to} D$, $X \underset{E_2^1}{\Rightarrow} E_2^1$,
$E_2^1 \to \varepsilon$, $E_2^1 \to X$,

2: $D \underset{D_1^2}{\to} +F$, $X \underset{D_1^2}{\Rightarrow} D_1^2$,
$D_1^2 \underset{D_2^2}{\to} D$, $X \underset{D_2^2}{\Rightarrow} D_2^2$,
$D_2^2 \to \varepsilon$, $D_2^2 \to X$,

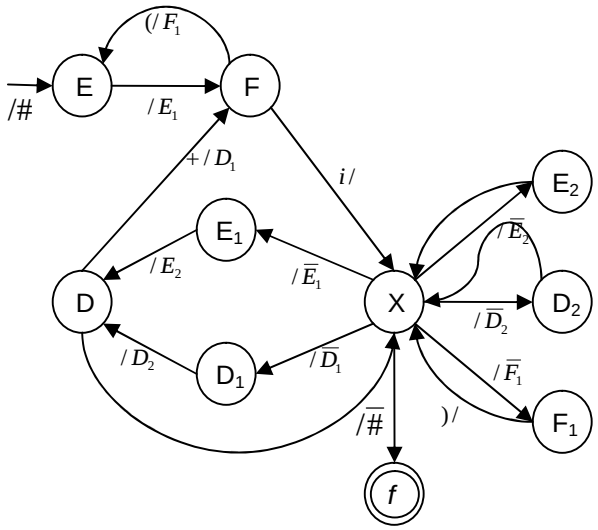
3: $D \to \varepsilon$, $D \to X$,

4: $F \underset{F_1^4}{\to} (E$, $X \underset{F_1^4}{\Rightarrow} F_1^4$,
$F_1^4 \to)$, $F_1^4 \to)X$,

5: $F \to i$, $F \to iX$.

Nyní by následoval krok 2, avšak ten tuto gramatiku nijak nezmění, protože G_2' obsahuje pouze pravě regulární pravidla.

Vykonáním kroku 3 získáme zásobníkový automat, ten můžete vidět na následujícím obrázku. Jeho hrany jsou ohodnoceny dvojicí (vstup/akce se zásobníkem). Horní indexy nových nonterminálů zde k odlišení nonterminálů od sebe nepotřebujeme a proto je zanedbáme.



Obrázek 3.2: Automat přijímající jazyk výrazů generovaný gramatikou G_2

Tento automat splňuje podmínku determinizmu a s jeho pomocí tedy můžeme deterministicky překládat jazyk generovaný původní LL(1) gramatikou G_2.

Pokud bychom si označili přechody automatu čísly původních pravidel (např. 1 pro přechod $E \to F$, 2 pro $D \to F$ atd.) a při překladu si tato čísla zapisovali, získáme levý rozklad pomocí původních pravidel gramatiky G_2.

Konec příkladu

3.3 Příklad 3

Pravidla bezkontextové gramatiky ve tvaru $A \to w_1 A_1 ... w_{n-2} A_{n-2} \varepsilon A_{n-1} \varepsilon$ lze převádět na GIGIG tak, aby výsledný počet pravidel byl ještě menší.Výsledek G_2" by byl následující:

1: $E \underset{D}{\to} F$, $X \underset{\overline{D}}{\to} D$,

2: $D \underset{D}{\to} + F$, $X \underset{\overline{D}}{\to} D$,

3: $D \to \varepsilon$, $D \to X$,

4: $F \underset{F_1^4}{\to} ($, $X \underset{\overline{F_1^4}}{\to} F_1^4$,

$F_1^4 \to)$, $F_1^4 \to) X$,

5: $F \to i$, $F \to iX$.

Získaná gramatika G_2" generuje stejný jazyk jako předchozí gramatika G_2 a podobně jako v předchozích případech by se dále použil krok 2 a krok 3 pro získání redukovaného zásobníkového automatu.

