Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Diplomová práce

Vyhledávání a indexování v neseřazených stromech

Bc. Vojtěch Sobota

Vedoucí práce: prof. Ing. Bořivoj Melichar, DrSc.

9. května 2013

Poděkování

Mé díky patří panu prof. Bořivoji Melicharovi za jeho cennou zpětnou vazbu, kterou poskytoval při psaní mé diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 9. května 2013 .....

Odkaz na tuto práci

Sobota, Vojtěch. *Vyhledávání a indexování v neseřazených stromech.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

Abstract

This thesis contains a description of a deterministic attributed pushdown automaton accepting unordered trees in prefix notation. Furthermore it describes ways of extending such an automaton for subtree matching and for indexing of unordered trees. Required algorithms are presented as well as examples of automata. The thesis is based on results achieved for ordered trees as part of the algorithmic discipline arbology.

Keywords Tree, subtree, prefix notation, pushdown automaton, attributed pushdown automaton, pattern matching, indexing.

Abstrakt

Práce obsahuje popis deterministického atributovaného zásobníkového automatu, který přijímá stromy v prefixovém zápisu s libovolným seřazením

následníků. Dále popisuje postupy, jak takový automat rozšířit pro vyhledávání podstromů v prefixovém zápisu stromu a pro indexování neseřazených stromů. Jsou uvedeny potřebné algoritmy a pro názornost rovněž příklady automatů. Práce staví na výsledcích dosažených pro seřazené stromy v rámci algoritmické disciplíny arbologie.

Klíčová slova Strom, podstrom, prefixový zápis, zásobníkový automat, atributovaný zásobníkový automat, vyhledávání, indexování.

Obsah

Seznam obrázků

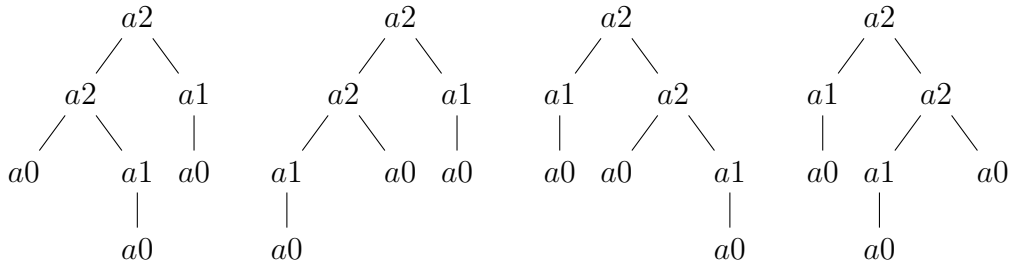
Seznam tabulek

Úvod

Algoritmy vyhledávání vzorů stromů, indexování stromů, hledání opakujících se podstromů apod. byly několik posledních desetiletí předmětem intenzivního studia. Stromové struktury se v přírodě a ve vědách, jako je informatika, matematika nebo například výpočetní biologie, objevují často a figurují v problémech, kde bývá klíčové mít schopnost s nimi efektivně operovat. Jako příklady takových problémů uveďme implementaci funkcionálních jazyků, systémy přepisování matematických výrazů, optimalizace kódu v překladačích, výběr kódu nebo dokazování vět [2].

V roce 2008 byla některými členy Pražského stringologického klubu založena algoritmická disciplína arbologie zabývající se algoritmy pracujícími nad stromy. Arbologie staví na teorii formálních jazyků a na dobře známých principech algoritmů ze stringologie, kde základním elementem je řetězec a kde se jako výpočetní model používá konečný automat. Arbologie analogicky pracuje s lineárními zápisy stromů [1]. Bylo dokázáno, že lineární zápis stromu lze vygenerovat pomocí bezkontextové gramatiky, vhodným modelem algoritmů pro zpracování lineárního zápisu stromu tedy může být zásobníkový automat [5], zachycující také rekurzivní podstatu stromů.

Nejběžnějšími lineárními reprezentacemi stromů jsou prefixový a postfixový zápis. Uvažujme nyní prefixový zápis stromu. Automaty, které přijímají prefixové zápisy seřazených stromů, vyhledávají vzory seřazených stromů v prefixových zápisech, a automaty, které reprezentují index seřazeného stromu již byly popsány v [5]. V článku [3] byl popsán algoritmus pro výpočet opakování podstromů v neseřazených stromech v lineárním čase. Jedna z dosud neúplně probádaných oblastí je však problematika efektivního použití zásobníkových automatů na neseřazené stromy. Neseřazeným stromem v podstatě myslíme množinu stromů vzniklých permutováním následníků uzlů jednoho konkrétního stromu. Obrázek 0.1 ilustruje



Obrázek 0.1: Čtyři možná uspořádání neseřazeného stromu

čtyři možná uspořádání neseřazeného stromu.

Zmíněné permutování následníků vnáší do problému určitou míru složitosti. S přihlédnutím na paměťovou složitost nelze k problému přistupovat tak, že bychom do stavů automatu zakódovali veškerá možná uspořádání následníků. Jakým způsobem je možno se tomuto problému vyhnout, je popsáno v bakalářské práci [6], kde byl představen zásobníkový automat přijímající prefixové zápisy neseřazených stromů. V této práci věnujeme kapitolu 2 shrnutí výsledků dosažených v [6] (poněkud změníme formalismus, ale princip algoritmů a automatu zůstane stejný), na tyto výsledky poté navážeme a představíme automat pro vyhledávání v neseřazeném stromu a také rozebereme problém indexování neseřazených stromů.

Kapitola 1

Základní definice

V této kapitole uvedeme definice základních pojmů převzaté z [5].

1.1 Abeceda

Abeceda je konečná neprázdná množina *symbolů*. *Ohodnocená abeceda* je konečná neprázdná množina symbolů, které mají definovanou *aritu*. Uvažujme ohodnocenou abecedu $\mathcal{A}$, arita symbolu $a \in \mathcal{A}$ je celé nezáporné číslo a značí se $arity(a)$. Předpokládáme, že $\mathcal{A}$ obsahuje alespoň jeden symbol arity 0. V příkladech používáme čísla na konci identifikátorů symbolů. Například $a2$ je symbol arity 2.

1.2 Strom, prefixový zápis stromu

Orientovaný graf G je dvojice (N, R), kde N je množina uzlů a R je množina seznamů hran, přičemž každý prvek R má tvar $((f, g_1), (f, g_2), \ldots, (f, g_n))$, kde $f, g_1, g_2, \ldots, g_n \in N$, $n \geq 0$. Tento prvek znamená, že pro uzel f je v grafu n hran vystupujících z uzlu f a vstupujících do uzlu g_1, do uzlu g_2 atd.

Posloupnost uzlů $(f_0, f_1, \ldots, f_n)$, $n \geq 1$, je *cesta* délky n z uzlu f_0 do uzlu f_n za předpokladu, že existuje hrana, která vystupuje z uzlu f_{i-1} a vstupuje do uzlu f_i, $1 \leq i \leq n$. *Cyklus* je cesta $(f_0, f_1, \ldots, f_n)$, kde $f_0 = f_n$. *DAG* (z anglického „directed acyclic graph“) je graf, který nemá žádný cyklus. *Ohodnocení* grafu $G = (N, R)$ symboly (ohodnocené) abecedy $\mathcal{A}$ je zobrazení z N do $\mathcal{A}$. V příkladech se na uzly odkazujeme pomocí symbolů, kterými jsou ohodnoceny.

Uvažujme uzel f, jeho *výstupní stupeň* je počet různých dvojic $(f, g) \in R$, kde $g \in N$. Analogicky *vstupní stupeň* uzlu f je počet různých dvojic $(g, f) \in R$, kde $g \in N$.

Strom je acyklický souvislý graf. Kterýkoli uzel stromu může být zvolen jako *kořen* stromu. Strom, který má kořen, se nazývá *kořenový strom*.

Strom může být *orientovaný*. *Kořenový orientovaný strom* t je DAG $t = (N, R)$ se zvláštním uzlem $r \in N$, zvaným *kořen*, takový, kde uzel r má vstupní stupeň 0, všechny ostatní uzly mají vstupní stupeň 1 a existuje pouze jedna cesta z uzlu r do každého uzlu $f \in N$, kde $f \neq r$.

Ohodnocený (kořenový, orientovaný) strom je strom, jehož všechny uzly $f \in N$ jsou ohodnoceny nějakým symbolem $a \in \mathcal{A}$, kde $\mathcal{A}$ je abeceda.

Uspořádaný (ohodnocený, kořenový, orientovaný) strom je strom ohodnocený symboly z ohodnocené abecedy $\mathcal{A}$ tak, že výstupní stupeň uzlu f ohodnoceného symbolem $a \in \mathcal{A}$ je $arity(a)$. Uzly ohodnocené symboly arity 0 se nazývají *listy*.

Prefixový zápis $pref(t)$ stromu t je lineární zápis daného stromu, do kterého je při procházení daným stromem do hloubky každý uzel zaznamenán právě jednou, a to při první návštěvě uzlu. Formální, rekurzivní definice je následující:

1. $pref(a) = a$, jestliže a je list stromu,
2. $pref(t) = a\ pref(b_1)\ pref(b_2) \ldots pref(b_n)$, kde a je kořen stromu t a $b_1, b_2, \ldots, b_n$ jsou přímí následníci a.

1.3 Jazyk, konečný a zásobníkový automat

Jazyk L nad abecedou $\mathcal{A}$ je množina řetězců nad $\mathcal{A}$. Symbol $\mathcal{A}^*$ značí množinu všech řetězců nad $\mathcal{A}$ včetně prázdného řetězce, který se značí ε. Platí, že $L \subset \mathcal{A}^*$. Množina $\mathcal{A}^+$ je definována jako $\mathcal{A}^+ = \mathcal{A}^* \setminus \{\varepsilon\}$. Podobně, pro řetězec $x \in \mathcal{A}^*$, symbol x^m, $m \geq 0$, značí sřetězení m řetězců x s tím, že $x^0 = \varepsilon$. Množina x^* je definována jako $x^* = \{x^m : m \geq 0\}$ a $x^+ = x^* \setminus \{\varepsilon\}$.

Nedeterministický konečný automat (NFA z anglického „nondeterministic finite automaton“) je pětice $M = (Q, \mathcal{A}, \delta, q_0, F)$, kde Q je konečná množina *stavů*, $\mathcal{A}$ je *vstupní abeceda*, δ je zobrazení z množiny $Q \times \mathcal{A}$ do množiny konečných podmnožin množiny Q, $q_0 \in Q$ je počáteční stav a $F \subseteq Q$ je množina koncových (přijímajících) stavů. Konečný automat M je *deterministický* (DFA z anglického „deterministic finite automaton“), pokud $\delta(q, a)$ obsahuje nejvýše jeden prvek pro všechna $q \in Q$ a $a \in \mathcal{A}$.

Každý NFA je možné transformovat na ekvivalentní DFA [5]. Stavy DFA vznikají během transformace jako podmnožiny stavů NFA. Tyto podmno-

žiny se nazývají *d-podmnožiny*, a i když se jedná o standardní množiny, často se místo složených závorek ({ }) zapisují pomocí hranatých závorek ([ ]).

Nedeterministický zásobníkový automat (nedeterministický PDA z anglického „pushdown automaton“) je sedmice $M = (Q, \mathcal{A}, G, \delta, q_0, Z_0, F)$, kde Q je konečná množina *stavů*, $\mathcal{A}$ je *vstupní abeceda*, G je *zásobníková abeceda*, δ je zobrazení z množiny $Q \times (\mathcal{A} \cup \{\varepsilon\}) \times G$ do množiny konečných podmnožin množiny $Q \times G^*$, $q_0 \in Q$ je počáteční stav, $Z_0 \in G$ je počáteční symbol v zásobníku a $F \subseteq Q$ je množina koncových (přijímajících) stavů. Trojice $(q, w, x) \in Q \times \mathcal{A}^* \times G^*$ značí konfiguraci zásobníkového automatu. V tomto textu budeme psát vrchol zásobníku na levé straně. Počáteční konfigurace automatu je trojice (q_0, w, Z_0) pro vstupní řetězec $w \in \mathcal{A}^*$.

Relace $\vdash_M \subset (Q \times \mathcal{A}^* \times G^*) \times (Q \times \mathcal{A}^* \times G^*)$ je *přechod* zásobníkového automatu M. Platí, že $(q, aw, \alpha\beta) \vdash_M (p, w, \gamma\beta)$, pokud $(p, \gamma) \in \delta(q, a, \alpha)$. Tranzitivní uzávěr, tranzitivní a reflexivní uzávěr a k-tá mocnina relace $\vdash_M$ se značí $\vdash_M^+$, $\vdash_M^*$, resp. $\vdash_M^k$.

Nechť $\delta(q, a, \alpha)$ obsahuje dvojici (p, β), kde $p, q \in Q$, $a \in \mathcal{A} \cup \{\varepsilon\}$, $\alpha \in G$, $\beta \in G^*$. Potom zápis $\alpha \mapsto \beta$ se používá pro operaci odebrání α z vrcholu zásobníku a následné přidání β na vrchol zásobníku. Tato operace se nazývá *zásobníková operace*.

Zásobníkový automat M je *deterministický* zásobníkový automat (deterministický PDA), pokud platí:

1. $|\delta(q, a, Z)| \leq 1$ pro všechna $q \in Q$, $a \in \mathcal{A}$, $Z \in G$ a $\delta(q, \varepsilon, Z) = \emptyset$, nebo
2. $\delta(q, a, Z) = \emptyset$ pro všechna $a \in \mathcal{A}$ a $|\delta(q, \varepsilon, Z)| \leq 1$.

Zásobníkový automat je tzv. *vstupem řízený*, pokud každá z jeho zásobníkových operací je daná pouze příslušným vstupním symbolem.

Jazyk L přijímaný zásobníkovým automatem M je definován dvěma odlišnými způsoby:

1. *Přijímání koncovým stavem:*

$$L(M) = \{x : (q_0, x, Z_0) \vdash_M^* (q, \varepsilon, \gamma) \wedge x \in \mathcal{A}^* \wedge \gamma \in G^* \wedge q \in F\}.$$

2. *Přijímání prázdným zásobníkem:*

$$L_\varepsilon(M) = \{x : (q_0, x, Z_0) \vdash_M^* (q, \varepsilon, \varepsilon) \wedge x \in \mathcal{A}^* \wedge q \in Q\}.$$

Pokud zásobníkový automat přijímá jazyk prázdným zásobníkem, pak množina koncových stavů F je prázdná množina.

Kapitola 2

Problém přijímání stromů s neseřazenými následníky

V této kapitole v podstatě shrneme výsledky dosažené v bakalářské práci [6]. Jako výpočetní model byl v [6] použit zásobníkový automat. V zájmu čitelnosti modelu a intuitivního použití zásobníku zde přistoupíme k problému trochu jinak; rozšíříme definici zásobníkového automatu a přesuneme část původní funkce zásobníku na specializované struktury nového automatu. Ukážeme algoritmus pro vytvoření takového ne nutně deterministického automatu a rovněž algoritmus pro determinizaci takto vzniklého automatu.

2.1 Atributovaný zásobníkový automat

Definice 2.1. *Uspořádaná posloupnost čísel* je taková posloupnost celých čísel $a_1, a_2, \ldots, a_n$, kde $n \geq 0$ a pro všechna $i \in \{1, 2, \ldots, n\}$ platí $a_i \leq a_{i+1}$. Prázdnou posloupnost, tj. posloupnost, kde $n = 0$, budeme značit stejně jako prázdnou množinu, tedy $\emptyset$. Pro uspořádanou posloupnost čísel s, symbol s^m, $m \geq 0$, značí m-prvkový vektor, jehož každý prvek je roven posloupnosti s. Pro vektor uspořádaných posloupností čísel s, symbol s^m, $m \geq 0$, značí $m|s|$-prvkový vektor, jehož prvky jsou tvořeny m opakováními prvků vektoru s.

Definice 2.2. Množina $seq(X)$, kde X je množina celých čísel, je množina všech uspořádaných posloupností obsahujících pouze čísla z X:

$$seq(X) = \{y : y \text{ je uspořádaná posloupnost čísel}, \\ y = (a_1, a_2, \ldots, a_n), a_i \in X, i = 1, 2, \ldots, n, n \geq 0\}.$$

Algoritmus 2.1 Insert

Vstup: Uspořádaná posloupnost čísel *sequence*, celé číslo n.
Výstup: Do posloupnosti *sequence* je zařazeno číslo n.

```
1: prev ← null
2: curr ← sequence
3: while curr ≠ null and curr.val < n do
4:     prev ← curr
5:     curr ← curr.next
6: end while
7: elem ← New-Element
8: elem.val ← n
9: elem.next ← curr
10: if prev ≠ null then
11:     prev.next ← elem
12: else
13:     sequence ← elem
14: end if
```

Definice 2.3. Definujme *nedeterministický atributovaný zásobníkový automat* takto:

$$M = (Q, \mathcal{A}, G, N, \delta, \delta_T, q_0, Z_0, s_0, F),$$

kde Q je konečná množina *stavů*, $\mathcal{A}$ je *vstupní abeceda*, G je *zásobníková abeceda*, N je konečná množina celých čísel (každý stav $q \in Q$ má atribut, který v sobě udržuje uspořádanou posloupnost čísel $s_q \in seq(N)$), δ je zobrazení z množiny $Q \times (\mathcal{A} \cup \{\varepsilon\}) \times G$ do množiny konečných podmnožin množiny $Q \times G^*$, δ_T je zobrazení z množiny $Q \times seq(N) \times G$ do množiny konečných podmnožin množiny $Q \times N \times G^*$, $q_0 \in Q$ je počáteční stav, $Z_0 \in G$ je počáteční symbol v zásobníku, $s_0 \in seq(N)^{|Q|}$ je počáteční nastavení atributů stavů a $F \subseteq Q$ je množina koncových (přijímajících) stavů. Čtveřice $(q, w, x, s) \in Q \times \mathcal{A}^* \times G^* \times seq(N)^{|Q|}$ značí konfiguraci atributovaného zásobníkového automatu. Počáteční konfigurace automatu je čtveřice (q_0, w, Z_0, s_0) pro vstupní řetězec $w \in \mathcal{A}^*$.

Definice nedeterministického atributovaného zásobníkového automatu se od standardního nedeterministického zásobníkového automatu v zásadě liší přidáním atributů $s_q \in seq(N)$ a také přidáním zobrazení δ_T, představujícího zvláštní ε-přechody, jejichž provedení je závislé na hodnotě zmiňovaného atributu aktuálního stavu a které modifikují tento atribut počátečního a koncového stavu přechodu.

