VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

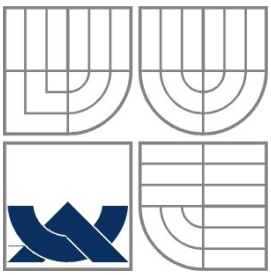
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PARALELNÍ HLUBOKÉ ZÁSOBNÍKOVÉ AUTOMATY

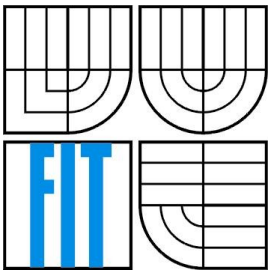
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PETER SOLÁR
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PARALELNÍ HLUBOKÉ ZÁSOBNÍKOVÉ AUTOMATY

PARALLEL DEEP PUSHDOWN AUTOMATA

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

Peter Solár

VEDOUCÍ PRÁCE
SUPERVISOR

prof. RNDr. Alexander Meduna, Csc.

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Solár Peter**
Obor: Informační technologie
Téma: **Paralelní hluboké zásobníkové automaty**
Kategorie: Překladače

Pokyny:

1. Seznamte se detailně s hlubokými zásobníkovými automaty.
2. Zaveďte paralelní hluboké zásobníkové automaty. Studujte jejich vlastnosti. Porovnejte je s vlastnostmi klasických hlubokých zásobníkových automatů.
3. Studujte užití navržených automatů(např. kompilátory, lingvistika, mikrobiologie). Navrhněte vhodnou aplikaci založenou na nich. Testujte ji.
4. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Meduna, A.: Automata and Languages, Springer, London, 2000

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Meduna Alexander, prof. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Peter Solár**
Id studenta: 84146
Bytem: Palackého 1462/32, 750 02 Přerov
Narozen: 26. 02. 1985, Přerov
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Paralelní hluboké zásobníkové automaty
Vedoucí/školitel VŠKP: Meduna Alexander, prof. RNDr., CSc.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☒ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato práce představuje *paralelní hluboké zásobníkové automaty* jako paralelní verzi hlubokých zásobníkových automatů. Jsou založeny na pravidlech, podle kterých může automat provést expanzi současně až *n* nejvýše položených neterminálních symbolů na vrcholu zásobníku pouze jednou aplikací pravidla. Podmínkou je, aby se na zásobníku vyskytoval dostatečný počet neterminálů. Hlavní výhoda použití paralelních hlubokých zásobníkových automatů spočívá v rychlejším rozhodování.

Klíčová slova

formální jazyky, automaty, zásobníkové automaty, hluboké zásobníkové automaty, paralelní hluboké zásobníkové automaty, gramatiky, bezkontextové gramatiky, n-omezené stavové gramatiky, stavové gramatiky

Abstract

This thesis introduces *parallel deep pushdown automata* as the parallel version of the deep pushdown automata. They are based on the rules, where the automaton can expand *n* topmost non-terminals in only one derivation step if there are enough non-terminals on the pushdown. The main advantage rests in a fact, that parallel automaton can makde a decission faster.

Keywords

formal languages, automata, pushdown automata, deep pushdown automata, parallel deep pushdown automata, grammars, context-free grammars, n-limited state grammars, state grammars

Citace

Peter Solár: Paralelní hluboké zásobníkové automaty, bakalářská práce, Brno, FIT VUT v Brně, 2007

Paralelní hluboké zásobníkové automaty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením prof. RNDr. Alexandra Meduny, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Peter Solár
14.5. 2007

Poděkování

Rád bych poděkoval panu prof. Alexandru Medunovi za přínosné konzultace a pomoc při řešení této bakalářské práce.

Obsah

Obsah ... 1
1 Úvod ... 2
2 Matematický základ ... 3
3 Jazyky ... 4
3.1 Chomského klasifikace ... 4
4 Gramatiky ... 5
4.1 Bezkontextová gramatika ... 5
4.2 n-omezené stavové gramatiky ... 5
5 Automaty ... 7
5.1 Konečný automat ... 7
5.1.1 Definice ... 7
5.2 Zásobníkový automat ... 7
5.2.1 Definice ... 8
5.3 Hluboký zásobníkový automat ... 8
5.3.1 Definice ... 8
5.3.2 Příklad ... 9
6 Paralelní hluboký zásobníkový automat ... 10
6.1 Neformální definice ... 10
6.2 Formální definice ... 10
6.3 Konfigurace a přechody ... 11
6.4 Příklad ... 12
7 Vlastnosti ... 13
7.1 Síla ... 13
7.2 Převod hlubokého zásobníkového automatu na paralelní verzi ... 17
7.3 Rychlost ... 17
7.3.1 Příklad ... 18
8 Aplikace ... 19
9 Závěr ... 22
Literatura ... 23
Seznam příloh ... 24

1 Úvod

V této práci, věnované teorii formálních jazyků, bude prezentováno paralelní rozšíření hlubokých zásobníkových automatů, které jsou zobecněnou verzí zásobníkových automatů. Toto téma jsem si vybral protože existuje pouze malé množství modifikací automatů, jako jednoho z modelů pro popis jazyků, ve srovnání s množstvím modifikací gramatik (druhý model). Než dojdeme k tomuto rozšíření, bude potřeba si zopakovat základní vědomosti, bez nichž by toto rozšíření nemuselo být pochopeno. Přestože je převážná část této práce teoretická, nesmíme zapomínat ani na část praktickou. Tou je vytvoření aplikace simulující chování tohoto prezentovaného modifikovaného automatu.

Co vlastně najdete v této práci? Ve druhé kapitole si připomeneme některé použité symboly a termíny z teorie formálních jazyků. Následně se podíváme na klasifikaci jazyků a ukážeme si dva základní modely pro jejich popis – gramatiky a automaty. Obě skupiny si v dalších kapitolách probereme podrobněji, především pak automaty, jichž se tato práce týká. V šesté kapitole zavedeme paralelní hluboké zásobníkové automaty. Následovat budou vlastnosti těchto nově zavedených automatů, kdy si dokážeme že jsou stejně silné jako neparalelní verze, ale na rozdíl od nich pracují rychleji. Osmá kapitola je věnována praktické části této práce a je zaměřena především na rozdíly implementace paralelního a neparalelního automatu. V závěru bude nastíněn další možný vývoj založený především na omezeních této modifikace.

2 Matematický základ

V následujícím textu bude použito poměrně velké množství matematických symbolu a termínů. Proto by bylo vhodné si je na úvod připomenout a vysvětlit.

Písmenem I budeme označovat množinu kladných celých čísel. Pro každou množinu Q můžeme zavést *kardinalitu* $card(Q)$, tedy počet prvků této množiny.

Abeceda je neprázdná konečná množina prvků, nazývaných *symboly*. Nechť Σ je abeceda, potom ε je *prázdný řetězec* nad abecedou Σ (řetězec délky 0) a pokud x je řetězec nad abecedou Σ, a a je symbol $a \in \Sigma$, potom xa je také řetězec nad abecedou Σ. *Konkatenace* dvou řetězců je jejich spojení za vzniku nového řetězce, viz. xa v předchozí větě. Výsledkem konkatenace jakéhokoliv řetězce s prázdným řetězcem je původní řetězec. Mějme abecedu V. Z jejích symbolů můžeme konkatenací vytvořit množinu neprázdných řetězců V^+. Pokud bychom chtěli přidat k této množině i prázdný řetězec (ε), vznikla by nám množina V^*, pro kterou platí $V^* = V^+ + \{\varepsilon\}$.

