VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA SYSTÉMECH HLUBOKÝCH ZÁSOBNÍKOVÝCH AUTOMATŮ

PARSING BASED ON DEEP PUSHDOWN AUTOMATA SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAKUB ŠOUSTAR**
AUTHOR

VEDOUCÍ PRÁCE **Prof. RNDr. ALEXANDER MEDUNA, CSc.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19167/2016/xsoust02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Šoustar Jakub, Bc.**
Obor: Informační systémy
Téma: **Syntaktická analýza založená na systémech hlubokých zásobníkových automatů**
Parsing Based on Deep Pushdown Automata Systems
Kategorie: Překladače

Pokyny:

1. Seznamte se s vybranými metodami syntaktické analýzy, automatovými systémy a hlubokými zásobníkovými automaty.
2. Zaveďte systémy hlubokých zásobníkových automatů.
3. Navrhněte modifikované metody syntaktické analýzy, které jsou založeny na systémech z bodu 2.
4. Zkoumejte vlastnosti metod z bodu 3. Porovnejte jejich sílu s klasickými metodami syntaktické analýzy.
5. Uvažujte řadu syntaktických struktur, včetně struktur, které nejsou bezkontextové. Popište jejich analýzu prostřednictvím modifikovaných metod z bodu 3.
6. Aplikujte metody z bodu 3 v kompilátorech. Aplikaci zaměřte na analýzu syntaktických struktur, které nejsou bezkontextové.
7. Implementujte aplikaci navrženou v bodě 6 a testujte ji.
8. Zhodnoťte dosažené výsledky. Diskutujte další vývoj projektu.

Literatura:

- Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Volume 1-3, Springer, 1997, ISBN 3-540-60649-1
- Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools (2nd Edition), Pearson Education, 2006, ISBN 0-321-48681-1
- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% 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 nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), 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 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá hlubokými zásobníkovými automaty a zavádí jejich modifikaci nazvanou řízený hluboký zásobníkový automat. Dále jsou v této práci představeny distribuované systémy hlubokých zásobníkových automatů a paralelně komunikující systémy řízených hlubokých zásobníkových automatů. Jsou zkoumány vlastnosti a vyjadřovací síla těchto automatových systémů a je zavedeno několik variant těchto systémů. Pro jednu z variant paralelně komunikujících systémů je dokázáno, že disponuje stejnou vyjadřovací silou, jakou mají Turingovy stroje. Na základě těchto automatových systémů je zavedena metoda syntaktické analýzy.

Abstract

This thesis investigates deep pushdown automata and introduces their modification called controlled deep pushdown automata. Distributed deep pushdown automata systems and parallel communicating deep pushdown automata systems are described. Their accepting power and properties are investigated and several variants are introduced. This thesis proves that the accepting power of one such variant of parallel communicating deep pushdown automata systems is equal to the accepting power of Turing machines. A method for syntactical analysis based on the previously introduced automata systems is described.

Klíčová slova

syntaktická analýza, hluboké zásobníkové automaty, distribuované systémy automatů, paralelně komunikující systémy automatů, determinismus

Keywords

parsing, deep pushdown automata, distributed automata systems, parallel communicating automata systems, determinism

Citace

ŠOUSTAR, Jakub. *Syntaktická analýza založená na systémech hlubokých zásobníkových automatů*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Meduna Alexander.

Syntaktická analýza založená na systémech hlubokých zásobníkových automatů

Prohlášení

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

.....

Jakub Šoustar

21. května 2017

Obsah

Kapitola 1

Úvod

Jazyky jsou základním nástrojem pro vzájemnou komunikaci. Lidmi používané jazyky můžeme jistým způsobem vnímat jako živé entity — neustále se vyvíjejí, dochází ke změnám ve významu slov, mění se způsoby, jakými jsou používány, a v neposlední řadě jazyky také neustále vznikají a zanikají. Jedná se symbiotický vztah, neboť člověk není schopen komunikace bez jazyka a jazyk existuje právě díky lidem, kteří jej používají. Společně se také navzájem ovlivňují — struktura jazyka se přizpůsobuje našim potřebám a ten zase na oplátku formuje naše vnímání okolního světa.

Potíží s přirozenými jazyky je právě jejich nezastavitelný vývoj. Komunikace mezi dvěma stranami je zcela závislá na schopnosti vzájemného porozumění si při použití nestálého média — přirozeného jazyka. Nejenom, že neznalost jakéhokoliv společného jazyka prakticky vylučuje možnost komunikace, ale také odlišnosti v úrovni znalosti, použitém dialektu a dalších aspektech mohou způsobit potíže při domluvě. Například význam některého slova se mohl v některých geografických lokalitách změnit. Nebo některý dialekt obsahuje zcela nové slovo. Tyto a mnohé další problémy mohou způsobit ztrátu nebo špatnou interpretaci informace, kterou se snažíme předat. Navíc se nemusí vždy jednat pouze o nevinná nedorozumění. V mnoha situacích může mít i drobná chyba dalekosáhlé následky.

Právě pro všechny tyto a další nesnáze spojené s přirozenými jazyky zůstává jejich plnohodnotné použití v počítačových systémech stále jen vzdáleným snem. Z tohoto důvodu věnuje teoretická informatika důkladnou pozornost formálním jazykům a prostředkům pro jejich popisování a rozpoznávání. Jako formální označujeme takové jazyky, které popisujeme pomocí formálních prostředků, kterými jim vtiskujeme přesně a formálně definovanou strukturu. Každý formální jazyk je tvořený množinou řetezců, které se skládají ze symbolů. Řetězce a symboly ve formálních jazycích zastávají stejnou roli jako věty a písmena v přirozených jazycích. Podoba těchto řetězců je přesně definována pravidly, která jsou dána konkrétním formálním jazykem. Mohou například určovat, z jakých symbolů se může řetězec skládat, nebo jaké vztahy musejí platit mezi použitými symboly. Takto exaktně popsané jazyky jsou již běžně používány v počítačových systémech. Jako příklad uveďmě programovací jazyky, značkovací jazyky, datové formáty, komunikační protokoly a další.

Jedním z modelů sloužících pro specifikování formálních jazyků jsou gramatiky. Tyto konstrukce využívají konečné sady pravidel k vygenerování řetězce daného formálního jazyka. Toto generování probíhá přepisováním symbolů v řetězci, který je zpočátku často tvořen pouhým jedním symbolem, dokud nedojde k získání řetězce z daného jazyka. Důležitou vlastností gramatik je jejich schopnost popsat pomocí konečné množiny pravidel jazyky tvořené nekonečným počtem možných řetězců. Tyto jazyky označujeme za nekonečné. Gramatiky také seskupujeme do tříd podle tvaru jejich pravidel a případně i dalších

vlastností. Takto získané třídy používáme pro klasifikování formálních jazyků — přiřazujeme je do tříd podle gramatik, kterými je dokážeme popsat. Jazyky v jedné třídě často sdílí mnoho podobných vlastností.

Dalším prostředkem, který používáme při práci s formálními jazyky, jsou automaty. Automaty jsou teoretická zařízení, která rozhodují, zda řetězec symbolů patří do daného jazyka nebo ne. Přirozeně takto tvoří protějšek gramatikám a vzájemně se s nimi doplňují. Automaty opět disponují konečnou množinou pravidel, která řídí jejich činnost. Významnou součástí automatů je paměť, která je používaná v průběhu rozhodování o náležitosti řetězce do jazyka. Podobně jako můžeme gramatiky členit podle tvaru pravidel, tak můžeme rozlišovat i automaty podle typu použité paměti či pamětí. A stejně jako v případě gramatik můžeme pomocí tříd automatů klasifikovat formální jazyky.

Současný vývoj v informatice, zejména zvýšený zájem o paralelizaci a distribuci výpočtů, se projevuje i ve formálních jazycích. Jsou zaváděny modely systémů, které kombinují stávající automaty a gramatiky do celků rozšířených o další řídící prvky. Takové systémy jsou jednou z možností, jak formálně definovat prostředky pro distribuované a paralelní operace s formálními jazyky. Tyto systémy také často svými schopnostmi přesahují schopnosti jejich dílčích částí.

1.1 Logické členění práce

V první části této práce zavedeme základní pojmy a definice, se kterými budeme následně pracovat. Dále se budeme věnovat studiu hlubokých zásobníkových automatů, které vycházejí z tradičních zásobníkových automatů — disponují konečným stavovým řízením a zásobníkovou pamětí. Na rozdíl od těch tradičních jsou ale také schopny pracovat se symboly, které se nacházejí hlouběji v jejich zásobníku. Tato schopnost jim propůjčuje větší sílu, než jakou má jejich původní vzor. Na základě těchto automatů zavedeme jejich modifikací, která umožní řídit jejich činnost pomocí aktuálního vstupního symbolu. Tuto modifikaci budeme nazývat řízeným hlubokým zásobníkovým automatem. Analogicky budeme označovat původní verzi těchto automatů za neřízenou. Také dokážeme, že neřízené hluboké zásobníkové automaty jsou zvláštním případem těch řízených.

Dále na základě řízených hlubokých zásobníkových automatů zavedeme dva druhy systémů, které pracují s těmito automaty. Nejdříve zavedeme distribuované (CD) systémy, ve kterých jednotlivé dílčí automaty pracují sekvenčně tak, že je vždy aktivní právě jeden. U těchto systémů zavedeme několik režimů, které udávají způsob, jak spolu jednotlivé automaty spolupracují. Druhým typem systému, který zavedeme, jsou paralelně komunikující (PC) systémy. Jednotlivé automaty v těchto systémech pracují paralelně a jsou vzájemně synchronizované. Jak již jejich název napovídá, automaty v těchto systémech vzájemně komunikují. U obou typů zavedených typů systémů budeme zkoumat jejich vlastnosti a jejich vyjadřovací sílu. Také zavedeme několik variant těchto systémů. V případě jedné z variant PC systémů také dokážeme, že disponuje stejnou vyjadřovací silou jako Turingovy stroje.

Nakonec navrhneme metodu syntaktické analýzy, která je založena na těchto systémech. Jádrem navrženého analyzátoru bude vždy distribuovaný nebo paralelně komunikující systém řízených hlubokých zásobníkových automatů. Při návrhu syntaktického analyzátoru se především zaměříme na řešení problémů specifických jednotlivým systémům a také na možnost nedeterministické analýzy. Navržený analyzátor bude nakonec implementován demonstrační aplikací.

Kapitola 2

Základní definice a pojmy

Tuto kapitolu věnujeme shrnutí některých důležitých matematických pojmů, které budou v této práci použity. Následující definice i příklady jsou převzaty z [10]. Pro podrobnější informace odkazujeme čtenáře na tuto knihu.

2.1 Množiny a n-tice

Množinou M rozumíme soubor objektů bez struktury, s výjimkou členství. Tyto objekty nazýváme *prvky* množiny. Členství prvku x v množině M značíme $x \in M$. Zápisem $x \notin M$ říkáme, že x není členem množiny M. Množinu, která nemá žádné členy, značíme $\emptyset$ a nazýváme ji *prázdnou* množinou. Pokud má množina M konečný počet prvků, nazýváme ji *konečnou* množinou. V opačném případě ji nazýváme *nekonečnou* množinou. *Kardinalita* množiny M je počet jejích prvků, psáno $|M|$, případně $card(M)$.

Konkrétní množinu je možno definovat mnoha způsoby. Například množinu se třemi prvky x, y a z můžeme definovat jejich výčtem

$$M = \{x, y, z\}$$

V případech, kdy je význam zřejmý, lze použít výpustek. Například $M = \{0, 1, \ldots, 9\}$ je množina celých čísel od nuly do devíti. Je-li potřeba, je možné vyjádřit množinu M charakteristickou vlastností π. Potom M obsahuje všechny prvky, které splňují vlastnost π. Pro tento způsob definice používáme zápis

$$M = \{x \mid \pi(x)\}$$

Základními operacemi, které je možné s množinami provádět, jsou *sjednocení* ($\cup$), *průnik* ($\cap$) a *rozdíl* ($-$). Mějme množiny M_1 a M_2. Potom

$$M_1 \cup M_2 = \{x \mid x \in M_1 \text{ nebo } x \in M_2\}$$
$$M_1 \cap M_2 = \{x \mid x \in M_1 \text{ a } x \in M_2\}$$
$$M_1 - M_2 = \{x \mid x \in M_1 \text{ a } x \notin M_2\}$$

Pokud je každý prvek množiny N současně prvkem množiny M, N je *podmnožinou* M, značeno $N \subseteq M$. Zároveň je M *nadmnožinou* N. Pokud $N \subseteq M$ a současně $M - N \neq \emptyset$, N je *vlastní podmnožinou* M, psáno $N \subset M$. Také platí, že M je *vlastní nadmnožinou* N.

N-tice je uspořádaný seznam prvků, který může na rozdíl od množiny obsahovat stejný prvek vícekrát a ve kterém záleží na pořadí jednotlivých prvků. Konkrétní n-tice označujeme podle počtu jejich prvků — dvojice, trojice a tak dále. Jako příklad uveďme dvojici (a, b).

2.2 Binární relace a funkce

Kartézský součin množin M a N, značený $M \times N$, je množina dvojic definovaná jako

$$M \times N = \{(x, y) \mid x \in M \text{ a } y \in N\}$$

Binární relace R z množiny M do množiny N je libovolná podmnožina kartézského součinu M a N. Tedy

$$R \subseteq M \times N$$

Pokud je R konečná množina, nazýváme R *konečnou relací*. V ostatních případech označujeme R jako *nekonečnou relaci*. Binární relaci $R \subseteq M \times N$ nazýváme *funkcí*, pokud pro všechna $x \in M$ platí, že

$$card(\{y \mid y \in N \text{ a } (x, y) \in R\}) \leq 1$$

2.3 Řetězce a jazyky

Abeceda Σ je konečná neprázdná množina, jejíž prvky nazýváme *symboly*. *Řetězec* nebo také *slovo* nad Σ je konečná posloupnost symbolů ze Σ. *Prázdný řetězec* označovaný jako ε je řetězec, který není tvořen žádnými symboly. Množinu všech řetězců nad Σ včetně ε označujeme jako Σ^*. Množinu všech neprázdných řetězců nad Σ označujeme Σ^+ a platí, že $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Množina řetězců tvoří *jazyk*.

Množinu symbolů vyskytujících se v řetězci x označujeme jako $alph(x)$. Počet výskytů symbolů z množiny Y v řetězci x označujeme $occur(x, Y)$. Je-li $Y = \{a\}$, používáme zkrácený zápis $occur(x, a)$. Řetězec, který získáme obracením pořadí symbolů v řetězci x, značíme $reversal(x)$.

Mějme řetězec x nad abecedou Σ, tedy $x \in \Sigma^*$. Vyjádřeme x jako sekvenci jednotlivých symbolů $a_1 a_2 \ldots a_n$, kde $a_i \in \Sigma$ pro všechna $1 \leq i \leq n$. Je-li $n = 0$, pak $x = \varepsilon$. Poté definujeme *délku řetězce*, značenou $|x|$, jako $|x| = n$. Poznamenejme, že $|\varepsilon| = 0$.

Mějme dva řetězce x a y nad abecedou Σ. *Konkatenací* neboli také *zřetězením* x a y je řetězec xy. Pro libovolný řetězec x platí, že $\varepsilon x = x = x\varepsilon$. Můžeme-li vyjádřit x jako $x = uv$, kde $u, v \in \Sigma^*$, pak u nazýváme *prefixem* x a v nazýváme *sufixem* x. Současně jsou u a v *podřetězci* x.

Nechť x je řetězec nad abecedou Σ a $n \geq 0$ je celé číslo. Poté je *n-tou mocninou řetězce* x, označovanou x^n, řetězec nad Σ, rekurzivně definovaný jako

$$(1)\ x^0 = \varepsilon$$
$$(2)\ x^n = xx^{n-1} \text{ pro } n \geq 1$$

2.4 Gramatiky

Gramatiky jsou speciální struktury sloužící pro generování jazyků. Pomocí konečné množiny pravidel jsou schopny produkovat řetězce určitého jazyka. Jsme tedy pomocí nich schopni popsat i nekonečné jazyky, což je nemožné učinit výčtem všech řetězců takového jazyka. *Gramatika* je čtveřice

$$G = (N, T, P, S)$$

kde:

- N je abeceda *neterminálů*.
- T je abeceda *terminálů*, $N \cap T = \emptyset$.
- P je konečná relace z $(N \cup T)^* N (N \cup T)^*$ do $(N \cup T)^*$.
- $S \in N$ je *počáteční symbol*.