Příklad 2.1. Jako příklad předchozí definice nechť poslouží nedeterministický atributovaný zásobníkový automat $M = (\{0,1,2,3,4\}, \mathcal{A}, \{S,]\}, \{1,2\}, \delta, \delta_T, 0, S, \emptyset^5, \emptyset)$, kde zobrazení δ a δ_T jsou definována takto:

$$\begin{aligned} \delta(0, a2, S) &= (2, SS]) & \qquad \delta_T(2, (1,2),]) &= (1, 1, \varepsilon) \\ \delta(2, a0, S) &= (3,]) & \qquad \delta_T(3, \emptyset,]) &= (2, 1, \varepsilon) \\ \delta(2, a0, S) &= (4,]) & \qquad \delta_T(4, \emptyset,]) &= (2, 2, \varepsilon) \end{aligned}$$

Definice 2.4. Relace $\vdash_M \subset (Q \times \mathcal{A}^* \times G^* \times seq(N)^{|Q|}) \times (Q \times \mathcal{A}^* \times G^* \times seq(N)^{|Q|})$ je *přechod* atributovaného zásobníkového automatu M. Platí, že $(q, aw, \alpha\beta, s) \vdash_M (p, w, \gamma\beta, s)$, pokud $(p, \gamma) \in \delta(q, a, \alpha)$ a zároveň $\delta_T(q, s_q, \nu) = \emptyset$, kde ν je prefix $\alpha\beta$. Dále platí, že $(q, w, \alpha\beta, s) \vdash_M (p, w, \gamma\beta, s')$, kde s' vznikne z s zařazením celého čísla n do s_p a vyprázdněním s_q, pokud $(p, n, \gamma) \in \delta_T(q, s_q, \alpha)$. Tranzitivní uzávěr, tranzitivní a reflexivní uzávěr a k-tá mocnina relace $\vdash_M$ se značí $\vdash_M^+$, $\vdash_M^*$, resp. $\vdash_M^k$.

Příklad 2.2. Pro nedeterministický atributovaný zásobníkový automat M z příkladu 2.1 a pro vstupní řetězec $w = a2\ a0\ a0$ platí následující:

$$\begin{aligned} &(0, a2\ a0\ a0, S, \emptyset^5) \vdash_M (2, a0\ a0, SS], \emptyset^5) \vdash_M (3, a0,]S], \emptyset^5) \\ &\vdash_M (2, a0, S], (\emptyset, \emptyset, (1), \emptyset, \emptyset)) \vdash_M (4, \varepsilon,]], (\emptyset, \emptyset, (1), \emptyset, \emptyset)) \\ &\vdash_M (2, \varepsilon,], (\emptyset, \emptyset, (1,2), \emptyset, \emptyset)) \vdash_M (1, \varepsilon, \varepsilon, (\emptyset, (1), \emptyset, \emptyset, \emptyset)) \end{aligned}$$

Definice 2.5. Nechť $\delta_T(q, s, \alpha)$ obsahuje trojici (p, n, β), kde $q, p \in Q$, $s \in seq(N)$, $\alpha \in G$, $n \in N$, $\beta \in G^*$. Potom zápis $AT(s, n, \alpha, \beta)$ (z anglického „accept tree“) se používá pro složenou operaci, kterou lze rozepsat i tímto způsobem:

$$AT(s, n, \alpha, \beta) = (\varepsilon, s_q = s | \alpha \mapsto \beta, s_p := \textsc{Insert}(s_p, n), s_q := \emptyset).$$

Předchozí zápis ilustruje, k jakým dílčím operacím během přechodu dochází, resp. nedochází;

1. nedochází ke čtení symbolu vstupního řetězce,
2. musí být splněna podmínka $s_q = s$,
3. dochází k zásobníkové operaci $\alpha \mapsto \beta$,
4. do uspořádané posloupnosti čísel uložené v atributu s_p se zařazuje celé číslo n (v případě implementace posloupností pomocí spojových seznamů lze použít výše uvedený algoritmus 2.1),
5. vyprazdňuje se hodnota atributu s_q počátečního stavu.

Dále v tomto textu budeme používat zkrácený zápis $AT(s, n)$ v případě, že platí $\alpha =] \wedge \beta = \varepsilon$.

Definice 2.6. Atributovaný zásobníkový automat M je *deterministický* atributovaný zásobníkový automat, pokud platí:

1. a) $|\delta(q, a, Z)| \leq 1$ pro všechna $q \in Q$, $a \in \mathcal{A}$, $Z \in G$ a $\delta(q, \varepsilon, Z) = \emptyset$, nebo
   b) $\delta(q, a, Z) = \emptyset$ pro všechna $a \in \mathcal{A}$ a $|\delta(q, \varepsilon, Z)| \leq 1$.
2. $|\delta_T(q, s_q, Z)| \leq 1$ pro všechna $q \in Q$, $s_q \in seq(N)$ a $Z \in G$.

Jazyk L přijímaný atributovaným zásobníkovým automatem M je definován obdobně jako v případě standardního zásobníkového automatu dvěma způsoby:

1. *Přijímání koncovým stavem:*

$$L(M) = \{x : (q_0, x, Z_0, s_0) \vdash_M^* (q, \varepsilon, \gamma, s) \\ \wedge\, x \in \mathcal{A}^* \wedge \gamma \in G^* \wedge s \ \in seq(N)^{|Q|} \wedge q \in F\}.$$

2. *Přijímání prázdným zásobníkem:*

$$L_\varepsilon(M) = \{x : (q_0, x, Z_0, s_0) \vdash_M^* (q, \varepsilon, \varepsilon, s) \\ \wedge\, x \in \mathcal{A}^* \wedge s \ \in seq(N)^{|Q|} \wedge q \in Q\}.$$

Pokud atributovaný zásobníkový automat přijímá jazyk prázdným zásobníkem, pak množina koncových stavů F je prázdná množina.

2.2 Konstrukce automatu

Automaty konstruované algoritmem 2.2 do značné míry simulují procházení stromů do hloubky, což naznačuje i na první pohled evidentní podobnost topologie jejich přechodových diagramů s odpovídajícími stromy.

Každému uzlu stromu, jehož prefixový zápis je zadán na vstupu algoritmu, odpovídá jeden stav výsledného automatu, dalšími stavy automatu jsou počáteční stav a stav, ve kterém automat přijímá strom. V případě posledně jmenovaného stavu se formálně nejedná o koncový stav (automat přijímá strom prázdným zásobníkem) a tento stav by bylo možné sloučit s počátečním stavem. Automat pro vyhledávání popsaný v kapitole 3 ovšem explicitní koncový stav vyžaduje a snahou je v této kapitole ukázat znovu použitelný algoritmus.

Algoritmus 2.2 Konstrukce atributovaného zásobníkového automatu přijímajícího strom t v prefixovém zápisu $pref(t)$ s libovolným seřazením následníků.

Vstup: Strom t nad ohodnocenou abecedou $\mathcal{A}$; prefixový zápis $pref(t) = a_1 a_2 \dots a_n$, $n \geq 1$.

Výstup: Atributovaný zásobníkový automat $M(t) = (\{0, 1, 2, \dots, n+1\}, \mathcal{A}, \{S,]\}, N, \delta, \delta_T, 0, S, \emptyset^{|Q|}, \emptyset)$.

1: $maxarity \leftarrow 1$
2: $q \leftarrow 0$ ▷ Stav odpovídající kořenu prvního nezpracovaného podstromu
3: **for** $i \leftarrow 1$ **to** n **do**
4: $a \in \mathcal{A}$
5: ▷ Vytvoření dvojice přechodů
6: $p \leftarrow i + 1$
7: Vytvořit přechod $\delta(q, a_i, S) = (p, S^{arity(a_i)}])$, kde $S^0 = \varepsilon$.
8: **if** $q = 0$ **then**
9: $r \leftarrow 1$
10: **else**
11: $r \leftarrow q$
12: **end if**
13: Vytvořit přechod $\delta_T(p, (1, 2, \dots, arity(a_i)),]) = (r, |\delta(q, a, S)|, \varepsilon)$.
14: $q \leftarrow p$
15: ▷ Aktualizace proměnné $maxarity$
16: **if** $arity(a_i) > maxarity$ **then**
17: $maxarity \leftarrow arity(a_i)$
18: **end if**
19: ▷ Nastavení proměnné q postupnou kontrolou počtu zpracovaných podstromů
20: $b \leftarrow a_i$
21: **while** $q \neq 0$ **and** $|\delta(q, a, S)| = arity(b)$ **do**
22: $|\{r : \delta(r, b, S) = (q, S^{arity(b)})\}| = 1$
23: $q \leftarrow r$
24: **if** $q \neq 0$ **then**
25: $|\{c : \delta(u, c, S) = (q, S^{arity(c)})\}| = 1$, $u \in Q$
26: $b \leftarrow c$
27: **end if**
28: **end while**
29: **end for**
30: $N \leftarrow \{1, 2, \dots, maxarity\}$

Při přečtení prvního symbolu vstupního řetězce, tj. symbolu kořene stromu, dochází k přechodu z počátečního stavu automatu do stavu odpovídajícího kořeni přijímaného stromu. Přechody z množiny δ směřují ve smyslu procházení stromem do hloubky vždy o úroveň níž, zatímco přechody z množiny δ_T směřují vždy o jednu úroveň výš a představují příjem určitého podstromu.

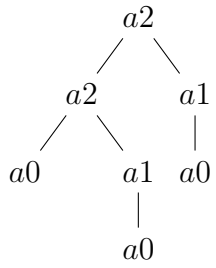
Automat čte symboly vstupního řetězce, přičemž v případě přečtení symbolu nulové arity dochází bez dalšího čtení symbolů k postupnému návratu až do stavu, který odpovídá kořeni dosud ne zcela zpracovaného podstromu. (Na podobném principu ostatně pracuje i samotný algoritmus 2.2.) Je tedy nutné udržovat v každém „vnitřním“ stavu automatu informaci o tom, které podstromy, jejichž kořeny jsou následníci daného uzlu, již byly zpracovány. K tomu slouží atributy stavů udržující uspořádané posloupnosti čísel. V rámci každého uzlu jsou následníkům přiřazeny jedinečné celočíselné identifikátory, které jsou při zpracování odpovídajících podstromů zařazovány do uspořádané posloupnosti udržované v daném uzlu. Přechod do stavu odpovídajícího předchůdci je potom podmíněn hodnotou atributu aktuálního stavu, kde je očekávána uspořádaná posloupnost identifikátorů všech následníků odpovídajícího uzlu.

Příjem celého stromu se vyznačuje atributem stavu, ve kterém automat přijímá strom, nastaveným na jednoprvkovou uspořádanou posloupnost (1), kde tento jediný prvek lze chápat jako identifikátor stromu, jehož prefixový zápis byl přečten.

Příklad 2.3. Uvažujme strom t_1, který je na obrázku 2.1, s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$. Atributovaný zásobníkový automat přijímající strom t_1 s libovolným seřazením následníků, který byl zkonstruován algoritmem 2.2, je deterministický atributovaný zásobníkový automat $M(t_1) = (\{0,1,2,3,4,5,6,7,8\}, \mathcal{A}, \{S,]\}, \{1,2\}, \delta, \delta_T, 0, S, \emptyset^9, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

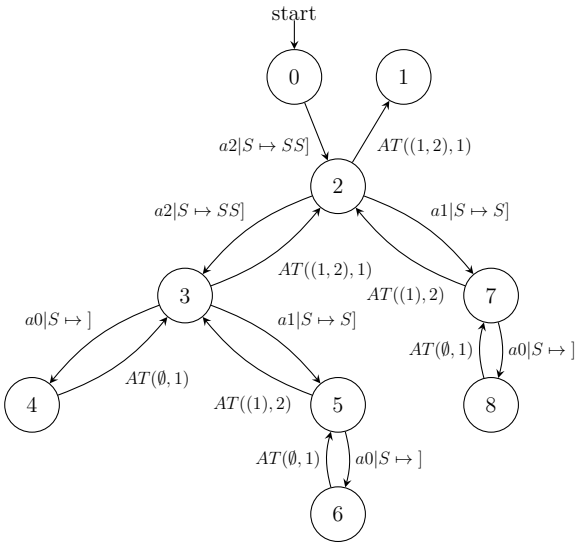
$$\begin{aligned}
\delta(0, a2, S) &= (2, SS]) & \qquad \delta_T(2, (1,2),]) &= (1, 1, \varepsilon)\\
\delta(2, a2, S) &= (3, SS]) & \qquad \delta_T(3, (1,2),]) &= (2, 1, \varepsilon)\\
\delta(2, a1, S) &= (7, S]) & \qquad \delta_T(4, \emptyset,]) &= (3, 1, \varepsilon)\\
\delta(3, a1, S) &= (5, S]) & \qquad \delta_T(5, (1),]) &= (3, 2, \varepsilon)\\
\delta(3, a0, S) &= (4,]) & \qquad \delta_T(6, \emptyset,]) &= (5, 1, \varepsilon)\\
\delta(5, a0, S) &= (6,]) & \qquad \delta_T(7, (1),]) &= (2, 2, \varepsilon)\\
\delta(7, a0, S) &= (8,]) & \qquad \delta_T(8, \emptyset,]) &= (7, 1, \varepsilon)
\end{aligned}$$

Přechodový diagram deterministického atributovaného zásobníkového automatu $M(t_1)$ je na obrázku 2.2.



$$pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$$

Obrázek 2.1: Strom t_1 a odpovídající prefixový zápis



Obrázek 2.2: Přechodový diagram deterministického atributovaného zásobníkového automatu $M(t_1)$ přijímajícího strom t_1 v prefixovém zápisu $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 2.3

Tabulka 2.1 znázorňuje posloupnost přechodů provedených deterministickým atributovaným zásobníkovým automatem $M(t_1)$ pro první a čtvrté uspořádání stromu t_1 na obrázku 0.1 v prefixovém zápisu.

stav	vstupní řetězec	zásobník	změny atributů
0	$a2\ a2\ a0\ a1\ a0\ a1\ a0$	S	
2	$a2\ a0\ a1\ a0\ a1\ a0$	$S\ S\ ]$	
3	$a0\ a1\ a0\ a1\ a0$	$S\ S\ ]\ S\ ]$	
4	$a1\ a0\ a1\ a0$	$]\ S\ ]\ S\ ]$	
3	$a1\ a0\ a1\ a0$	$S\ ]\ S\ ]$	$s_3 \leftarrow (1)$, $s_4 \leftarrow \emptyset$
5	$a0\ a1\ a0$	$S\ ]\ ]\ S\ ]$	
6	$a1\ a0$	$]\ ]\ ]\ S\ ]$	
5	$a1\ a0$	$]\ ]\ S\ ]$	$s_5 \leftarrow (1)$, $s_6 \leftarrow \emptyset$
3	$a1\ a0$	$]\ S\ ]$	$s_3 \leftarrow (1,2)$, $s_5 \leftarrow \emptyset$
2	$a1\ a0$	$S\ ]$	$s_2 \leftarrow (1)$, $s_3 \leftarrow \emptyset$
7	$a0$	$S\ ]\ ]$	
8	ε	$]\ ]\ ]$	
7	ε	$]\ ]$	$s_7 \leftarrow (1)$, $s_8 \leftarrow \emptyset$
2	ε	$]$	$s_2 \leftarrow (1,2)$, $s_7 \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (1)$, $s_2 \leftarrow \emptyset$

stav	vstupní řetězec	zásobník	změny atributů
0	$a2\ a1\ a0\ a2\ a1\ a0\ a0$	S	
2	$a1\ a0\ a2\ a1\ a0\ a0$	$S\ S\ ]$	
7	$a0\ a2\ a1\ a0\ a0$	$S\ ]\ S\ ]$	
8	$a2\ a1\ a0\ a0$	$]\ ]\ S\ ]$	
7	$a2\ a1\ a0\ a0$	$]\ S\ ]$	$s_7 \leftarrow (1)$, $s_8 \leftarrow \emptyset$
2	$a2\ a1\ a0\ a0$	$S\ ]$	$s_2 \leftarrow (2)$, $s_7 \leftarrow \emptyset$
3	$a1\ a0\ a0$	$S\ S\ ]\ ]$	
5	$a0\ a0$	$S\ ]\ S\ ]\ ]$	
6	$a0$	$]\ ]\ S\ ]\ ]$	
5	$a0$	$]\ S\ ]\ ]$	$s_5 \leftarrow (1)$, $s_6 \leftarrow \emptyset$
3	$a0$	$S\ ]\ ]$	$s_3 \leftarrow (2)$, $s_5 \leftarrow \emptyset$
4	ε	$]\ ]\ ]$	
3	ε	$]\ ]$	$s_3 \leftarrow (1,2)$, $s_4 \leftarrow \emptyset$
2	ε	$]$	$s_2 \leftarrow (1,2)$, $s_3 \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (1)$, $s_2 \leftarrow \emptyset$

Tabulka 2.1: Posloupnost přechodů deterministického atributovaného zásobníkového automatu $M(t_1)$ z příkladu 2.3 pro první a čtvrté uspořádání stromu t_1 na obrázku 0.1 v prefixovém zápisu

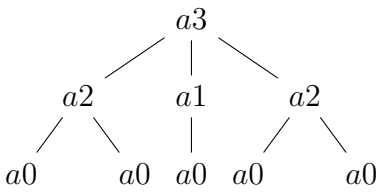
V [6] byla zmíněna také určitá optimalizace vytvářených automatů spočívající ve vynechání stavů odpovídajících listům stromu. Tyto stavy a přechody z/do nich bylo možné nahradit jediným přechodem – pouhou smyčkou ve stavu odpovídajícím předchůdci daného listu. Podobnou optimalizaci ovšem zde zavedený formalismus neumožňuje, protože zmíněný přechod by musel přečíst symbol vstupního řetězce a zároveň modifikovat atribut stavu. Absenci této optimalizace nepovažujeme za zásadní problém, protože ponecháváme formalismu a algoritmům jednoduchost a automatům názornost a topologickou podobnost s odpodvídajícími stromy.