Pro jakýkoliv řetězec $w \in V^*$, nám $|w|$ označuje jeho délku, tedy počet všech symbolů, které se v něm nacházejí. Dále můžeme zavést $alph(w)$ označující množinu všech symbolů, které se v daném řetězci w vyskytují. Pro každou množinu symbolů $W \subseteq V$, funkce $occur(w, W)$ značí počet výskytů symbolů z množiny W v řetězci w. Pro $i = 1, \ldots, |w|$, $[w, i, W]$ označuje i -tý výskyt některého symbolu z množiny W v řetězci w. Pokud takový výskyt neexistuje, pak $[w, i, W] = 0$. Například $[aabbc, 2, \{a, c\}]$ označuje podtržené $\underline{a}$ v $a\underline{a}bbc$. Pro každé $i \geq 0$, je $prefix(w, i)$ předpona řetězce w o délce i pokud je délka řetězce $|w| \geq i$ nebo $prefix(w, i) = w$ pokud je řetězec kratší, $i > |w|$.

Pojmem *relace* nazýváme libovolbý vztah mezi skupinou prvků množin. *Zobrazení* je předpis jakým způsobem jednoznačně přiřazovat prvkům jedné množiny prvky jiné množiny. *Homomorfizmus* je zobrazení z jedné algebraické struktury do jiné stejného typu. Příkladem může být Morseova abeceda, kde je každému písmenu přiřazena sekvence $\{\bullet, -\}^*$. Homomorfizmus f nad V^* je takové zobrazení, kdy $f(A) \in \{A, \varepsilon\}$ pro všechna $A \in V$. *Bijektivní zobrazení* (bijekce) přiřazuje každému prvku první množiny vždy právě jeden prvek druhé množiny.

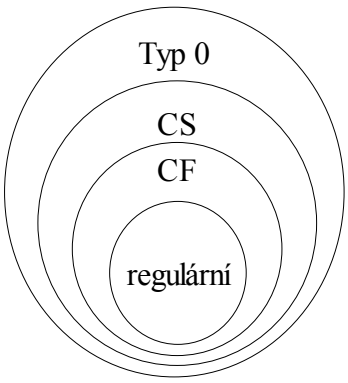
3 Jazyky

Jazyk je z formálního hlediska podmnožina množiny všech možných řetězců nad danou abecedou. Máme-li abecedu Σ a jazyk L, můžeme tuto skutečnost zapsat jako $L \subseteq \Sigma^*$. Jazyky dělíme na konečné a nekonečné podle toho, zda obsahuje konečný počet řetězců či nikoliv. Příkladem nekonečného jazyka může být takováto situace – máme abecedu $\Sigma = \{a, b, c\}$ a jazyk definovaný $L = \{x | ab \text{ je podřetězec } x\}$. Mezi konečné jazyky patří i $L = \emptyset$, jeho kardinalita je 0 (žádný řetězec), a $L = \{\varepsilon\}$ s kardinalitou 1 (jeden prázdný řetězec).

Jazyky můžeme popsat dvěma základními modely. Prvním modelem jsou gramatiky. Ty podle svých gramatických pravidel generují řetězce daného jazyka. Druhým modelem jsou automaty. Automaty na rozdíl od gramatik řetězce nevytvářejí, ale podle svých pravidel rozhodují, zda vstupní řetězec patří do jazyka popisovaného tímto automatem.

3.1 Chomského klasifikace

Noam Chomsky zavedl v roce 1956 hierarchii formálních gramatik generujících rekurzivně spočetné formální jazyky. Tyto gramatiky rozdělil do čtyř základních typů. Těmi nejobecnějšími jsou gramatiky typu 0, které mohou být rozpoznávány Turingovým strojem. Turingův stroj je složen z konečného automatu, pravidel přechodové funkce a teoreticky nekonečné pásky pro zápis mezivýsledků. Podmnožinou gramatik typu 0 jsou kontextové gramatiky (CSG), známé také pod označením typ 1, podle nichž jsou pojmenovány i kontextové jazyky (dále CS). K jejich přijetí stačí omezenější verze Turingova stroje, tzv. lineárně ohraničený Turingův stroj (už nemá nekonečnou pásku, její délka je lineárně závislá na délce vstupního řetězce). Bezkontextové gramatiky (CFG) jsou podmnožinou kontextových, zřídka se u nich setkáme s označením typ 2. S bezkontextovými jazyky (CF) se setkáváme poměrně často, například do této skupiny patří programovací jazyky, pro něž v každém překladači existuje syntaktický analyzátor což v praxi není nic jiného než zásobníkový automat. Jazyky rozpoznávané zásobníkovým automatem označujeme PD. Modifikace, hluboký a paralelní hluboký, zapíšeme ${}_{deep}PD$ resp. ${}_{par.deep}PD$. V případě přijetí pomocí vyprázdnění zásobníku označíme tento jazyk jako ${}^{empty}PD$ resp. ${}^{empty}_{deep}PD$ resp. ${}^{empty}_{par.deep}PD$. Posledním typem dle Chomského klasifikace jsou regulární gramatiky a jejich regulární jazyky. S nimi se v praxi setkáváme téměř nejčastěji, patří sem veškeré regulární a výpočetní výrazy. Jsou přijímány konečnými automaty. Každý typ této klasifikace je přímou nadmnožinou dalšího typu (viz. obrázek).



4 Gramatiky

Gramatiky jsou jedním ze stavebních prvků teorie formálních jazyků. Jádrem každé gramatiky jsou takzvaná gramatická pravidla, jejichž pomocí dochází ke generování řetězců patřících do jazyka, který tato gramatika popisuje. V minulosti bylo na rozdíl od automatů zavedeno velké množství modifikací. My si uvedeme dvě základní, které jsou pro tuto práci zajímavé.

4.1 Bezkontextová gramatika

Bezkontextové gramatiky jsou schopny generovat jazyky typu 2 podle Chomského klasifikace, tedy jazyky bezkontextové. Příkladem mohou být programovací jazyky, které právě do této skupiny patří. V porovnání s regulárními výrazy jsou tedy silnější. Typickým jazykem, který je generován bezkontextovou gramatikou je $L(G)=\{a^n b^n | n \geq 1\}$.

Formálně je bezkontextová gramatika čtveřice $G=(N, T, P, S)$,

kde

N je množina neterminálních symbolů

T je množina terminálních symbolů, $T \cap N = \emptyset$

$S \in N$ je počáteční neterminál

P je množina gramatických pravidel tvaru $A \to x$, $A \in N$, $x \in (N \cup T)^*$

4.2 n-omezené stavové gramatiky

Tyto gramatiky jsou silnější než předchozí. Mají tu zajímavou vlastnost, že tvoří nekonečnou hierarchii tříd jazyků mezi jazyky bezkontextovými a kontextovými.

Stavová gramatika je pětice $G=(V, W, T, P, S)$,

kde

V je úplná abeceda,

W je konečná množina stavů,

$T \subseteq V$ je abeceda terminálních symbolů,

$S \in (V-T)$ je startující symbol

a $P \subseteq (W \times (V-T)) \times (W \times V^+)$ je konečná relace. Přehledněji zapisujeme pravidla ve tvaru $(q, A) \to (p, v) \in P$ místo matematicky přesnějšího $(q, A, p, v) \in P$.