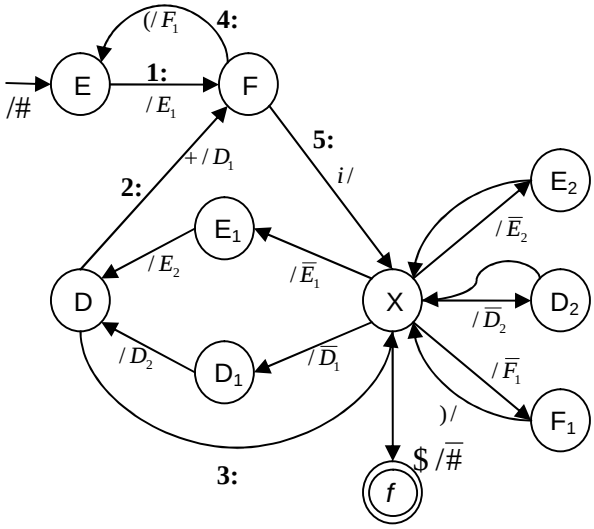
Konec příkladu

3.4 Příklad 4 – vytváření LL(1) překladové tabulky

V tomto příkladě si naznačíme postup, který nám umožní z RZA získat LL(1) překladovou tabulku. Vycházíme z její definice v [Ces92].

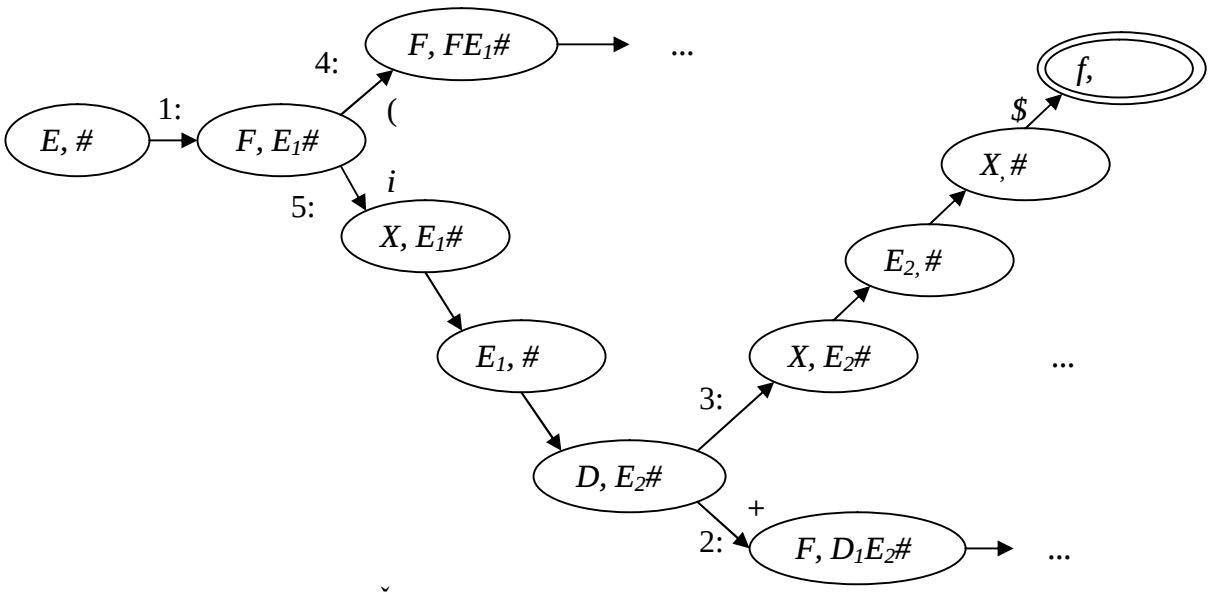
Mějme gramatiku G_2 z příkladu 2 a získaný RZA. U automatu si nejprve označíme přechody korespondující s původními pravidly – přechody odpovídající vždy prvním vytvořeným pravidlům GIGIG z původního pravidla.

$1: E \to FD$
$2: D \to +FD$
$3: D \to \varepsilon$
$4: F \to (E)$
$5: F \to i$



Obrázek 3.3: Bezkontextová gramatika a RZA pro překlad výrazů s označenými přechody

Nyní si sestrojíme automat představující stavový prostor uvedeného RZA. Jeden stav nového automatu bude popsán dvojicí (*původní stav, obsah zásobníku*). Tento automat bude nekonečný, ale je možné generování stavů omezit tak, že dva stavy (q,α) a (q,β), kde platí $\alpha = \mu^+\nu \wedge \beta = \mu\nu$, kde $\mu \in \Gamma^+, \nu \in \Gamma^*$, budeme považovat za ekvivalentní. Za takovéhoto předpokladu již bude automat konečný.