Příklad 2.4. Uvažujme strom t_2, který je na obrázku 2.3, s prefixovým zápisem $pref(t_2) = a3\ a2\ a0\ a0\ a1\ a0\ a2\ a0\ a0$. Atributovaný zásobníkový automat přijímající strom t_2 s libovolným seřazením následníků, který byl zkonstruován algoritmem 2.2, je nedeterministický atributovaný zásobníkový automat $M(t_2) = (\{0,1,2,3,4,5,6,7,8,9,10\}, \mathcal{A}, \{S,]\}, \{1,2,3\}, \delta, \delta_T, 0, S, \emptyset^{11}, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$
\begin{aligned}
\delta(0, a3, S) &= (2, SSS]) & \qquad \delta_T(2, (1,2,3),]) &= (1, 1, \varepsilon)\\
\delta(2, a2, S) &= (3, SS]) & \qquad \delta_T(3, (1,2),]) &= (2, 1, \varepsilon)\\
\delta(2, a2, S) &= (8, SS]) & \qquad \delta_T(4, \emptyset,]) &= (3, 1, \varepsilon)\\
\delta(2, a1, S) &= (6, S]) & \qquad \delta_T(5, \emptyset,]) &= (3, 2, \varepsilon)\\
\delta(3, a0, S) &= (4,]) & \qquad \delta_T(6, (1),]) &= (2, 2, \varepsilon)\\
\delta(3, a0, S) &= (5,]) & \qquad \delta_T(7, \emptyset,]) &= (6, 1, \varepsilon)\\
\delta(6, a0, S) &= (7,]) & \qquad \delta_T(8, (1,2),]) &= (2, 3, \varepsilon)\\
\delta(8, a0, S) &= (9,]) & \qquad \delta_T(9, \emptyset,]) &= (8, 1, \varepsilon)\\
\delta(8, a0, S) &= (10,]) & \qquad \delta_T(10, \emptyset,]) &= (8, 2, \varepsilon)
\end{aligned}
$$

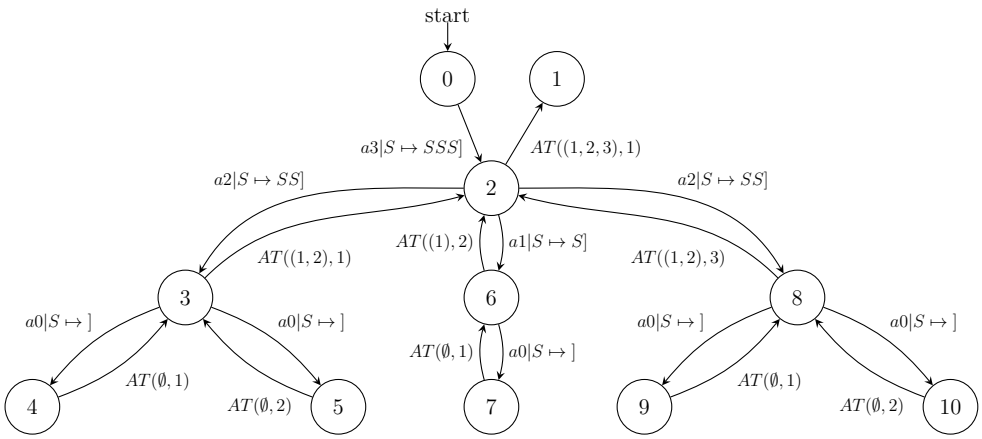
Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M(t_2)$ je na obrázku 2.4.

I když automat z příkladu 2.3, jehož přechodový diagram je na obrázku 2.2, je deterministický, obecně algoritmus 2.2 nevytváří deterministické automaty. Nedeterminismus vzniká v případě, kdy alespoň dva následníci určitého uzlu jsou označeni stejným symbolem, nedochází potom ke splnění první z podmínek uvedených v definici deterministického atributovaného zásobníkového automatu. Automat z příkladu 2.4, jehož přechodový diagram je na obrázku 2.4, není deterministický, tuto skutečnost způsobují přechody ze stavů 2, 3 a 8.



$$pref(t_2) = a3\ a2\ a0\ a0\ a1\ a0\ a2\ a0\ a0$$

Obrázek 2.3: Strom t_2 a odpovídající prefixový zápis



Obrázek 2.4: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M(t_2)$ přijímajícího strom t_2 v prefixovém zápisu $pref(t_2) = a3\ a2\ a0\ a0\ a1\ a0\ a2\ a0\ a0$ s libovolným seřazením následníků z příkladu 2.4

2.3 Determinizace automatu

Obecný algoritmus pro transformaci nedeterministického PDA na deterministický PDA neexistuje. Existují ovšem tři třídy nedeterministických PDA, kde pro každou tuto třídu je znám algoritmus pro transformaci na ekvivalentní deterministický PDA. Jednou z těchto tříd je třída vstupem řízených zásobníkových automatů. Algoritmus transformace vstupem řízeného nedeterminictického PDA pracuje na stejném principu jako transformace nedeterministického konečného automatu na ekvivalentní deterministický konečný automat, kde stavy výsledného DFA jsou d-podmnožiny stavů původního NFA [5]. Zde ovšem nepracujeme s žádnou známou třídou zásob-

níkových automatů, pracujeme s vlastní definicí deterministického a nedeterministického automatu, a je tedy rovněž potřeba definovat vlastní transformační algoritmus.

Algoritmus 2.3.1 Replace (část 1/2)

Vstup: Uspořádaná posloupnost čísel *sequence*, celé číslo n, celé číslo m.
Výstup: Z posloupnosti *sequence* jsou vyřazeny všechny výskyty čísla n a číslo m je zařazeno tolikrát, kolik bylo výskytů čísla n.

```
1:  ▷ Inicializace
2:  found ← false
3:  ins ← null
4:  prev ← null
5:  curr ← sequence
6:  ▷ Hledání prvního výskytu n
7:  while curr ≠ null and curr.val < n do
8:      ▷ Současné hledání pozice pro zařazení m
9:      if not found and curr.val ≥ m then
10:         found ← true
11:         ins ← prev
12:     end if
13:     prev ← curr
14:     curr ← curr.next
15: end while
16: ▷ Zjištění počtu výskytů n
17: count ← 0
18: while curr ≠ null and curr.val = n do
19:     count ← count + 1
20:     curr ← curr.next
21: end while
22: ▷ Vyřazení výskytů n
23: if prev ≠ null then
24:     prev.next ← curr
25: else
26:     sequence ← curr
27: end if
```

Algoritmus 2.4.1–2.4.4 nevytváří nový automat na základě automatu na vstupu, ale tento vstupní nedeterministický automat modifikuje. Algoritmus pracuje ve dvou fázích. První fáze slučuje stavy, do kterých vedou přechody způsobující nedeterminismus, výsledné stavy jsou pro názornost v diagramech značeny stejným způsobem jako d-podmnožiny vznikající při

Algoritmus 2.3.2 Replace (část 2/2)

```
28: if found then
29:     if ins ≠ null then
30:         curr ← ins.next
31:     else
32:         curr ← sequence
33:     end if
34: else
35:     ▷ Případné dokončení hledání pozice pro zařazení m
36:     ins ← prev
37:     while curr ≠ null and curr.val < m do
38:         ins ← curr
39:         curr ← curr.next
40:     end while
41: end if
42: ▷ Zařazení m
43: for i ← 1 to count do
44:     elem ← New-Element
45:     elem.val ← m
46:     elem.next ← curr
47:     curr ← elem
48: end for
49: if ins ≠ null then
50:     ins.next ← curr
51: else
52:     sequence ← curr
53: end if
```

determinizaci konečných automatů. Pohledem na přechodový diagram automatu napodobující přijímaný strom se tato fáze algoritmu jeví jako pracující shora dolů. Ve druhé fázi dochází ke sjednocování $AT(s, n)$ přechodů, které představují přijímání stejných podstromů (bez ohledu na uspořádání následníků) a které způsobují, že automat je nyní nedeterministický z důvodu nesplnění druhé z podmínek uvedených v definici deterministického atributovaného zásobníkového automatu. S přihlédnutím na podobnost přechodového diagramu automatu a odpovídajícího stromu se modifikace automatu ve druhé fázi naopak šíří zdola nahoru.

Algoritmus 2.4.1 Transformace nedeterministického atributovaného zásobníkového automatu na odpovídající deterministický atributovaný zásobníkový automat (část 1/4 – vstup, výstup).

Vstup: Nedeterministický atributovaný zásobníkový automat $M(t) = (Q, \mathcal{A}, G, N, \delta, \delta_T, 0, Z_0, \emptyset^{|Q|}, \emptyset)$, kde $Q = \{0, 1, 2, \ldots, n\}$, $G = \{S,]\}$ a $n \geq 2$. Automat musí splňovat následující podmínky:

1. Uspořádání stavů automatu vzhledem k jejich číselnému označení je takové, že:
   a) pokud $\delta(q, a, \alpha) = (p, \beta)$, pak platí $q < p$, kde $q, p \in Q$, $a \in \mathcal{A}$, $\alpha \in G$ a $\beta \in G^*$,
   b) pokud $\delta_T(q, s, \alpha) = (p, k, \beta)$, pak platí $q > p$, kde $q, p \in Q$, $s \in seq(N)$, $\alpha \in G$, $k \in N$ a $\beta \in G^*$.
2. Pro stav $q = 0$ neexistuje přechod, který lze zapsat $\delta_T(q, s, \alpha) = (p, k, \beta)$ nebo $\delta_T(p, s, \alpha) = (q, k, \beta)$, kde $p \in Q$, $s \in seq(N)$, $\alpha \in G$, $k \in N$ a $\beta \in G^*$.
3. Pro stav $q = 1$ neexistuje přechod, který lze zapsat $\delta(q, a, \alpha) = (p, \beta)$, $\delta(p, a, \alpha) = (q, \beta)$ nebo $\delta_T(q, s, \alpha) = (p, k, \beta)$, kde $p \in Q$, $a \in \mathcal{A}$, $\alpha \in G$, $\beta \in G^*$, $s \in seq(N)$ a $k \in N$.
4. Do každého stavu $q \in Q$, $q \neq 0$, $q \neq 1$, vede právě jeden přechod, který lze zapsat $\delta(p, a, S) = (q, S^{arity(a)}])$, kde $p \in Q$, $a \in \mathcal{A}$ a $S^0 = \varepsilon$.
5. Z každého stavu $q \in Q$, $q \neq 0$, $q \neq 1$, vede právě jeden přechod, který lze zapsat $\delta_T(q, s,]) = (p, k, \varepsilon)$, kde $p \in Q$, $s \in seq(N)$ a $k \in N$.
6. Neexistuje přechod, který lze zapsat $(q, \varepsilon, \alpha) = (p, \beta)$, kde $q, p \in Q$, $\alpha \in G$ a $\beta \in G^*$.

Výstup: Odpovídající deterministický atributovaný zásobníkový automat $M_d(t) = (Q', \mathcal{A}, G, N', \delta', \delta'_T, 0, Z_0, \emptyset^{|Q'|}, \emptyset)$.

Algoritmus 2.4.2 Transformace nedeterministického atributovaného zásobníkového automatu na odpovídající deterministický atributovaný zásobníkový automat (část 2/4 – první fáze, inicializace).

```
1: Q' ← Q, N' ← N, δ' ← δ, δ'_T ← δ_T
2: ▷ Pokud q < p, pak (q, a) má vyšší prioritu než (p, b), q, p ∈ Q', a, b ∈ A.
3: queue ← New-Priority-Queue
4: for all q ∈ Q' do
5:     for all a ∈ A do
6:         if |δ'(q, a, S)| > 1 then
7:             Enqueue(queue, (q, a))
8:         end if
9:     end for
10: end for
```

Příklad 2.5. Uvažujme nedeterministický atributovaný zásobníkový automat $M(t_2)$ z příkladu 2.4, jehož přechodový diagram je na obrázku 2.4. Odpovídající deterministický atributovaný zásobníkový automat, který byl zkonstruován algoritmem 2.4.1–2.4.4 z nedeterministického atributovaného zásobníkového automatu $M(t_2)$, je deterministický atributovaný zásobníkový automat $M_d(t_2) = (\{0,1,2,3,5,6,10\}, \mathcal{A}, \{S,]\}, \{1,2,3\}, \delta, \delta_T, 0, S, \emptyset^7, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

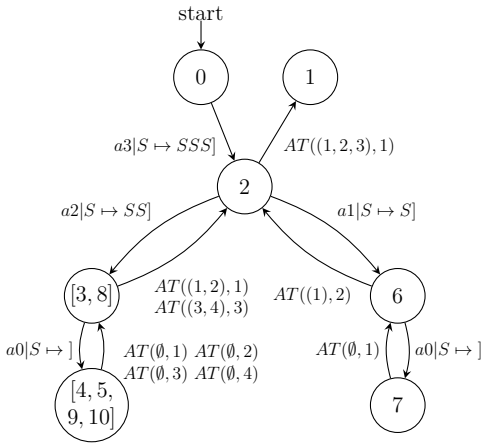
$$\begin{aligned}
\delta(0, a3, S) &= (2, SSS]) & \delta_T(2, (1,1,2),]) &= (1, 1, \varepsilon)\\
\delta(2, a2, S) &= (3, SS]) & \delta_T(3, (1,1),]) &= (2, 1, \varepsilon)\\
\delta(2, a1, S) &= (6, S]) & \delta_T(4, \emptyset,]) &= (3, 1, \varepsilon)\\
\delta(3, a0, S) &= (4,]) & \delta_T(6, (1),]) &= (2, 2, \varepsilon)\\
\delta(6, a0, S) &= (7,]) & \delta_T(7, \emptyset,]) &= (6, 1, \varepsilon)
\end{aligned}$$

Přechodový diagram deterministického atributovaného zásobníkového automatu $M_d(t_2)$ je na obrázku 2.6, přičemž přechodový diagram mezivýsledku algoritmu 2.4.1–2.4.4 po jeho první fázi a před jeho druhou fází je na obrázku 2.5. Označení stavů 3 a 4 je v těchto dvou diagramech následující: $3 = [3, 8]$ a $4 = [4, 5, 9, 10]$.

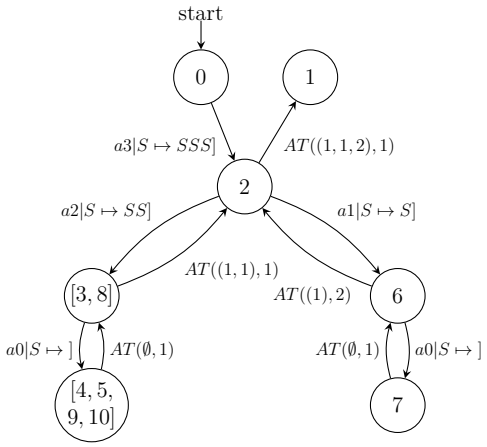
Tabulka 2.2 znázorňuje posloupnost přechodů provedených deterministickým atributovaným zásobníkovým automatem $M_d(t_2)$ pro uspořádání stromu t_2 v prefixovém zápisu $pref(t'_2) = a3\ a1\ a0\ a2\ a0\ a0\ a2\ a0\ a0$.

Algoritmus 2.4.3 Transformace nedeterministického atributovaného zásobníkového automatu na odpovídající deterministický atributovaný zásobníkový automat (část 3/4 – první fáze, pokračování).

11: **while not** EMPTY($queue$) **do**
12: $(q, a) \leftarrow$ DEQUEUE($queue$)
13: **if** $q \in Q'$ **then**
14: $R \leftarrow \{p : (p, \alpha) \in \delta'(q, a, S)\}$, $\alpha \in G^*$
15: $r \leftarrow$ MIN(R) ▷ Pro názornost značíme v diagramech tyto stavy jako d-podmnožiny obsahující stavy z R.
16: $i \leftarrow 0$
17: **for all** $p \in R$ **do**
18: $s \in seq(N)$, $k \in N$
19: Zrušit přechod $\delta'(q, a, S) = (p, S^{arity(a)})$, kde $S^0 = \varepsilon$.
20: Zrušit přechod $\delta'_T(p, s,]) = (q, k, \varepsilon)$.
21: $t \leftarrow (i \cdot arity(a) + 1, i \cdot arity(a) + 2, \ldots, (i + 1) \cdot arity(a))$
22: Vytvořit přechod $\delta'_T(r, t,]) = (q, k, \varepsilon)$, $(q, k, \varepsilon) \in \delta_T(p, s,])$.
23: $j \leftarrow i \cdot arity(a) + 1$
24: **for all** $(v, \beta) \in \delta(p, b, S)$, $v \in Q$, $\beta \in G^*$, $b \in \mathcal{A}$ **do**
25: Zrušit přechod $\delta'(p, b, S) = (v, \beta)$.
26: Vytvořit přechod $\delta'(r, b, S) = (v, \beta)$.
27: Zrušit přechod $\delta'_T(v, s,]) = (p, k, \varepsilon)$.
28: Vytvořit přechod $\delta'_T(v, s,]) = (r, j, \varepsilon)$, $(p, k, \varepsilon) \in \delta_T(v, s,])$.
29: $N' \leftarrow N' \cup \{j\}$
30: $j \leftarrow j + 1$
31: **end for**
32: $i \leftarrow i + 1$
33: **end for**
34: Vytvořit přechod $\delta'(q, a, S) = (r, S^{arity(a)}])$, kde $S^0 = \varepsilon$.
35: $Q' \leftarrow Q' \setminus (R \setminus \{r\})$
36: **for all** $b \in \mathcal{A}$ **do**
37: **if** $|\delta'(r, b, S)| > 1$ **then**
38: ENQUEUE($queue, (r, b)$)
39: **end if**
40: **end for**
41: **end if**
42: **end while**



Obrázek 2.5: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M'(t_2)$ přijímajícího strom t_2 v prefixovém zápisu $pref(t_2) = a3\ a2\ a0\ a0\ a1\ a0\ a2\ a0\ a0$ s libovolným seřazením následníků, který je mezivýsledkem algoritmu 2.4.1–2.4.4, z příkladu 2.5



Obrázek 2.6: Přechodový diagram deterministického atributovaného zásobníkového automatu $M_d(t_2)$ přijímajícího strom t_2 v prefixovém zápisu $pref(t_2) = a3\ a2\ a0\ a0\ a1\ a0\ a2\ a0\ a0$ s libovolným seřazením následníků z příkladu 2.5

Algoritmus 2.4.4 Transformace nedeterministického atributovaného zásobníkového automatu na odpovídající deterministický atributovaný zásobníkový automat (část 4/4 – druhá fáze).