Každý prvek $(u, v) \in P$ nazýváme *pravidlem* a používáme pro něj zápis $u \to v \ \in P$. Mějme gramatiku $G = (N, T, P, S)$. Relace *přímé derivace* v G, označovaná $\Rightarrow_G$, je aplikace pravidla z P a je definovaná jako

$$x_1 u x_2 \Rightarrow_G y_1 v y_2$$

právě pokud $u \to v \in P$ a kde $x_1, x_2, y_1, y_2 \in (N \cup T)^*$. Dále nechť $\Rightarrow_G^*$ značí tranzitivní a reflexivní uzávěr $\Rightarrow_G$ a $\Rightarrow_G^+$ značí tranzitivní uzávěr $\Rightarrow_G$. Nechť $x \in (N \cup T)^*$, poté $S \Rightarrow_G^* x$ je *derivace* x a x nazýváme *větnou formou*. Pokud je $x \in T^*$, pak x nazýváme *větou*.

Jazyk L, který je generovaný gramatikou G, značíme $L(G)$ a definujeme jej jako

$$L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$$

Chomského hierarchie

Noam Chomsky v [1] popsal čtyřstupňovou hierarchii gramatik. Tyto úrovně jsou vytvořeny aplikací omezení na tvar pravidel gramatik v dané úrovni. Chomského hierarchii používáme pro klasifikaci jazyků. Množinu jazyků ve stejné úrovni označujeme jako *rodinu jazyků*. Vztah mezi rodinami jazyků je znázorněn na obrázku 2.1.

- *Neomezené gramatiky* jsou všechny gramatiky, jejichž pravidla odpovídají tvaru uvedenému v definici gramatiky ze sekce 2.4. Tyto gramatiky generují rodinu *rekurizvně spočetných jazyků*, které značíme **RE**. Tento typ gramatik disponuje největší generativní silou — generují největší rodinu jazyků ze všech dále uvedených gramatik.
- *Kontextově senzitivní gramatiky* definují rodinu *kontextově senzitivních jazyků*, označovanou **CS**. Gramatiky tohoto typy mají pravidla ve tvaru

$$x_1 A x_2 \to x_1 y x_2, \text{ kde } A \in N, y \in (N \cup T)^+ \text{ a } x_1, x_2 \in (N \cup T)^*$$

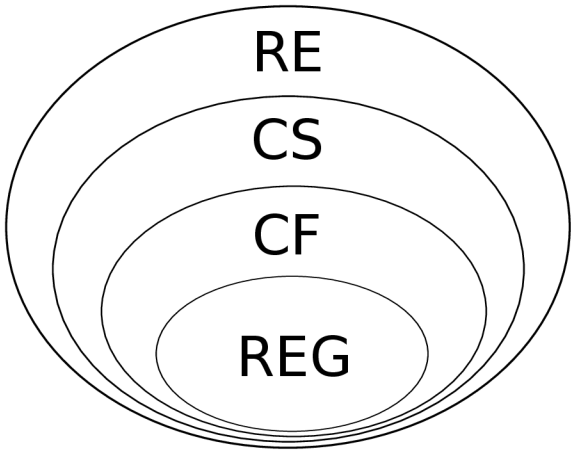
- *Bezkontextové gramatiky* generují rodinu bezkontextových jazyků, kterou označujeme **CF**. Pravidla gramatik tohoto typu jsou omezena na tvar

$$A \to x, \text{ kde } A \in N \text{ a } x \in (N \cup T)^*$$

- *Regulární gramatiky* jsou nejslabším typem gramatik této hierarchie. Rodinu jazyků, které generují, nazýváme *regulárními jazyky* a zkráceně ji označujeme jako **REG**. Regulární gramatiky mají pravidla tvaru

$$A \to aB \text{ nebo } A \to a, \text{ kde } A, B \in N \text{ a } a \in T$$

Věta 2.1. **REG** $\subset$ **CF** $\subset$ **CS** $\subset$ **RE**. Pro více informací o této větě odkazujeme čtenáře na [1].



Obrázek 2.1: Chomského hierarchie [1]

2.5 Automaty

Automaty tvoří přirozený protějšek gramatik a jsou dalším nástrojem, který můžeme použít pro definování jazyků. Na rozdíl od gramatik, které generují řetězce daného jazyka, automat tyto řetězce *přijímá*. Automat přijme řetězec právě tehdy, patří-li do daného jazyka. Všechny ostatní řetězce jsou automatem *odmítnuty*. Množina všech řetězců, které automat přijímá, tvoří *jazyk automatu* a říkáme, že automat *přijímá* tento jazyk. Automaty jsou obzvláště důležité pro syntaktickou analýzu, neboť formálně popisují její výpočetní model.

V této sekci si nejprve představíme dva základní typy automatů — konečný automat a zásobníkový automat. Konečné automaty jsou schopny rozpoznávat všechny řetězce regulárních jazyků (viz [10]). Zásobníkový automat je oproti konečnému automatu navíc rozšířen o další paměť, takzvaný zásobník. Tento typ automatu je schopen rozpoznávat všechny řetězce bezkontextových jazyků (viz [10]). Nakonec představíme ještě variantu zásobníkového automatu, která disponuje dvěma zásobníky. Díky druhému zásobníku u této varianty pozorujeme větší vyjadřovací sílu, než jak je tomu v případě verze s jedním zásobníkem. Konkrétně tento automat rozpoznává rekurzivně vyčíslitelné jazyky (viz [6]).

Konečný automat

Konečný automat je pětice

$$A = (Q, \Sigma, \delta, q_0, F)$$

kde:

- Q je konečná množina *stavů*.
- Σ je *vstupní abeceda*.
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ je konečná relace.
- $q_0 \in Q$ je *počáteční stav*.
- $F \subseteq Q$ je množina *koncových stavů*.

Relaci δ označujeme jako množinu *přechodových pravidel* a její prvky nazýváme *přechodovými pravidly*. Namísto $(q, a, p) \in \delta$ píšeme $qa \to p \in \delta$.

Konfigurace konečného automatu A je dvojice $(q, x) \in Q \times \Sigma^*$. Relaci *přechodu*, kterou někdy také nazýváme *krokem*, A značíme $\vdash_A$ a definujeme ji jako

$$(q, ax) \vdash_A (p, x)$$

právě pokud $qa \to p \in \delta$ a zároveň (q, ax) a (p, x) jsou konfigurace A. Dále nechť $\vdash_A^*$ značí tranzitivní a reflexivní uzávěr $\vdash_A$ a $\vdash_A^+$ značí tranzitivní uzávěr $\vdash_A$. Jazyk přijímaný A značíme $L(A)$ a definujeme jej jako

$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash_A^* (f, \varepsilon), f \in F\}$$

Zásobníkový automat

Zásobníkový automat je sedmice

$$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

kde:

- Q je konečná množina *stavů*.
- Σ je *vstupní abeceda*.
- Γ je *zásobníková abeceda*.
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$ je konečná relace.
- $q_0 \in Q$ je *počáteční stav*.
- $Z_0 \in \Gamma$ je *počáteční zásobníkový symbol*.
- $F \subseteq Q$ je množina *koncových stavů*.

Stejně jako u konečných automatů nazýváme relaci δ množinou *přechodových pravidel* a pro její prvky používáme označení *přechodová pravidla*. Podobně také namísto $(q, a, Z, p, \alpha) \in \delta$ píšeme $qaZ \to p\alpha \in \delta$. Nadále budeme při práci s řetězci symbolů představujících zásobník předpokládat, že nejlevější symbol se nachází na vrcholu zásobníku. Analogicky bude nejpravější symbol řetězce umístěn na dně zásobníku.

Konfigurace zásobníkového automatu A je trojice $(q, x, \alpha) \in Q \times \Sigma^* \times \Gamma^*$. Relaci *přechodu*, respektive *kroku*, A značíme $\vdash_A$ a definujeme ji jako

$$(q, ax, Z\alpha) \vdash_A (p, x, \gamma\alpha)$$

právě pokud $qaZ \to p\gamma \in \delta$ a zároveň $(q, ax, Z\alpha)$ a $(p, x, \gamma\alpha)$ jsou konfigurace A. Dále nechť $\vdash_A^*$ značí tranzitivní a reflexivní uzávěr $\vdash_A$ a $\vdash_A^+$ značí tranzitivní uzávěr $\vdash_A$. Jazyk přijímaný A dosažením koncového stavu značíme $L_f(A)$ a definujeme jej jako

$$L_f(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_A^* (f, \varepsilon, \alpha), f \in F, \alpha \in \Gamma^*\}$$

Jazyk přijímaný A vyprázdněním zásobníku značíme $L_e(A)$ a je definován jako

$$L_e(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_A^* (q, \varepsilon, \varepsilon), q \in Q\}$$

Jazyk přijímaný A dosažením koncového stavu a vyprázdněním zásobníku značíme $L_{ef}(A)$ a definujeme jej jako

$$L_{ef}(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_A^* (f, \varepsilon, \varepsilon), f \in F\}$$

Dvouzásobníkový automat

Dvouzásobníkový automat je osmice

$$A = (Q, \Sigma, \Gamma, \delta, q^0, Z_1^0, Z_2^0, F)$$

kde:

- Q je konečná množina *stavů*.
- Σ je *vstupní abeceda*.
- Γ je *zásobníková abeceda*.
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma \times Q \times \Gamma^* \times \Gamma^*$ je konečná relace.
- $q^0 \in Q$ je *počáteční stav*.
- $Z_1^0 \in \Gamma$ je *počáteční zásobníkový symbol* prvního zásobníku.
- $Z_2^0 \in \Gamma$ je *počáteční zásobníkový symbol* druhého zásobníku.
- $F \subseteq Q$ je množina *koncových stavů*.

U tohoto automatu budeme používat, neuvedeme-li jinak, stejné názvosloví, jaké jsme zavedli pro zásobníkový automat. Dále budeme namísto $(q, a, Z_1, Z_2, p, \alpha_1, \alpha_2) \in \delta$ psát $qaZ_1Z_2 \to p\alpha_1\alpha_2 \in \delta$.

Konfigurace dvouzásobníkového automatu A je čtveřice $(q, x, \alpha_1, \alpha_2) \in Q \times \Sigma^* \times \Gamma^* \times \Gamma^*$. Relaci *přechodu* neboli *kroku* A značíme $\vdash_A$ a definujeme ji jako

$$(q, ax, Z_1\alpha_1, Z_2\alpha_2) \vdash_A (p, x, \gamma_1\alpha_1, \gamma_2\alpha_2)$$

právě pokud $qaZ_1Z_2 \to p\gamma_1\gamma_2 \in \delta$ a zároveň $(q, ax, Z_1\alpha_1, Z_2\alpha_2)$ a $(p, x, \gamma_1\alpha_1, \gamma_2\alpha_2)$ jsou konfigurace A. Dále nechť $\vdash_A^*$ značí tranzitivní a reflexivní uzávěr $\vdash_A$ a $\vdash_A^+$ značí tranzitivní uzávěr $\vdash_A$. Jazyk přijímaný A dosažením koncového stavu značíme $L_f(A)$ a definujeme jej jako

$$L_f(A) = \{w \in \Sigma^* \mid (q^0, w, Z_1^0, Z_2^0) \vdash_A^* (f, \varepsilon, \alpha_1, \alpha_2), f \in F, \alpha_1, \alpha_2 \in \Gamma^*\}$$

Jazyk přijímaný A vyprázdněním obou zásobníků značíme $L_e(A)$ a je definován jako

$$L_e(A) = \{w \in \Sigma^* \mid (q^0, w, Z_1^0, Z_2^0) \vdash_A^* (q, \varepsilon, \varepsilon, \varepsilon), q \in Q\}$$

Jazyk přijímaný A dosažením koncového stavu a vyprázdněním obou zásobníků značíme $L_{ef}(A)$ a definujeme jej jako

$$L_{ef}(A) = \{w \in \Sigma^* \mid (q^0, w, Z_1^0, Z_2^0) \vdash_A^* (f, \varepsilon, \varepsilon, \varepsilon), f \in F\}$$

Kapitola 3

Hluboké zásobníkové automaty

Běžný zásobníkový automat během provádění svých přechodů pracuje pouze s jediným symbolem na vrcholu svého zásobníku. Při každém kroku je tento symbol automatem přečten a následně nahrazen řetězcem zásobníkových symbolů libovolné délky. *Hluboký zásobníkový automat* představuje modifikaci zásobníkového automatu, která je schopna ze zásobníku přečíst m-tý, $m \geq 1$, zásobníkový symbol od vrcholu a nahradit jej neprázdným řetězcem zásobníkových symbolů. Takovou operaci nazýváme *expanzí* symbolu v *hloubce* m. Tento typ automatu byl poprvé představen v [9].

Hluboký zásobníkový automat, který si v této kapitole postupně představíme, je značně inspirován použitím zásobníkového automatu při vytváření syntaktického analyzátoru pracujícího shora dolů z bezkontextové gramatiky. Zásobníkový automat, který je součástí takového analyzátoru, provede v každém svém kroku jednu ze dvou možných operací v závislosti na typu symbolu, který se nachází na vrcholu jeho zásobníku. Jedná-li se o neterminální symbol, pak provede jeho expanzi, čímž simuluje provedení odpovídající derivace v gramatice. Pokud je symbol na vrcholu zásobníku terminální, automat jej porovná s aktuálním symbolem na svém vstupu. Při shodě je takový symbol odstraněn ze zásobníku a dojde k přesunu čtecí hlavy automatu na následující vstupní symbol. V opačném případě dochází k zastavení automatu, protože není schopen provést žádný další krok. Vstupní řetězec je přijat, pokud byl celý přečten a zároveň je zásobník automatu prázdný.

3.1 Definice a příklady

Nejprve definujeme hluboký zásobníkový automat a poté se podrobněji zaměříme na některé jeho vlastnosti. V následujících částech této kapitoly se poté budeme zabývat vyjadřovací silou a také determinismem v tomto typu automatu.

Definice 3.1. *Hluboký zásobníkový automat* je sedmice

$$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

kde:

- Q je konečná množina *stavů*.
- $\Sigma \subseteq \Gamma$ je *vstupní abeceda*.
- Γ je *zásobníková abeceda*, $\mathbb{N}$, Q a Γ jsou po dvou disjunktní, $\Gamma - \Sigma$ obsahuje symbol *dna zásobníku* značený $\#$.

- $\delta \quad \subseteq \quad (\mathbb{N} \times Q \times [\Gamma - (\Sigma \cup \{\#\})] \times Q \times [\Gamma - \{\#\}]^+)$
  $\cup\, (\mathbb{N} \times Q \times \{\#\} \times Q \times [(\Gamma - \{\#\})^*\{\#\}])$ je konečná relace.
- $q_0 \in Q$ je *počáteční stav*.
- $Z_0 \in \Gamma$ je *počáteční zásobníkový symbol*.
- $F \subseteq Q$ je množina *koncových stavů*.

Opět budeme relaci δ označovat jako množinu *přechodových pravidel* a její prvky budeme nazývat *přechodovými pravidly*. Namísto zápisu $(m, q, Z, p, \alpha) \in \delta$ budeme psát $mqZ \to p\alpha \in \delta$. Dále předpokládáme, neuvedeme-li jinak, že zásobník A je vždy inicializován řetězcem $Z_0\#$ pokud $Z_0 \neq \#$. Symbol # budeme používat pro označení dna zásobníku i ve zbývajících částech této práce.

O $mqZ \to p\alpha$ říkáme, že jde o pravidlo *hloubky* m. Pokud je $n \in \mathbb{N}$ nejmenší celé číslo takové, že všechna pravidla A jsou nanejvýš hloubky n, říkáme, že A je *hloubky* n a píšeme ${}_nA$. Hluboké zásobníkové automaty, které provádí expanze nanejvýš v hloubce n, kde $n \geq 1$, jsou ekvivalentní n-limitovaným stavovým gramatikám (viz [9, 10]).

Definice 3.2. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. *Konfigurace* A je trojice $(q, x, \alpha) \in Q \times \Sigma^* \times [(\Gamma - \{\#\})^*\{\#\}]$, kde:

- q je aktuální stav.
- x je doposud nepřečtená část vstupního řetězce.
- α je obsah zásobníku zakončený symbolem dna zásobníku.