Každému řetězci $z \in V^*$ nastavíme množinu $states_G(z)=\{q | (q, B) \to (p, v) \in P\}$, kde $B \in (V-T) \cap alph(z)$, tedy množina neterminálů vyskytujících se v tomto řetězci, $v \in V^+$ je neprázdný řetězec a $q, p \in W$ jsou stavy. Pokud máme pravidlo $(q, A) \to (p, v) \in P$, řetězce

$x, y \in V^*$ a množinu $states_G(x) \cap \{q\} = \emptyset$, pak gramatika G provede *derivační krok* z (q, xAy) do (p, xvy). Tuto skutečnost symbolicky zapisujeme $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$. Dodáme-li kladné celé číslo n splňující podmínku $occur(xA, V - T) \leq n$, říkáme, že $(q, xAy) \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$ je n-omezené, symbolicky zapsáno $(q, xAy)_n \Rightarrow (p, xvy)$ $[(q, A) \rightarrow (p, v)]$. Jestli nehrozí možnost záměny, můžeme zkráceně psát $(q, xAy) \Rightarrow (p, xvy)$ a $(q, xAy)_n \Rightarrow (p, xvy)$, tedy bez určení podle kterého pravidla k derivaci došlo.

Obvyklým způsobem můžeme rozšířit derivační krok $\Rightarrow$ na sérii m derivačních kroků $\Rightarrow^m$, pro celé číslo $m \geq 0$, určující počet těchto kroků. Dále můžeme zavést $\Rightarrow^+$, kdy je proveden alespoň jeden derivační krok a $\Rightarrow^*$, kdy nemusí být proveden žádný derivační krok (pokud není proveden žádný, musí být obě strany stejné).

Mějme $n \in I$ a $\nu, \varpi \in (W \times V^+)$. Abychom vyjádřili, že je každý derivační krok v $\nu \Rightarrow^m \varpi$, $\nu \Rightarrow^+ \varpi$ a $\nu \Rightarrow^* \varpi$ n-omezený, píšeme často symboly derivací jako $\nu_n \Rightarrow^m \varpi$, $\nu_n \Rightarrow^+ \varpi$ a $\nu_n \Rightarrow^* \varpi$. Zápisem $strings(\nu_n \Rightarrow^* \varpi)$ vyjadřujeme množinu všech řetězců vyskytujících se v derivaci $\nu_n \Rightarrow^* \varpi$. Jazyk, generovaný gramatikou G, $L(G)$ je definován jako $L(G) = \{w \in T^* | (q, S) \Rightarrow^* (p, w), p, q \in W\}$. Kromě toho můžeme definovat jazyk generovaný n-omezenou gramatikou pro všechna $n \geq 1$ jako $L(G, n) = \{w \in T^* | (q, S)_n \Rightarrow^* (p, w), p, q \in W\}$. Derivace tvaru $(q, S)_n \Rightarrow^* (p, w)$, kde $p, q \in W$ a $w \in T^*$, reprezentuje úspěšné n-omezené generování řetězce w v gramatice G.

5 Automaty

V této kapitole se podíváme na vývoj automatů. Automat na rozdíl od gramatiky slouží především k rozhodování, zda daný řetězec patří do zadaného jazyka či nikoliv. Vzhledem k tomu, že tato práce má představit modifikovaný automat, neškodilo by si uvést, jak se postupně automaty jako modely pro popis jazyků vyvíjely, jaké měly vlastnosti a jaké jazyky přijímaly. Nejstarším typem automatů je konečný automat.

5.1 Konečný automat

Konečný automat je výpočetní model využívaný ke studiu formálních jazyků. Popisuje velice jednoduchý počítač přecházející mezi několika stavy pouze na základě aktuálního vstupního symbolu, získaného ze vstupní pásky. Tento automat tedy nepoužívá žádnou další paměť. Veškeré rozhodování je založeno pouze na znalosti aktuálního stavu a aktuálního vstupního symbolu. Vzhledem ke své jednoduchosti je schopen rozpoznat pouze nejjednodušší, tzv. regulární, jazyky a je tedy vhodným nástrojem například pro zpracování regulárních výrazů.

5.1.1 Definice

Konečný automat je uspořádaná pětice $M=(Q,\Sigma,R,s,F)$,

kde

Q je konečná množina stavů,

Σ je konečná množina vstupních symbolů, nazývaná vstupní abeceda,

$s\in Q$ je startující (počáteční) stav,

$F\subseteq Q$ je konečná množina koncových stavů

a R je konečná množina pravidel. Čistě matematicky je R relací z $Q\times(\Sigma\cup\{\varepsilon\})$ do Q. Místo zápisu $(pa,q)\in R$ se častěji používá tvar $(pa\to q)\in R$, $p,q\in Q$, $a\in(\Sigma\cup\{\varepsilon\})$.

5.2 Zásobníkový automat

Zásobníkový automat je rozšířením konečného automatu. Toto rozšíření spočívá především v přidání zásobníku jako paměti. Zásobník dává tomuto automatu možnost rozhodovat se nejen na základě vstupního symbolu a stavu, ale také podle symbolu umístěného na vrcholu tohoto zásobníku. Tímto rozšířením bylo dosaženo výrazného zvětšení množiny rozpoznávaných jazyků a to až na množinu tzv. bezkontextových jazyků, podle Chomského klasifikace se jedná o jazyky typu 2. Se zásobníkem

mohou být prováděny dvě základní operace – expanze a vyjmutí. Expanze spočívá v nahrazení nevstupního symbolu nacházejícího se na vrcholu zásobníku řetězcem symbolů zásobníkové abecedy a přechodu do jiného stavu. Obě činnosti odpovídají právě použitému pravidlu. K vyjmutí může dojít pouze v situaci, kdy je na vrcholu zásobníku symbol ze vstupní abecedy a zároveň je stejný symbol na vstupu automatu. Daný symbol bude odebrán z vrcholu zásobníku a zároveň bude přečten ze vstupu. Přečtením se rozumí posun čtecí hlavy po vstupní pásce na následující symbol.

5.2.1 Definice

Zásobníkový automat je uspořádaná sedmice $M=(Q,\Sigma,\Gamma,R,s,S,F)$,

kde Q je konečná množina stavů,

Σ je vstupní abeceda,

Γ je konečná množina zásobníkových symbolů, nazývaná zásobníková abeceda. Platí, že $\Sigma\subset\Gamma$ a $\Gamma-\Sigma$ obsahuje speciální symbol $\#$, značící dno zásobníku,

stav $s\in Q$ je startující stav,

$S\in\Gamma$ je počáteční zásobníkový symbol,

$F\subseteq Q$ je konečná množina koncových stavů

a R je konečná množina pravidel tvaru $(Apa\rightarrow wq)\in R$, $A\in\Gamma$, $p,q\in Q$, $a\in(\Sigma\cup\{\varepsilon\})$, $w\in\Gamma^*$.

5.3 Hluboký zásobníkový automat

Hluboký zásobníkový automat je rozšířenou verzí klasického zásobníkového automatu. Stejně jako on, používá ke své činnosti zásobník, nad nímž provádí dvě základní operace (expanzi a vyjmutí). Hlavním rozdílem je však fakt, že nekontroluje pouze nejhornější zásobníkový symbol, ale může provádět veškeré operace i hlouběji (omezeno podle typu automatu). Tímto je dosaženo zvýšení tzv. síly, tento automat je schopen generovat i některé kontextové jazyky, na rozdíl od základních zásobníkových automatů schopných generovat jen třídu bezkontextových jazyků. Tato obecná verze je nedeterministická a nepovoluje vymazávací pravidla (přepis neterminálu prázdným řetězcem).

5.3.1 Definice

Hluboký zásobníkový automat je uspořádaná sedmice $M=(Q,\Sigma,\Gamma,R,s,S,F)$,

kde

Q je konečná množina stavů,

Σ je vstupní abeceda,

Γ je konečná množina zásobníkových symbolů, nazývaná zásobníková abeceda, $\Sigma\subset\Gamma$, $\Gamma-\Sigma=\{\#\}$,

stav $s \in Q$ je startující stav,

$S \in \Gamma$ je počáteční zásobníkový symbol,

$F \subseteq Q$ je konečná množina koncových stavů

a R je konečná množina pravidel tvaru $(mqA \to pv) \in R$, $A \in \Gamma$, $p, q \in Q$, $0 < m \leq n$, $v \in \Gamma^*$.