Obrázek 3.4: Část stavového automatu RZA pro překlad výrazů

V uvedeném automatu máme na přechodech jak informaci o číslu pravidla tak i o terminálech, které se čtou ze vstupu.

LL(1) překladová tabulka nám na základě symbol na vstupu a symbolu na vrcholu zásobníku určí, které pravidlo se má použít. Zde bude pouze naznačeno, jak na základě stavového prostoru RZA vytvoří, případně zájemce o další detaily odkazuje například na [Ces92].

Budeme brát jednotlivé stavy z automat na obrázku 3.4. Pro každý stav prozkoumáme cestu z něj vedoucí, vždy si zapamatujeme číslo pravidla a prohledáváme stavový prostor dokud nenarazíme na hranu či hrany označené terminálem. Získáme několik dvojic (***pravidlo***, ***terminál***) a na jejich základě si naplníme překladovou tabulku. Pro uvedenou část stavového automatu z obrázku 3.4 získáme tuto část překladové tabulky:

	i	(	+	$
E	1:	1:		
F	5:	4:		
D			2:	3:

Tabulka 3.1: Část LL(1) překladové tabulky získané na základě RZA

Z důvodu rozsáhlosti stavového automatu není možné ukázat vytvoření celé tabulky, ale dalo by se ukázat, že výsledek je stejný, jako při výpočtu množin FIRST a FOLLOW.

3.5 Příklad 5 – zpracování atributů

Mějme gramatiku G_2, stejnou, jako v příkladě 2. Oproti příkladu 2 si zde doplníme sémantické akce, které nám umožní vypočítat překládaný výraz.

Sémantické akce pracují s dalším pomocným zásobníkem, na něj si ukládáme hodnoty atributů. Jsou zde dva typy sémantických akcí. Zásobník mějme označen jako ***st*** a k jeho prvkům můžeme libovolně přistupovat: ***st[0]*** je prvek na vrcholu, ***st[n]*** je ***n***-tý prvek od vrcholu zásobníku. Další operace se zásobníkem jsou ***push(a)***, ta přidá ***a*** na vrchol a dále ***pop(n)***, která odebere ***n*** prvků z vrcholu zásobníku.

První typ jsou akce, které se vykonají vždy při vygenerování terminálu. Na zásobník si uložíme hodnotu atributu terminálu, pokud nás tato hodnota zajímá. V případě, že pro nás není důležitá, uložíme si na vrchol zásobníku znak $\perp$, který nám představuje prvek zásobníku bez hodnoty.

Druhý typ akcí se vykonává vždy po ukončení rozvoje pravidla. V těchto akcí se vždy provádějí 3 činnosti: 1) výpočet nové hodnoty atributu, 2) odstranění již nepotřebných hodnot atributů ze zásobníku (vždy odstraňujeme takový počet prvků ze zásobníku, kolik má dané pravidlo na pravé straně symbolů) a 3) uložení nové hodnoty atributu na zásobník.

Gramatika G_2 má tyto pravidla a k nim přiřazené sémantické akce:

$1: E \rightarrow FD$	{ *a := st[0]; pop(2); push(a);* }
$2: D \rightarrow +$ { *push($\perp$);* } FD	{ *a := st[-3] + st[-1]; pop(3); push(a);* }
$3: D \rightarrow \varepsilon$	{ *a := $\perp$; pop(0); push(a);* }
$4: F \rightarrow ($ { *push($\perp$);* } E) { *push($\perp$);* }	{ *a := st[-1]; pop(3); push(a);* }
$5: F \rightarrow i$ { *push(i.value);* }	{ *a := st[0]; pop(1); push(a);* }

Indexy pro přístup k syntetizovaným a zděděným atributům na zásobníku jsou počítány od konce pravidla počínaje nulou. Například u pravidla 4 se k syntetizovanému atributu nonterminálu *E* dostaneme pomocí *st[-1]*. Pro *st[-n]* platí, že pokud je $n < k$, kde *k* je počet symbolů na pravé straně pravidla, u kterého je *st[-n]* použito v sémantické akci, pak *st[-n]* představuje hodnotu syntetizovaného atributu, jinak *st[-n]* představuje hodnotu atributu zděděného.