43: **for all** $q \in Q'$ **in descending order do**
44: **for all** $s \in seq(N')$ **do**
45: **if** $|\delta'_T(q, s,])| > 1$ **then**
46: $|\{p : (p, k, \varepsilon) \in \delta'_T(q, s,])\}| = 1$, $k \in N'$
47: $K \leftarrow \{k : (p, k, \varepsilon) \in \delta'_T(q, s,])\}$
48: $k_{min} \leftarrow \textsc{Min}(K)$
49: $K \leftarrow K \setminus \{k_{min}\}$
50: **for all** $k \in K$ **do**
51: $m \geq 1$
52: Zrušit přechod $\delta'_T(q, s,]) = (p, k, \varepsilon)$.
53: $V \leftarrow \{v : v \in seq(N') \wedge v = (a_1, a_2, \ldots, a_m) \wedge (\exists i : a_i = k)\}$
54: $|\{t : t \in V \wedge \delta'_T(p, t,]) \neq \emptyset\}| = 1$
55: $|\{(r, l, \varepsilon) : (r, l, \varepsilon) \in \delta'_T(p, t,])\}| = 1$
56: Zrušit přechod $\delta'_T(p, t,]) = (r, l, \varepsilon)$.
57: $t \leftarrow \textsc{Replace}(t, k, k_{min})$ ▷ Ukázka implementace funkce Replace – algoritmus 2.3.1–2.3.2
58: Vytvořit přechod $\delta'_T(p, t,]) = (r, l, \varepsilon)$.
59: **end for**
60: **end if**
61: **end for**
62: **end for**

stav	vstupní řetězec	zásobník	změny atributů
0	$a3\ a1\ a0\ a2\ a0\ a0\ a2\ a0\ a0$	S	
2	$a1\ a0\ a2\ a0\ a0\ a2\ a0\ a0$	$S\ S\ S\ ]$	
6	$a0\ a2\ a0\ a0\ a2\ a0\ a0$	$S\ ]\ S\ S\ ]$	
7	$a2\ a0\ a0\ a2\ a0\ a0$	$]\ ]\ S\ S\ ]$	
6	$a2\ a0\ a0\ a2\ a0\ a0$	$]\ S\ S\ ]$	$s_6 \leftarrow (1),\ s_7 \leftarrow \emptyset$
2	$a2\ a0\ a0\ a2\ a0\ a0$	$S\ S\ ]$	$s_2 \leftarrow (2),\ s_6 \leftarrow \emptyset$
3	$a0\ a0\ a2\ a0\ a0$	$S\ S\ ]\ S\ ]$	
4	$a0\ a2\ a0\ a0$	$]\ S\ ]\ S\ ]$	
3	$a0\ a2\ a0\ a0$	$S\ ]\ S\ ]$	$s_3 \leftarrow (1),\ s_4 \leftarrow \emptyset$
4	$a2\ a0\ a0$	$]\ ]\ S\ ]$	
3	$a2\ a0\ a0$	$]\ S\ ]$	$s_3 \leftarrow (1,1),\ s_4 \leftarrow \emptyset$
2	$a2\ a0\ a0$	$S\ ]$	$s_2 \leftarrow (1,2),\ s_3 \leftarrow \emptyset$
3	$a0\ a0$	$S\ S\ ]\ ]$	
4	$a0$	$]\ S\ ]\ ]$	
3	$a0$	$S\ ]\ ]$	$s_3 \leftarrow (1),\ s_4 \leftarrow \emptyset$
4	ε	$]\ ]\ ]$	
3	ε	$]\ ]$	$s_3 \leftarrow (1,1),\ s_4 \leftarrow \emptyset$
2	ε	$]$	$s_2 \leftarrow (1,1,2),\ s_3 \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (1),\ s_2 \leftarrow \emptyset$

Tabulka 2.2: Posloupnost přechodů deterministického atributovaného zásobníkového automatu $M_d(t_2)$ z příkladu 2.5 pro uspořádání stromu t_2 v prefixovém zápisu $pref(t'_2) = a3\ a1\ a0\ a2\ a0\ a0\ a2\ a0\ a0$

Kapitola 3

Automat pro vyhledávání

V této kapitole ukážeme jednoduchý algoritmus, který na základě prefixového zápisu stromu vytvoří nedeterministický atributovaný zásobníkový automat pro vyhledávání tohoto stromu s libovolným seřazením následníků. Takový automat vyhledává daný strom jako podstrom jiného stromu v prefixovém zápisu. Princip konstrukce automatu vychází z konstrukce NFA pro vyhledávání řetězců. Uvedeme také definice potřebné pro popis deterministického automatu pro vyhledávání a algoritmus pro transformaci nedeterministického automatu pro vyhledávání na odpovídající deterministický automat.

3.1 Algoritmus konstrukce automatu

Algoritmus 3.1 se skládá ze tří kroků; nejprve se pomocí algoritmu 2.2 zkonstruuje nedeterministický atributovaný zásobníkový automat přijímající daný strom, poté se tento automat zdeterminizuje pomocí algoritmu 2.4.1–2.4.4 a nakonec se pro každý symbol vstupní abecedy přidá přechod ve tvaru smyčky v počátečním stavu 0. Tato vyhledávací smyčka způsobuje další nedeterminismus, odstranění tohoto typu nedeterminismu je popsáno dále v této kapitole. Automaty vytvářené algoritmem 3.1 přijímají koncovým stavem.

Důvodem pro aplikaci algoritmu 2.4.1–2.4.4 je potřeba nalezení stejných podstromů, jejichž kořeny jsou přímí následníci určitého uzlu (takové podstromy musí mít přiřazen stejný identifikátor pro zařazování do posloupnosti uchovávané v atributu stavu odpovídajícímu zmíněnému společnému uzlu) a to je přesně úloha, kterou tento algoritmus řeší a kterou tedy nemusíme zohledňovat v další fázi konstrukce deterministického automatu pro vyhledávání.

Algoritmus 3.1 Konstrukce nedeterministického atributovaného zásobníkového automatu pro vyhledávání stromu t s prefixovým zápisem $pref(t)$ s libovolným seřazením následníků.

Vstup: Strom t nad ohodnocenou abecedou $\mathcal{A}$; prefixový zápis $pref(t) = a_1 a_2 \dots a_n$, $n \geq 1$.

Výstup: Nedeterministický atributovaný zásobníkový automat $M_{ns}(t) = (Q, \mathcal{A}, \{S,]\}, N, \delta, \delta_T, 0, S, \emptyset^{|Q|}, F)$.

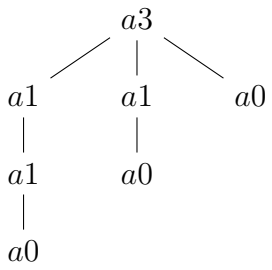
1: Vytvořit atributovaný zásobníkový automat $M(t)$ pomocí algoritmu 2.2.
2: Transformovat atributovaný zásobníkový automat $M(t)$ pomocí algoritmu 2.4.1–2.4.4 na odpovídající deterministický atributovaný zásobníkový automat a tento automat označit $M_{ns}(t)$.
3: **for all** $a \in \mathcal{A}$ **do**
4: Vytvořit přechod $\delta(0, a, S) = (0, S^{arity(a)}])$, kde $S^0 = \varepsilon$.
5: **end for**
6: $F \leftarrow \{1\}$

3.2 Příklady nedeterministických automatů

Příklad 3.1. Uvažujme strom t_4, který je na obrázku 3.1, s prefixovým zápisem $pref(t_4) = a3\ a1\ a1\ a0\ a1\ a0\ a0$. Atributovaný zásobníkový automat pro vyhledávání stromu t_4 s libovolným seřazením následníků, který byl zkonstruován algoritmem 3.1, je nedeterministický atributovaný zásobníkový automat $M_{ns}(t_4) = (\{0, 1, 2, 3, 4, 5, 7, 8\}, \mathcal{A}, \{S,]\}, \{1, 2, 3\}, \delta, \delta_T, 0, S, \emptyset^8, \{1\})$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

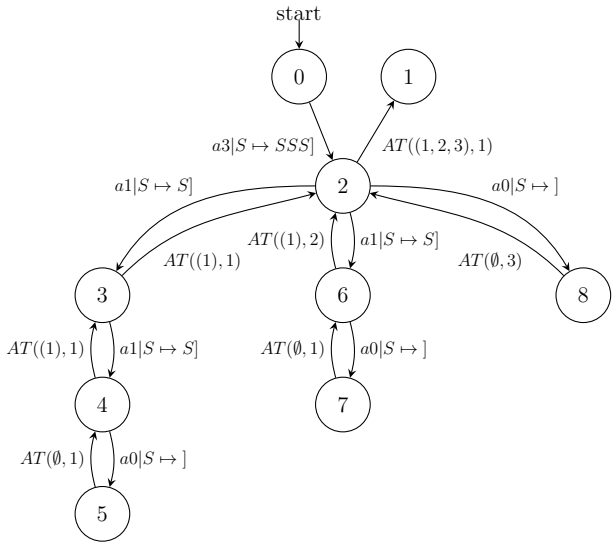
$$\begin{aligned}
\delta(0, a3, S) &= (0, SSS]) & \delta_T(2, (1, 2, 3),]) &= (1, 1, \varepsilon)\\
\delta(0, a3, S) &= (2, SSS]) & \delta_T(3, (1),]) &= (2, 1, \varepsilon)\\
\delta(0, a1, S) &= (0, S]) & \delta_T(3, (2),]) &= (2, 2, \varepsilon)\\
\delta(0, a0, S) &= (0,]) & \delta_T(4, (1),]) &= (3, 1, \varepsilon)\\
\delta(2, a1, S) &= (3, S]) & \delta_T(5, \emptyset,]) &= (4, 1, \varepsilon)\\
\delta(2, a0, S) &= (8,]) & \delta_T(7, \emptyset,]) &= (3, 2, \varepsilon)\\
\delta(3, a1, S) &= (4, S]) & \delta_T(8, \emptyset,]) &= (2, 3, \varepsilon)\\
\delta(3, a0, S) &= (7,])\\
\delta(4, a0, S) &= (5,])
\end{aligned}$$

Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ns}(t_4)$ je na obrázku 3.4, přičemž přechodový diagram nedeterministického atributovaného zásobníkového automatu $M(t_4)$, který je mezivýsledkem algoritmu 3.1 po vykonání jeho prvního kroku, je na obrázku 3.2 a



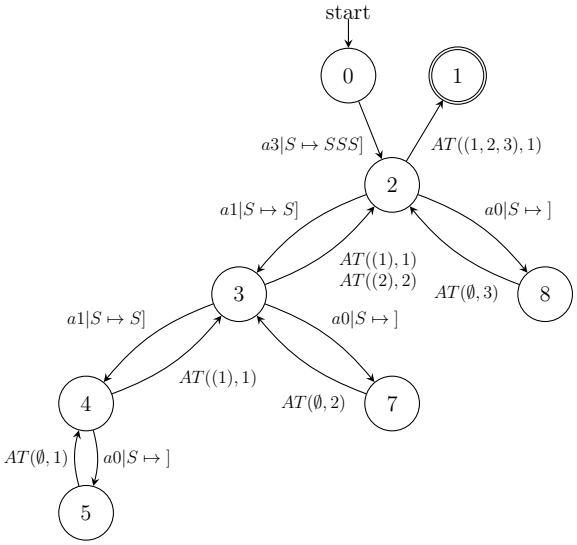
$$pref(t_4) = a3\ a1\ a1\ a0\ a1\ a0\ a0$$

Obrázek 3.1: Strom t_4 a odpovídající prefixový zápis

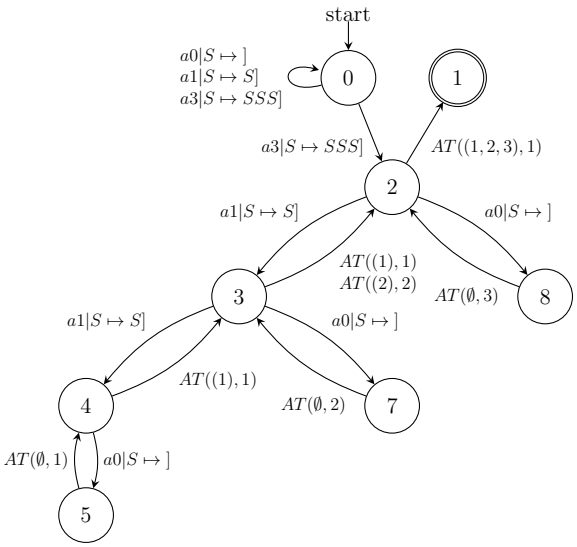


Obrázek 3.2: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M(t_4)$ přijímajícího strom t_4 v prefixovém zápisu $pref(t_4) = a3\ a1\ a1\ a0\ a1\ a0\ a0$ s libovolným seřazením následníků, který je mezivýsledkem algoritmu 3.1 po vykonání jeho prvního kroku, z příkladu 3.1

diagram deterministického atributovaného zásobníkového automatu $M_d(t_4)$, který je mezivýsledkem algoritmu 3.1 po vykonání jeho druhého kroku, je na obrázku 3.3.



Obrázek 3.3: Přechodový diagram deterministického atributovaného zásobníkového automatu $M_d(t_4)$ přijímajícího strom t_4 v prefixovém zápisu $pref(t_4) = a3\ a1\ a1\ a0\ a1\ a0\ a0$ s libovolným seřazením následníků, který je mezivýsledkem algoritmu 3.1 po vykonání jeho druhého kroku, z příkladu 3.1



Obrázek 3.4: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ns}(t_4)$ pro vyhledávání stromu t_4 s prefixovým zápisem $pref(t_4) = a3\ a1\ a1\ a0\ a1\ a0\ a0$ s libovolným seřazením následníků z příkladu 3.1

Příklad 3.2. Uvažujme strom t_1, který je na obrázku 2.1, s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$. Atributovaný zásobníkový automat pro vyhledávání stromu t_1 s libovolným seřazením následníků, který byl zkonstruován algoritmem 3.1, je nedeterministický atributovaný zásobníkový automat $M_{ns}(t_1) = (\{0,1,2,3,4,5,6,7,8\}, \mathcal{A}, \{S,]\}, \{1,2\}, \delta, \delta_T, 0, S, \emptyset^9, \{1\})$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

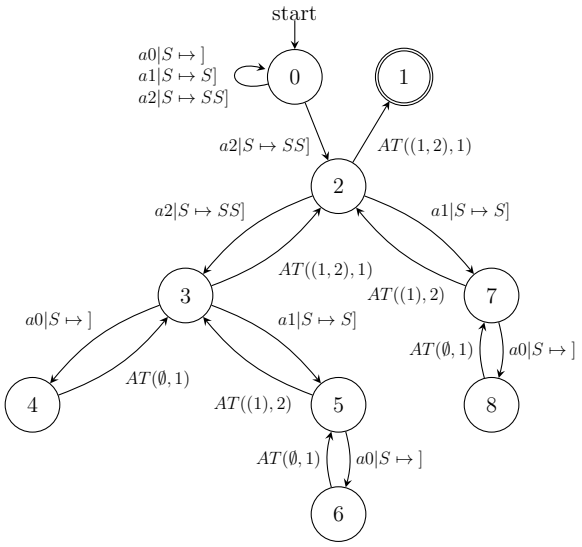
$$\begin{aligned}
\delta(0, a2, S) &= (0, SS]) & \delta_T(2, (1,2),]) &= (1, 1, \varepsilon)\\
\delta(0, a2, S) &= (2, SS]) & \delta_T(3, (1,2),]) &= (2, 1, \varepsilon)\\
\delta(0, a1, S) &= (0, S]) & \delta_T(4, \emptyset,]) &= (3, 1, \varepsilon)\\
\delta(0, a0, S) &= (0,]) & \delta_T(5, (1),]) &= (3, 2, \varepsilon)\\
\delta(2, a2, S) &= (3, SS]) & \delta_T(6, \emptyset,]) &= (5, 1, \varepsilon)\\
\delta(2, a1, S) &= (7, S]) & \delta_T(7, (1),]) &= (2, 2, \varepsilon)\\
\delta(3, a1, S) &= (5, S]) & \delta_T(8, \emptyset,]) &= (7, 1, \varepsilon)\\
\delta(3, a0, S) &= (4,])\\
\delta(5, a0, S) &= (6,])\\
\delta(7, a0, S) &= (8,])
\end{aligned}$$

Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ns}(t_1)$ je na obrázku 3.5.

3.3 Rozšířený atributovaný zásobníkový automat

Než přistoupíme k popisu algoritmu pro transformaci nedeterministického automatu pro vyhledávání na odpovídající deterministický automat, je potřeba zavést pojem rozšířeného atributovaného zásobníkového automatu. Důležitou vlastností automatů vytvářených algoritmem 3.1 je, že zásobníkové operace všech přechodů představovaných zobrazením δ jsou jednoznačně dané pouze příslušným vstupním symbolem a zásobníkové operace všech přechodů představovaných zobrazením δ_T jsou stejné (analogie se vstupem řízeným zásobníkovým automatem). Díky tomu není potřeba provádět větvení zásobníku, nadále postačí jediný jednoduchý zásobník.

Pokud X je množina, *potenční množina* množiny X, tj. množina všech podmnožin množiny X, se značí $\mathcal{P}(X)$.



Obrázek 3.5: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ns}(t_1)$ pro vyhledávání stromu t_1 s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 3.2

Definice 3.1. Definujme *nedeterministický rozšířený atributovaný zásobníkový automat* takto:

$$M = (Q, Q_s, \mathcal{A}, G, N, D, \delta, \delta_T, q_0, Z_0, s_0, F),$$

kde Q je konečná množina *stavů*, Q_s je konečná množina *stavů výchozího automatu*, $\mathcal{A}$ je *vstupní abeceda*, G je *zásobníková abeceda*, N je konečná množina celých čísel, $D = \{1, 2, \ldots, d\}$, $d \geq 1$, d je *počet paralelních vyhledávání* (každý stav výchozího automatu $q \in Q_s$ má atribut, který v sobě udržuje d uspořádaných posloupností čísel $s_q^{[i]} \in seq(N)$, $i \in D$), δ je zobrazení z množiny $Q \times (\mathcal{A} \cup \{\varepsilon\}) \times G$ do množiny konečných podmnožin množiny $Q \times G^* \times \bigcup_{e \in D} D^e$, δ_T je zobrazení z množiny $Q \times \mathcal{P}(Q_s \times D \times seq(N)) \times G$ do množiny konečných podmnožin množiny $Q \times \mathcal{P}(Q_s \times D \times N) \times G^* \times \bigcup_{e \in D} D^e$, $q_0 \in Q$ je počáteční stav, $Z_0 \in G$ je počáteční symbol v zásobníku, $s_0 \in seq(N)^{|Q_s|}$ je počáteční nastavení atributů stavů výchozího automatu a $F \subseteq Q$ je množina koncových (přijímajících) stavů. Čtveřice $(q, w, x, s) \in Q \times \mathcal{A}^* \times G^* \times seq(N)^{d|Q_s|}$ značí konfiguraci rozšířeného atributovaného zásobníkového automatu. Počáteční konfigurace automatu je čtveřice (q_0, w, Z_0, s_0^d) pro vstupní řetězec $w \in \mathcal{A}^*$.

Definice nedeterministického rozšířeného atributovaného zásobníkového automatu se od standardního nedeterministického atributovaného zásobníkového automatu liší v tom, že kromě množiny stavů Q je přítomna navíc množina stavů výchozího automatu Q_s, na které se vážou atributy udržující uspořádané posloupnosti čísel, a to v několika paralelních instancích pro každý stav $q \in Q_s$. Zobrazení δ a δ_T umožňují v rámci $i \in D$ permutovat množiny atributů $s_q^{[i]}$, $q \in Q_s$.