U hlubokých zásobníkových automatů rozlišujeme dva typy přechodů, které vycházejí z operací syntaktického analyzátoru popsaného v úvodu této kapitoly. Prvním je přechod expanzí, při kterém dochází k nahrazení zásobníkového symbolu v hloubce m neprázdným řetězcem symbolů, pro nějaké $m \geq 1$. Druhým typem je přechod přečtením symbolu, při kterém hluboký zásobníkový automat porovnává symbol na vrcholu svého zásobníku s aktuálním vstupním symbolem.

Definice 3.3. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. A provede *přechod*, nebo také *krok*, mezi konfiguracemi u a v *přečtením* symbolu a, značeno

$$u \vdash_p v$$

právě pokud $u = (q, ax, a\alpha)$ a $v = (q, x, \alpha)$, kde $q \in Q$, $a \in \Sigma$, $x \in \Sigma^*$ a $\alpha \in \Gamma^*$. Dále nechť $\vdash_p^*$ značí tranzitivní a reflexivní uzávěr $\vdash_p$ a $\vdash_p^+$ značí tranzitivní uzávěr $\vdash_p$.

Definice 3.4. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. A provede *přechod* neboli také *krok* mezi konfiguracemi u a v *expanzí* zásobníkového symbolu Z v *hloubce* m, značeno

$$u \vdash_e v$$

právě pokud $u = (q, x, \alpha Z\gamma)$, $v = (p, x, \alpha\beta\gamma)$ a $mqZ \to p\beta \in \delta$, kde $q, p \in Q$, $x \in \Sigma^*$, $Z \in \Gamma$, $\alpha, \beta, \gamma \in \Gamma^*$ a $occur(\alpha, \Gamma - \Sigma) = m - 1$. Dále nechť $\vdash_e^*$ značí tranzitivní a reflexivní uzávěr $\vdash_e$ a $\vdash_e^+$ značí tranzitivní uzávěr $\vdash_e$.

Chceme-li zdůraznit, že A provedl krok $u \vdash_e v$ podle přechodového pravidla $mqZ \to p\beta$, píšeme

$$u \vdash_e v\ [mqZ \to p\beta]$$

Obecně budeme za přechod hlubokého zásobníkového automatu považovat provedení kroku přečtením nebo expanzí symbolu. Nadále budeme předpokládat, že pokud může hluboký zásobníkový automat ze své aktuální konfigurace provést více než jeden druh přechodu, vždy nejdříve vykoná přechod přečtením symbolu.

Definice 3.5. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. A provede *přechod* neboli také *krok* mezi konfiguracemi u a v, značeno

$$u \vdash v$$

právě pokud A provede $u \vdash_p v$ nebo $u \vdash_e v$. Dále nechť $\vdash^*$ značí tranzitivní a reflexivní uzávěr $\vdash$ a $\vdash^+$ značí tranzitivní uzávěr $\vdash$.

Povšimněme si, že automat během přechodu přečtením symbolu provádí pouze porovnání symbolu na vrcholu zásobníku s aktuálním vstupním symbolem. Nedochází ke změně stavu automatu a s výjimkou odstranění úspěšně porovnaného symbolu nedochází k žádným změnám obsahu zásobníku. Při přechodu expanzí symbolu může automat nejenom změnit svůj stav, ale také samozřejmě dochází ke změně obsahu jeho zásobníku. Během provádění tohoto přechodu ale hluboký zásobníkový automat nijak nepracuje se vstupním řetězcem. Toto přesné vymezení rolí jednotlivých typů přechodů a jejich možných dopadů na konfiguraci automatu odlišuje hluboké zásobníkové automaty od obyčejných zásobníkových automatů. Běžný zásobníkový automat definuje pouze jeden typ přechodu, který v sobě kombinuje aspekty obou variant přechodů hlubokých zásobníkových automatů.

Z těchto omezení a také z toho, že hluboký zásobníkový automat při přechodu expandováním symbolu může nahradit expandovaný symbol pouze neprázdným řetězcem symbolů, vyplývá další vlastnost týkající se práce se zásobníkem. Jediným prostředkem hlubokého zásobníkového automatu pro redukci obsahu svého zásobníku je odebírání symbolu na jeho vrholu pomocí přechodu přečtením symbolu.

Definice 3.6. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. Jazyk přijímaný A dosažením koncového stavu, značený $L_f(A)$, je definován jako

$$L_f(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0\#) \vdash^* (f, \varepsilon, \alpha\#), f \in F, \alpha \in (\Gamma^* - \{\#\})\}$$

Jazyk přijímaný A vyprázdněním zásobníku, značený $L_e(A)$, je definován jako

$$L_e(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0\#) \vdash^* (q, \varepsilon, \#), q \in Q\}$$

Tuto sekci zakončíme příkladem hlubokého zásobníkového automatu, na kterém demonstrujeme jeho činnost při přijímání daného vstupního řetězce. Zároveň na tomto příkladu ukážeme, že hluboké zásobníkové automaty jsou silnější v porovnání s těmi bežnými.

Příklad 3.1. Mějme hluboký zásobníkový automat

$${}_2A = (\{s, f, q_1, q_2, q_f, p_1, p_2, p_f\}, \{a, b\}, \{S, X, \#\}, \delta, s, S, \{f\})$$

jehož množina přechodových pravidel δ obsahuje těchto následujících devět pravidel:

$$\begin{array}{lll} 1sS \to q_1XX & 1q_1X \to q_2aX & 1p_1X \to p_2bX \\ & 1q_1X \to q_faX & 1p_1X \to p_fb \\ & 2q_2X \to q_1aX & 2p_2X \to p_1bX \\ & 2q_fX \to p_1aX & 1p_fX \to fb \end{array}$$

Pro vstupní řetězec *abbabb* provede A následující přechody:

$$\begin{aligned}
(s, abbabb, S\#) &\vdash_e (q_1, abbabb, XX\#) && [1sS \to q_1XX] \\
&\vdash_e (q_f, abbabb, aXX\#) && [1q_1X \to q_faX] \\
&\vdash_p (q_f, bbabb, XX\#) \\
&\vdash_e (p_1, bbabb, XaX\#) && [2q_fX \to p_1aX] \\
&\vdash_e (p_2, bbabb, bXaX\#) && [1p_1X \to p_2bX] \\
&\vdash_p (p_2, babb, XaX\#) \\
&\vdash_e (p_1, babb, XabX\#) && [2p_2X \to p_1bX] \\
&\vdash_e (p_f, babb, babX\#) && [1p_1X \to p_fb] \\
&\vdash_p (p_f, abb, abX\#) \\
&\vdash_p (p_f, bb, bX\#) \\
&\vdash_p (p_f, b, X\#) \\
&\vdash_e (f, b, b\#) && [1p_fX \to fb] \\
&\vdash_p (f, \varepsilon, \#)
\end{aligned}$$

Jazyky přijímané tímto automatem dosažením koncového stavu a vyprázdněním zásobníku jsou stejné, platí tedy, že $L_f({}_2A) = L_e({}_2A)$. Konkrétně tento automat přijímá jazyk $\{a^nb^ma^nb^m \mid n, m \geq 1\}$, který není bezkontextový — patří do množiny jazyků ležících v $\mathbf{CS} - \mathbf{CF}$. Pro takovýto jazyk není možné sestrojit zásobníkový automat, který by jej přijímal.

3.2 Vyjadřovací síla

V příkladě 3.1 jsme si představili hluboký zásobníkový automat ${}_2A$, který přijímá jazyk $\{a^nb^ma^nb^m \mid n, m \geq 1\}$, který není bezkontextový. Je tedy patrné, že tyto automaty jsou silnější než běžné zásobníkové automaty. Vyjadřovací síla hlubokých zásobníkových automatů byla zkoumána v [9] a také v [10]. Výsledky těchto zkoumání stručně prezentujeme v podobě několika vět. Důkazy těchto vět, které zde neuvádíme, je možné nalézt ve výše uvedené literatuře.

Definice 3.7. Pro každé celé číslo $k \geq 1$, $\mathbf{DPDA}_k^f$ značí rodinu jazyků přijímaných hlubokými zásobníkovými automaty hloubky i dosažením koncového stavu, kde $1 \leq i \leq k$. Podobně $\mathbf{DPDA}_k^e$ značí rodinu jazyků přijímaných hlubokými zásobníkovými automaty hloubky i vyprázdněním zásobníku, kde $1 \leq i \leq k$.

Věta 3.1. Pro každé celé číslo $k \geq 1$ platí, že

$$\mathbf{DPDA}_k^f = \mathbf{DPDA}_k^e \subset \mathbf{DPDA}_{k+1}^f = \mathbf{DPDA}_{k+1}^e$$

Věta 3.2. $\mathbf{DPDA}_1^f = \mathbf{DPDA}_1^e = \mathbf{CF}$.

Věta 3.3. Pro každé celé číslo $k \geq 1$ platí, že $\mathbf{DPDA}_k^f = \mathbf{DPDA}_k^e \subset \mathbf{CS}$.

Důsledkem těchto vět je, že hluboké zásobníkové automaty definují nekonečnou hierarchii jazyků, které se nacházejí mezi bezkontextovými jazyky a kontextově senzitivními jazyky (viz [9, 10]). Vždy ovšem existuje takový kontextově senzitivní jazyk, který nemůže být přijat žádným hlubokým zásobníkovým automatem hloubky k, pro všechna celá čísla $k \geq 1$.

3.3 Determinismus

V této sekci se zaměříme na determinismus v hlubokých zásobníkových automatech, který je obzvláště zajímavý z pohledu jejich praktického nasazení v syntaktické analýze. Schopnost těchto automatů provádět expanze v různých hloubkách jejich zásobníku také umožňuje rozlišit několik variant jejich determinismu.

Determinismus v hlubokých zásobníkových automatech zkoumáme ve spojitosti s přechody prováděnými expandovámím symbolu. Právě u těchto přechodů může docházet k situacím, kdy automat může z jedné konfigurace provést přechod expanzí podle více různých přechodových pravidel. Kroky automatu prováděné přečtením symbolu jsou ze své definice deterministické. Tento krok je hlubokým zásobníkovým automatem proveden kdykoliv jej může provést. Podmínka proveditelnosti tohoto kroku je daná shodou aktuálního symbolu na vstupu a symbolu na vrcholu zásobníku. Tato podmínka může v libovolné konfiguraci automatu platit nanejvýše pro jeden symbol.

Zaměříme se na dva konkrétní typy determinismu popsané v [9]. Jako první definujeme striktní formu determinismu podobnou té, kterou známe z tradičních zásobníkových automatů. Striktně deterministický hluboký zásobníkový automat může ze své libovolné konfigurace provést nanejvýš jeden přechod. Druhý typ, který si představíme, je determinismus s ohledem na hloubku prováděných expanzí. Tato varianta je slabší formou striktního determinismu, neboť pouze diktuje, že všechny přechody expanzí, které může automat provést z libovolné konfigurace, musejí být stejné hloubky.

Definice 3.8. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. Říkáme, že A je *striktně deterministický* pokud pro všechna jeho přechodová pravidla $mqZ \to p\alpha \in \delta$ platí, že

$$card(\{nqZ \to o\beta \mid nqZ \to o\beta \in \delta, o \in Q, \beta \in \Gamma^+\}) \leq 1$$

Označíme-li A jako *deterministický* bez bližšího upřesnění o jaký typ determinismu se jedná, předpokládáme, že se jedná o striktně deterministický automat.

Definice 3.9. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je hluboký zásobníkový automat. Říkáme, že A je *deterministický s ohledem na hloubku svých expanzí* pokud pro všechny jeho stavy $q \in Q$ platí, že

$$card(\{m \mid mqZ \to p\alpha \in \delta, p \in Q, Z \in \Gamma, \alpha \in \Gamma^+\}) \leq 1$$

Povšimněme si, že automat z příkladu 3.1 je deterministický s ohledem na hloubku svých expanzí, ale již nesplňuje podmínku striktního determinismu.

Pro praktické nasazení hlubokých zásobníkových automatů je nejvíce zajímavý striktní determinismus. Syntaktický analyzátor založený na automatu, který není striktně deterministický, musí být rozšířen o dodatečnou rozhodovací logiku, která vybírá z množiny použitelných pravidel to, podle kterého bude provedena expanze v daném kroku. Dalším množným přístupem je vytvořit analyzátor, který je schopen zkoumat všechny proveditelné expanze. Tyto operace ovšem zvyšují cenu prováděné analýzy.

Hluboký zásobníkový automat při expandování symbolu nepracuje se svým vstupem. Proto je v jeho libovolné konfiguraci množina pravidel, podle kterých může provést přechod expanzí, omezena pouze jeho stavem a obsahem zásobníku. Obecně se tato pravidla mezi sebou mohou odlišovat pouze svojí hloubkou. U těchto automatů se proto můžeme, v porovnání s běžnými zásobníkovými automaty, setkat s větší mírou nedeterminismu, protože tyto automaty disponují méně prostředky pro jeho redukci a naopak zavádějí jeho další možný zdroj v podobě hloubky přechodových pravidel.

Kapitola 4

Řízené hluboké zásobníkové automaty

Hluboké zásobníkové automaty, které byly postupně představeny v předchozí kapitole, kopírují model syntaktického analyzátoru pracujícího shora dolů založeného na zásobníkových automatech. Tento analyzátor, stejně jako hluboký zásobníkový automat, postupně generuje na svém zásobníku řetězec daného jazyka a porovnává jej se vstupním řetězcem. Zásadní odlišností mezi hlubokými a bežnými zásobníkovými automaty je způsob, jakým nakládají se vstupním řetězcem.

Zatímco hluboký zásobníkový automat řetězec symbolů na svém vstupu pouze postupně porovnává s obsahem svého zásobníku, běžný zásobníkový automat je navíc schopen volit pravidlo, které použije pro provedení následujícího přechodu, na základě aktuálního vstupního symbolu. Tuto schopnost je možno využít při návrhu syntaktického analyzátoru tak, aby byl deterministický — to je důležité zejména v praxi.

V této kapitole zavedeme právě takovou modifikaci hlubokého zásobníkového automatu, která umožní *řídit* výběr přechodového pravidla také pomocí aktuálního vstupního symbolu. Tuto variantu budeme nazývat *řízený hluboký zásobníkový automat*. Původní typ hlubokého zásobníkového automatu, který jsme zavedli v kapitole 3, budeme dále označovat jako *neřízený*. Tento nový typ automatu definujeme tak, abychom mohli libovolný neřízený hluboký zásobníkový automat považovat za speciální instanci řízené varianty.

4.1 Definice a příklady

Nejdříve zavedeme řízený hluboký zásobníkový automat a poté definujeme odlišnosti v jeho chování a vlastnostech vůči neřízené variantě. Závěr této sekce věnujeme demonstraci řízeného hlubokého zásobníkového automatu pomocí příkladu, na kterém také znázorníme přínos tohoto nového typu.

Definice 4.1. *Řízený hluboký zásobníkový automat* je sedmice

$$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

kde:

- Q je konečná množina *stavů*.
- $\Sigma \subseteq \Gamma$ je *vstupní abeceda*.

- Γ je *zásobníková abeceda*, $\mathbb{N}$, Q a Γ jsou po dvou disjunktní, $\Gamma - \Sigma$ obsahuje symbol *dna zásobníku* značený $\#$.
- $\delta \subseteq (\mathbb{N} \times Q \times [\Sigma \cup \{\varepsilon\}] \times [\Gamma - (\Sigma \cup \{\#\})] \times Q \times [\Gamma - \{\#\}]^+)$
  $\cup (\mathbb{N} \times Q \times [\Sigma \cup \{\varepsilon\}] \times \{\#\} \times Q \times [(\Gamma - \{\#\})^*\{\#\}])$ je konečná relace.
- $q_0 \in Q$ je *počáteční stav*.
- $Z_0 \in \Gamma$ je *počáteční zásobníkový symbol*.
- $F \subseteq Q$ je množina *koncových stavů*.