Nyní každé pravidlo transformujeme podle kroku 1, získáme gramatiku G_2' s níže uvedenými pravidly. Sémantické akce prvního typu zůstávají u terminálů a akce druhého typu se posunou k posledním vytvořeným pravidlům, tedy pravidlům GIG představují ukončení rozvoje původních pravidel (tedy pravidla ve tvaru $A \to w$ nebo $A \to wX$, kde $A \in N, w \in T^*$ a *X* je speciální nonterminál).

1: $E \underset{E_1^1}{\to} F$, $X \underset{E_1^1}{\to} E_1^1$,

$E_1^1 \underset{E_2^1}{\to} D$, $X \underset{E_2^1}{\to} E_2^1$,

$E_2^1 \to \varepsilon$ { *a := st[0]; pop(2); push(a);* },

$E_2^1 \to X$ { *a := st[0]; pop(2); push(a);* },

2: $D \underset{D_1^2}{\to} +$ { *push(⊥);* } F, $X \underset{D_1^2}{\to} D_1^2$,

$D_1^2 \underset{D_2^2}{\to} D$, $X \underset{D_2^2}{\to} D_2^2$,

$D_2^2 \to \varepsilon$ { *a := st*[-3] + *st*[-1]*; pop*(3)*; push*(*a*)*;* },

$D_2^2 \to X$ { *a := st*[-3] + *st*[-1]*; pop*(3)*; push*(*a*)*;* },

3: $D \to \varepsilon$ { *a :=* ⊥*; pop*(0)*; push(a);* },

$D \to X$ { *a :=* ⊥*; pop(0); push(a);* },

4: $F \underset{F_1^4}{\to} ($ { *push*(⊥)*;* } E, $X \underset{F_1^4}{\to} F_1^4$,

$F_1^4 \to)$ { *push*(⊥)*;* } { *a := st[-1]; pop(3); push(a);* },

$F_1^4 \to)$ { *push(⊥);* } X { *a := st[-1]; pop(3); push(a);* },

5: $F \to i$ { *push(i.value);* } { *a := st[0]; pop(1); push(a);* },

$F \to i$ { *push(i.value);* } X { *a := st[0]; pop(1); push(a);* }.

Stejně jako v příkladu 2 vytvoříme redukovaný zásobníkový automat *M*,
$M = (\{E, F, D, E_1, E_2, D_1, D_2, F_1, X\}, \{(,), \mathrm{i}, +\}, \{E_1, E_2, D_1, D_2\}, \delta, E, \varepsilon, \{f\})$,
kde přechodová funkce δ je určena pravidly gramatiky G_2.

Nyní si můžeme pomocí automatu přeložit řetězec "*i + i*", kde první symbol *i* atribut bude mít hodnotu atributu *value* rovnu 1 a druhý symbol *i* rovnu 2. Pomocí automatu tedy počítáme výsledek výrazu 1 + 2.

Ukážeme si posloupnost konfigurací RZA včetně obsahu zásobníku atributů *st.* Pod každou konfigurací jsou uvedeny akce uveden stav zásobníku a pod přechody sémantické akce, které se zásobníkem atributů manipulují.

Krok	Stav	Vstup	Zásobník zásobníkového automatu	Zásobník atributů	Provedené sémantické akce
1	*E*	i+i			
2	*F*	i+i	E_1		
3	*X*	+i	E_1	1 1	*push(i.value);* *a:= st[0]; pop(1); push(a);*
4	E_1	+i	ε	1	
5	*D*	+i	E_2	1	
6	*F*	i	D_1, E_2	$\perp$,1	*push($\perp$);*
7	*X*		D_1, E_2	2,$\perp$,1 2,$\perp$,1	*push(i.value);* *a := st[0]; pop(1); push(a);*
8	D_1		E_2	2,$\perp$,1	
9	*D*		D_2, E_2	2,$\perp$,1	
10	*X*		D_2, E_2	$\perp$,2,$\perp$,1	*a := $\perp$; pop(0); push(a);*
11	D_2		E_2	$\perp$,2,$\perp$,1	
12	*X*		E_2	3,1	*a := st[-3] + st[-1]; pop(3); push(a);*
13	E_2			3,1	
14	*X*			3	*a := st[0]; pop(2); push(a);*
15	*f*			3	

Tabulka 3.2: Příklad zpracování atributů

Automat skončil v koncovém stavu s prázdným vstupem a zásobníkem, tedy vstupní řetězec v pořádku přijal. Na zásobníku atributů máme jediný prvek s hodnotou 3 a ten je požadovaným výsledkem.