5.3.2 Příklad

Uvažujme hluboký zásobníkový automat ${}_2M = (\{s, p, q, f\}, \{a, b, c\}, \{A, S, \#\}, R, s, S, \{f\})$, který má v R následující pravidla:

[1] $1\text{sS} \to qAA$,

[2] $1\text{qA} \to paAb$,

[3] $1\text{qA} \to fab$,

[4] $2\text{pA} \to qAc$,

[5] $1\text{fA} \to fc$.

Se vstupním řetězcem $aaabbbccc$ provede automat M tyto kroky

$(s, aaabbbccc, S\#) \;{}_e\Rightarrow (q, aaabbbccc, AA\#)$ [1]

${}_e\Rightarrow (p, aaabbbccc, aAbA\#)$ [2]

${}_p\Rightarrow (p, aabbbccc, AbA\#)$

${}_e\Rightarrow (q, aabbbccc, AbAc\#)$ [4]

${}_e\Rightarrow (p, aabbbccc, aAbbAc\#)$ [2]

${}_p\Rightarrow (p, abbbccc, AbbAc\#)$

${}_e\Rightarrow (q, abbbccc, AbbAcc\#)$ [4]

${}_e\Rightarrow (f, abbbccc, abbbAcc\#)$ [3]

${}_p\Rightarrow (f, bbbccc, bbbAcc\#)$

${}_e\Rightarrow (f, bbbccc, bbbccc\#)$ [5]

${}_p\Rightarrow (f, bbccc, bbccc\#)$

${}_p\Rightarrow (f, bccc, bccc\#)$

${}_p\Rightarrow (f, ccc, ccc\#)$

${}_p\Rightarrow (f, cc, cc\#)$

${}_p\Rightarrow (f, c, c\#)$

${}_p\Rightarrow (f, \varepsilon, \#)$

tedy platí $(s, aabbcc, S\#) \Rightarrow^{*} (f, \varepsilon, \#)$ a zadaný řetězec byl přijat po 16 krocích (z toho bylo 7 expanzí a 9 vyjmutí). Dále si můžeme všimnout, že jazyk přijímaný tímto automatem $L({}_2M) = \{a^n b^n c^n | n \geq 1\}$ patří do skupiny jazyků definované jako $CS - CF$.

6 Paralelní hluboký zásobníkový automat

V této části bude uvedena neformální i formální definice paralelního hlubokého zásobníkového automatu, ukážeme si co je u tohoto automatu konfigurace a jakým způsobem může mezi jednotlivými konfiguracemi přecházet. Na závěr bude uveden příklad postupného derivování zadaného řetězce konkrétním automatem až k jeho přijetí. Stejně jako u hlubokých zásobníkových automatů i tato obecná verze funguje nedeterministicky a nepovoluje vymazávací pravidla.

6.1 Neformální definice

Paralelní hluboký zásobníkový automat se od hlubokého zásobníkového automatu odlišuje tím, že v jednom kroku může provést expanzi několika nejhornějších neterminálních symbolů na zásobníku. Toho je dosaženo především tvarem pravidel. Každé z nich je složeno z několika jednodušších akcí spočívajících v přepsání neterminálního symbolu v dané hloubce řetězcem. První neterminál v pravidle odpovídá nejhornějšímu neterminálu na zásobníku, druhý odpovídá druhému neterminálu na zásobníku, atd. až do maximální hloubky pravidla, která je menší nebo rovna maximální hloubce ve které může dojít, specifické pro tento automat. Číslování neterminálů na zásobníku se shoduje se stavem před použitím pravidla. Pravidlo můžeme použít jen v případě, kdy je na zásobníku dostatečný počet neterminálů a kdy jejich uspořádání souhlasí s uspořádáním v pravidle.

6.2 Formální definice

Paralelní hluboký zásobníkový automat je sedmice

$$_{deep}M_{par}=(Q,\Sigma,\Gamma,R,s,S,F),$$

kde mají jednotlivé symboly následující význam:

deep	maximální hloubka, ve které může dojít k nahrazení
Q	konečná množina stavů
Σ	vstupní abeceda
Γ	zásobníková abeceda
	$\Sigma\subseteq\Gamma, \#\in(\Gamma-\Sigma)$ # je speciální zásobníkový symbol značící dno zásobníku
s	startující stav
S	počáteční zásobníkový symbol $S\in(\Gamma-\Sigma)$
F	konečná množina koncových stavů $F\subseteq Q$

R konečná množina pravidel tvaru

$$p(A_{1,}A_{2,}...A_i) \rightarrow q(v_{1,}v_{2,}...v_i)$$

kde

$p \in Q$	výchozí stav (před použitím pravidla)
$q \in Q$	koncový stav (po použití pravidla)
$A_j \in \Gamma - (\# \cup \Sigma)$	neterminální symboly
$v_j \in \Gamma^*$	řetězec složený z terminálních a neterminálních symbolů
$i \leq deep$	maximální hloubka daného pravidla, pravidlo je možné použít pouze v případě, že je na zásobníku nejméně i neterminálních symbolů
$1 \leq j \leq i$	hloubka daného neterminálního symbolu
$i, j, deep \in I$	kladná celá čísla

6.3 Konfigurace a přechody

Konfigurace paralelního hlubokého zásobníkového automatu musí obsahovat všechny potřebné údaje se kterými automat pracuje. Těmito údaji jsou aktuální stav, stav zásobníku (řetězec zásobníkových symbolů) a nezpracovaná část vstupního řetězce. Formálně je to trojice z $Q \times \Sigma^* \times (\Gamma - \{\#\})^* \{\#\}$. Zjednodušeně můžeme říci, že se dá zapsat následovně:

$$(aktuální\ stav, nepřečtená\ část\ vstupního\ řetězce, stav\ zásobníku)$$

Označme χ množinu všech možných konfigurací daného automatu. Dále pak označme $x, y \in \chi$. Přechody mezi jednotlivými konfiguracemi značíme $x \Rightarrow y$ a pokud chceme zkrátit zápis, například přechod ze startovní konfigurace do koncové bez určení přesného počtu kroků, můžeme použít $x \Rightarrow^* y$. Přesný počet kroků můžeme zapsat obdobně, jen s tím rozdílem, že nahradíme znak $*$ číslem vyjadřujícím počet daných přechodů.

Automat může provádět dvě operace ovlivňující stav zásobníku. První operací je expanze, označovaná $x_e \Rightarrow y$, kdy dojde k nahrazení některého (případně několika) nevstupního zásobníkového symbolu řetězcem zásobníkových symbolů. Tento řetězec může být i prázdný (ε). Jinými slovy, při expanzi dochází k aplikaci pravidla. Druhou operací je vyjmutí terminálního symbolu z vrcholu zásobníku. To nastane pouze za podmínky, že je shodný s aktuálním vstupním symbolem. Tato činnost bývá označována $x_p \Rightarrow y$. Při vyjmutí v žádném případě nedochází k použití pravidla a ani ke změně aktuálního stavu. Nedílnou součástí je současné přečtení vstupního symbolu (posun čtecí hlavy). Operaci vyjmutí také můžeme nalézt pod anglickým výrazem *pop*.