Z této definice je zřejmé, že řízený hluboký zásobníkový automat se bude od neřízeného odlišovat pouze tvarem svých přechodových pravidel. Proto budeme nadále předpokládat, neuvedeme-li jinak, že pro řízené hluboké zásobníkové automaty analogicky platí stejné definice, které jsme uvedli v kapitole 3. Obdobně budeme používat již zavedené názvosloví i pro tuto modifikaci. V případě přechodových pravidel budeme nadále místo zápisu $(m, q, a, Z, p, \alpha) \in \delta$ používat tvaru $mqaZ \to p\alpha \in \delta$.

Před bližším zkoumáním této varianty je třeba upravit některé definice zavedené v kapitole 3 tak, aby zohlednily změnu tvaru přechodových pravidel. Nejpodstatnější je následující úprava definice přechodu expanzí symbolu.

Definice 4.2. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je řízený hluboký zásobníkový automat. A provede *přechod* neboli také *krok* mezi konfiguracemi u a v *expanzí* symbolu Z v *hloubce* m, značeno

$$u \vdash_e v$$

právě pokud $u = (q, ax, \alpha Z\gamma)$, $v = (p, ax, \alpha\beta\gamma)$ a $mqaZ \to p\beta \in \delta$, kde $q, p \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$, $x \in \Sigma^*$, $Z \in \Gamma$, $\alpha, \beta, \gamma \in \Gamma^*$ a $occur(\alpha, \Gamma - \Sigma) = m - 1$. Dále nechť $\vdash_e^*$ značí tranzitivní a reflexivní uzávěr $\vdash_e$ a $\vdash_e^+$ značí tranzitivní uzávěr $\vdash_e$.

Chceme-li zdůraznit, že A provedl krok $u \vdash_e v$ podle přechodového pravidla $mqaZ \to p\beta$, píšeme

$$u \vdash_e v \ [mqaZ \to p\beta]$$

Povšimněme si, že řízený hluboký zásobníkový automat při přechodu expanzí symbolu neupravuje vstupní řetězec — pouze nahlédne na jeho první symbol. Nadále tedy platí, že vstupní řetězec je možné číst pouze pomocí přechodů přečtením symbolu. V následující části této kapitoly ukážeme, že zachování této vlastnosti nám umožní nahlížet na neřízené hluboké zásobníkové automaty jako na speciální případ těch řízených.

Dále budeme opět rozlišovat dva typy determinismu — striktní formu a také tu, která se týká hloubky prováděných expanzí.

Definice 4.3. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je řízený hluboký zásobníkový automat. Říkáme, že A je *striktně deterministický*, pokud pro všechna jeho přechodová pravidla $mqaZ \to p\alpha \in \delta$ platí, že

$$card(\{nqaZ \to o\beta \mid nqaZ \to o\beta \in \delta, a \in (\Sigma \cup \{\varepsilon\}), o \in Q, \beta \in \Gamma^+\}) \leq 1$$

Definice 4.4. Nechť $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je řízený hluboký zásobníkový automat. Říkáme, že A je *deterministický s ohledem na hloubku svých expanzí*, pokud pro všechny jeho stavy $q \in Q$ platí, že

$$card(\{m \mid mqaZ \to p\alpha \in \delta, p \in Q, Z \in \Gamma, \alpha \in \Gamma^+\}) \leq 1$$

Tuto sekci uzavřeme demonstrací přínosu řízené varianty hlubokých zásobníkových automatů. Pro tyto účely využijeme jako předlohu automat uvedný v příkladě 3.1. Připomeňme, že tento automat přijímal jazyk $\{a^n b^m a^n b^m \mid n, m \geq 1\}$. V případě tohoto jazyka je nemožné sestrojit takový neřízený hluboký zásobníkový automat, který by jej přijímal a zároveň byl striktně deterministický. Důvodem je, že neřízené automaty postrádají prostředky pro určení hranice mezi symboly a a b ve vstupním řetězci. Zkonstruovat řízený hluboký zásobníkový automat přijímající stejný jazyk, který jsou současně striktně deterministický, je ovšem triviální.

Příklad 4.1. Na základě automatu definovaném v příkladě 3.1 sestrojíme řízený hluboký zásobníkový automat

$$_2A = (\{s, f, q_1, q_2, p_1, p_2\}, \{a, b, \$\}, \{S, X, \#\}, \delta, s, S, \{f\})$$

Pro potřeby tohoto příkladu rozšíříme vstupní abecedu Σ o pomocný symbol \$, který bude označovat konec vstupního řetězce. Množina přechodových pravidel δ obsahuje následující pravidla:

$$\begin{array}{llll} 1saS \to q_1aXX & 1q_1aX \to q_2aX & 1p_1bX \to p_2bX & 1f\$X \to f\$ \\ & 2q_2\varepsilon X \to q_1aX & 2p_2\varepsilon X \to p_1bX & \\ & 1q_1bX \to p_2bX & 1p_1aX \to fa & \end{array}$$

Pro vstupní řetězec $abbabb\$$ provede A následující přechody:

$$\begin{array}{rll} (s, abbabb\$, S\#) & \vdash_e (q_1, abbabb\$, aXX\#) & [1saS \to q_1aXX] \\ & \vdash_p (q_1, bbabb\$, XX\#) & \\ & \vdash_e (p_2, bbabb\$, bXX\#) & [1q_1bX \to p_2bX] \\ & \vdash_p (p_2, babb\$, XX\#) & \\ & \vdash_e (p_1, babb\$, XbX\#) & [2p_2\varepsilon X \to p_1bX] \\ & \vdash_e (p_2, babb\$, bXbX\#) & [1p_1bX \to p_2bX] \\ & \vdash_p (p_2, abb\$, XbX\#) & \\ & \vdash_e (p_1, abb\$, XbbX\#) & [2p_2\varepsilon X \to p_1bX] \\ & \vdash_e (f, abb\$, abbX\#) & [1p_1aX \to fa] \\ & \vdash_p (f, bb\$, bbX\#) & \\ & \vdash_p (f, b\$, bX\#) & \\ & \vdash_p (f, \$, X\#) & \\ & \vdash_e (f, \$, \$\#) & [1f\$X \to f\$] \\ & \vdash_p (f, \varepsilon, \#) & \end{array}$$

Tento automat přijímá jazyk $\{a^n b^m a^n b^m\$ \mid n, m \geq 1\}$ a to jak dosažením koncového stavu, tak i vyprázdněním zásobníku. Zanedbáme-li pomocný symbol \$, jedná se o stejný jazyk, který je přijímaný automatem z příkladu 3.1 — symbol \$ do něj lze zavést triviální úpravou. Stejně jako v případě jeho předlohy, je i tento automat deterministický s ohledem na hloubku prováděných expanzí. Navíc ovšem splňuje i parametry striktního determinismu, což nebylo u neřízeného automatu možné.

4.2 Vztah mezi řízenými a neřízenými variantami

Závěr této kapitoly věnujeme bližšímu zkoumání vztahu mezi oběma typy hlubokých zásobníkových automatů, kterými jsme se doposud zabývali. Konkrétně dokážeme, že libovolný neřízený hluboký zásobníkový automat je možné převést na řízený tak, aby nově sestrojený automat přijímal stejný jazyk jako ten původní.

Tento důkaz je založen na několika vlastnostech modifikace, kterou jsme zavedli v této kapitole. Řízený hluboký zásobníkový automat může obsahovat taková přechodová pravidla, která porovnávají aktuální vstupní symbol s prázdným řetězcem. Jelikož takové porovnání bude vždy platit, jsou tato pravidla funkčně ekvivalentní pravidlům neřízeného hlubokého zásobníkového automatu. V obou případech totiž budou automaty provádět přechody expanzemi symbolů bez ohledu na své vstupní řetězce. Dále využijeme skutečnosti, že se přechody přečtením a expanzí symbolu u obou variant již dále neodlišují. Společně nám tyto vlastnosti umožní simulovat neřízené hluboké zásobníkové automaty těmi řízenými, což shrneme následující větou a jejím důkazem.

Věta 4.1. Pro každé $n \geq 1$ a každý neřízený hluboký zásobníkový automat ${}_nA$ existuje řízený hluboký zásobníkový automat ${}_nA'$ pro který platí, že $L_f({}_nA) = L_f({}_nA')$ a $L_e({}_nA) = L_e({}_nA')$.

Důkaz. Mějme neřízený hluboký zásobníkový automat ${}_nA = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Sestrojíme řízený hluboký zásobníkový automat

$$ {}_nA' = (Q, \Sigma, \Gamma, \delta', q_0, Z_0, F) $$

kde δ' vytvoříme tak, že pro každé přechodové pravidlo $mqZ \to p\alpha \in \delta$ přidáme $mq\varepsilon Z \to p\alpha$ do δ'. Potom pro libovolné dvě konfigurace u a v je automat ${}_nA'$ schopen provést přechod $u \vdash v$ právě tehdy, je-li krok $u \vdash v$ uskutečnitelný automatem ${}_nA$. □

Důležitým důsledkem věty 4.1 je také skutečnost, že řízené hluboké zásobníkové automaty jsou po stránce jejich vyjadřovací síly přinejmenším stejně silné jako ty neřízené. Proto pokud budeme ve zbytku této práce hovořit o hlubokých zásobníkových automatech bez uvedení jejich konkrétní varianty, budeme předpokládat, neuvedeme-li jinak, že se jedná o řízený typ.

Na konec této kapitoly uveďme jeden otevřený problém týkající se řízených hlubokých zásobníkových automatů. Tím je případná možnost, či nemožnost, jejich převodu na neřízený hluboký zásobníkový automat přijímající stejný jazyk. Pokud by byl takový převod možný, plynula by z něj, ve spojení s větou 4.1, ekvivalence obou variant hlubokých zásobníkových automatů.

Kapitola 5

CD systémy hlubokých zásobníkových automatů

V této kapitole zavedeme *distribuované systémy*, neboli také *CD systémy*, hlubokých zásobníkových automatů. Označení *CD systémy* pochází z anglického *cooperating distributed systems* a nadále budeme všechny tyto termíny považovat za vzájemně zaměnitelné. Tyto systémy založíme na podobných systémech skládajících se ze zásobníkových automatů představených v [3]. Distribuované systémy byly nejprve zkoumány v oblasti gramatik, viz například [2]. Stejně jako v případě gramatických systémů bylo zjištěno, že síla automatových systémů často dalece přesahuje sílu jejich jednotlivých komponent. Například CD systémy zásobníkových automatů zkoumané v [7] jsou stejně silné jako Turingovy stroje. Protože hluboké zásobníkové automaty již disponují značnou vyjadřovací silou, budeme se jejich distribuovanými systémy zabývat především jako prostředkem pro dekompozici složitějších problémů na několik menších podproblémů.

Distribuovaný systém hlubokých zásobníkových automatů se skládá z množiny dílčích automatů tohoto typu, které nazýváme *komponentami* daného systému. Všechny komponenty sdílejí jedinou čtecí hlavu a pracují se společným vstupním řetězcem. Z tohoto důvodu může být v libovolném okamžiku aktivní vždy právě jedna komponenta, která má jako jediná přístup ke čtecí hlavě a vstupnímu řetězci. Dále jsou již všechny komponenty nezávislé na ostatních — každá má vlastní množinu stavů, zásobník a přechodová pravidla.

Jednotlivé komponenty distribuovaného systému hlubokých zásobníkových automatů společně pracují podle daného *řídícího protokolu* na přijmutí, či odmítnutí vstupního řetězce. Ten je systémem přijat právě tehdy, je-li přijat všemi komponentami. Řídící protokol obecně klade omezení na počet přechodů, které právě aktivní komponenta může, nebo naopak musí, vykonat před tím, než bude možné aktivovat další komponentu. Konkrétní forma těchto restrikcí poté závisí na zvoleném protokolu.

Chování těchto systémů si můžeme představit na příkladu z reálného života. Skupina žáků u tabule společně řeší zadaný úkol tak, že vždy právě jeden z nich má křídu a píše na tabuli. Ostatní zatím vyčkávají na pokyn učitele, aby se s pracujícím žákem vystřídali.

5.1 Definice a příklady

V následujícím textu postupně definujeme distribuované systémy hlubokých zásobníkových automatů a také režimy, podle kterých pracují jejich komponenty. Tuto sekci poté uzavřeme příkladem takovéhoto systému.

Definice 5.1. *Distribuovaný systém hlubokých zásobníkových automatů* stupně n, zkráceně $CDDPAS(n)$, pro nějaké $n \geq 1$, je $(n+2)$-tice

$$\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$$

kde:

- Σ je *vstupní abeceda.*
- $A_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i^0, Z_i^0, F_i)$, $1 \leq i \leq n$, je hluboký zásobníkový automat.
- λ_0, $1 \leq \lambda_0 \leq n$, určuje počáteční aktivní komponentu A_{λ_0}.

Nadále budeme automat A_i, $1 \leq i \leq n$, nazývat i-tou *komponentou* systému $\mathcal{A}$. Pokud je $k \in \mathbb{N}$ nejmenší celé číslo takové, že všechny komponenty $\mathcal{A}$ jsou nanejvýš hloubky k, říkáme, že $\mathcal{A}$ je *hloubky* k a tuto skutečnost můžeme zdůraznit zápisem ${}_k\mathcal{A}$.

Definice 5.2. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$. *Konfigurací* $\mathcal{A}$ je $(2n+2)$-tice

$$(x, q_1, \alpha_1, q_2, \alpha_2, \ldots, q_n, \alpha_n, \lambda)$$

kde:

- $x \in \Sigma^*$ je doposud nepřečtená část vstupního řetězce.
- Pro všechna i, $1 \leq i \leq n$, $A_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i^0, Z_i^0, F_i)$, je $q_i \in Q_i$ aktuální stav komponenty A_i a $\alpha_i \in \Gamma_i^*$ je obsah jejího zásobníku.
- λ, $1 \leq \lambda \leq n$, identifikuje aktivní komponentu A_λ.

Přechod distribuovaného systému hlubokých zásobníkových automatů definujeme jako provedení kroku jeho aktivní komponentou. Ostatní komponenty setrvávájí při jeho realizaci ve stejném stavu a také nedochází k žádným změnám na jejich zásobníku. Nemůže-li aktivní komponenta provést přechod, může dojít, v závislosti na konkrétním systému, k jeho úplnému zastavení, anebo pouze aktivaci další komponenty.

Definice 5.3. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$. *Přechod* neboli také *krok* mezi dvěma konfiguracemi $\mathcal{A}$ provedený komponentou A_i, $1 \leq i \leq n$, značíme $\vdash_i$ a definujeme jej jako

$$(x, q_1, \alpha_1, \ldots, q_i, \alpha_i, \ldots, q_n, \alpha_n, i) \vdash_i (y, q_1, \alpha_1, \ldots, p, \beta, \ldots, q_n, \alpha_n, i)$$

právě pokud hluboký zásobníkový automat tvořící komponentu A_i provede přechod

$$(x, q_i, \alpha_i) \vdash (y, p, \beta)$$

kde $x, y \in \Sigma^*$ a pro všechna j, $1 \leq j \leq n$, $A_j = (Q_j, \Sigma, \Gamma_j, \delta_j, q_j^0, Z_j^0, F_j)$, platí, že $q_j \in Q_j$, $\alpha_j \in \Gamma_j^*$, a současně $p \in Q_i$, $\beta \in \Gamma_i^*$. Dále nechť $\vdash_i^k$ značí k-tou mocninu $\vdash_i$, $\vdash_i^*$ značí tranzitivní a reflexivní uzávěr $\vdash_i$ a $\vdash_i^+$ značí tranzitivní uzávěr $\vdash_i$.

Zásadní součástí libovolného CD systému hlubokých zásobníkových automatů je použitý řídící protokol, který jsme zmínili v úvodu této kapitoly. Stejně jako v [3] budeme rozlišovat celkem čtyři takovéto protokoly, které budeme nadále označovat za *derivační módy.* V kontextu distribuovaných systémů hlubokých zásobníkových automatů budeme považovat *derivaci* za sekvenci přechodů provedených určitou komponentou.

Definice 5.4. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$ a C_1, C_2 jsou dvě konfigurace $\mathcal{A}$. Rozlišujeme čtyři *derivační módy*, podle kterých může C_1 *derivovat* C_2.