Definice 3.2. Relace $\vdash_M \subset (Q \times \mathcal{A}^* \times G^* \times seq(N)^{d|Q_s|}) \times (Q \times \mathcal{A}^* \times G^* \times seq(N)^{d|Q_s|})$ je *přechod* rozšířeného atributovaného zásobníkového automatu M. Platí, že $(q, aw, \alpha\beta, s) \vdash_M (p, w, \gamma\beta, s')$, kde $s_r'^{[i]} = s_r^{[k_i]}$, $i \in D$, $r \in Q_s$, pokud $(p, \gamma, (k_1, k_2, \ldots, k_e)) \in \delta(q, a, \alpha)$, kde $e \in D$, a zároveň $\delta_T(q, P, \nu) = \emptyset$, kde $P \subset \{(r, i, s_r^{[i]}) : r \in Q_s, i \in D\}$ a ν je prefix $\alpha\beta$. Dále platí, že $(q, w, \alpha\beta, s) \vdash_M (p, w, \gamma\beta, s')$, kde $s_r''^{[i]} = s_r^{[k_i]}$, $i \in D$, $r \in Q_s$ a s' vznikne z s'' zařazením celého čísla n do $s_t''^{[j]}$ pro každé $(t, j, n) \in R$ a vyprázdněním $s_r^{[i]}$ pro každé $(r, i, s_r^{[i]}) \in P$, pokud $(p, R, (k_1, k_2, \ldots, k_e), \gamma) \in \delta_T(q, P, \alpha)$, kde $e \in D$, a zároveň neexistuje P' takové, že $\delta_T(q, P') \neq \emptyset \wedge |P'| > |P|$, $P, P' \subset \{(r, i, s_r^{[i]}) : r \in Q_s, i \in D\}$. Tranzitivní uzávěr, tranzitivní a reflexivní uzávěr a k-tá mocnina relace $\vdash_M$ se značí $\vdash_M^+$,$\vdash_M^*$, resp. $\vdash_M^k$.

Definice 3.3. Rozšířený atributovaný zásobníkový automat M je *deterministický* rozšířený atributovaný zásobníkový automat, pokud platí:

1. a) $|\delta(q, a, Z)| \leq 1$ pro všechna $q \in Q$, $a \in \mathcal{A}$, $Z \in G$ a $\delta(q, \varepsilon, Z) = \emptyset$, nebo
   b) $\delta(q, a, Z) = \emptyset$ pro všechna $a \in \mathcal{A}$ a $|\delta(q, \varepsilon, Z)| \leq 1$.
2. $|\delta_T(q, P, Z)| \leq 1$ pro všechna $q \in Q$, $P \subset (Q_s \times D \times seq(N))$ a $Z \in G$.
3. Pro všechna $q \in Q$ a $Z \in G$ platí:
   a) existuje nejvýše jedna množina $P \subset (Q_s \times D \times seq(N))$ taková, že $\delta_T(q, P, Z) \neq \emptyset$, nebo
   b) takových množin je více, P_q je jejich sjednocení a pro každé $P \in \mathcal{P}(P_q)$, $P \neq \emptyset$, platí, že $\delta_T(q, P, Z) \neq \emptyset$.

Za zmínku stojí třetí bod v předchozí definici. Podmínka uvedená v tomto bodě zaručí, že všechny přechody představované množinou δ_T, jejichž počáteční stav je $q \in Q$ a kde požadovaný symbol na vrcholu zásobníku je $Z \in G$, obsáhnou všechny podmnožiny trojic $(p, i, s_p^{[i]}) \in (Q_s \times D \times seq(N))$, které se nachází ve výčtech trojic všech těchto přechodů (v případě že takový přechod není pouze jeden). Pokryjí se tak všechny možné kombinace splnění hodnot uspořádaných posloupností čísel, čímž se předejde nedeterminismu.

Jazyk L přijímaný rozšířeným atributovaným zásobníkovým automatem M je definován obdobně jako v případě standardního atributovaného zásobníkového automatu dvěma způsoby:

1. *Přijímání koncovým stavem:*

$$L(M) = \{x : (q_0, x, Z_0, s_0^d) \vdash_M^* (q, \varepsilon, \gamma, s) \\ \wedge\, x \in \mathcal{A}^* \wedge \gamma \in G^* \wedge s \ \in seq(N)^{d|Q_s|} \wedge q \in F\}.$$

2. *Přijímání prázdným zásobníkem:*

$$L_\varepsilon(M) = \{x : (q_0, x, Z_0, s_0^d) \vdash_M^* (q, \varepsilon, \varepsilon, s) \\ \wedge\, x \in \mathcal{A}^* \wedge s \ \in seq(N)^{d|Q_s|} \wedge q \in Q\}.$$

Pokud rozšířený atributovaný zásobníkový automat přijímá jazyk prázdným zásobníkem, pak množina koncových stavů F je prázdná množina.

3.4 Algoritmus determinizace automatu

Následuje výpis algoritmu, který je specificky navržený pro determinizaci atributovaného zásobníkového automatu pro vyhledávání a jehož výstupem je odpovídající deterministický rozšířený atributovaný zásobníkový automat. Algoritmus 3.2.1–3.2.4 pracuje se dvěma frontami stavů a skončí, když jsou obě fronty prázdné. Stav se zařadí do fronty *queuea*, pokud je potřeba zkonstruovat přechody vedoucí z tohoto stavu odpovídající zobrazení δ, naopak stav se zařadí do fronty *queueb*, pokud je potřeba zkonstruovat přechody vedoucí z tohoto stavu odpovídající zobrazení δ_T.

Cyklus obsluhující frontu *queuea* (část 2/4 výpisu algoritmu) pracuje na podobném principu jako determinizace konečných automatů. V těle cyklu obsluhujícího frontu *queueb* se nejdříve provádí rozhodnutí, jak vytvořit množinu V množin dvojic $(p, s) \in Q \times seq(N)$, na základě které se dále vytvářejí přechody odpovídající zobrazení δ_T. Významná je množina $B \subset Q$, která obsahuje stavy odpovídající uzlům, které leží na nejdelší cestě z kořene stromu, jejíž všechny uzly jsou ohodnocené symbolem, kterým je ohodnocen kořen. Pokud je potřeba vytvořit přechody ze stavu, jehož d-podmnožina je podmnožinou množiny B, potom stačí uvažovat pouze nejvyšší stav $p \in Q$ z této d-podmnožiny. Důvodem je fakt, že není možné, aby došlo k přijetí podstromu bez předchozího přijetí některého z jeho dílčích podstromů. Algoritmus vychází z toho, že d-podmnožiny všech stavů, u kterých lze provést popsanou optimalizaci, jsou podmnožiny množiny B.

Algoritmus 3.2.1 Transformace nedeterministického atributovaného zásobníkového automatu pro vyhledávání na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 1/4 – vstup, výstup).

Vstup: Nedeterministický atributovaný zásobníkový automat $M(t) = (Q, \mathcal{A}, G, N, \delta, \delta_T, 0, Z_0, s_0, \{1\})$, kde $\{0,1,2\} \subset Q$ a $G = \{S,]\}$. Automat musí splňovat následující podmínky:

1. Uspořádání stavů automatu vzhledem k jejich číselnému označení je takové, že:
   a) pokud $\delta(q, a, \alpha) = (p, \beta)$, pak platí $q < p$, nebo $q = p = 0$, kde $q, p \in Q$, $a \in \mathcal{A}$, $\alpha \in G$ a $\beta \in G^*$,
   b) pokud $\delta_T(q, s, \alpha) = (p, k, \beta)$, pak platí $q > p$, kde $q, p \in Q$, $s \in seq(N)$, $\alpha \in G$, $k \in N$ a $\beta \in G^*$.
2. Pro všechna $a \in \mathcal{A}$ platí, že $(0, S^{arity(a)}]) \in \delta(0, a, S)$, kde $S^0 = \varepsilon$.
3. Pro stav $q = 0$ neexistuje přechod, který lze zapsat $\delta_T(q, s, \alpha) = (p, k, \beta)$ nebo $\delta_T(p, s, \alpha) = (q, k, \beta)$, kde $p \in Q$, $s \in seq(N)$, $\alpha \in G$, $k \in N$ a $\beta \in G^*$.
4. Pro stav $q = 1$ neexistuje přechod, který lze zapsat $\delta(q, a, \alpha) = (p, \beta)$, $\delta(p, a, \alpha) = (q, \beta)$ nebo $\delta_T(q, s, \alpha) = (p, k, \beta)$, kde $p \in Q$, $a \in \mathcal{A}$, $\alpha \in G$, $\beta \in G^*$, $s \in seq(N)$ a $k \in N$.
5. Do každého stavu $q \in Q$, $q \neq 0$, $q \neq 1$, vede právě jeden přechod, který lze zapsat $\delta(p, a, S) = (q, S^{arity(a)}])$, kde $p \in Q$, $a \in \mathcal{A}$ a $S^0 = \varepsilon$.
6. Pro každý stav $q \in Q$, $q \neq 0$, $q \neq 1$, existuje právě jeden stav $p \in Q$, do kterého vede alespoň jeden přechod, který lze zapsat $\delta_T(q, s,]) = (p, k, \varepsilon)$, kde $s \in seq(N)$ a $k \in N$.
7. Neexistuje přechod, který lze zapsat $(q, \varepsilon, \alpha) = (p, \beta)$, kde $q, p \in Q$, $\alpha \in G$ a $\beta \in G^*$.

Výstup: Odpovídající deterministický rozšířený atributovaný zásobníkový automat $M_d(t) = (Q', Q, \mathcal{A}, G, N, D, \delta', \delta'_T, q_I, Z_0, s_0, F')$.

Algoritmus 3.2.2 Transformace nedeterministického atributovaného zásobníkového automatu pro vyhledávání na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 2/4 – vlastní algoritmus).

1: $q_I \leftarrow [0]$, $Q' \leftarrow \{[0]\}$
2: $B \leftarrow \emptyset$, $Q'_{enqb} \leftarrow \emptyset$
3: $position[0\,..\,\textsc{Max}(Q)] \leftarrow$ New-Array
4: $queuea \leftarrow$ New-Queue, $queueb \leftarrow$ New-Queue
5: Enqueue($queuea, q_I$)
6: **while not** Empty($queuea$) **or not** Empty($queueb$) **do**
7: **while not** Empty($queuea$) **do**
8: $q' \leftarrow$ Dequeue($queuea$)
9: **for all** $a \in \mathcal{A}$ **do**
10: $q'' \leftarrow \emptyset$, $i \leftarrow 1$
11: **for all** $p \in q'$ **in ascending order do**
12: **for all** $(q, \beta) \in \delta(p, a, \alpha)$, $q \in Q$, $\beta \in G^*$, $\alpha \in G$ **do**
13: $q'' \leftarrow q'' \cup \{q\}$
14: $\alpha' \leftarrow \alpha$, $\beta' \leftarrow \beta$
15: $position[q] \leftarrow i$
16: **end for**
17: $i \leftarrow i + 1$
18: **end for**
19: $i \leftarrow 1$
20: **for all** $q \in q''$ **in ascending order do** ▷ Nyní $q'' \neq \emptyset$
21: $k_i \leftarrow position[q]$
22: $i \leftarrow i + 1$
23: **end for**
24: **if not** $q'' \in Q'$ **then**
25: $Q' \leftarrow Q' \cup \{q''\}$, $D \leftarrow D \cup \{|q''|\}$
26: **if** $arity(a) > 0$ **then**
27: Enqueue($queuea, q''$)
28: **else**
29: Enqueue($queueb, q''$)
30: **end if**
31: **if** $2 \in q''$ **then**
32: $B \leftarrow B \cup q''$
33: **end if**
34: **end if**
35: Vytvořit přechod $\delta'(q', a, \alpha') = (q'', \beta', (k_1, k_2, \ldots, k_{|q''|}))$.
36: **end for**
37: **end while**

Algoritmus 3.2.3 Transformace nedeterministického atributovaného zásobníkového automatu pro vyhledávání na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 3/4 – vlastní algoritmus, pokračování).

38: **while not** EMPTY($queueb$) **do**
39: $q' \leftarrow$ DEQUEUE($queueb$)
40: **if** $q' \subset B$ **then**
41: $V \leftarrow \mathcal{P}(\{(p,s) : p = \text{MAX}(q') \wedge \delta_T(p,s,]) \neq \emptyset\}) \setminus \emptyset$
42: **else**
43: $q \in Q$, $\alpha \in G$, $\beta \in G^*$
44: $V_{base} \leftarrow \{(p,s) : p \in q' \wedge \delta_T(p,s,]) \neq \emptyset\}$
45: $|\{a : (p,\beta) \in \delta(q,a,\alpha)\}| = 1$, $p \in q'$, $p \neq 0$
46: **if** $arity(a) > 1$ **or** $|V_{base}| > |q'| - 1$ **then**
47: $V \leftarrow \mathcal{P}(V_{base}) \setminus \{\emptyset\}$
48: **else**
49: $V \leftarrow \{V_{base}\}$
50: **end if**
51: **end if**
52: **for all** $V_{sub} \in V$ **do**
53: $q'' \leftarrow \{0\}$, $i \leftarrow 1$
54: $position[0] \leftarrow 1$
55: **for all** $p \in q'$ **in ascending order do**
56: **if** $(p,s) \in V_{sub}$, $s \in seq(N)$ **then**
57: $|\{q : (q,l,\varepsilon) \in \delta_T(p,s,])\}| = 1$, $l \in N$
58: $q'' \leftarrow q'' \cup \{q\}$
59: $position[q] \leftarrow i$
60: **end if**
61: $i \leftarrow i + 1$
62: **end for**

Algoritmus 3.2.4 Transformace nedeterministického atributovaného zásobníkového automatu pro vyhledávání na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 4/4 – vlastní algoritmus, pokračování).

63: $P \leftarrow \emptyset$, $R \leftarrow \emptyset$, $i \leftarrow 1$
64: **for all** $q \in q''$ **in ascending order do**
65: **for all** $(p, s) \in V_{sub}$ **do**
66: **if** $(q, l, \varepsilon) \in \delta_T(p, s,])$, $l \in N$ **then**
67: $P \leftarrow P \cup \{(p, i, s)\}$
68: $R \leftarrow R \cup \{(q, i, l)\}$
69: **end if**
70: **end for**
71: $k_i \leftarrow position[q]$
72: $i \leftarrow i + 1$
73: **end for**
74: **if** $1 \in q''$ **then**
75: $Q' \leftarrow Q' \cup \{q''\}$, $F' \leftarrow \{q''\}$
76: **else**
77: **if not** $q'' \in Q'$ **then**
78: $Q' \leftarrow Q' \cup \{q''\}$
79: Enqueue$(queuea, q'')$
80: **end if**
81: **if not** $q'' \in Q'_{enqb}$ **then**
82: $Q'_{enqb} \leftarrow Q'_{enqb} \cup \{q''\}$
83: Enqueue$(queueb, q'')$
84: **end if**
85: **end if**
86: Vytvořit přechod $\delta'_T(q', P,]) = (q'', R, (k_1, k_2, \ldots, k_{|q''|}), \varepsilon)$.
87: **end for**
88: **end while**
89: **end while**

Množina V se kromě krajních případů, kdy arita symbolu, kterým jsou ohodnoceny uzly odpovídající stavům z d-podmnožiny zpracovávaného stavu, je rovna 0, případně 1, vytváří jako potenční množina výchozí množiny přechodů proto, aby došlo ke splnění třetí podmínky z definice deterministického rozšířeného atributovaného zásobníkového automatu. V případě řádky 41 je vytváření potenční množiny z hlediska funkčnosti automatu zbytečné a je nutné pouze pro splnění zmíněné definice. Vytváření těchto potenčních množin přechodů (a případně zbytečných stavů) je zjevnou slabinou tohoto algoritmu, resp. definice deterministického rozšířeného atributovaného zásobníkového automatu. Je možné, že určité předzpracování stromu (např. výpočet repetic podstromů, který lze provést v lineárním čase [3]) a vytvoření vhodných pomocných datových struktur by umožnilo identifikovat potřebné přechody a stavy a vyhnout se tak vytváření potenční množiny.

3.5 Příklady deterministických automatů

Příklad 3.3. Uvažujme nedeterministický atributovaný zásobníkový automat $M_{ns}(t_4)$ z příkladu 3.1, jehož přechodový diagram je na obrázku 3.4. Odpovídající deterministický rozšířený atributovaný zásobníkový automat, který byl zkonstruován algoritmem 3.2.1–3.2.4 z nedeterministického atributovaného zásobníkového automatu $M_{ns}(t_4)$, je deterministický rozšířený atributovaný zásobníkový automat $M_{ds}(t_4) = (\{[0], [0,1], [0,2], [0,3], [0,4], [0,5], [0,7], [0,8]\}, \{0,1,2,3,4,5,7,8\}, \mathcal{A}, \{S,]\}, \{1,2,3\}, \{1,2\}, \delta, \delta_T, [0], S, \emptyset^8, \{[0,1]\})$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$
\begin{aligned}
\delta([0], a3, S) &= ([0,2], SSS], (1,1)) \\
\delta([0], a1, S) &= ([0], S], (1)) \\
\delta([0], a0, S) &= ([0],], (1)) \\
\delta([0,2], a3, S) &= ([0,2], SSS], (1,1)) \\
\delta([0,2], a1, S) &= ([0,3], S], (1,2)) \\
\delta([0,2], a0, S) &= ([0,8],], (1,2)) \\
\delta([0,3], a3, S) &= ([0,2], SSS], (1,1)) \\
\delta([0,3], a1, S) &= ([0,4], S], (1,2)) \\
\delta([0,3], a0, S) &= ([0,7],], (1,2)) \\
\delta([0,4], a3, S) &= ([0,2], SSS], (1,1)) \\
\delta([0,4], a1, S) &= ([0], S], (1)) \\
\delta([0,4], a0, S) &= ([0,5],], (1,2))
\end{aligned}
$$

	stav	$a3$	$a1$	$a0$	δ_T
$\rightarrow$	$[0]$	$[0,2]$	$[0]$	$[0]$	–
	$[0,2]$	$[0,2]$	$[0,3]$	$[0,8]$	$[0,1]$
	$[0,3]$	$[0,2]$	$[0,4]$	$[0,7]$	$[0,2]$
	$[0,8]$	–	–	–	$[0,2]$
	$[0,4]$	$[0,2]$	$[0]$	$[0,5]$	$[0,3]$
	$[0,7]$	–	–	–	$[0,3]$
	$[0,5]$	–	–	–	$[0,4]$
$\leftarrow$	$[0,1]$	–	–	–	–

Tabulka 3.1: Tabulka přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_4)$ pro vyhledávání stromu t_4 s prefixovým zápisem $pref(t_4) = a3\ a1\ a1\ a0\ a1\ a0\ a0$ s libovolným seřazením následníků z příkladu 3.3

$$\begin{aligned}
\delta_T([0,2],\{(2,2,(1,2,3))\},]) &= ([0,1],\{(1,2,1)\},\varepsilon,(1,2)) \\
\delta_T([0,3],\{(3,2,(1))\},]) &= ([0,2],\{(2,2,1)\},\varepsilon,(1,2)) \\
\delta_T([0,3],\{(3,2,(2))\},]) &= ([0,2],\{(2,2,2)\},\varepsilon,(1,2)) \\
\delta_T([0,3],\{(3,2,(1)),(3,2,(2))\},]) &= ([0,2],\{(2,2,1),(2,2,2)\},\varepsilon,(1,2)) \\
\delta_T([0,4],\{(4,2,(1))\},]) &= ([0,3],\{(3,2,1)\},\varepsilon,(1,2)) \\
\delta_T([0,5],\{(5,2,\emptyset)\},]) &= ([0,4],\{(4,2,1)\},\varepsilon,(1,2)) \\
\delta_T([0,7],\{(7,2,\emptyset)\},]) &= ([0,3],\{(3,2,2)\},\varepsilon,(1,2)) \\
\delta_T([0,8],\{(8,2,\emptyset)\},]) &= ([0,2],\{(2,2,3)\},\varepsilon,(1,2))
\end{aligned}$$

Tabulka 3.1 je tabulka přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_4)$, která pouze znázorňuje množiny stavů, do kterých vedou přechody odpovídající zobrazení δ pro každý stav a každý symbol vstupní abecedy, resp. přechody odpovídající zobrazení δ_T pro každý stav automatu.