6.4 Příklad

Mějme paralelní hluboký zásobníkový automat ${}_3M_{par}=(Q,\Sigma,\Gamma,R,s,S,F)$,

$Q=\{s,p,f\}$,

$\Sigma=\{a,b,c\}$,

$\Gamma=\{a,b,c,A,S,\#\}$,

$F=\{f\}$,

$R=\{[1]sS\to pAAA,[2]p(A,A,A)\to p(aA,bA,cA),[3]p(A,A,A)\to f(a,b,c)\}$.

S řetězcem $aabbcc\in L=\{a^n b^n c^n, n\geq 1\}\subset CS-CF$ provede automat M následující kroky (číslo v hranatých závorkách určuje použité pravidlo):

$$
\begin{aligned}
(s,aabbcc,S\#) &\;{}_e\Rightarrow (p,aabbcc,AAA\#)\ [1]\\
&\;{}_e\Rightarrow (p,aabbcc,aAbAcA\#)\ [2]\\
&\;{}_p\Rightarrow (p,abbcc,AbAcA\#)\\
&\;{}_e\Rightarrow (f,abbcc,abbcc\#)\ [3]\\
&\;{}_e\Rightarrow (f,bbcc,bbcc\#)\\
&\;{}_p\Rightarrow (f,bcc,bcc\#)\\
&\;{}_p\Rightarrow (f,cc,cc\#)\\
&\;{}_p\Rightarrow (f,c,c\#)\\
&\;{}_p\Rightarrow (f,\varepsilon,\#)
\end{aligned}
$$

zkráceně $(s,aabbcc,S\#)\Rightarrow^{*}(f,\varepsilon,\#)$, což znamená, že zadaný řetězec byl přijat.

7 Vlastnosti

V této kapitole se podíváme na vlastnosti paralelních hlubokých zásobníkových automatů. Nejprve si ukážeme, že paralelní hluboký zásobníkový automat je stejně silný jako neparalelní verze.

7.1 Síla

Lemma 1:

Pro každou stavovou gramatiku G a pro všechna $n \geq 1$ existuje paralelní hluboký zásobníkový automat hloubky n, ${}_nM$, který přijímá stejný jazyk jako generuje tato gramatika, platí tedy $L(G,n) = L({}_nM)$.

Důkaz:

Mějme stavovou gramatiku $G = (V, W, T, P, S)$, číslo $n \geq 1$ a množinu neterminálních symbolů $N = V - T$. Definujme homomorfizmus f nad řetězcem tvořeným symboly úplné abecedy a dna zásobníku, $(\{\#\} \cup V)^*$, jako $f(A) = A$ pro každé $A \in \{\#\} \cup N$, a $f(a) = \varepsilon$ pro každé $a \in T$. Zaveďme paralelní hluboký zásobníkový automat hloubky *n*

$$ {}_nM = (Q, \Sigma, \Gamma, R, s, S, \{\$\}) ,$$

kde

$$Q = \{s, \$\} \cup \{\langle p, u \rangle | p \in W, u \in prefix(N^*\{\#^n\}, n) |u| \leq n\} ,$$

$$\Sigma = T,$$

$$\Gamma = \{\#\} \cup V$$

a R je konstruováno následujícím postupem:

1, vytvoříme množinu R_a

2, pro každé $(p,S) \rightarrow (q,x) \in P$, $p, q \in W$, $x \in V^+$ přidáme $1sS \rightarrow \langle p, S \rangle S$ do R_a

3, pokud máme gramatické pravidlo $(p, A) \rightarrow (q, x) \in P$, $\langle p, uAv \rangle \in Q$, $p, q \in W$, $A \in N$, $x \in V^+$, $u \in N^*$, $v \in N^*\{\#\}^*$, $|uAv| = n$, $p \notin_G states(u)$, přidáme do R_a $|uA| \langle p, uAv \rangle A \rightarrow \langle q, prefix(uf(x)v, n) \rangle x$

4, pokud platí $A \in N$, $p \in W$, $u \in N^*$, $v \in \{\#\}^*$, $|uv| \leq n-1$, $p \notin_G states(u)$, pak přidáme do R_a pravidla $|uA| \langle p, uv \rangle A \rightarrow \langle p, uAv \rangle A$ a $|uA| \langle p, uv \rangle \# \rightarrow \langle p, uv \rangle \#$

5, pro každý stav $q \in W$ přidáme $1 \langle q, \#^n \rangle \# \rightarrow \$\#$ do R_a

6, vytvořit množinu R_b

7, pro všechny stavy $q \in Q$, každý neterminál $A \in N$, a každé celé číslo i, $0 < i < n$ přidat pravidlo $iqA \rightarrow qA$ do R_b

8, všechna pravidla $mr \in R_a \cup R_b$ hloubky m přidat do R_m

9, $m = n$

10, z R_m odebrat všechna pravidla nacházející se i v R_b, $R_m = R_m - R_b$

11, do R přidat paralelní pravidla $R_1 \times R_2 \times R_3 ... \times R_m$ (zapsána v požadovaném tvaru), taková kdy cílový stav pravidla hloubky i je shodný s výchozím stavem pravidla hloubky $i+1$ $(1 < i \leq m)$. Výchozí stav paralelního pravidla je shodný s výchozím stavem pravidla hloubky 1 a cílový stav je shodný s cílovým stavem pravidla hloubky m.

12, $m = m - 1$

13, opakovat od bodu 10, dokud $m\,! = 0$

Tento algoritmus nejprve vytvoří množinu pravidel R_a stejným způsobem jako by ji vytvořil klasický hluboký zásobníkový automat (body 1-5). Následně se vytvoří množina pravidel R_b přepisujících ve všech hloubkách a stavech všechny neterminály na ně samotné ($iqA \rightarrow qA$). Může se zdát, že takováto pravidla nemají žádný přínos, ale pokud se zamyslíme nad tvarem pravidel paralelních hlubokých zásobníkových automatů ($p(A, A, A) \rightarrow p(aA, bA, cA)$), kdy první neterminál je přepsán na první řetězec, druhý neterminál na druhý řetězec, atd., tak již nějaký smysl mají. To především v situaci, kdy v množině R_a bude pravidlo přepisující neterminál v hloubce i, ale budou chybět pravidla například pro hloubku $i-1$. V takovém případě by již paralelní verze byla výrazně slabší, což by rozhodně nebylo ideální. Tento krok nám zajistí možnost mít i pravidla typu $p(A, A, A, A) \rightarrow p(A, A, A, w)$, což je přesně to co potřebujeme. Následně již můžeme pravidla z těchto dvou množin skládat dohromady, tak aby již byla požadovaného tvaru a především aby splňovala podmínku o navazujících stavech. Ještě zde nastává jedno drobné omezení – poslední (nejhlubší) použité „podpravidlo“ musí být z množiny R_a. Pravidla tak mohou mít různé délky a v žádném případě je nesmíme doplňovat pravidly z R_b tak aby byla delší. Tedy nesmí existovat pravidla tvaru $p(A, B, B, B, B) \rightarrow p(w, B, B, B, B)$, kde nahrazujeme jen první neterminál, ale abychom toto nahrazení mohli provést, museli bychom mít na zásobníku nejméně pět dalších neterminálů a ještě k tomu přesně takových, jako jsou v pravidle uvedeny. Samozřejmě je toto nevhodné pravidlo pouze ilustrační příklad, přepis na řetězec by rovněž mohl nastat i u jiného neterminálu.