- *Derivací tvořenou nepokračovatelnou sekvencí kroků*, značeno $C_1 \vdash_{\mathcal{A}}^{t} C_2$, právě pokud $C_1 \vdash_i^* C_2$ pro nějaké $1 \leq i \leq n$, a současně neexistuje žádná konfigurace C', pro kterou by platilo $C_2 \vdash_i C'$.
- *Derivací s právě k kroky*, značeno $C_1 \vdash_{\mathcal{A}}^{=k} C_2$, právě pokud $C_1 \vdash_i^k C_2$ pro nějaké $1 \leq i \leq n$.
- *Derivací s alespoň k kroky*, značeno $C_1 \vdash_{\mathcal{A}}^{\geq k} C_2$, právě pokud $C_1 \vdash_i^l C_2$ pro nějaké $l \geq k$ a $1 \leq i \leq n$.
- *Derivací s nejvýše k kroky*, značeno $C_1 \vdash_{\mathcal{A}}^{\leq k} C_2$, právě pokud $C_1 \vdash_i^l C_2$ pro nějaké $l \leq k$ a $1 \leq i \leq n$.

Definice 5.5. Množinu *derivačních módů* distribuovaných systémů hlubokých zásobníkových automatů značíme M a definujeme ji jako

$$M = \{t\} \cup \{= k, \geq k, \leq k \mid k \geq 1\}$$

Pomocí derivací můžeme lépe popsat činnost CD systémů hlubokých zásobníkových automatů. Aktivní komponenta systému se vždy pokusí provést derivaci, během níž může systém uskutečnit několik přechodů komponentou — jejich počet závisí na derivačním módu systému. Nemůže-li komponenta dokončit derivaci, dochází k zastavení systému. Po úspěšném provedení derivace dochází k výběru a aktivaci další komponenty. Pokud je systém tvořen více než jednou komponentou, je tento výběr obecně nedeterministický. Komponenta pro aktivaci se vždy vybírá z množiny komponent, pomocí kterých může systém přejít z aktuální konfigurace do libovolné jiné. Neexistuje-li taková komponenta, je systém zastaven.

Definice 5.6. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$, kde pro všechna i, $1 \leq i \leq n$ je $A_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i^0, Z_i^0, F_i)$. Jazyk přijímaný $\mathcal{A}$ v derivačním módu $Y \in M$ dosažením koncového stavu všemi komponentami značíme $L_f(\mathcal{A}, Y)$ a definujeme jej jako

$$\begin{aligned} L_f(\mathcal{A}, Y) = \{w \mid (w, q_1^0, Z_1^0\#, \ldots, q_n^0, Z_n^0\#, \lambda_0) \ (\vdash_{\mathcal{A}}^{Y})^* \ (\varepsilon, f_1, \alpha_1, \ldots, f_n, \alpha_n, \lambda), \\ w \in \Sigma^*, f_i \in F_i, \alpha_i \in \Gamma^*, 1 \leq i, \lambda \leq n\} \end{aligned}$$

Jazyk přijímaný $\mathcal{A}$ v derivačním módu $Y \in M$ vyprázdněním zásobníků všech jeho komponent značíme $L_e(\mathcal{A}, Y)$ a definujeme jej jako

$$\begin{aligned} L_e(\mathcal{A}, Y) = \{w \mid (w, q_1^0, Z_1^0\#, \ldots, q_n^0, Z_n^0\#, \lambda_0) \ (\vdash_{\mathcal{A}}^{Y})^* \ (\varepsilon, q_1, \varepsilon, \ldots, q_n, \varepsilon, \lambda), \\ w \in \Sigma^*, q_i \in Q_i, 1 \leq i, \lambda \leq n\} \end{aligned}$$

Stávající část této kapitoly zakončíme příkladem CD systému hlubokých zásobníkových automatů. Tento systém bude pracovat v t-módu — jeho komponenty budou po aktivování pracovat tak dlouho, jak jen to bude možné.

Příklad 5.1. Mějme CD systém hlubokých zásobníkových automatů $\mathcal{A} = (\Sigma, A_1, A_2, 1)$, $\Sigma = \{a, b, c, d\}$, který je tvořen následujícími komponentami:

$$A_1 = (\{s_1, f_1, o_1, p_1, q_1, r_1\}, \Sigma, \{S_1, X_1, \#\}, \delta_1, s_1, S_1, \{f_1\})$$

$$A_2 = (\{s_2, f_2, o_2, p_2, q_2, r_2\}, \Sigma, \{S_2, X_2, \#\}, \delta_2, s_2, S_2, \{f_2\})$$

Množina přechodových pravidel první komponenty A_1 je tvořena těmito pravidly:

$$\begin{array}{lll} 1s_1aS_1 \to o_1aX_1X_1 & 1o_1aX_1 \to p_1aX_1 & 1q_1bX_1 \to r_1bX_1 \\ & 2p_1\varepsilon X_1 \to o_1aX_1 & 2r_1\varepsilon X_1 \to q_1bX_1 \\ & 1o_1bX_1 \to r_1bX_1 & 1q_1cX_1 \to f_1a \end{array}$$

Komponenta A_2 tohoto systému pracuje podle následujících přechodových pravidel:

$$\begin{array}{lll} 1s_2cS_2 \to o_2cX_2X_2 & 1o_2cX_2 \to p_2cX_2 & 1q_2dX_2 \to r_2dX_2 \\ & 2p_2\varepsilon X_2 \to o_2cX_2 & 2r_2\varepsilon X_2 \to q_2dX_2 \\ & 1o_2dX_2 \to r_2dX_2 & 1q_2aX_2 \to f_2c \end{array}$$

První aktivovanou komponentou systému je A_1. Potom tento distribuovaný systém hlubokých zásobníkových automatů pracující v t-módu přijímá dosažením koncového stavu ve všech svých komponentách následující jazyk, který není bezkontextový:

$$L_f(\mathcal{A}, t) = \{a^n b^m c^l d^k a^n b^m c^l d^k \mid n, m, l, k \geq 1\}$$

Komponenta A_1 na svém zásobníku generuje řetězec $a^n b^m a^n b^m$, $n, m \geq 1$, a průběžně jej porovnává se vstupním řetězcem. Když na vstupu detekuje symbol c, přejde A_1 do koncového stavu. V tomto okamžiku je na zásobníku A_1 řetězec $a^n b^m$, $n, m \geq 1$, a tato komponenta již nemůže provést žádný další přechod. Proto dojde k aktivaci komponenty A_2, která analogickým způsobem zpracuje část vstupu tvořenou symboly c a d. Po zpracování této části dojde k opětovné aktivaci automatu A_1, který již pouze porovná obsah svého zásobníku se vstupním řetězcem. Poté je naposledy aktivována komponenta A_2, která porovná zbývající část vstupního řetězce s obsahem svého zásobníku.

5.2 Determinismus

U distribuovaných systémů hlubokých zásobníkových automatů budeme determinismus zkoumat ve dvou formách. Jednou z nich je pochopitelně determinismus na úrovni jednotlivých komponent. Derivační mód, podle kterého každý CD systém hlubokých zásobníkových automatů pracuje, pak dává vzniknout druhému typu determinismu, který zasahuje celý systém jako celek. Samotný mechanismus, podle kterého dochází k aktivaci a deaktivaci jednotlivých komponent, může být totiž nedeterministický.

Nejdříve se budeme zabývat první variantou determinismu CD systémů hlubokých zásobníkových automatů, která přirozeně vyplývá z determinismu jejich komponent. V systému tvořeném pouze striktně deterministickými komponentami je přechod mezi libovolnými dvěma konfiguracemi tohoto systému vždy deterministický. Totéž platí i pro všechny derivace, protože každá se skládá právě ze sekvence kroků provedených stejnou komponentou.

Definice 5.7. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$. Říkáme, že $\mathcal{A}$ je *deterministický s ohledem na své komponenty* právě tehdy, pokud jsou všechny jeho komponenty A_i, $1 \leq i \leq n$, striktně deterministické (viz 4.3).

Ovšem i systém, který je deterministický s ohledem na své komponenty, může provádět nedeterministická rozhodnutí při výběru komponenty pro aktivaci. Připomeňme, že k tomuto výběru dochází mezi provedením dvou derivací. Proto budeme v těchto systémech

rozlišovat i druhou, striktnější, formu determinismu, která navíc zohlední proces výběru komponenty k aktivování. Pro její definici ale nejdříve musíme zavést pomocný pojem takzvané *Y-dosažitelné* (viz [5]) konfigurace, pro libovolné $Y \in M$.

Definice 5.8. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$ pracující v derivačním módu $Y \in M$, kde pro všechna i, $1 \leq i \leq n$, je $A_i = (Q_i, \Sigma, \Gamma_i, \delta_i, q_i^0, Z_i^0, F_i)$. O libovolné konfiguraci u tohoto systému říkáme, že je *Y-dosažitelná* pokud existuje takové $w \in \Sigma^*$, pro které

$$(w, q_1^0, Z_1^0\#, \ldots, q_n^0, Z_n^0\#, \lambda_0) \ (\vdash_{\mathcal{A}}^{Y})^* \ u$$

Jinými slovy, konfigurace nějakého systému $\mathcal{A}$ je Y-dosažitelná, pokud se $\mathcal{A}$ pracující v Y-módu, $Y \in M$, při zpracování nějakého vstupního řetězce může v této konfiguraci vyskytnout. S využitím determinismu s ohledem na komponenty systému a také pojmu Y-dosažitelnosti zavedeme takzvaný *Y-determinismus* (viz [5]), pro $Y \in M$.

Definice 5.9. Nechť $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$ je $CDDPAS(n)$ pracující v derivačním módu $Y \in M$. Říkáme, že $\mathcal{A}$ je *Y-deterministický* právě tehdy, je-li deterministický s ohledem na své komponenty a současně pro všechny jeho Y-dosažitelné konfigurace u a v platí, že

$$card(\{i \mid 1 \leq i \leq n, u \vdash_i v\}) \leq 1$$

Pro systém z příkladu 5.1 platí, že je t-deterministický. Lze triviálně ověřit, že obě jeho komponenty jsou striktně deterministické. Komponenty tohoto systému jsou navíc pomocí řízených pravidel navrženy tak, že každá zpracovává odlišné části vstupního řetězce. Proto tento systém může ze své libovolné konfigurace přejít pouze pomocí jedné komponenty v závislosti na právě zpracovávaném vstupním symbolu.

Na závěr poznamenejme, že u $\leq k$-deterministických a $\geq k$-deterministických je rozhodnutí o přesném počtu přechodů, provedených v rámci jedné derivace, nadále nedeterministické.

5.3 Vyjadřovací síla

Kapitolu o distribuovaných systémech hlubokých zásobníkových automatů uzavřeme krátkou poznámkou týkající se jejich vyjadřovací síly. Ta prozatím zůstává otevřeným problémem. Předpokládáme, že podobně jako v případě podobných modelů distribuovaných systémů zkoumaných v [3, 7] bude vyjadřovací schopnost těchto systému větší v porovnání se samostatnými hlubokými zásobníkovými automaty.

Zejména model zavedený [7] dosahoval shodné vyjadřovací síly, jakou mají Turingovy stroje, a to již u systémů tvořených dvěma zásobníkovými automaty. V tomto modelu mají komponenty krom sdíleného vstupního řetězce také společný stav a každá komponenta může přejít do stavu, který způsobí předání řízení jiné komponentě. Taková interakce těmto systémům umožňuje simulovat zásobníkový automat se dvěma zásobníky, který je dále schopen simulovat Turingův stroj (viz [6]). Stejnému postupu v případě hlubokých zásobníkových automatů brání několik jejich základních vlastností, zejména těch, které omezují možnosti manipulace se zásobníkem při vkládání a odstraňování symbolů.

Kapitola 6

PC systémy hlubokých zásobníkových automatů

Následující kapitolu věnujeme zavedení *PC systémů* (z anglického *parallel communicating systems*) neboli *paralelně komunikujících systémů* hlubokých zásobníkových automatů. Všechna tato označení budeme nadále považovat za vzájemně zaměnitelná. Těmito systémy navážeme na PC systémy konečných automatů představené v [8] a zejména také na PC systémy zásobníkových automatů zavedené v [4], které použijeme jako vzor. Paralelně komunikující systémy byly také zkoumány v případě gramatik — PC gramatické systémy byly poprvé definovány v [11].

Stejně jako tomu bylo u distribuovaných systémů, jsou i paralelně komunikující systémy tvořeny několika dílčími hlubokými zásobníkovými automaty, které nazýváme *komponentami* systému. V rámci systému spolu komponenty sdílejí jak vstupní, tak i zásobníkovou abecedu. Každá z komponent má vlastní zásobník a množinu přechodových pravidel. Dále také pracuje s vlastním vstupním řetězcem, který je v počáteční konfiguraci systému shodný pro všechny komponenty. Všechny dílčí automaty systému pracují paralelně. Činnosti jednotlivých komponent mohou být vzájemně *synchronizované* — hovoříme o *synchronizovaných PC systémech*. V takovýchto systémech je vždy současně všemi komponentami proveden právě jeden přechod. Toto omezení neplatí pro *nesynchronizované PC systémy*, kterým se v této práci již nebudeme nadále věnovat.

Jak již napovídá jejich samotný název, komponenty paralelně komunikujících automatů si mohou vzájemně předávat informace. Komunikace komponent je realizována takzvaným *komunikačním krokem* systému. Předmětem komunikace jsou obsahy jednotlivých komponentních zásobníků — z tohoto důvodu sdílí celý systém zásobníkovou abecedu. Komunikační krok je vyvolán kdykoliv je na vrcholu zásobníku alespoň jedné komponenty nalezen některý ze speciálních *dotazovacích symbolů*. Pro každou komponentu systému existuje právě jeden takovýto symbol, který se na ni jednoznačně odkazuje. Během komunikačního kroku dochází k nahrazování dotazovacích symbolů obsahy zásobníků příslušných komponent. Výskyt dotazovacího symbolu na vrcholu zásobníku některé komponenty tedy můžeme vnímat jako vznesení žádosti na získání aktuálního obsahu zásobníku komponenty, na kterou se odkazuje dotazovací symbol.

6.1 Definice a příklady

Jako první definujeme samotné paralelně komunikující systémy hlubokých zásobníkových automatů. U těchto systémů můžeme rozlišovat dvě jejich varianty podle toho, které komponenty mohou v daném systému vyvolat komunikační krok. Další varianty rozlišíme v závislosti na chování komponent po ukončení komunikačního kroku. Na základě těchto variant poté rozlišíme celkem čtyři hlavní typy PC systémů hlubokých zásobníkových automatů. V samotném závěru této části uvedeme příklad jednoho takového systému.

Definice 6.1. *Paralelně komunikující systém hlubokých zásobníkových automatů* stupně n, zkráceně $PCDPAS(n)$, pro nějaké $n \geq 1$, je $(n+3)$-tice

$$\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$$

kde:

- Σ je *vstupní abeceda.*
- $K \subseteq \{K_1, K_2, \ldots, K_n\} \subseteq \Gamma$ je množina *dotazovacích symbolů.*
- Γ je *zásobníková abeceda.*
- $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$, $1 \leq i \leq n$, je hluboký zásobníkový automat.

Nadále budeme automat A_i, $1 \leq i \leq n$, nazývat i-tou *komponentou* systému $\mathcal{A}$. Pokud je $k \in \mathbb{N}$ nejmenší celé číslo takové, že všechny komponenty $\mathcal{A}$ jsou nanejvýš hloubky k, říkáme, že $\mathcal{A}$ je *hloubky* k a tuto skutečnost můžeme zdůraznit zápisem ${}_k\mathcal{A}$. O dotazovacím symbolu $K_i \in K$ říkáme, že se *odkazuje* na komponentu A_i, pro nějaké $1 \leq i \leq n$. Beze ztráty na obecnosti budeme komponentu A_1 libovolného $PCDPAS(n)$ považovat za *hlavní* nebo také *řídící* komponentu daného systému.

Definice 6.2. Nechť $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$ je $PCDPAS(n)$, kde pro všechna i, $1 \leq i \leq n$, je $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$. *Konfigurací* $\mathcal{A}$ je $3n$-tice

$$(q_1, x_1, \alpha_1, q_2, x_2, \alpha_2, \ldots, q_n, x_n, \alpha_n)$$

kde:

- $q_i \in Q_i$ je aktuální stav komponenty A_i.
- $x_i \in \Sigma^*$ je doposud nepřečtená část vstupního řetězce komponenty A_i.
- $\alpha_i \in \Gamma^*$ je obsah zásobníku komponenty A_i.

Nyní zavedeme *centralizovanou* variantu PC systémů hlubokých zásobníkových automatů. Za centralizovaný označujeme každý takový systém, ve kterém může být komunikační krok vyvolán pouze jeho řídící komponentou.