4 Závěr

Problém, který mě dovedl k uvedenému řešení spočíval v tom, jak nějakým způsobem vložit do konečného automatu část, která by byla schopná překládat obecné výrazy. Jedna z cest bylo nějak "regularizovat" bezkontextovou gramatiku. V kombinaci s globálně indexovanými gramatikami se objevilo uvedené řešení, které nám umožní libovolnou bezkontextovou gramatiku převést na redukovaný zásobníkový automat ještě s tou výhodou, že určitou podtřídu jazyků LL(1) lze překládat takovýmto automatem i deterministicky. Také je vhodný pro zpracování atributů, jak ukazuje příklad v kapitole 3.5. Nevýhodou je, že pro získání deterministického redukovaného zásobníku potřebujeme vědět, zda je či není původní gramatika typu LL(1). Z RZA se sice LL(1) tabulka dá získat, ale časová složitost jejího výpočtu bude nejspíše exponenciální.

Práce také ukazuje souvislost konečných automatů a bezkontextových jazyků a možnosti reprezentovat zásobníkový automat graficky, která nemusí být u bezkontextových jazyků na první pohled patrná.

Dále, pokud si vezmeme asi tu nejhrubější klasifikaci vyčíslitelných problémů, tak konečné automaty bez jakéhokoli zásobníku odpovídají regulárním jazykům – stačí nám jediná proměnná, kde si pamatujeme aktuální stav. Pokud ke konečnému automatu přidáme jeden zásobník, tak se jedná o třídu problémů pro jejichž vyřešení si musíme něco plánovat. A nakonec máme konečné automaty se dvěma zásobníky, potřebujeme si plánovat dvě věci nezávisle na sobě a dostáváme se na sílu Turingových strojů.

Z této klasifikace nám vypadly kontextové jazyky, které jsou charakterizovány také jako Turingovsky rozhodnutelné problémy. Ty je možné řešit lineárně omezeným Turingovým strojem, kde určitým způsobem omezíme délku pracovní pásky, resp. maximální hloubku zásobníků u konečného automatu se dvěma zásobníky. Otázkou pak je, jakou sílu by měl konečný automat s jedním zásobníkem, pokud bychom mu také podobným způsobem omezili velikost zásobníku – zda by šlo stále o třídu bezkontextových jazyků a nebo by se tato síla omezila.

Co se týká aplikací uvedeného postupu, pak se, kromě již uvedené substituce redukovaného zásobníkového automatu do konečného automatu, nabízí například nějaké využití v hardware, kde samotný konečný automat je využíván poměrně často a realizace takovéhoto jednoduššího zásobníku také není velký problém. Také by bylo možné zjistit, zda to, že nad zásobníkem provádíme pouze jednodušší operace než je tomu u klasického zásobníkového automatu, má nějakou výhodu například ve větší rychlosti překladu.

Literatura

[Aho68] Aho, A. V.: Indexed grammars - an extension of context-free grammars. *Journal of the Association for Computing Machinery*, 15(4):647–671, 1968. Dokument dostupný na WWW: <http://portal.acm.org/citation.cfm?id=321488&dl=ACM&coll=portal>.

[Cas04] Castano, J. M.: *Global Index Languages*. PhD. Thesis, The Faculty of the Graduate School of Arts and Sciences, Brandeis University, 2004. Dokument dostupný na WWW: <http://www.cs.brandeis.edu/~jcastano/thesis3.pdf>.

[Ces92] Češka, M.: *Gramatiky a jazyky*. FIT VUT v Brně, 1992. Dokument dostupný na WWW: <http://www.fit.vutbr.cz/study/courses/TI1/public/Texty/ti.pdf>.