Základní princip:

Paralelní hluboký zásobníkový automat ${}_nM_{par}$ simuluje n -omezenou derivaci gramatiky G. Vždy si do stavu zaznamená prvních n neterminálů vyskytujících se v aktuální větné formě a pokud jich je méně než n, jsou zbývající pozice doplněny mřížkami ($\#$). ${}_nM_{par}$ na zásobníku simuluje derivační kroky a zároveň si do svého stavu zaznamenává nově vygenerované neterminální symboly. Když gramatika G úspěšně dokončí generování řetězce terminálních symbolů, automat ${}_nM_{par}$ dokončí čtení řetězce, vyprázdní zásobník a přejde do koncového stavu.

Lemma 2:

Pro každé $n \geq 1$ a každý paralelní hluboký zásobníkový automat ${}_nM_{par}$ existuje stavová gramatika, která popisuje stejný jazyk jako tento automat a platí tedy $L(G,n) = L({}_nM_{par})$.

Důkaz:

Mějme $n \geq 1$ a ${}_nM_{par} = (Q, \Sigma, \Gamma, R, s, S, F)$. Zaveďme dva nové symboly Z a $\$$, které se nevyskytují v žádné součásti tohoto automatu. Písmenem V označme množinu všech zásobníkových symbolů, $V = \Gamma$, písmenem T označme množinu terminálních symbolů, $T = \Sigma$ a písmenem N označme množinu neterminálních symbolů, tedy $N = V - T$. Vytvořme nové množiny $C = \{\langle q, i, \triangleright\rangle | q \in Q, 1 \leq i \leq n-1\}$, $D = \{\langle q, i, \triangleleft\rangle | q \in Q, 0 \leq i \leq n-1\}$, abecedu W takovou, že její kardinalita je stejná jako kardinalita V, $card(V) = card(W)$. Dále pro všechna i splňující podmínku $1 \leq i \leq n$ vytvořme abecedu U_i, takovou, že $card(U_i) = card(N)$. Beze ztráty na obecnosti rozumějme, že V, Q a všechny nově zavedené množiny a abecedy jsou navzájem disjunktní. Sjednoťme všechny abecedy U_i do U, $U = \cup_{i=1}^{n} U_i$. Nyní zaveďme bijekce ${}_ig$ z N do U_i pro všechna $1 \leq i \leq n$ a bijekci h z V do W. V tomto okamžiku můžeme definovat stavovou gramatiku

$$G = (V \cup W \cup U \cup \{Z\}, Q \cup C \cup D \cup \{\$\}, T, P, Z),$$

kde množinu gramatických pravidel P můžeme zkonstruovat následujícím způsobem:

1, přidáme pravidlo $(s, Z) \rightarrow (\langle s, 1, \triangleright\rangle, h(S))$

2, pro všechny $q \in Q$, $A \in N$, $1 \leq i \leq n-1$, $x \in V^+$ přidáme

a, $(\langle q, i, \triangleright\rangle, A) \rightarrow (\langle q, i, \triangleright\rangle, {}_ig(A))$

b, $(\langle q, i, \triangleleft\rangle, {}_ig(A)) \rightarrow (\langle q, i, \triangleleft\rangle, A)$

3, pro pravidlo $p(A_1, A_2, \ldots A_i) \rightarrow q(v_1 Y_1, v_2 Y_2, \ldots, v_i Y_i)$, $p, q \in Q$, $m = 1, \ldots, i$, $A \in N$, $v_m \in V^*$, $Y_m \in V$, přidáme

$(\langle p, i, \triangleright\rangle, A) \rightarrow (\langle q, i-1, \triangleleft\rangle, vY)$

a $(\langle p, i, \triangleright\rangle, h(A)) \rightarrow (\langle q, i-1, \triangleleft\rangle, vh(Y))$

4, pro všechny stavy $q \in Q$ a neterminály $A \in N$ přidáme $(\langle q, 0, \lhd \rangle, A) \rightarrow (\langle q, i, \rhd \rangle, A)$ a $(\langle q, 0, \lhd \rangle, h(Y)) \rightarrow (\langle q, 1, \rhd \rangle, h(Y))$

5, pro všechny koncové stavy $q \in F$ a vstupní symboly $a \in T$ přidáme $(\langle q, 0, \lhd \rangle, h(A)) \rightarrow (\$, a)$

Základní princip:

Gramatika G simuluje aplikaci pravidla $p(A_1, A_2, \ldots A_i) \rightarrow q(v_1 Y_1, v_2 Y_2, \ldots, v_i Y_i)$ tak, že prochází větnou formu zleva doprava. Při tom počítá neterminální symboly. Pokud najde i-tý neterminál, porovná jej s neterminálem pro hloubku i daného pravidla a pokud jsou shodné, nahradí jej řetězcem $v_i Y_i$. Směr procházení se změní a hledá se neterminál hloubky $i-1$. Souhlasí-li neterminál ve větné formě a v pravidle, dojde k nahrazení. Tato situace se opakuje, až dokud se nedojde na začátek větné formy (bez dalších změn směru). Pokud se podařilo úspěšně nahradit všech i neterminálních symbolů dojde ke změně stavu na q a otočení směru procházení. Následně dochází k simulaci dalších pravidel. Pokud však dojde k nějaké chybě, jako je nemožnost přepsání některého neterminálu, obnoví se stav větné formy do podoby před použitím pravidla. Po úspěšném přijetí řetězce dojde k použití pravidla z bodu 5 předchozího postupu, a vygenerování řetězce.

Teorém 1:

Pro každé $n \geq 1$ a každý jazyk L, platí že je generovaný n-omezenou stavovou gramatikou G, $L = L(G, n)$ jen v případě, že je přijímán paralelním hlubokým zásobníkovým automatem ${}_n M_{par}$, $L = L({}_n M_{par})$.

Toto tvrzení je založeno na spojení výše uvedených lemmat.

Teorém 2:

Paralelní hluboký zásobníkový automat je stejně silný jako neparalelní, ${}_{par.deep}PD_n = {}_{deep}PD_n$

U tohoto tvrzení vycházíme z předchozího teorému a z vlastností hlubokých zásobníkových automatů, pro něž platí prakticky stejně znějící teorém.

Teorém 3:

$${}^{empty}_{par.deep}PD_n = {}_{par.deep}PD_n \subset {}_{par.deep}PD_{n+1} = {}^{empty}_{par.deep}PD_{n+1}$$

Opět vychází z výše uvedených lemmat a z vlastnosti n-omezených stavových gramatik, generujících nekonečnou hierarchii tříd jazyků mezi třídou bezkontextových a kontextových, kdy každá třída je nedílnou součástí své nadtřídy.

Teorém 4:

$${}_{par.deep}PD_1 = {}^{empty}_{par.deep}PD_1 = CF$$

Pokud omezíme hloubku zásobníku na hodnotu 1, stane se z jakéhokoliv hlubokého zásobníkového automatu běžný zásobníkový automat, jehož síla stačí pouze na bezkontextové jazyky.

Teorém 5:

Pro každé $n \geq 1$ platí ${}_{par.deep}PD_n = {}^{empty}_{par.deep}PD_n \subset CS$

Obdobně jako u teorému 3 je za základ použita nekonečná hierarchie tříd jazyků mezi bezkontextovými a kontextovými jazyky generovaná n-omezenými stavovými gramatikami.

7.2 Převod hlubokého zásobníkového automatu na paralelní verzi

Nyní si ukážeme jakým způsobem můžeme převést klasický hluboký zásobníkový automat na paralelní verzi. Vyjdeme při tom z postupů uvedených v lemmatu 1 této práce a lemmatu 1 z článku prof. Meduny [2]. Jako zdrojový automat nám poslouží ten, který najdeme v kapitole popisující hluboké zásobníkové automaty na straně 9.