Definice 6.3. Nechť $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$ je $PCDPAS(n)$, kde pro všechna i, $1 \leq i \leq n$ je $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$. Říkáme, že $\mathcal{A}$ je *centralizovaný* PC systém hlubokých zásobníkových automatů právě tehdy, platí-li pro všechna j, $2 \leq j \leq n$, že

$$card(\{mqaZ \to p\alpha \mid mqaZ \to p\alpha \in \delta_j, occur(\alpha, K) \geq 1\}) = 0$$

Dále budeme také rozlišovat takzvané *navracející* se systémy. U tohoto typu PC systémů hlubokých zásobníkových automatů je každý komunikační krok následován další operací. Během ní jsou všechny zásobníky, jejichž obsah byl komunikován, *navráceny* do počátečního stavu. Poznamenejme, že se jedná o operaci na úrovni PC systému, která je vždy provedena bez ohledu na případná omezení týkající se práce se zásobníky, která mohou klást jednotlivé komponenty.

Definice 6.4. Nechť $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$ je $PCDPAS(n)$, kde pro všechna i, $1 \leq i \leq n$, je $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$. Říkáme, že $\mathcal{A}$ je *navracející se* PC systém hlubokých zásobníkových automatů, pokud po ukončení komunikačního kroku, ve kterém si komponenta A_j vyžádala obsah zásobníku komponenty A_i, pro nějaká $1 \leq i, j \leq n$, dojde k nahrazení obsahu zásobníku komponenty A_i řetězcem $Z_i^0\#$.

Na základě definic 6.3 a 6.4 poté můžeme rozlišovat celkem čtyři hlavní typy PC systémů hlubokých zásobníkových automatů. Všechny tyto varianty včetně označení, která pro ně budeme dále používat, uvádíme v následujícím přehledu:

- *RC-PCDPAS*(n) značí navracející se centralizovaný systém hlubokých zásobníkových automatů stupně n.
- *R-PCDPAS*(n) představuje navracející se systém hlubokých zásobníkových automatů stupně n.
- *C-PCDPAS*(n) označuje centralizovaný systém hlubokých zásobníkových automatů stupně n.
- *PCDPAS*(n) značí systém hlubokých zásobníkových automatů stupně n.

Bez ohledu na konkrétní variantu rozlišujeme u PC systémů hlubokých zásobníkových automatů dva způsoby, jakými může systém provést přechod mezi svými konfiguracemi. Prvním je samozřejmě přechod tvořený krokem v komponentách systému. Každá z komponent tento krok vykonává zcela individuálně. Je ovšem nezbytné, aby byl proveden *všemi* komponentami — pokud některá komponenta není schopna přechodu, dochází k zastavení celého systému. Druhým způsobením je provedení komunikačního kroku, který je vyvolán přítomností dotazovacího symbolu na vrcholu zásobníku libovolné komponenty. Během tohoto kroku zůstavají všechny komponenty neaktivní a systém manipuluje s jejich zásobníky. Komunikačnímu kroku se budeme blíže věnovat po tom, co si jej definujeme.

Definice 6.5. Nechť $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$ je $PCDPAS(n)$, kde pro všechna i, $1 \leq i \leq n$, je $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$. *Přechod* neboli také *krok* mezi dvěma konfiguracemi $\mathcal{A}$ značíme $\vdash$ a definujeme jej jako

$$(q_1, x_1, Z_1\alpha_1, \ldots, q_n, x_n, Z_n\alpha_n) \vdash (p_1, y_1, \beta_1, \ldots, p_n, y_n, \beta_n)$$

kde $q_j, p_j \in Q_j$, $x_j, y_j \in \Sigma^*$, $Z_j \in \Gamma$, $\alpha_j, \beta_j \in \Gamma^*$, $1 \leq j \leq n$, právě pokud platí právě jedna z následujících podmínek:

1. Pro všechna i, $1 \leq i \leq n$, platí, že $Z_i \notin K$ a komponenta A_i provede přechod $(q_i, x_i, Z_i\alpha_i) \vdash (p_i, y_i, \beta_i)$.
2. Pro všechna taková i, $1 \leq i \leq n$, kde $Z_i = K_j$ a $Z_j \notin K$, $\beta_i = Z_j\alpha_j\alpha_i$, $K_j \in K$, $1 \leq j \leq n$. Pokud je $\mathcal{A}$ navracející se systém, tak po ukončení komunikačního kroku platí, že $\beta_j = Z_j^0\#$. Pro všechna ostatní r, $1 \leq r \leq n$, je $\beta_r = Z_r\alpha_r$ a pro všechna t, $1 \leq t \leq n$, je $y_t = x_t$ a $p_t = q_t$. Symbol $\#$ značící dno zásobníku je z komunikace vždy vyjmut.

Dále nechť $\vdash^*$ značí tranzitivní a reflexivní uzávěr $\vdash$ a $\vdash^+$ značí tranzitivní uzávěr $\vdash$.

Upřesněme nyní některé aspekty komunikačního kroku. Vytvoří-li komunikační symboly nacházející se na vrcholech komponentních zásobníků kruhovou závislost, dochází k zastavení systému. Považujme za výjimku případ, kdy se dotazovací symbol odkazuje na tutéž komponentu, na jejímž zásobníku se nachází. V této situaci dojde k nahrazení tohoto symbolu zbývající částí daného zásobníku namísto zastavení systému. PC systém může provést více po sobě jdoucích komunikačních kroků v případě, že nelze všechny dotazy kvůli jejich vzájemným vlastnostem obsloužit v průběhu jednoho komunikačního kroku. Nakonec zdůrazněme, že v případě navracejících se systémů musí být obsah libovolného zásobníku v rámci jednoho komunikačního kroku odeslán *všem* komponentám, které si jej vyžádaly, a to před tím, než dojde k jeho navrácení do počátečního stavu.

Definice 6.6. Nechť $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$ je $PCDPAS(n)$, kde pro všechna i, $1 \leq i \leq n$, je $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$. Jazyk přijímaný $\mathcal{A}$ dosažením koncového stavu všemi komponentami značíme $L_f(\mathcal{A})$ a definujeme jej jako

$$L_f(\mathcal{A}) = \{w \mid (q_1^0, w, Z_1^0\#, \ldots, q_n^0, w, Z_n^0\#) \vdash^* (f_1, \varepsilon, \alpha_1, \ldots, f_n, \varepsilon, \alpha_n), \\ w \in \Sigma^*, f_i \in F_i, \alpha_i \in \Gamma^*, 1 \leq i \leq n\}$$

Jazyk přijímaný $\mathcal{A}$ vyprázdněním zásobníků všech jeho komponent značíme $L_e(\mathcal{A})$ a definujeme jej jako

$$L_e(\mathcal{A}) = \{w \mid (q_1^0, w, Z_1^0\#, \ldots, q_n^0, w, Z_n^0\#) \vdash^* (q_1, \varepsilon, \varepsilon, \ldots, q_n, \varepsilon, \varepsilon), \\ w \in \Sigma^*, q_i \in Q_i, 1 \leq i \leq n\}$$

Na závěr této sekce uvedeme příklad jednoho paralelně komunikujícího systému hlubokých zásobníkových automatů.

Příklad 6.1. Nechť $\mathcal{A} = (\Sigma, \{K_1, K_2\}, \Gamma, A_1, A_2)$ je $PCDPAS(2)$, kde $\Sigma = \{a, b, c\}$ a $\Gamma = \{X, \#\}$. Tento systém je tvořen následujícími komponentami:

$$A_1 = (\{s_1, p_1, f_1\}, \Sigma, \Gamma, \delta_1, s_1, X, \{f_1\})$$
$$A_2 = (\{s_2, p_2, f_2\}, \Sigma, \Gamma, \delta_2, s_2, X, \{f_2\})$$

Množiny přechodových pravidel jednotlivých komponent jsou dány následovně:

$$\delta_1 = \{1s_1aX \to p_1aXK_2, 1p_1aX \to p_1aXb, 1p_1bX \to f_1b\}$$
$$\delta_2 = \{1s_2aX \to p_2aX, 1p_2aX \to p_2aXc, 1p_2bX \to f_2K_1c\}$$

Tento decentralizovaný a nenavracející se PC systém přijímá dosažením koncových stavů ve svých komponentách i vyprázdněním jejich zásobníků jazyk $\{a^nb^nc^n \mid n \geq 1\}$, který není bezkontextový. Obě jeho komponenty postupně čtou ze svých vstupů řetězec symbolů a a zároveň na svých zásobnících generují řetězce stejné délky, které jsou tvořeny symboly b, v případě první komponenty, nebo symboly c, jedná-li se o druhou komponentu. Druhá komponenta při dosažení prvního symbolu b na vstupu vyvolá komunikační krok, pomocí něhož na svůj zásobník získá řetězec b^n, $n \geq 1$, vygenerovaný první komponentou. Podobně se zachová i komponenta A_1, která vyvolá komunikační krok po tom, co ze svého zásobníku odstraní poslední symbol b. Od komponenty A_2 získá řetězec c^n, $n \geq 1$, který porovná se zbývající částí svého vstupní řetězce. Pro tento jazyk je možné sestrojit i samostatný hluboký zásobníkový automat, který jej přijímá — jeho konstrukci zde neuvádíme a čtenáře případně odkazujeme na příklad uvedený v [10]. Systém $\mathcal{A}$ z tohoto příkladu využívá paralelního generování řetězců b^n a c^n, $n \geq 1$, k tomu, aby urychlil zpracování vstupního řetězce.

6.2 Vyjadřovací síla

V této sekci se budeme věnovat vyjadřovací síle paralelně komunikujících systémů hlubokých zásobníkových automatů. Konkrétně se zaměříme na navracející se verze těchto systémů — dokážeme, že takovýto systém tvořený čtyřmi komponentami je schopen rozpoznávat třídu rekurzivně vyčíslitelných jazyků. Při dokazování následující věty budeme vycházet z důkazu podobného tvrzení zkoumaného v [4].

Věta 6.1. Pro každý jazyk $L \in \mathbf{RE}$ existuje *R-PCDPAS*(4) $\mathcal{A}$ takový, že $L_f(\mathcal{A}) = L\$$, kde \$ je symbol značící konec vstupního řetězce, který se nevyskytuje v L.

Důkaz. Nejprve ukážeme, že pro libovolný dvouzásobníkový automat M je možné sestrojit takový *R-PCDPAS*(4) $\mathcal{A}$, pro který platí, že $L_f(\mathcal{A}) = L_f(M)\$$.

Mějme dvouzásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q^0, Z_1^0, Z_2^0, F)$. Beze ztráty na obecnosti předpokládáme, že $\Sigma \cap \Gamma = \emptyset$. V následujících třech krocích sestavíme pomocné množiny Γ_r, Q_c a Q':

1. Pro každé přechodové pravidlo $qaZ_1Z_2 \to p\alpha_1\alpha 2 \in \delta$, kde $q, p \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$, $Z_1, Z_2 \in \Gamma$, $\alpha_1, \alpha_2 \in \Gamma^*$, přidáme do Γ_r symbol $\langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2 \rangle$.
2. Pro všechna $q \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$ a $Z \in \Gamma$ přidáme $\langle q, a, Z \rangle$ do Q_c.
3. Pro všechny stavy $q \in Q$ přidáme q' do Q'.

Dále nechť $\Gamma_{\mathcal{A}} = \Gamma \cup \Gamma_r \cup \{Z_0^0, X, \#\}$, $\Sigma_{\mathcal{A}} = \Sigma \cup \{\$\}$ a $\mathcal{A}$ je následující *R-PCDPAS*(4):

$$\mathcal{A} = (\Sigma_{\mathcal{A}}, \{K_1, K_4\}, \Gamma_{\mathcal{A}}, A_1, A_2, A_3, A_4)$$

Jednotlivé komponenty $\mathcal{A}$ zkonstruujeme následovně:

$$\begin{aligned}
A_1 &= (Q \cup Q' \cup \{f_1\}, \Sigma_{\mathcal{A}}, \Gamma_{\mathcal{A}}, \delta_1, q^0, Z_0^0, \{f_1\}) \\
A_2 &= (Q \cup Q_c \cup \{f_2\}, \Sigma_{\mathcal{A}}, \Gamma_{\mathcal{A}}, \delta_2, q^0, Z_1^0, \{f_2\}) \\
A_3 &= (Q \cup Q_c \cup \{f_3\}, \Sigma_{\mathcal{A}}, \Gamma_{\mathcal{A}}, \delta_3, q^0, Z_2^0, \{f_3\}) \\
A_4 &= (\{q_4^0, f_4\}, \Sigma_{\mathcal{A}}, \Gamma_{\mathcal{A}}, \delta_4, q_4^0, \#, \{f_4\})
\end{aligned}$$

Množiny přechodových pravidel všech komponent $\mathcal{A}$ definujeme takto:

1. Pro všechny symboly $\langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2 \rangle \in \Gamma_r$ vytvoříme přechodová pravidla

$$\begin{aligned}
1qaZ_0^0 &\to p'a\langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2 \rangle \in \delta_1 \\
1qaZ_1 &\to \langle q, a, Z_1 \rangle aK_1 \in \delta_2 \\
1qaZ_2 &\to \langle q, a, Z_2 \rangle aK_1 \in \delta_3
\end{aligned}$$

2. Pro každý symbol $\langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2 \rangle \in \Gamma_r$ sestrojíme přechodová pravidla

$$\begin{aligned}
1p'Z_0^0 &\to pZ_0^0 \in \delta_1 \\
1\langle q, a, Z_1 \rangle \varepsilon \langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2 \rangle &\to p\beta_1 \in \delta_2 \\
1\langle q, a, Z_2 \rangle \varepsilon \langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2 \rangle &\to p\beta_2 \in \delta_3
\end{aligned}$$

kde $\beta_i = \alpha_i$ pokud $\alpha_i \neq \varepsilon$, anebo $\beta_i = K_4$ je-li $\alpha_i = \varepsilon$, pro $1 \leq i \leq 2$.

3. Pro všechna $f \in F$ a $Z \in (\Gamma \cup \{Z_0^0, \#\})$ zavedeme pravidla

$$1f\$Z \to f_i\$Z \in \delta_i$$
$$1f_i\varepsilon Z \to f_i Z \in \delta_i$$

kde $1 \leq i \leq 3$.

4. Množina přechodových pravidel δ_4 komponenty A_4 obsahuje pravidla

$$1q_4^0\varepsilon\# \to q_4^0\# \qquad 1q_4^0\varepsilon\# \to f_4X\# \qquad 1f_4aX \to f_4aX$$

kde $a \in (\Sigma_{\mathcal{A}} \cup \{\varepsilon\})$.

Uvažujme, že se M nachází ve stavu q a na vrcholech jeho zásobníků se nacházejí symboly Z_1 a Z_2. Přechodem z této konfigurace se M dostane do stavu q, přečte symbol $a \in (\Sigma \cup \{\varepsilon\})$ ze svého vstupu a vrcholy Z_1 a Z_2 svých zásobníků nahradí řetězci α_1 a α_2. Jak si dále popíšeme, tento přechod lze simulovat v $\mathcal{A}$.

Komponenta A_1 nacházející se ve stavu q nahradí symbol Z_0^0 na vrcholu svého zásobníku řetězcem $a\langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2\rangle$, který určuje symbol, jenž má být přečten ze vstupu, a také pravidlo zvolené k simulaci. Ve stejnou chvíli se obě komponenty A_2 a A_3 nacházejí ve stavu q a symboly X a Y jsou vrcholy jejich zásobníků. Obě komponenty současně přejdou do stavů $\langle q, x, X\rangle$ a $\langle q, y, Y\rangle$ a vrcholy svých zásobníků nahradí řetězci xK_1 a yK_1. Symboly x a y představují symboly určené k přečtení ze vstupu. Pokud je některý ze symbolů a, x nebo y různý od ε, musí příslušná komponenta ve svém příštím kroku přečíst tento symbol ze svého vstupu. Takový krok musí být vždy proveden současně všemi třemi komponentami — v ostatních případech dojde k zastavení systému. Následně dojde k vyvolání komunikačního kroku, na jehož konci se na vrcholech zásobníků A_2 a A_3 nachází symbol simulovaného pravidla $\langle q, a, Z_1, Z_2, p, \alpha_1, \alpha_2\rangle$. Další přechod systému je možný pouze tehdy, pokud platí $a = x = y$, $X = Z_1$ a $Y = Z_2$. Pokud jsou tyto podmínky splněny, A_2 a A_3 nahradí symbol simulovaného pravidla na vrcholech svých zásobníků řetězci α_1 a α_2 a spolu s A_1 přejdou do stavu p, což značí připravenost systému k simulování dalšího kroku M.