Tabulka 3.2 znázorňuje posloupnost přechodů provedených deterministickým rozšířeným atributovaným zásobníkovým automatem $M_{ds}(t_4)$ pro náhodně uspořádaný strom v prefixovém zápisu, jehož jedním podstromem je také strom t_4.

Přechod $\delta_T([0,3],\{(3,2,(1)),(3,2,(2))\},]) = ([0,2],\{(2,2,1),(2,2,2)\}, \varepsilon,(1,2))$ je příklad přechodu, který vznikl z důvodu vytváření potenční množiny výchozí množiny přechodů a který je zbytečný. Není možné, aby platilo $s_3^{[2]} = (1) \wedge s_3^{[2]} = (2)$. V tomto příkladě nedošlo k vytvoření zbytečných stavů.

stav	vstupní řetězec/změny atributů	zásobník
$[0]$	$a3\ a3\ a1\ a0\ a1\ a1\ a0\ a0\ a0\ a0$	S
$[0,2]$	$a3\ a1\ a0\ a1\ a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[1]})$	$S\ S\ S\ ]$
$[0,2]$	$a1\ a0\ a1\ a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[1]})$	$S\ S\ S\ ]\ S\ S\ ]$
$[0,3]$	$a0\ a1\ a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$S\ ]\ S\ S\ ]\ S\ S\ ]$
$[0,7]$	$a1\ a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$]\ ]\ S\ S\ ]\ S\ S\ ]$
$[0,3]$	$a1\ a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_3^{[2]} \leftarrow (2),\ s_7^{[2]} \leftarrow \emptyset$	$]\ S\ S\ ]\ S\ S\ ]$
$[0,2]$	$a1\ a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_2^{[2]} \leftarrow (2),\ s_3^{[2]} \leftarrow \emptyset$	$S\ S\ ]\ S\ S\ ]$
$[0,3]$	$a1\ a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$S\ ]\ S\ ]\ S\ S\ ]$
$[0,4]$	$a0\ a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$S\ ]\ ]\ S\ ]\ S\ S\ ]$
$[0,5]$	$a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$]\ ]\ ]\ S\ ]\ S\ S\ ]$
$[0,4]$	$a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_4^{[2]} \leftarrow (1),\ s_5^{[2]} \leftarrow \emptyset$	$]\ ]\ S\ ]\ S\ S\ ]$
$[0,3]$	$a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_3^{[2]} \leftarrow (1),\ s_4^{[2]} \leftarrow \emptyset$	$]\ S\ ]\ S\ S\ ]$
$[0,2]$	$a0\ a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_2^{[2]} \leftarrow (1,2),\ s_3^{[2]} \leftarrow \emptyset$	$S\ ]\ S\ S\ ]$
$[0,8]$	$a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$]\ ]\ S\ S\ ]$
$[0,2]$	$a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_2^{[2]} \leftarrow (1,2,3),\ s_8^{[2]} \leftarrow \emptyset$	$]\ S\ S\ ]$
$[0,1]$	$a0\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_1^{[2]} \leftarrow (1),\ s_2^{[2]} \leftarrow \emptyset$	$S\ S\ ]$

Tabulka 3.2: Posloupnost přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_4)$ pro vyhledávání stromu t_4 z příkladu 3.3 pro vstupní řetězec $a3\ a3\ a1\ a0\ a1\ a1\ a0\ a0\ a0\ a0$

Příklad 3.4. Uvažujme nedeterministický atributovaný zásobníkový automat $M_{ns}(t_1)$ z příkladu 3.2, jehož přechodový diagram je na obrázku 3.5. Odpovídající deterministický rozšířený atributovaný zásobníkový automat, který byl zkonstruován algoritmem 3.2.1–3.2.4 z nedeterministického atributovaného zásobníkového automatu $M_{ns}(t_1)$, je deterministický rozšířený atributovaný zásobníkový automat $M_{ds}(t_1) = (\{[0], [0,1], [0,2], [0,2,3], [0,3], [0,4], [0,5], [0,5,7], [0,6], [0,6,8], [0,7], [0,8]\}, \{0,1,2,3,4,5,6,7,8\}, \mathcal{A}, \{S,]\}, \{1,2\}, \{1,2,3\}, \delta, \delta_T, [0], S, \emptyset^{12}, \{[0,1]\})$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$
\begin{aligned}
\delta([0], a2, S) &= ([0,2], SS], (1,1)) \\
\delta([0], a1, S) &= ([0], S], (1)) \\
\delta([0], a0, S) &= ([0],], (1)) \\
\delta([0,2], a2, S) &= ([0,2,3], SS], (1,1,2)) \\
\delta([0,2], a1, S) &= ([0,7], S], (1,2)) \\
\delta([0,2], a0, S) &= ([0],], (1)) \\
\delta([0,2,3], a2, S) &= ([0,2,3], SS], (1,1,2)) \\
\delta([0,2,3], a1, S) &= ([0,5,7], S], (1,3,2)) \\
\delta([0,2,3], a0, S) &= ([0,4],], (1,3)) \\
\delta([0,3], a2, S) &= ([0,2], SS], (1,1)) \\
\delta([0,3], a1, S) &= ([0,5], S], (1,2)) \\
\delta([0,3], a0, S) &= ([0,4],], (1,2)) \\
\delta([0,5], a2, S) &= ([0,2], SS], (1,1)) \\
\delta([0,5], a1, S) &= ([0], S], (1)) \\
\delta([0,5], a0, S) &= ([0,6],], (1,2)) \\
\delta([0,5,7], a2, S) &= ([0,2], SS], (1,1)) \\
\delta([0,5,7], a1, S) &= ([0], S], (1)) \\
\delta([0,5,7], a0, S) &= ([0,6,8],], (1,2,3)) \\
\delta([0,7], a2, S) &= ([0,2], SS], (1,1)) \\
\delta([0,7], a1, S) &= ([0], S], (1)) \\
\delta([0,7], a0, S) &= ([0,8],], (1,2)) \\
\delta_T([0,2], \{(2,2,(1,2))\},]) &= ([0,1], \{(1,2,1)\}, \varepsilon, (1,2)) \\
\delta_T([0,2,3], \{(3,2,(1,2))\},]) &= ([0,2], \{(2,2,1)\}, \varepsilon, (1,3)) \\
\delta_T([0,3], \{(3,2,(1,2))\},]) &= ([0,2], \{(2,2,1)\}, \varepsilon, (1,2)) \\
\delta_T([0,4], \{(4,2,\emptyset)\},]) &= ([0,3], \{(3,2,1)\}, \varepsilon, (1,2))
\end{aligned}
$$

	stav	$a2$	$a1$	$a0$	δ_T
$\rightarrow$	$[0]$	$[0,2]$	$[0]$	$[0]$	–
	$[0,2]$	$[0,2,3]$	$[0,7]$	$[0]$	$[0,1]$
	$[0,2,3]$	$[0,2,3]$	$[0,5,7]$	$[0,4]$	$[0,2]$
	$[0,7]$	$[0,2]$	$[0]$	$[0,8]$	$[0,2]$
	$[0,5,7]$	$[0,2]$	$[0]$	$[0,6,8]$	$[0,2,3]$
	$[0,4]$	–	–	–	$[0,3]$
	$[0,8]$	–	–	–	$[0,7]$
	$[0,6,8]$	–	–	–	$[0,5,7]$
	$[0,3]$	$[0,2]$	$[0,5]$	$[0,4]$	$[0,2]$
$\leftarrow$	$[0,1]$	–	–	–	–
	$[0,5]$	$[0,2]$	$[0]$	$[0,6]$	$[0,3]$
	$[0,6]$	–	–	–	$[0,5]$

Tabulka 3.3: Tabulka přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_1)$ pro vyhledávání stromu t_1 s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 3.4

$$\delta_T([0,5],\{(5,2,(1))\},]) = ([0,3],\{(3,2,2)\},\varepsilon,(1,2))$$
$$\delta_T([0,5,7],\{(5,3,(1)),(7,2,(1))\},]) = ([0,2,3],\{(2,2,2),(3,3,2)\},\varepsilon,(1,3,2))$$
$$\delta_T([0,6],\{(6,2,\emptyset)\},]) = ([0,5],\{(5,2,1)\},\varepsilon,(1,2))$$
$$\delta_T([0,6,8],\{(6,2,\emptyset),(8,3,\emptyset)\},]) = ([0,5,7],\{(5,2,1),(7,3,1)\},\varepsilon,(1,2,3))$$
$$\delta_T([0,7],\{(7,2,(1))\},]) = ([0,2],\{(2,2,2)\},\varepsilon,(1,2))$$
$$\delta_T([0,8],\{(8,2,\emptyset)\},]) = ([0,7],\{(7,2,1)\},\varepsilon,(1,2))$$

Tabulka 3.3 je tabulka přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_1)$, která pouze znázorňuje množiny stavů, do kterých vedou přechody odpovídající zobrazení δ pro každý stav a každý symbol vstupní abecedy, resp. přechody odpovídající zobrazení δ_T pro každý stav automatu.

Tabulka 3.4 znázorňuje posloupnost přechodů provedených deterministickým rozšířeným atributovaným zásobníkovým automatem $M_{ds}(t_1)$ pro náhodně uspořádaný strom v prefixovém zápisu, jehož jedním podstromem je také strom t_1.

stav	vstupní řetězec/změny atributů	zásobník
$[0]$	$a2\ a1\ a0\ a2\ a2\ a1\ a0\ a0\ a1\ a0$	S
$[0,2]$	$a1\ a0\ a2\ a2\ a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[1]})$	$S\ S\ ]$
$[0,7]$	$a0\ a2\ a2\ a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$S\ ]\ S\ ]$
$[0,8]$	$a2\ a2\ a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]})$	$]\ ]\ S\ ]$
$[0,7]$	$a2\ a2\ a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_7^{[2]} \leftarrow (1),\ s_8^{[2]} \leftarrow \emptyset$	$]\ S\ ]$
$[0,2]$	$a2\ a2\ a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_2^{[2]} \leftarrow (2),\ s_7^{[2]} \leftarrow \emptyset$	$S\ ]$
$[0,2,3]$	$a2\ a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[1]}, s^{[2]})$	$S\ S\ ]\ ]$
$[0,2,3]$	$a1\ a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[1]}, s^{[2]})$	$S\ S\ ]\ S\ ]\ ]$
$[0,5,7]$	$a0\ a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[3]}, s^{[2]})$	$S\ ]\ S\ ]\ S\ ]\ ]$
$[0,6,8]$	$a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}, s^{[3]})$	$]\ ]\ S\ ]\ S\ ]\ ]$
$[0,5,7]$	$a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}, s^{[3]}),$ $s_5^{[2]} \leftarrow (1),\ s_6^{[2]} \leftarrow \emptyset,\ s_7^{[3]} \leftarrow (1),\ s_8^{[3]} \leftarrow \emptyset$	$]\ S\ ]\ S\ ]\ ]$
$[0,2,3]$	$a0\ a1\ a0$ $s \leftarrow (s^{[1]}, s^{[3]}, s^{[2]}),$ $s_2^{[2]} \leftarrow (2),\ s_7^{[2]} \leftarrow \emptyset,\ s_3^{[3]} \leftarrow (2),\ s_5^{[3]} \leftarrow \emptyset$	$S\ ]\ S\ ]\ ]$
$[0,4]$	$a1\ a0$ $s \leftarrow (s^{[1]}, s^{[3]})$	$]\ ]\ S\ ]\ ]$
$[0,3]$	$a1\ a0$ $s \leftarrow (s^{[1]}, s^{[2]}),\ s_3^{[2]} \leftarrow (1,2),\ s_4^{[2]} \leftarrow \emptyset$	$]\ S\ ]\ ]$

(pokračování na další straně)

Tabulka 3.4: Posloupnost přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_1)$ pro vyhledávání stromu t_1 z příkladu 3.4 pro vstupní řetězec $a2\ a1\ a0\ a2\ a2\ a1\ a0\ a0\ a1\ a0$

(pokračování z předchozí strany)

stav	vstupní řetězec/změny atributů	zásobník
$[0,2]$	$a1\ a0$	$S\ ]\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]}),\ s_2^{[2]} \leftarrow (1),\ s_3^{[2]} \leftarrow \emptyset$	
$[0,7]$	$a0$	$S\ ]\ ]\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]})$	
$[0,8]$	ε	$]\ ]\ ]\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]})$	
$[0,7]$	ε	$]\ ]\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]}),\ s_7^{[2]} \leftarrow (1),\ s_8^{[2]} \leftarrow \emptyset$	
$[0,2]$	ε	$]\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]}),\ s_2^{[2]} \leftarrow (1,2),\ s_7^{[2]} \leftarrow \emptyset$	
$[0,1]$	ε	$]$
	$s \leftarrow (s^{[1]}, s^{[2]}),\ s_1^{[2]} \leftarrow (1),\ s_2^{[2]} \leftarrow \emptyset$	

Tabulka 3.4: Posloupnost přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{ds}(t_1)$ pro vyhledávání stromu t_1 z příkladu 3.4 pro vstupní řetězec $a2\ a1\ a0\ a2\ a2\ a1\ a0\ a0\ a1\ a0$

Kapitola 4

Automaty přijímající podstromy

V této kapitole ukážeme dvě varianty konstrukce automatu, který přijímá všechny podstromy stromu zadaného v prefixovém zápisu s libovolným seřazením následníků. Takový automat představuje index daného stromu a je obdobou faktorového automatu známého ze stringologie, kde takový automat přijímá všechny podřetězce určitého řetězce. Pro každou variantu ukážeme příklady nedeterministických automatů a také odpovídající deterministické automaty.

4.1 První varianta automatu

4.1.1 Algoritmus konstrukce automatu

Pro konstrukci automatu přijímajícího všechny podstromy určitého stromu s libovolným seřazením následníků lze použít postup analogický k jednomu z možných postupů vytváření NFA přijímajícího všechny podřetězce určitého řetězce, kde se výsledný nedeterministický automat skládá z několika lineárních posloupností stavů a přechodů, které sdílejí společný počáteční stav [4]. Zde sdílejí skupiny stavů odpovídající jednotlivým podstromům daného stromu počáteční stav a stav, ve kterém automat přijímá podstrom. Každý podstrom je tedy reprezentován vlastní skupinou stavů, kde topologie přechodů mezi těmito stavy odpovídá topologii daného podstromu.

Algoritmus konstrukce popsaného automatu 4.1.1–4.1.2 je podobný algoritmu 2.2. Zásadní rozdíl oproti algoritmu 2.2 spočívá v udržování celočíselné proměnné (*numopentrees*), která určuje kolik podstromů je aktuálně ne zcela přečtených, a tedy kolik stavů je v dané iteraci pro aktuální symbol vstupního řetězce nutno vytvořit. Odlišností je, že na základě délky vstupu nejsme schopni určit počet stavů výsledného automatu, a tudíž algoritmus

přidává stavy v průběhu zpracovávání vstupu. Algoritmus pracuje v kvadraticky omezeném čase vzhledem k délce vstupního řetězce, tedy počtu uzlů stromu, jehož prefixový zápis je zadán na vstupu algoritmu. Rovněž počet stavů výsledného nedeterministického automatu je kvadraticky omezen vzhledem k délce vstupního řetězce.

Příjem konkrétního podstromu se vyznačuje atributem stavu, ve kterém automat přijímá podstrom, nastaveným na jednoprvkovou uspořádanou posloupnost obsahující celočíselný identifikátor tohoto podstromu, přičemž v případě přečtení prefixového zápisu celého stromu je tento identifikátor roven 1.

Algoritmus 4.1.1 Konstrukce nedeterministického atributovaného zásobníkového automatu přijímajícího všechny podstromy stromu t s prefixovým zápisem $pref(t)$ s libovolným seřazením následníků (část 1/2).

Vstup: Strom t nad ohodnocenou abecedou $\mathcal{A}$; prefixový zápis $pref(t) = a_1 a_2 \dots a_n$, $n \geq 1$.

Výstup: Nedeterministický atributovaný zásobníkový automat $M_{ni}(t) = (Q, \mathcal{A}, \{S,]\}, N, \delta, \delta_T, 0, S, \emptyset^{|Q|}, \emptyset)$.