Zaveďme ${}_2M_{par} = (\{s, p, q, f\}, \{a, b, c\}, \{A, S, \#\}, R, s, S, \{f\})$. Do tohoto bodu je definice stejná. Rozdíl nastane až v množině pravidel R, kterou vytvoříme následovně:

1, Pravidla původního (neparalelního) automatu vložíme do množiny R_a.

$R_a = \{1\mathrm{sS} \to qAA, 1\mathrm{qA} \to paAb, 1\mathrm{qA} \to fab, 2\mathrm{pA} \to qAc, 1\mathrm{fA} \to fc\}$

2, Vytvoříme množinu R_b do které vložíme pomocná pravidla $1\mathrm{sA} \to sA$, $1\mathrm{sS} \to sS$, $1\mathrm{pA} \to pA$, $1\mathrm{pS} \to pS$, $1\mathrm{qA} \to qA$, $1\mathrm{qS} \to qS$, $1\mathrm{fA} \to fA$, $1\mathrm{fS} \to fS$. Všechna mají hloubku 1 protože má platit podmínka $0 < i < n$.

3, Pravidla si rozdělíme podle hloubky do dvou množin (maximální hloubka je 2). Množina R_1 tedy obsahuje pravidla $1\mathrm{sS} \to qAA$, $1\mathrm{qA} \to paAb$, $1\mathrm{qA} \to fab$, $1\mathrm{fA} \to fc$, $1\mathrm{sA} \to sA$, $1\mathrm{sS} \to sS$, $1\mathrm{pA} \to pA$, $1\mathrm{pS} \to pS$, $1\mathrm{qA} \to qA$ a $1\mathrm{qS} \to qS$. Množina R_2 obsahuje pouze jediné pravidlo $2\mathrm{pA} \to qAc$.

4, Nyní již můžeme provést kombinace těchto pravidel z čehož nám vyjde množina R obsahující $s(S) \to q(AA)$, $q(A) \to p(aAb)$, $q(A) \to f(ab)$, $f(A) \to f(c)$ pro hloubku 1 a $q(A, A) \to q(aAb, Ac)$, $p(A, A) \to q(A, Ac)$, $p(S, A) \to q(S, Ac)$ pro hloubku 2.

7.3 Rychlost

Paralelní hluboký zásobníkový automat je obecně schopen přijmout řetězec rychleji. Toto tvrzení však neplatí pro automat hloubky 1, protože se z něj, stejně jako z neparalelní verze, stává běžný zásobníkový automat. Zrychlení proti neparalelní verzi je především dáno tvarem pravidla, kdy v závislosti na hloubce můžeme provést současně přepis více neterminálů, což běžný hluboký

zásobníkový automat nedokáže. Bohužel nelze obecně prohlásit jak velké toto zrychlení bude, protože velmi záleží na pravidlech a hlavně na vstupním řetězci. Dále je zde důležité především to, že k zrychlení dochází pouze při expanzi (použití pravidla), vyjímání ze zásobníku probíhá stejně jako u neparalelní verze. Nejlépe bude si toto zrychlení předvést na příkladu, na kterém si můžeme počet kroků porovnat, proto je použit stejný příklad jako u předvedení hlubokých zásobníkových automatů na straně 9. Tento automat jsme si na straně 18 již také převedli na paralelní.

7.3.1 Příklad

Paralelní hluboký zásobníkový automat ${}_2M_{par}=(\{s,p,q,f\},\{a,b,c\},\{A,S,\#\},R,s,S,\{f\})$, který má v R následující pravidla:

[1] $s(S)\rightarrow q(AA)$,

[2] $q(A)\rightarrow p(aAb)$,

[3] $q(A)\rightarrow f(ab)$,

[4] $f(A)\rightarrow f(c)$,

[5] $q(A,A)\rightarrow q(aAb,Ac)$,

[6] $p(A,A)\rightarrow q(A,Ac)$,

[7] $p(S,A)\rightarrow q(S,Ac)$,

se vstupním řetězcem $aaabbbccc$ provede tyto kroky

$(s,aaabbbccc,S\#)$ ${}_e\Rightarrow(q,aaabbbccc,AA\#)$ [1]

${}_e\Rightarrow(q,aaabbbccc,aAbAc\#)$ [5]

${}_p\Rightarrow(q,aabbbccc,AbAc\#)$

${}_e\Rightarrow(q,aabbbccc,aAbbAcc\#)$ [5]

${}_p\Rightarrow(q,abbbccc,AbbAcc\#)$

${}_e\Rightarrow(f,abbbccc,abbbAcc\#)$ [3]

${}_p\Rightarrow(f,bbbccc,bbbAcc\#)$

${}_p\Rightarrow(f,bbccc,bbAcc\#)$

${}_p\Rightarrow(f,bccc,bAcc\#)$

${}_p\Rightarrow(f,ccc,Acc\#)$

${}_e\Rightarrow(f,ccc,ccc\#)$ [4]

${}_p\Rightarrow(f,cc,cc\#)$

${}_p\Rightarrow(f,c,c\#)$

${}_p\Rightarrow(f,\varepsilon,\#)$

tedy $(s, aabbcc, S\#) \Rightarrow^{*} (f, \varepsilon, \#)$ což znamená že zadaný řetězec byl přijat po 14 krocích (z toho bylo 5 expanzí a 9 vyjmutí). To potvrzuje, že paralelní verze hlubokého zásobníkového automatu opravdu pracuje rychleji. U tohoto řetězce se jedná o úsporu 2 expanzí.

8 Aplikace

Praktickou částí, jak již bylo v úvodu naznačeno, je implementace paralelního hlubokého zásobníkového automatu. Po několika konzultacích s vedoucím práce, prof. Alexandrem Medunou, a s Jiřím Viktorinem, který vypracovává bakalářskou práci na téma „Prediktivní syntaktická analýza s hlubokými zásobníky“ jsme došli k závěru, že by bylo vhodnější vytvořit jednu aplikaci, která by byla schopna pracovat jako paralelní i neparalelní hluboký zásobníkový automat. Tato aplikace by následně mohla sloužit i pro výukové účely.

Pro vytvoření aplikace byl vybrán programovací jazyk C# a program funguje nad rozhraním .NET Framework 2.0. Vstupní konfigurace je uložena v souborech formátu xml. Tyto soubory obsahují všechny potřebné stavy, vstupní i nevstupní zásobníkové symboly, pravidla tvořená podpravidly, jejichž levá strana musí být složena pouze z nevstupních zásobníkových symbolů, a také vstupní řetězec, který musí být tvořen pouze vstupními symboly. Vzhledem k tomu, že se již od prvních návrhů počítalo s implementací obou verzí, nebylo následně prakticky potřeba předělávat větší množství zdrojového kódu. V dalším textu se budeme věnovat především částem, které bylo potřeba upravit výhradně kvůli paralelní verzi a větším částem na kterých jsem pracoval.