Úspěšné ukončení simulace je v prvních třech komponentách indikováno dosažením jednoho z koncových stavů M a nalezením značky \$ na vstupu. V takové situaci přejde každá z komponent A_1, A_2 a A_3 do svého jediného koncového stavu a také přečte značku konce vstupu. Ve svém konečném stavu potom tyto komponenty setrvají, dokud A_4 nezpracuje zbývající část svého vstupu, viz dále.

Během této simulace jednotlivých přechodů M udržuje pomocný automat A_4 svůj zásobník prázdný. Provedením dotazu na jeho obsah potom může komponenta A_2 nebo A_3 odstranit symbol z vrcholu svého zásobníku, což je operace, které není hluboký zásobníkový automat sám schopen. Jakmile již není dále potřeba, A_4 nedeterministicky přejde do svého koncového stavu a začne postupně číst svůj vstupní řetězec. Od této chvíle způsobí dotaz směřující na A_4 zastavení celého systému.

Každý přechod M je tedy možné simulovat sekvencí přechodů v $\mathcal{A}$. Podoba této sekvence je pevně dána použitým přechodovým pravidlem automatu M. Systém $\mathcal{A}$ zkonstruovaný podle výše uvedeného postupu přijme vstup $w\$$ právě tehdy, je-li řetězec w přijat simulovaným automatem M a pokud všechny komponenty $\mathcal{A}$ za pomoci značky \$ dosáhnou svých koncových stavů. Protože víme, že dvouzásobníkové automaty rozpoznávají rekurzivně vyčíslitelné jazyky (viz [6]), platí i tvrzení věty. □

6.3 Determinismus

Kapitolu o paralelně komunikujících systémech hlubokých zásobníkových automatů zakončíme zkoumáním determinismu v těchto systémech. Stejně jako tomu bylo již u distribuovaných systémů, můžeme určit systémy, které jsou deterministické s ohledem na své komponenty. Na rozdíl od CD systémů ovšem již nemůžeme zkoumat determinismus v mechanismech samotného systému — proces komunikačního kroku je pevně definován a pokud lze tento krok provést, je vždy proveden deterministicky. Pro úplnost ovšem uveďme, že v případě nesynchronizovaných PC systémů je možné zkoumat také *rychlost* jeho komponent, danou počtem přechodů provedených jednotlivými komponentami mezi dvěma komunikačními kroky.

Definice 6.7. Nechť $\mathcal{A} = (\Sigma, K, \Gamma, A_1, \ldots, A_n)$ je $PCDPAS(n)$. Říkáme, že $\mathcal{A}$ je *deterministický* právě tehdy, pokud jsou všechny jeho komponenty A_i, $1 \leq i \leq n$, striktně deterministické (viz 4.3).

Dále se zaměříme na vlastnost paralelně komunikujících systémů, který plyne ze samotné jejich podstaty — tedy synchronizovaného paralelního běhu jejich komponent. Nedeterminismus ve více komponentách totiž může mít silný vliv na celý systém. V následujícím textu se budeme zabývat množinami konfigurací, do kterých může přejít komponenta nebo systém z dané výchozí konfigurace. U všech takových výchozích konfigurací budeme nadále předpokládat, že nemohou v daném systému vyvolat komunikační krok.

Uvažujme $PCDPAS(n)$ $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$, kde pro všechna i, $1 \leq i \leq n$, definujeme jeho komponenty jako $A_i = (Q_i, \Sigma, \Gamma, \delta_i, q_i^0, Z_i^0, F_i)$. Dále nechť (q_i, x_i, α_i) je konfigurace komponenty A_i. Množinu konfigurací, do kterých může A_i přejít z dané konfigurace, vyjádříme následovně:

$$\Delta_i(q_i, x_i, \alpha_i) = \{(p_i, y_i, \beta_i) \mid (q_i, x_i, \alpha_i) \vdash (p_i, y_i, \beta_i), p_i \in Q_i, y_i \in \Sigma^*, \beta_i \in \Gamma^*\}$$

Mějme konfiguraci $u = (q_1, x_1, \alpha_1, \ldots, q_n, x_n, \alpha_n)$ systému $\mathcal{A}$. Množinu konfigurací do kterých je systém $\mathcal{A}$ z konfigurace u schopen přejít můžeme vyjádřit takto:

$$\Delta_{\mathcal{A}}(u) = \{(p_1, y_1, \beta_1, \ldots, p_n, y_n, \beta_n) \mid (p_i, y_i, \beta_i) \in \Delta_i(q_i, x_i, \alpha_i), 1 \leq i \leq n\}$$

Pozorujeme tedy, že pokud současně alespoň dvě komponenty $\mathcal{A}$ provedou z dané konfigurace nedeterministický přechod, celý systém $\mathcal{A}$ musí přejít do jedné z konfigurací daných možnými kombinacemi jednotlivých konfigurací, do kterých mohou přejít jeho komponenty. Počet konfigurací, do kterých může $\mathcal{A}$ přejít z konfigurace $u = (q_1, x_1, \alpha_1, \ldots, q_n, x_n, \alpha_n)$, je vyjádřen následujícím vztahem:

$$card(\Delta_{\mathcal{A}}(u)) = \prod_{i=1}^{n} card(\Delta_i(q_i, x_i, \alpha_i))$$

Kapitola 7

Syntaktická analýza

V této kapitole se budeme věnovat návrhu takových metod syntaktické analýzy, které budou založeny na systémech hlubokých zásobníkových automatů. Tyto systémy jsme představili v kapitolách 5 a 6. Aspektům navrhovaných metod, které jsou nezávislé na konkrétním typu použitého systému hlubokých zásobníkových automatů, se budeme věnovat v následujících dvou sekcích. Částmi, které jsou specifické pro daný systém se poté budeme zabývat odděleně.

Nejprve se zaměříme na základní strukturu samotného syntaktického analyzátoru a následně navrhneme řešení pro práci s nedeterministickými systémy hlubokých zásobníkových automatů. Ve zbývajících sekcích této kapitoly se budeme zabývat problémy, které jsou specifické pro daný typ automatového systému — například výběr komponenty pro aktivaci u distribuovaných systémů.

7.1 Základní struktura

Tuto sekci věnujeme návrhu základní struktury syntaktického analyzátoru založeného na hlubokých zásobníkových automatech a jejich systémech. U takovéhoto analyzátoru budeme rozlišovat jeho dvě základní části — *řídící jednotku* a jednotlivé *komponenty*. Budeme uvažovat pouze komponenty tvořené řízenými hlubokými zásobníkovými automaty, protože tato varianta disponuje oproti jejich neřízené verzi více prostředky, které usnadňují návrh deterministických automatů.

Nejdůležitější částí syntaktického analyzátoru je jeho řídící jednotka — je zodpovědná za řízení celého procesu syntaktické analýzy. Protože pracujeme s automatovými systémy, bude tato jednotka plnit celkem dvě úlohy, přičemž první souvisí s již zmíněným řízením průběhu syntaktické analýzy. Její druhou úlohou bude sloužit jako kontrolní prvek použitého systému hlubokých zásobníkových automatů. Bude tedy zodpovědná za operace, jako je provedení komunikačního kroku u PC systémů nebo aktivace komponent u CD systémů. Poznamenejme, že *řídící komponenta*, kterou rozlišujeme u PC systémů, nesouvisí s *řídící jednotkou* syntaktického analyzátoru.

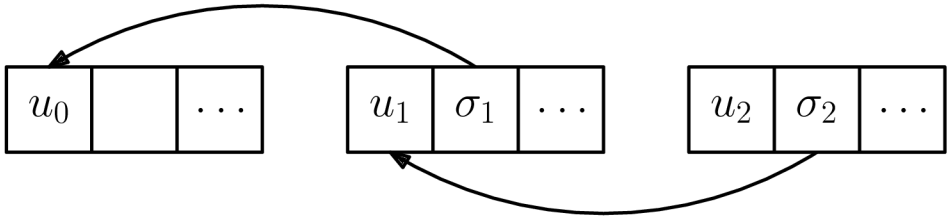
Proces syntaktické analýzy má dva hlavní výstupy. Prvním je informace o tom, zda byl zkoumaný vstup analyzátorem přijat nebo ne. Potvrzení přijetí daného řetězce je samozřejmě indikováno tak, že dojde k jeho přijetí použitým systémem. Odmítnutí vstupu ovšem není možné takto přesně vymezit — kromě zastavení systému může také například dojít k jeho uváznutí v nekonečné smyčce. Pro některé vstupy je také možné, že analyzátor nebude schopen dosáhnout jakéhokoliv výsledku. To může být mimo jiné zapříčiněno vyčer-

páním dostupných prostředků. Druhým výstupem analyzátoru je tradičně seznam pravidel, jejichž postupné použití vedlo k přijetí či odmítnutí daného vstupu. Protože u hlubokých zásobníkových automatů rozlišujeme i přechod přečtením symbolu, který se realizuje bez použití pravidel, zobecníme podobu tohoto výstupu. Namísto seznamu použitých pravidel bude výstup tvořen seznamem konfigurací, mezi kterými systém postupně přecházel. Relevantní konfigurace systému i jednotlivých komponent budou označeny tak, aby byl jednoznačně určen způsob, jakým byla daná konfigurace dosažena. Takový seznam můžeme pozorovat u příkladu 4.1.

Za vytváření tohoto seznamu bude zodpovědná řídící jednotka, která jej bude v průběhu syntaktické analýzy sestavovat podle přechodů, které provede použitý systém a jeho komponenty. Proto dále zavedeme pomocnou konstrukci takzvaného *obrazu* stavu systému. Uvážíme-li systém $\mathcal{A}$ tvořený n komponentami, je *obrazem* jeho stavu $(n+3)$-tice

$$(u, \sigma, \tau_{\mathcal{A}}, \tau_1, \tau_2, \ldots, \tau_n)$$

kde u je konfigurace $\mathcal{A}$, σ je *předcházející* obraz stavu systému, $\tau_{\mathcal{A}}$ je *systémová značka* a τ_i, $1 \leq i \leq n$, jsou *komponentní značky*. Konfigurace jednotlivých komponent jsou dány konfigurací celého systému. Předcházející obraz stavu systému σ zachycuje systém $\mathcal{A}$ v konfiguraci v, přičemž platí, že systém $\mathcal{A}$ přešel z konfigurace v do u právě jedním krokem. Vztah mezi jednotlivými obrazy je znázorněn na obrázku 7.1. Pomocí jednotlivých *značek* je přesně určen způsob, jakým došlo k přechodu mezi u a v. Značka se může vztahovat k určité komponentě nebo celému systému. V tomto návrhu neklademe žádné požadavky na jejich podobu, pouze zavádíme jejich význam. Díky vazbě mezi jednotlivými obrazy je řídící jednotka schopna kdykoli zrekonstruovat sled přechodů vedoucích z počáteční konfigurace systému až do té aktuální.



Obrázek 7.1: Vazby mezi obrazy stavů systému

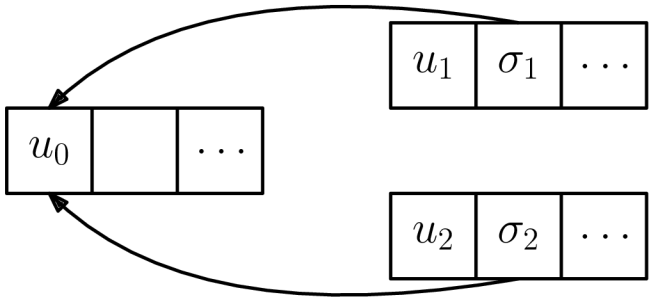
7.1.1 Označení konce vstupu

Uveďme nyní metodu práce se vstupním řetězcem, která může pro některé jazyky usnadnit návrh automatových systémů, které je přijímají. Touto metodou je použití speciální značky, která určuje konec vstupního řetězce. V příkladě 4.1 je k tomuto účelu použit symbol \$. Pro tyto účely může syntaktický analyzátor zajistit, že se na konci analyzovaného vstupního řetězce vždy nachází značka konce vstupu, kterou je možné libovolně číst, ale není možné ji ze vstupu plně odstranit, podobně jako je tomu u symbolu dna zásobníku u hlubokých zásobníkových automatů. Při použití této metody je nutné zajistit, aby byl vstupní řetězec tvořený pouze tímto symbolem považován za přečtený při vyhodnocování toho, zda byl vstup přijat či nikoli. Je také samozřejmé, že tento symbol se může ve vstupním řetězci vyskytnout pouze na jeho konci.

7.2 Nedeterminismus

Hluboké zásobníkové automaty i jejich systémy mohou být nedeterministické. Proto navrhneme metodu, která umožní syntaktickému analyzátoru pracovat s nedeterministickými automaty a jejich systémy. V sekci 7.1 jsme zavedli takzvaný *obraz* stavu systému, pomocí něhož jsme schopni zachytit nejen aktuální stav syntaktického analyzátoru, ale také kompletní historii výpočtů, které vedly ke stávajícímu stavu. Protože se jednotlivé obrazy odkazují vždy na přecházející stav systému, můžeme nedeterministické přechody modelovat stromem odpovídajících obrazů stavů systému. Na obrázku 7.2 je znázorněn nedeterministický přechod z konfigurace u_0 do konfigurace u_1 nebo u_2. Každý listový uzel takového stromu představuje aktuální stav analýzy v příslušné *linii* nebo také *větvi* výpočtů syntaktického analyzátoru.

Je nezbytně nutné, aby syntaktický analyzátor zkoumal všechny možné linie — pro přijetí jeho vstupu postačí, když bude přijat v libovolné z nich. Naopak vstupní řetězec je možné odmítnout pouze tehdy, dojde-li k jeho odmítnutí ve všech větvích. Poznamenejme také, že pokud při provádění analýzy v rámci jedné větve dojde k zastavení systému, nemusí nutně dojít k zastavení celého analyzátoru. Jednotlivé linie jsou totiž na sobě nezávislé, a proto odmítnutí vstupu v jedné z nich nevylučuje jeho přijmutí v jiné větvi. Z tohoto důvodu musí řídící jednotka pokračovat zkoumáním zbývajících linií — k zastavení analyzátoru může dojít pouze tehdy, pokud neexistuje žádná větev, ve které je možné pokračovat v analýze.



Obrázek 7.2: Strom obrazů stavů systému

Mluvíme tedy o problému prohledávání stavového prostoru (viz [12]). Syntaktický analyzátor hledá takovou konfiguraci použitého automatového systému, ve které je přijat zkoumaný vstup. Tuto konfiguraci hledá v prostoru, který je reprezentován stromem všech obrazů stavů systému, jenž byly zaznamenány řídící komponentou v průběhu činnosti syntaktického analyzátoru.

Je zřejmé, že s každou zkoumanou větví stavového prostoru dochází k nárůstu náročnosti celého procesu syntaktické analýzu. V této souvisloti upozorněme na způsob, jakým u PC systémů ovlivňuje nedeterminismus komponent celý systém (viz 6.3).

7.3 CD systémy hlubokých zásobníkových automatů

V této sekci se zaměříme na použití distribuovaných systémů hlubokých zásobníkových automatů v navrhovaném syntaktickém analyzátoru. Nejprve zavedeme upravené verze módů $\leq k$ a $\geq k$, u kterých již nebude možné nedeterministicky zvolit konkrétní počet kroků, které

aktivní komponenta provede. Později se budeme věnovat problematice výběru komponenty k aktivování.

7.3.1 Derivační módy

Distribuovaný systém hlubokých zásobníkových automatů vždy pracuje v jistém derivačním módu, viz 5.4 a 5.5. Pro použití v syntaktické analýze se jako nejvhodnější jeví dva z těchto módů — t-mód a $= k$-mód. Důvodem jsou jednoznačné požadavky těchto módů na počet přechodů, které musí aktivní komponenta systému v rámci jedné derivace provést před tím, než bude možné aktivovat další komponentu. V případě $= k$-módu je tento požadavek explicitní — aktivní komponenta musí provést právě k přechodů. U systémů, které operují v t-módu je aktivace další komponenty možná právě tehdy, když stávající aktivní komponenta již není schopna žádného dalšího přechodu.