1: $Q \leftarrow \{0, 1\}$
2: $maxarity \leftarrow 1$
3: $q \leftarrow 0$ ▷ Stav odpovídající kořenu prvního nezpracovaného podstromu
4: $numopentrees \leftarrow 0$
5: **for** $i \leftarrow 1$ **to** n **do**
6: $\quad a \in \mathcal{A}$
7: $\quad t \leftarrow (1, 2, \dots, arity(a_i))$
8: $\quad$ ▷ Přidání stavů a vytvoření dvojic přechodů pro všechny dosud ne zcela zpracované podstromy
9: $\quad$ **for** $j \leftarrow 0$ **to** $numopentrees - 1$ **do**
10: $\quad\quad p \leftarrow |Q|$
11: $\quad\quad Q \leftarrow Q \cup \{p\}$
12: $\quad\quad$ Vytvořit přechod $\delta(q + j, a_i, S) = (p, S^{arity(a_i)}])$, kde $S^0 = \varepsilon$.
13: $\quad\quad$ Vytvořit přechod $\delta_T(p, t,]) = (q + j, |\delta(q + j, a, S)|, \varepsilon)$.
14: $\quad$ **end for**
15: $\quad$ ▷ Přidání stavu a vytvoření dvojice přechodů pro nový podstrom
16: $\quad p \leftarrow |Q|$
17: $\quad Q \leftarrow Q \cup \{p\}$
18: $\quad$ Vytvořit přechod $\delta(0, a_i, S) = (p, S^{arity(a_i)}])$, kde $S^0 = \varepsilon$.
19: $\quad$ Vytvořit přechod $\delta_T(p, t,]) = (1, |\delta(0, a, S)|, \varepsilon)$.
20: $\quad q \leftarrow i + 1$
21: $\quad numopentrees \leftarrow numopentrees + 1$

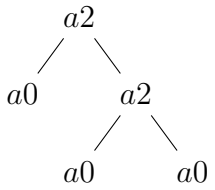
Algoritmus 4.1.2 Konstrukce nedeterministického atributovaného zásobníkového automatu přijímajícího všechny podstromy stromu t s prefixovým zápisem $pref(t)$ s libovolným seřazením následníků (část 2/2).

22: ▷ Aktualizace proměnné $maxarity$
23: **if** $arity(a_i) > maxarity$ **then**
24: $maxarity \leftarrow arity(a_i)$
25: **end if**
26: ▷ Nastavení proměnné q postupnou kontrolou počtu zpracovaných podstromů
27: $b \leftarrow a_i$
28: **while** $q \neq 0$ **and** $|\delta(q, a, S)| = arity(b)$ **do**
29: $|\{r : \delta(r, b, S) = (q, S^{arity(b)})\}| = 1$
30: $q \leftarrow r$
31: $numopentrees \leftarrow numopentrees - 1$
32: **if** $q \neq 0$ **then**
33: $|\{c : \delta(u, c, S) = (q, S^{arity(c)})\}| = 1$, $u \in Q$
34: $b \leftarrow c$
35: **end if**
36: **end while**
37: **end for**
38: $N \leftarrow \{1, 2, \ldots, maxarity\}$

4.1.2 Příklady nedeterministických automatů

Příklad 4.1. Uvažujme strom t_3, který je na obrázku 4.1, s prefixovým zápisem $pref(t_3) = a2\ a0\ a2\ a0\ a0$. Atributovaný zásobníkový automat přijímající všechny podstromy stromu t_3 s libovolným seřazením následníků, který byl zkonstruován algoritmem 4.1.1–4.1.2, je nedeterministický atributovaný zásobníkový automat $M_{ni}(t_3) = (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}, \mathcal{A}, \{S,]\}, \{1, 2\}, \delta, \delta_T, 0, S, \emptyset^{13}, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$\begin{aligned}
\delta(0, a2, S) &= (2, SS]) & \delta_T(2, (1, 2),]) &= (1, 1, \varepsilon)\\
\delta(0, a2, S) &= (6, SS]) & \delta_T(3, \emptyset,]) &= (2, 1, \varepsilon)\\
\delta(0, a0, S) &= (4,]) & \delta_T(4, \emptyset,]) &= (1, 2, \varepsilon)\\
\delta(0, a0, S) &= (9,]) & \delta_T(5, (1, 2),]) &= (2, 2, \varepsilon)\\
\delta(0, a0, S) &= (12,]) & \delta_T(6, (1, 2),]) &= (1, 3, \varepsilon)
\end{aligned}$$



$$pref(t_3) = a2\ a0\ a2\ a0\ a0$$

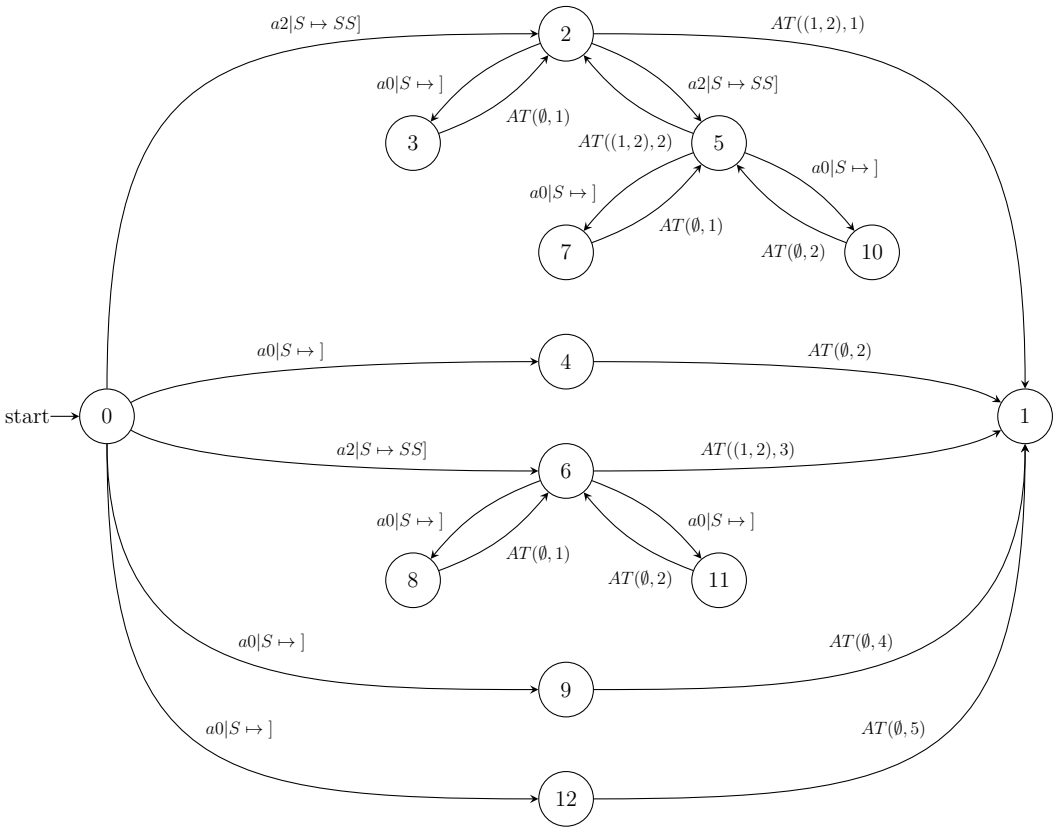
Obrázek 4.1: Strom t_3 a odpovídající prefixový zápis

$$\begin{aligned}
\delta(2, a2, S) &= (5, SS]) & \delta_T(7, \emptyset,]) &= (5, 1, \varepsilon)\\
\delta(2, a0, S) &= (3,]) & \delta_T(8, \emptyset,]) &= (6, 1, \varepsilon)\\
\delta(5, a0, S) &= (7,]) & \delta_T(9, \emptyset,]) &= (1, 4, \varepsilon)\\
\delta(5, a0, S) &= (10,]) & \delta_T(10, \emptyset,]) &= (5, 2, \varepsilon)\\
\delta(6, a0, S) &= (8,]) & \delta_T(11, \emptyset,]) &= (6, 2, \varepsilon)\\
\delta(6, a0, S) &= (11,]) & \delta_T(12, \emptyset,]) &= (1, 5, \varepsilon)
\end{aligned}$$

Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ni}(t_3)$ je na obrázku 4.2.

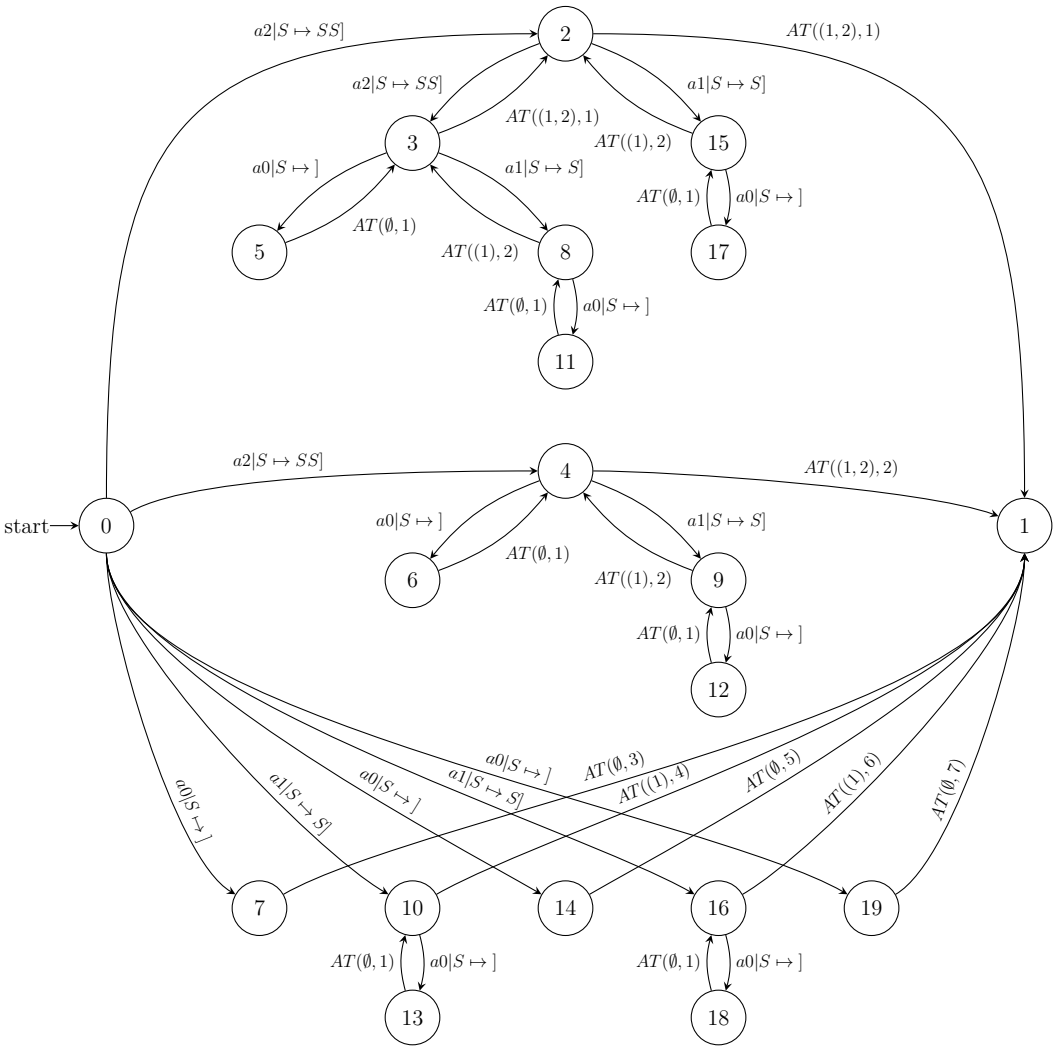
Příklad 4.2. Uvažujme strom t_1, který je na obrázku 2.1, s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$. Atributovaný zásobníkový automat přijímající všechny podstromy stromu t_1 s libovolným seřazením následníků, který byl zkonstruován algoritmem 4.1.1–4.1.2, je nedeterministický atributovaný zásobníkový automat $M_{ni}(t_1) = (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}, \mathcal{A}, \{S,]\}, \{1, 2\}, \delta, \delta_T, 0, S, \emptyset^{20}, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$\begin{aligned}
\delta(0, a2, S) &= (2, SS]) & \delta_T(2, (1, 2),]) &= (1, 1, \varepsilon)\\
\delta(0, a2, S) &= (4, SS]) & \delta_T(3, (1, 2),]) &= (2, 1, \varepsilon)\\
\delta(0, a1, S) &= (10, S]) & \delta_T(4, (1, 2),]) &= (1, 2, \varepsilon)\\
\delta(0, a1, S) &= (16, S]) & \delta_T(5, \emptyset,]) &= (3, 1, \varepsilon)\\
\delta(0, a0, S) &= (7,]) & \delta_T(6, \emptyset,]) &= (4, 1, \varepsilon)\\
\delta(0, a0, S) &= (14,]) & \delta_T(7, \emptyset,]) &= (1, 3, \varepsilon)\\
\delta(0, a0, S) &= (19,]) & \delta_T(8, (1),]) &= (3, 2, \varepsilon)
\end{aligned}$$



Obrázek 4.2: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ni}(t_3)$ přijímajícího všechny podstromy stromu t_3 s prefixovým zápisem $pref(t_3) = a2\ a0\ a2\ a0\ a0$ s libovolným seřazením následníků z příkladu 4.1

$$\begin{aligned}
\delta(2, a2, S) &= (3, SS]) & \delta_T(9, (1),]) &= (4, 2, \varepsilon)\\
\delta(2, a1, S) &= (15, S]) & \delta_T(10, (1),]) &= (1, 4, \varepsilon)\\
\delta(3, a1, S) &= (8, S]) & \delta_T(11, \emptyset,]) &= (8, 1, \varepsilon)\\
\delta(3, a0, S) &= (5,]) & \delta_T(12, \emptyset,]) &= (9, 1, \varepsilon)\\
\delta(4, a1, S) &= (9, S]) & \delta_T(13, \emptyset,]) &= (10, 1, \varepsilon)\\
\delta(4, a0, S) &= (6,]) & \delta_T(14, \emptyset,]) &= (1, 5, \varepsilon)\\
\delta(8, a0, S) &= (11,]) & \delta_T(15, (1),]) &= (2, 2, \varepsilon)\\
\delta(9, a0, S) &= (12,]) & \delta_T(16, (1),]) &= (1, 6, \varepsilon)\\
\delta(10, a0, S) &= (13,]) & \delta_T(17, \emptyset,]) &= (15, 1, \varepsilon)\\
\delta(15, a0, S) &= (17,]) & \delta_T(18, \emptyset,]) &= (16, 1, \varepsilon)\\
\delta(16, a0, S) &= (18,]) & \delta_T(19, \emptyset,]) &= (1, 7, \varepsilon)
\end{aligned}$$



Obrázek 4.3: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ni}(t_1)$ přijímajícího všechny podstromy stromu t_1 s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 4.2

Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{ni}(t_1)$ je na obrázku 4.3.

4.1.3 Odpovídající deterministické automaty

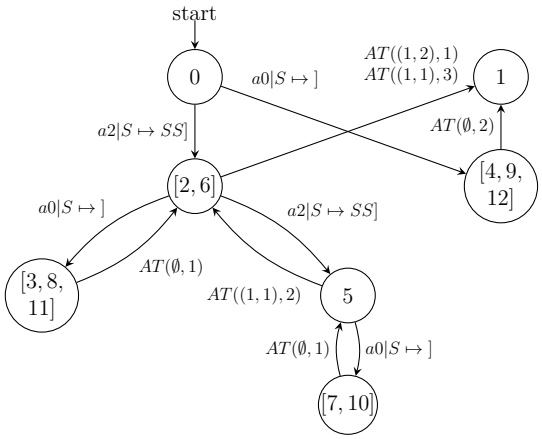
Ukazuje se, že pro determinizaci automatů přijímajících podstromy vytvořených algoritmem 4.1.1–4.1.2 postačuje algoritmus 2.4.1–2.4.4 uvedený v kapitole 2. Tyto automaty totiž splňují podmínky kladené na vstup tohoto algoritmu. Platí, že počet stavů se během této transformace může pouze snížit. Opakování symbolů v prefixovém zápisu stromu vede k redukci počtu stavů při determinizaci automatů vytvořených algoritmem 4.1.1–4.1.2. Zde jsou uvedeny automaty, které vznikly transformací automatů uvedených v předchozí sekci.

Příklad 4.3. Uvažujme nedeterministický atributovaný zásobníkový automat $M_{ni}(t_3)$ z příkladu 4.1, jehož přechodový diagram je na obrázku 4.2. Odpovídající deterministický atributovaný zásobníkový automat, který byl zkonstruován algoritmem 2.4.1–2.4.4 z nedeterministického atributovaného zásobníkového automatu $M_{ni}(t_3)$, je deterministický atributovaný zásobníkový automat $M_{di}(t_3) = (\{0,1,2,3,4,5,7\}, \mathcal{A}, \{S,]\}, \{1,2\}, \delta, \delta_T, 0, S, \emptyset^7, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$\begin{aligned}
\delta(0, a2, S) &= (2, SS]) & \delta_T(2, (1,1),]) &= (1, 3, \varepsilon)\\
\delta(0, a0, S) &= (4,]) & \delta_T(2, (1,2),]) &= (1, 1, \varepsilon)\\
\delta(2, a2, S) &= (5, SS]) & \delta_T(3, \emptyset,]) &= (2, 1, \varepsilon)\\
\delta(2, a0, S) &= (3,]) & \delta_T(4, \emptyset,]) &= (1, 2, \varepsilon)\\
\delta(5, a0, S) &= (7,]) & \delta_T(5, (1,1),]) &= (2, 2, \varepsilon)\\
& & \delta_T(7, \emptyset,]) &= (5, 1, \varepsilon)
\end{aligned}$$

Přechodový diagram deterministického atributovaného zásobníkového automatu $M_{di}(t_3)$ je na obrázku 4.4. Označení stavů 2, 3, 4 a 7 je v tomto diagramu následující: $2 = [2,6]$, $3 = [3,8,11]$, $4 = [4,9,12]$ a $7 = [7,10]$.

Tabulka 4.1 znázorňuje posloupnost přechodů provedených deterministickým atributovaným zásobníkovým automatem $M_{di}(t_3)$ pro dva různé náhodně uspořádané podstromy stromu t_3 v prefixovém zápisu.