Mým prvním úkolem bylo vytvoření struktury xml souboru. Po několika nepoužitelných pokusech jsme se nakonec shodli na následující skladbě. Mezi symboly /* a */ je uveden komentář, tři tečky (…) za ukončující značkou položky znamenají možnost více takových položek.

```
<deeppushdownautomaton>
  <states> /* množina stavů, musí být jeden startovní a alespoň jeden koncový, y=ano n=ne*/
    <state start="y" final="n">p</state> ...
  </states>
  <terminals> /* množina vstupních symbolů */
    <terminal>a</terminal> ...
  </terminals>
  <nonterminals> /* množina nevstupních zásobníkových symbolů.*/
    <nonterminal>A</nonterminal> ...
  </nonterminals>

  <rules version="standard"> /* množina pravidel, pouze pro standardní verzi */
   <rule d="1" s1="p" s2="q"> /* pravidlo d= hloubka, s1 = výchozí a s2 = koncový stav */
     <left>A</left> /* levá strana pravidla, uvnitř jeden neterminální symbol */
     <right> /* pravá strana pravidla, řetězec symbolů, závisí na pořadí! */
       <r>A</r>
       <r>A</r> ...
```

```
      </right>
    </rule> ...
  </rules>

  <rules version="parallel"> /* množina pravidel, pouze pro paralelní verzi */
    <rule s1="p" s2="q"> /* pravidlo  s1 = výchozí a s2 = koncový stav */
      <sub>  /* podpravidlo  pro hloubku 1 */
        <left>A</left>
        <right><r>a</r> ... </right> /* stejné jako u neparalelní verze */
      </sub>
      <sub> /* podpravidlo  pro hloubku 2 */
        <left>A</left>
        <right><r>b</r> ... </right>
      </sub> ...
    </rule> ...
  </rules>

  <inputstring> /* vstupní řetězec, záleží na pořadí symbolů! */
    <i>a</i><i>a</i> ... /* jednotlivé symboly */
  </inputstring>
</deeppushdownautomaton>
```

V jednom okamžiku může být použita pouze množina paralelních nebo množina standardních pravidel. Správně načtena budou jen pravidla s korektními stavy a existujícími symboly. Obdobně to platí i pro vstupní řetězec, kde jsou chybné symboly taktéž přeskakovány. Aby byla konfigurace načtena, musí spolehlivě existovat alespoň jeden prvek z každé této skupiny (stav, terminál, neterminál, pravidlo) a vstupní řetězec nesmí být prázdný. Jak je patrné z této struktury, v některých případech závisí na pořadí jednotlivých značek, hlavně u použití řetězců ze symbolů. Zpracování konfiguračního souboru je zajištěno metodou `XMLinput()` třídy `C_PDAConf`.

Asi nejdůležitějším bodem bylo vytvoření třídy `C_Subrule` obsahující hloubku, symbol na levé straně podpravidla a řetězec pravé strany. Tato třída by pro neparalelní verzi nebyla vůbec potřeba, protože by bylo možné ji implementovat přímo do třídy `C_Rule`. Místo toho je však v `C_Rule` seznam těchto podpravidel. Obecně lze o neparalelním hlubokém zásobníkovém automatu prohlásit, že je zjednodušenou verzí paralelního, která má pouze jedno podpravidlo. Dalším prvkem přidaným do třídy `C_Rule` je logická hodnota, zda je toto pravidlo paralelní (a tedy je nutné projít více podpravidel).

V tomto okamžiku jsme si ukázali všechny potřebné úpravy tříd a zaměříme se na úpravy metod pro aplikaci pravidla.

Výběr pravidel je prováděn především na základě aktuálního stavu a nejprve je vybráno první pravidlo které lze použít. Toto ovšem není pro paralelní automat vhodné, neboť by jsme se museli spolehnout na konfigurační soubor, který by musel mít upřednostněna hlubší pravidla a až po nich mít pravidla v menší hloubce. Takovýto postup by byl implementačně nejjednodušší, ale neměli by jsme kontrolu, zda je zvoleno opravdu to nejvhodnější pravidlo. Také víme, že tento automat pracuje nedeterministicky a tudíž může existovat hned několik způsobů jak příjmout řetězec. Obdobným řešením mohlo být přeuspořádání seznamu pravidel podle hloubky ihned po načtení. Zadá-li uživatel pravidla v nějakém pořadí, pravděpodobně k tomu má nějaký důvod. Proto nebyla ani tato možnost použita a podle hloubky je tříděn až seznam vybraných pravidel. Je to sice mírně pomalejší, avšak z uživatelského hlediska zcela nejpřijatelnější postup. K tomuto třídění dochází v metodě `SortByDeep()` třídy `C_Rules`. Je volána pouze z `UseableRules()` třídy `C_Config` (třída pro konfiguraci automatu, ve smyslu definice uvedené na straně).

Do zmíněné metody `UseableRules()` muselo být přidáno zacyklení, které zajistí kontrolu zda všechny neterminální symboly na levých stranách pravidla odpovídají těm na zásobníku.

Úplně nejdůležitější změna nastala opět ve třídě `C_Config` a to v metodě `UseRule()`. V případě že chceme aplikovat paralelní pravidlo, je nutné postupovat od jeho nejhlubšího podpravidla k podpravidlu hloubky 1. Souvisí to s požadavkem, kdy se číslování neterminálů musí shodovat s jejich umístěním před aplikací pravidla. Při postupu k nejhlubšímu se totiž vystavujeme riziku přepsání jednoho neterminálu několika a jejich přečíslování. Uhlídání této situace by bylo mnohem složitější než použití zmiňovaného postupu, kde k něčemu podobnému nemůže dojít.

Kvůli nedeterminismu ve výběru pravidel je v tomto programu implementován bactracking, který v případě nevhodně zvoleného pravidla vrátí konfiguraci do stavu kdy došlo k tomuto výběru a vybere jiné pravidlo. Jestliže lze řetězec nějakým způsobem příjmout, tak bude tento způsob vždy nalezen. S touto metodou a tříděním pravidel při výběru dochází k poslednímu drobnému problému, který musel být vyřešen. Je jím potřeba si pamatovat která pravidla byla použita. Stejně jako u standardní verze si bactracking pamatuje číslo pravidla, ale rozdíl je v tom, že už po seřazení podel hloubky nejsou jejich čísla vzestupné. Tato poslední změna nastala ve třídě `C_Rules` v implementaci metody `GetRulesBy()`.

Po aplikování zde uvedených změn je program schopen pracovat korektně i s paralelními pravidly. Žádná z uvedených změn neměla zásadní vliv na činnost neparalelní verze.

Aplikaci včetně zdrojových kódů a příkladů na vyzkoušení neleznete na přiloženém CD.

9 Závěr

V této práci bylo prezentováno nové rozšíření hlubokých zásobníkových automatů. Podle očekávání došlo použitím paralelismu k výraznému urychlení jejich činnosti a to hlavně bez vlivu na přijímací síly automatu. Nevýhodou by mohla být potřeba většího počtu pravidel pro vyjádření automatu ekvivalentního k původnímu neparalelnímu. V praktické části této práce byla provedena modifikace hlubokého zásobníkového automatu z bakalářské práce Jiřího Viktorina (Jiří Viktorin: Prediktivní syntaktická analýza s hlubokými zásobníky, bakalářská práce, Brno, FIT VUT v Brně, 2007).

Další vývoj této práce by mohl být zaměřen hlavně na determinismus. Tato práce uvádí obecnou verzi paralelních hlubokých zásobníkových automatů, které pracují nedeterministicky. Zavedení deterministické verze by mohlo mít významný přínos především pro praktické využití. Při takovéto úpravě je ale velice pravděpodobné, že dojde k jejich oslabení podobně jako tomu je u běžných zásobníkových automatů.

Jinou další možnou modifikací neuvažovanou v této práci je zavedení pravidel dovolujících přepis nevstupního zásobníkového symbolu prázdným řetězcem (tzv. ε -pravidlo). Zde ale opět může dojít ke změně síly takovéhoto automatu.

Literatura

[1] Meduna, A. *Automata and Languages*. London, Springer 2000.

[2] Meduna, A. Deep pushdown automata. Acta Informatica, 98, 2006, s. 114-124.

[3] *Chomsky hierarchy* [online]. <http://en.wikipedia.org/wiki/Chomsky_hierarchy> (květen 2007)

Seznam příloh

Příloha 1. CD s programem, zdrojovými kódy a elektronickou verzí tohoto dokumentu