Na druhou stranu $\leq k$-mód a $\geq k$-mód pouze ukládají, aby počet kroků provedených aktivní komponentou spadal do intervalu, který je dán konkrétní hodnotou k. Tyto celočíselné intervaly jsou $\langle 0, k\rangle$, v případě $\leq k$-módu, nebo $\langle k, \infty)$, jedná-li se o $\geq k$-mód. Výběr konkrétního počtu kroků z těchto intervalů je proveden nedeterministicky, což je z pohledu syntaktické analýzy nežádoucí vlastnost.

Z tohoto důvodu budeme dále pracovat s modifikacemi těchto módů, které vzniknou kombinací příslušného módu s t-módem. Pro tyto varianty budeme používat označení $\preceq k$ a $\succeq k$. Komponenta systému, který pracuje v jednom z těchto módů, provede během jedné derivace l kroků tak, jako by pracovala podle t-módu. Současně ovšem musí platit, že $l \leq k$, jedná-li se o $\preceq k$-mód, nebo $l \geq k$, v případě $\succeq k$-módu. Pokud daný systém pracuje podle $\preceq k$-módu a jeho aktivní komponenta provede $l = k$ přechodů, dojde k aktivaci další komponenty bez ohledu na to, zda stávající aktivní komponenta byla schopna dalšího přechodu. Tyto módy tedy můžeme také vnímat jako rozšíření t-módu o horní, nebo dolní hranici počtu kroků provedených v jedné derivaci.

Pro syntaktickou analýzu založenou na systémech distribuovaných hlubokých zásobníkových automatů budeme dále uvažovat následující množinu přechodových módů, podle kterých mohou tyto systémy operovat:

$$M = \{t\} \cup \{= k, \preceq k, \succeq k \mid k \geq 1\}$$

7.3.2 Řídící jednotka

Pokud bude v syntaktickém analyzátoru použit distribuovaný systém hlubokých zásobníkových automatů, musí řídící jednotka analyzátoru zastat také funkci kontrolního prvku použitého systému. Tento kontrolní prvek například zodpovídá za předávání řízení mezi jednotlivými komponentami (formou deaktivace stávající aktivní komponenty a následné aktivace další komponenty) a také za samotný proces výběru komponenty určené k aktivaci. Dále také musí zaručit zastavení celého systému, dojde-li k porušení podmínek, které jsou dané zvoleným derivačním módem systému. Takovým porušením je například zastavení komponenty dříve, než provede požadovaný počet přechodů.

Uvažujme CD systém hlubokých zásobníkových automatů $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_n, \lambda_0)$, který se nachází v konfiguraci $u = (x, q_1, \alpha_1, q_2, \alpha_2, \ldots, q_n, \alpha_n, \lambda)$. Pokud již komponenta A_λ není schopna provést přechod z konfigurace u, případně pokud takovému přechodu brání omezení počtu přechodů dané derivačním módem, je nutné aktivovat další komponentu. Množinu všech komponent, které je možné v konfiguraci u aktivovat, získáme následovně:

$$next(\mathcal{A}, u) = \{A_i \mid u \vdash_i v, v \text{ je konfigurace } \mathcal{A}, 1 \leq i \leq n\}$$

Je-li tato množina prázdná, dochází k zastavení systému, neboť žádná z jeho komponent není schopna provést z konfigurace u přechod. V ostatních případech je vybrána komponenta A_i, která je následně aktivována. Tuto skutečnost můžeme vyjádřit tak, že kontrolní prvek provede přechod do konfigurace $u' = (x, q_1, \alpha_1, q_2, \alpha_2, \ldots, q_n, \alpha_n, i)$. Pomocí stromu obrazů stavů systému je syntaktický analyzátor schopen modelovat i nedeterministický výběr komponenty, pokud je možné aktivovat více komponent.

7.4 PC systémy hlubokých zásobníkových automatů

Pokud bude v syntaktickém analyzátoru použit PC systém hlubokých zásobníkových automatů, bude řídící jednotka analyzátoru také plnit funkci kontrolního prvku použitého systému. Tento prvek v první řadě zajišťuje synchronizaci komponent. Jak již víme, všechny komponenty PC systému vždy současně provedou právě jeden přechod, což musí být zaručeno právě kontrolním prvkem. Ten je také zodpovědný za zablokování systému, pokud některá z komponent není schopna dalšího přechodu.

Druhou a neméně důležitou úlohou kontrolního prvku je detekce a případné vykonání komunikačního kroku. Komunikační krok bude proveden tehdy, je-li na vrcholu zásobníku alespoň jedné komponenty dotazovací symbol.

Uvažme PC systém hlubokých zásobníkových automatů $\mathcal{A} = (\Sigma, K, \Gamma, A_1, A_2, \ldots, A_n)$ nacházející se v konfiguraci $u = (q_1, x_1, \alpha_1, q_2, x_2, \alpha_2, \ldots, q_n, x_n, \alpha_n)$. Množinu všech komponent, které v u vznášejí splnitelný dotaz na některou z komponent $\mathcal{A}$, lze vyjádřit následujícím způsobem:

$$recv(\mathcal{A}, u) = \{A_i \mid \alpha_i = K_j\beta_i, \alpha_j = Z_j\beta_j, K_j \in K, Z_j \notin K, 1 \leq i, j \leq n\}$$

Dále získáme množinu komponent, které budou v konfiguraci u komunikovat obsah svého zásobníku s některou komponentou $\mathcal{A}$, a to následovně:

$$send(\mathcal{A}, u) = \{A_i \mid \alpha_i = Z_i\beta_i, \alpha_j = K_i\beta_j, Z_i \notin K, K_i \in K, 1 \leq i, j \leq n\}$$

Pokud v takovém případě platí, že $recv(\mathcal{A}, u) = \emptyset$, pak došlo ke vzniku kruhového dotazu, který způsobí zastavení systému. Výjimkou je případ, kdy komponenta vznese dotaz na obsah svého vlastního zásobníku. V tomto případě je dotazovací symbol nahrazen zbývající částí obsahu zásobníku dané komponenty.

Samotný komunikační krok je poté proveden ve dvou až třech krocích v závislosti na tom, je-li $\mathcal{A}$ navracející se systém. V prvním kroku řídící jednotka získá obsahy zásobníků všech komponent z $send(\mathcal{A}, u)$, ze kterých odstraní symbol dna zásobníku. Takto upravenými obsahy zásobníků jsou poté v druhém kroku nahrazeny odpovídající komunikační symboly na vrcholech zásobníků komponent z $recv(\mathcal{A}, u)$. Jedná-li se o navracející se systém, jsou obsahy zásobníků komponent z $send(\mathcal{A}, u)$ nahrazeny příslušným počátečním zásobníkovým symbolem.

Kapitola 8

Demonstrační aplikace

Jako součást této práce byla na základě návrhu syntaktického analyzátoru z kapitoly 7.1 implementována demonstrační aplikace v jazyce Python. Tato aplikace je schopna na základě uživatelem poskytnuté definice distribuovaného nebo paraleleně komunikujícího systému simulovat činnost syntaktického analyzátoru, který pracuje s daným automatovým systémem. Pro demonstrační účely je aplikace také schopna simulovat samotný řízený hluboký zásobníkový automat. Soubor s definicí automatového systému, který má být použit syntaktickým analyzátorem, je ve formátu JSON. Sadu souborů s příklady je možné nalézt na přiloženém disku.

8.1 Struktura aplikace

Demonstrační aplikace se skládá ze tří hlavních částí. První je samozřejmě implementace řízeného hlubokého zásobníkového automatu. Při jeho implementaci jsme vycházeli z jeho formální definice, která byla zavedena v kapitolách 3 a 4. Protože tyto automaty budou použity jako komponenty v distribuovaném nebo paralelně komunikujícím systému, zohlednili jsme při jejich implementaci také ty operace, které nad nimi budou provádět kontrolní jednotky daných systémů. Uveďme jako příklad manipulaci s obsahem zásobníku při komunikačním kroku v paralelně komunikujícím systému nebo změny vstupního řetězce, které nastanou při neaktivitě komponenty distribuovaném systému.

Zbylé dvě části jsou řídící jednotky určené pro simulaci distribuovaných a paralelně komunikujících systémů. Obě tyto jednotky jsou schopny nedeterministické syntaktické analýzy, které se budeme věnovat v následující sekci. Dále každá z těchto jednotek implementuje jak operace specifické pro daný systém (viz kapitoly 5 a 6), tak metody z kapitoly 7 týkající se řídící jednotky. Demonstrační aplikace volí konkrétní typ řídící jednotky na základě uživatelem poskytnutého definičního souboru.

8.2 Nedeterministická analýza

Demonstrační aplikace je schopna provádět i nedeterministickou syntaktickou analýzu. Při její implementaci jsme vycházeli z metody popsané v sekci 7.2. Pro prohledávání stavového prostoru je použit algoritmus *prohledávání do šířky* (anglicky *breadth-first search*; viz [12]). Tato metoda je zvolena proto, že je *úplná* a také *optimální*. Protože je metoda *úplná* [12], je zaručeno, že existuje-li nějaké řešení, je touto metodou vždy nalezeno. Řešením v tomto kontextu máme samozřejmě na mysli takovou konfiguraci automatového systému, ve které

dojde k přijmutí vstupního řetězce. *Optimálnost* [12] této metody znamená, že je-li nalezeno řešení, je to vždy takové řešení, které se nachází nejblíže kořenovému uzlu prohledávaného stromu. Jinými slovy je touto metodou nalezeno takové řešení, při kterém simulovaný systém provedl nejmenší možný počet kroků.

Prohledávání stavového prostoru do šířky je implementováno pomocí fronty obrazů stavů systému, kterou udržuje řídící jednotka. Prvky této fronty představují listové uzly stromu, který je popsán v 7.2. Řídící jednotka pracuje tak, že z fronty vyjme obraz stavu systému a simuluje přechod systému z dané konfigurace. Pro každou konfiguraci, do které je systém schopen přejít, je vytvořen obraz stavu systému v dané konfiguraci. Tento obraz je následně vložen na konec fronty. Dojde-li při simulaci přechodu k zastavení systému, je daný obraz stavu systému vložen do seznamu neúspěšně zkoumaných větví.

Pokud je vstupní řetězec analyzátorem odmítnut, je v seznamu neúspěšně zkoumaných větví nalezena takzvaná *nejméně neúspěšná větev*, která bude součástí výstupu syntaktického analyzátoru. Míra neúspěchu větve je měřena délkou úspěšně přečtené části vstupního řetězce v okmažiku zastavení systému. Při odmítnutí vstupu je výstupem analyzátoru tedy taková výpočetní větev, ve které byla zpracována nejdelší část vstupního řetězce. Protože tato míra může být v závislosti na konkrétním použitém systému zavádějící, je analyzátor také schopen uživateli poskytnout všechny neúspěšně zkoumané větve stavového prostoru k dalšímu zkoumání.

Kapitola 9

Závěr

Prvním cílem této práce bylo studium distribuovaných a paralelně komunikujících automatových systémů a následné zavedení hlubokých zásobníkových automatů do těchto systémů. Při zkoumání samotných hlubokých zásobníkových automatů jsme identifikovali některé jejich vlastnosti, které by mohly zkomplikovat nasazení těchto automatů v syntaktické analýze. Z tohoto důvodu jsme zavedli takzvanou řízenou variantu hlubokého zásobníkového automatu, která je schopna při volbě přechodového pravidla zohlednit také její aktuální vstupní symbol. Dokázali jsme, že hluboké zásobníkové automaty jsou speciálním případem těch řízených.

Na základě řízených zásobníkových automatů jsme zavedli paralelně komunikující a distribuované systémy tvořené těmito automaty. Při jejich zavádění jsme vycházeli ze současného výzkumu v této oblasti, který naopak do značné míry kopíruje výzkum gramatických systémů. V případě navracející se varianty paralelně komunikujících systémů jsme dokázali, že vyjadřovací síla těchto systémů odpovídá síle Turingových strojů. Dále jsme také zkoumali různé formy determinismu v těchto systémech a to jak na úrovni jednotlivých komponent tak i systémových mechanismů.

Distribuované i paralelně komunikující systémy řízených hlubokých zásobníkových automatů jsme v závěru této práce použili při návrhu syntaktického analyzátoru založeného na těchto systémech. Tyto poznatky jsme využili při implementaci demonstrační aplikace.

9.1 Směřování dalšího výzkumu

Na závěr uvedeme několik zajímavých otevřených problémů, které se v rámci této práce nepodařilo zodpovědět. Prvním je přesné určení vyjadřovací síly distribuovaných systémů. Předpokládáme, že po vzoru podobně definovaných systémů dojde ke zvýšení jejich vyjadřovací síly oproti samotným zásobníkovým automatům.

Druhým otevřeným problémem je vyjadřovací síla nenavracejících se paralelně komunikujících systémů. Ukázali jsme, že navracející variantou jsme schopni simulovat Turingův stroj, ale při této konstrukci jsme pomocí návratu zásobníku do původního stavu po komunikačním kroku simulovali odstranění symbolu ze zásobníku, což je operace, které samotný hluboký zásobníkový automat není schopen.

Z toho plyne možnost zkoumání vlivu různých modifikací hlubokých zásobníkových automatů na vyjadřovací sílu jak těchto automatů samotných, tak i jejich systémů. Uveďme jako příklad možnost expanze zásobníkového symbolu prázdným řetězcem nebo kombinace kroku přečtením a expanze symbolu.

Literatura

[1] Chomsky, N.: Three models for the description of language. *IRE Transactions on Information Theory*, 1956: s. 113–124, ISSN 0096-1000.

[2] Csuhaj-Varju, E.; Dassow, J.: On Cooperating/Distributed Grammar Systems. *J. Inf. Process. Cybern.*, ročník 26, č. 1-2, Březen 1990: s. 49–63, ISSN 0863-0593.

[3] Csuhaj-Varjú, E.; Mitrana, V.; Vaszil, G.: Distributed Pushdown Automata Systems: Computational Power. In *Proceedings of the 7th International Conference on Developments in Language Theory*, DLT'03, Berlin, Heidelberg: Springer-Verlag, 2003, ISBN 3-540-40434-1, s. 218–229.

[4] Csuhaj-Varjú, E.; Martín-Vide, C.; Mitrana, V.; aj.: Parallel communicating pushdown automata systems. *International Journal of Foundations of Computer Science*, 2000: s. 631–650.

[5] Dassow, J.; Mitrana, V.: Stack Cooperation in Multistack Pushdown Automata. *Journal of Computer and System Sciences*, ročník 58, č. 3, 1999: s. 611 – 621, ISSN 0022-0000.

[6] Hopcroft, J. E.; Motwani, R.; Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*. Boston: Addison-Wesley, druhé vydání, 2001, ISBN 0201441241.

[7] Krithivasan, K.; Balan, M. S.; Harsha, P.: Distributed Processing in Automata. *Int. J. Found. Comput. Sci.*, ročník 10, č. 4, 1999: s. 443–464.

[8] Martín-Vide, C.; Mateescu, A.; Mitrana, V.: Parallel finite automata systems communicating by states. *International Journal of Foundations of Computer Science*, 2002: s. 733–749.

[9] Meduna, A.: Deep Pushdown Automata. *Acta Inf.*, ročník 42, č. 8-9, Duben 2006: s. 541–552, ISSN 0001-5903.

[10] Meduna, A.; Zemek, P.: *Regulated Grammars and Automata*. Springer US, 2014, ISBN 978-1-4939-0368-9, 694 s.

[11] Păun, G.; Sântean, L.: Parallel communicating grammar systems: the regular case. *An. Univ. Bucureşti, Ser. Matem.-Inform.*, 1989: s. 55–63.

[12] Russell, S. J.; Norvig, P.: *Artificial intelligence: a modern approach (3rd edition)*. Prentice Hall, 2009.

Přílohy

Příloha A

Obsah CD

Přiložený disk obsahuje:

- Text této práce ve formátu PDF.
- Zdrojové soubory a obrázky potřebné k přeložení této práce pomocí systému LaTeX.
- Zdrojové kódy demonstrační aplikace.
- Sadu příkladů pro demonstrační aplikaci.