Obrázek 4.4: Přechodový diagram deterministického atributovaného zásobníkového automatu $M_{di}(t_3)$ přijímajícího všechny podstromy stromu t_3 s prefixovým zápisem $pref(t_3) = a2\ a0\ a2\ a0\ a0$ s libovolným seřazením následníků z příkladu 4.3

Příklad 4.4. Uvažujme nedeterministický atributovaný zásobníkový automat $M_{ni}(t_1)$ z příkladu 4.2, jehož přechodový diagram je na obrázku 4.3. Odpovídající deterministický atributovaný zásobníkový automat, který byl zkonstruován algoritmem 2.4.1–2.4.4 z nedeterministického atributovaného zásobníkového automatu $M_{ni}(t_1)$, je deterministický atributovaný zásobníkový automat $M_{di}(t_1) = (\{0,1,2,3,5,6,7,8,9,10,11,12,13\}, \mathcal{A}, \{S,]\}, \{1, 2\}, \delta, \delta_T, 0, S, \emptyset^{13}, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

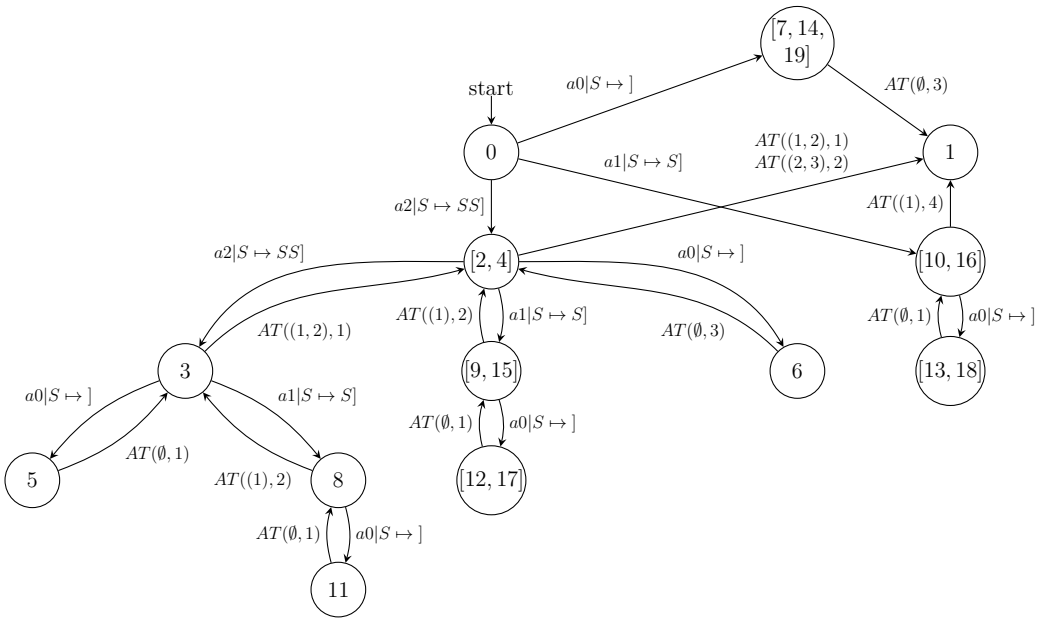
$$\begin{aligned}
\delta(0,a2,S) &= (2,SS]) & \delta_T(2,(1,2),]) &= (1,1,\varepsilon)\\
\delta(0,a1,S) &= (10,S]) & \delta_T(2,(2,3),]) &= (1,2,\varepsilon)\\
\delta(0,a0,S) &= (7,]) & \delta_T(3,(1,2),]) &= (2,1,\varepsilon)\\
\delta(2,a2,S) &= (3,SS]) & \delta_T(5,\emptyset,]) &= (3,1,\varepsilon)\\
\delta(2,a1,S) &= (9,S]) & \delta_T(6,\emptyset,]) &= (2,3,\varepsilon)\\
\delta(2,a0,S) &= (6,]) & \delta_T(7,\emptyset,]) &= (1,3,\varepsilon)\\
\delta(3,a1,S) &= (8,]) & \delta_T(8,(1),]) &= (3,2,\varepsilon)\\
\delta(3,a0,S) &= (5,]) & \delta_T(9,(1),]) &= (2,2,\varepsilon)\\
\delta(8,a0,S) &= (11,]) & \delta_T(10,(1),]) &= (1,4,\varepsilon)\\
\delta(9,a0,S) &= (12,]) & \delta_T(11,\emptyset,]) &= (8,1,\varepsilon)\\
\delta(10,a0,S) &= (13,]) & \delta_T(12,\emptyset,]) &= (9,1,\varepsilon)\\
& & \delta_T(13,\emptyset,]) &= (10,1,\varepsilon)
\end{aligned}$$

stav	vstupní řetězec	zásobník	změny atributů
0	$a2\ a2\ a0\ a0\ a0$	S	
2	$a2\ a0\ a0\ a0$	$S\ S\ ]$	
5	$a0\ a0\ a0$	$S\ S\ ]\ S\ ]$	
7	$a0\ a0$	$]\ S\ ]\ S\ ]$	
5	$a0\ a0$	$S\ ]\ S\ ]$	$s_5 \leftarrow (1)$, $s_7 \leftarrow \emptyset$
7	$a0$	$]\ ]\ S\ ]$	
5	$a0$	$]\ S\ ]$	$s_5 \leftarrow (1,1)$, $s_7 \leftarrow \emptyset$
2	$a0$	$S\ ]$	$s_2 \leftarrow (2)$, $s_5 \leftarrow \emptyset$
3	ε	$]\ ]$	
2	ε	$]$	$s_2 \leftarrow (1,2)$, $s_3 \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (1)$, $s_2 \leftarrow \emptyset$
stav	vstupní řetězec	zásobník	změny atributů
0	$a2\ a0\ a0$	S	
2	$a0\ a0$	$S\ S\ ]$	
3	$a0$	$]\ S\ ]$	
2	$a0$	$S\ ]$	$s_2 \leftarrow (1)$, $s_3 \leftarrow \emptyset$
3	ε	$]\ ]$	
2	ε	$]$	$s_2 \leftarrow (1,1)$, $s_3 \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (3)$, $s_2 \leftarrow \emptyset$

Tabulka 4.1: Posloupnost přechodů deterministického atributovaného zásobníkového automatu $M_{di}(t_3)$ z příkladu 4.3 pro dva různé náhodně uspořádané podstromy stromu t_3 v prefixovém zápisu

Přechodový diagram deterministického atributovaného zásobníkového automatu $M_{di}(t_1)$ je na obrázku 4.5. Označení stavů 2, 7, 9, 10, 12 a 13 je v tomto diagramu následující: $2 = [2,4]$, $7 = [7,14,19]$, $9 = [9,15]$, $10 = [10,16]$, $12 = [12,17]$ a $13 = [13,18]$.

Tabulka 4.2 znázorňuje posloupnost přechodů provedených deterministickým atributovaným zásobníkovým automatem $M_{di}(t_1)$ pro dva různé náhodně uspořádané podstromy stromu t_1 v prefixovém zápisu.



Obrázek 4.5: Přechodový diagram deterministického atributovaného zásobníkového automatu $M_{di}(t_1)$ přijímajícího všechny podstromy stromu t_1 s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 4.4

4.2 Druhá varianta automatu

4.2.1 Algoritmus konstrukce automatu

Při konstrukci zásobníkového automatu přijímajícího všechny podstromy určitého stromu s pevně daným uspořádáním následníků se postupuje tak, že se nedříve sestrojí automat přijímající daný strom a poté se vytvoří přechody z počátečního stavu do všech stavů odpovídajících uzlům stromu kromě stavu odpovídajícího kořeni stromu [5]. Uvedeme konstrukci nedeterministického atributovaného zásobníkového automatu přijímajícího všechny podstromy určitého stromu s libovolným seřazním následníků, jejíž princip je tentýž.

Stejně jako algoritmus 3.1 se i algoritmus 4.2 skládá ze tří kroků; nejprve se pomocí algoritmu 2.2 zkonstruuje nedeterministický atributovaný zásobníkový automat přijímající daný strom, poté se tento automat zdeterminizuje pomocí algoritmu 2.4.1–2.4.4 a nakonec se vytvoří přechody z počátečního stavu do všech stavů automatu $q \in Q$, $q > 2$. Přidáním těchto přechodů se automat stane nedeterministickým. Dále v této kapitole je popsáno, jak zkonstruovat odpovídající deterministický rozšířený atri-

stav	vstupní řetězec	zásobník	změny atributů
0	$a2\ a1\ a0\ a0$	S	
2	$a1\ a0\ a0$	$S\ S\ ]$	
9	$a0\ a0$	$S\ ]\ S\ ]$	
12	$a0$	$]\ ]\ S\ ]$	
9	$a0$	$]\ S\ ]$	$s_9 \leftarrow (1)$, $s_{12} \leftarrow \emptyset$
2	$a0$	$S\ ]$	$s_2 \leftarrow (2)$, $s_9 \leftarrow \emptyset$
6	ε	$]\ ]$	
2	ε	$]$	$s_2 \leftarrow (2,3)$, $s_6 \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (2)$, $s_2 \leftarrow \emptyset$

stav	vstupní řetězec	zásobník	změny atributů
0	$a1\ a0$	S	
10	$a0$	$S\ ]$	
13	ε	$]\ ]$	
10	ε	$]$	$s_{10} \leftarrow (1)$, $s_{13} \leftarrow \emptyset$
1	ε	ε	$s_1 \leftarrow (4)$, $s_{10} \leftarrow \emptyset$

Tabulka 4.2: Posloupnost přechodů deterministického atributovaného zásobníkového automatu $M_{di}(t_1)$ z příkladu 4.4 pro dva různé náhodně uspořádané podstromy stromu t_1 v prefixovém zápisu

butovaný zásobníkový automat. Přijetí podstromu se vyznačuje atributem stavu, ve kterém se automat nachází při vyprázdnění zásobníku, nastaveným na jednoprvkovou uspořádanou posloupnost obsahující celočíselný identifikátor tohoto podstromu.

Opět platí, že důvodem pro aplikaci algoritmu 2.4.1–2.4.4 je potřeba nalezení stejných podstromů, jejichž kořeny jsou přímí následníci určitého uzlu, protože takové podstromy musí mít přiřazen stejný identifikátor pro zařazování do posloupnosti uchovávané v atributu stavu odpovídajícímu zmíněnému společnému uzlu.

Algoritmus 4.2 Konstrukce nedeterministického atributovaného zásobníkového automatu přijímajícího všechny podstromy stromu t s prefixovým zápisem $pref(t)$ s libovolným seřazením následníků.

Vstup: Strom t nad ohodnocenou abecedou $\mathcal{A}$; prefixový zápis $pref(t) = a_1 a_2 \dots a_n$, $n \geq 1$.

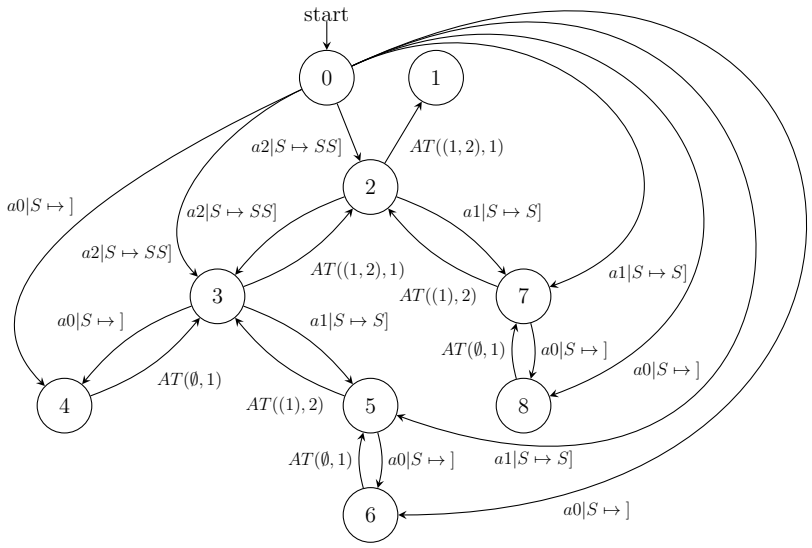
Výstup: Nedeterministický atributovaný zásobníkový automat $M_{nia}(t) = (Q, \mathcal{A}, \{S,]\}, N, \delta, \delta_T, 0, S, \emptyset^{|Q|}, \emptyset)$.

1: Vytvořit atributovaný zásobníkový automat $M(t)$ pomocí algoritmu 2.2.
2: Transformovat atributovaný zásobníkový automat $M(t)$ pomocí algoritmu 2.4.1–2.4.4 na odpovídající deterministický atributovaný zásobníkový automat a tento automat označit $M_{nia}(t)$.
3: **for all** $q \in Q$, $q > 2$ **do**
4: $\quad |\{a : (q, \beta) \in \delta(p, a, \alpha)\}| = 1$, $\beta \in G^*$, $p \in Q$, $\alpha \in G$
5: $\quad$ Vytvořit přechod $\delta(0, a, S) = (q, S^{arity(a)}])$, kde $S^0 = \varepsilon$.
6: **end for**

Příklad 4.5. Uvažujme strom t_1, který je na obrázku 2.1, s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$. Atributovaný zásobníkový automat přijímající všechny podstromy stromu t_1 s libovolným seřazením následníků, který byl zkonstruován algoritmem 4.2, je nedeterministický atributovaný zásobníkový automat $M_{nia}(t_1) = (\{0,1,2,3,4,5,6,7,8\}, \mathcal{A}, \{S,]\}, \{1,2\}, \delta, \delta_T, 0, S, \emptyset^9, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$\begin{aligned}
&\delta(0, a2, S) = (2, SS]) && \delta_T(2, (1,2),]) = (1, 1, \varepsilon)\\
&\delta(0, a2, S) = (3, SS]) && \delta_T(3, (1,2),]) = (2, 1, \varepsilon)\\
&\delta(0, a1, S) = (5, S]) && \delta_T(4, \emptyset,]) = (3, 1, \varepsilon)\\
&\delta(0, a1, S) = (7, S]) && \delta_T(5, (1),]) = (3, 2, \varepsilon)\\
&\delta(0, a0, S) = (4,]) && \delta_T(6, \emptyset,]) = (5, 1, \varepsilon)\\
&\delta(0, a0, S) = (6,]) && \delta_T(7, (1),]) = (2, 2, \varepsilon)\\
&\delta(0, a0, S) = (8,]) && \delta_T(8, \emptyset,]) = (7, 1, \varepsilon)\\
&\delta(2, a2, S) = (3, SS]) && \delta(3, a0, S) = (4,])\\
&\delta(2, a1, S) = (7, S]) && \delta(5, a0, S) = (6,])\\
&\delta(3, a1, S) = (5, S]) && \delta(7, a0, S) = (8,])
\end{aligned}$$

Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{nia}(t_1)$ je na obrázku 4.6.



Obrázek 4.6: Přechodový diagram nedeterministického atributovaného zásobníkového automatu $M_{nia}(t_1)$ přijímajícího všechny podstromy stromu t_1 s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 4.5

4.2.2 Algoritmus determinizace automatu

Následuje výpis algoritmu, který je velice podobný algoritmu 3.2.1–3.2.4 pro determinizaci automatů pro vyhledávání. Algoritmus 4.3.1–4.3.4, jehož výstupem je deterministický rozšířený atributovaný zásobníkový automat přijímající podstromy, také pracuje se dvěma frontami stavů a skončí, když jsou obě fronty prázdné. Stav se zařadí do fronty *queuea*, pokud je potřeba zkonstruovat přechody vedoucí z tohoto stavu odpovídající zobrazení δ, naopak stav se zařadí do fronty *queueb*, pokud je potřeba zkonstruovat přechody vedoucí z tohoto stavu odpovídající zobrazení δ_T.

Cyklus obsluhující frontu *queuea* (část 2/4 výpisu algoritmu) pracuje na stejném principu jako odpovídající cyklus algoritmu 3.2.1–3.2.4. Rozdílem je, že zde není jisté, že stav q'' vyjde pro všechna $a \in \mathcal{A}$ jako neprázdná množina, je proto nutné provést tento test. Další odlišností je nemožnost

Algoritmus 4.3.1 Transformace nedeterministického atributovaného zásobníkového automatu přijímajícího podstromy na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 1/4 – vstup, výstup).

Vstup: Nedeterministický atributovaný zásobníkový automat $M(t) = (Q, \mathcal{A}, G, N, \delta, \delta_T, 0, Z_0, s_0, \emptyset)$, kde $\{0, 1, 2\} \subset Q$ a $G = \{S,]\}$. Automat musí splňovat následující podmínky:

1. Uspořádání stavů automatu vzhledem k jejich číselnému označení je takové, že:
   a) pokud $\delta(q, a, \alpha) = (p, \beta)$, pak platí $q < p$, kde $q, p \in Q$, $a \in \mathcal{A}$, $\alpha \in G$ a $\beta \in G^*$,
   b) pokud $\delta_T(q, s, \alpha) = (p, k, \beta)$, pak platí $q > p$, kde $q, p \in Q$, $s \in seq(N)$, $\alpha \in G$, $k \in N$ a $\beta \in G^*$.
2. Pro stav $q = 0$ neexistuje přechod, který lze zapsat $\delta_T(q, s, \alpha) = (p, k, \beta)$ nebo $\delta_T(p, s, \alpha) = (q, k, \beta)$, kde $p \in Q$, $s \in seq(N)$, $\alpha \in G$, $k \in N$ a $\beta \in G^*$.
3. Pro stav $q = 1$ neexistuje přechod, který lze zapsat $\delta(q, a, \alpha) = (p, \beta)$, $\delta(p, a, \alpha) = (q, \beta)$ nebo $\delta_T(q, s, \alpha) = (p, k, \beta)$, kde $p \in Q$, $a \in \mathcal{A}$, $\alpha \in G$, $\beta \in G^*$, $s \in seq(N)$ a $k \in N$.
4. Pro stav $q = 2$ existuje právě jeden přechod, který lze zapsat $\delta(0, a, S) = (q, S^{arity(a)}])$, kde $a \in \mathcal{A}$ a $S^0 = \varepsilon$.
5. Do každého stavu $q \in Q$, $q > 2$, vedou právě dva přechody, z nichž jeden lze zapsat $\delta(p, a, S) = (q, S^{arity(a)}])$, kde $p \in Q$, $2 \leq p < q$, $a \in \mathcal{A}$, a druhý lze zapsat $\delta(0, a, S) = (q, S^{arity(a)}])$, kde $a \in \mathcal{A}$, $S^0 = \varepsilon$.
6. Pro každý stav $q \in Q$, $q \neq 0$, $q \neq 1$, existuje právě jeden stav $p \in Q$, do kterého vede alespoň jeden přechod, který lze zapsat $\delta_T(q, s,]) = (p, k, \varepsilon)$, kde $s \in seq(N)$ a $k \in N$.
7. Neexistuje přechod, který lze zapsat $(q, \varepsilon, \alpha) = (p, \beta)$, kde $q, p \in Q$, $\alpha \in G$ a $\beta \in G^*$.

Výstup: Odpovídající deterministický rozšířený atributovaný zásobníkový automat $M_d(t) = (Q', Q, \mathcal{A}, G, N, D, \delta', \delta'_T, q_I, Z_0, s_0, \varepsilon)$.

Algoritmus 4.3.2 Transformace nedeterministického atributovaného zásobníkového automatu přijímajícího podstromy na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 2/4 – vlastní algoritmus).

```
 1: q_I ← [0], Q' ← {[0]}
 2: Q'_enqb ← ∅
 3: position[0 .. MAX(Q)] ← NEW-ARRAY
 4: queuea ← NEW-QUEUE, queueb ← NEW-QUEUE
 5: ENQUEUE(queuea, q_I)
 6: while not EMPTY(queuea) or not EMPTY(queueb) do
 7:     while not EMPTY(queuea) do
 8:         q' ← DEQUEUE(queuea)
 9:         for all a ∈ A do
10:             q'' ← ∅, i ← 1
11:             for all p ∈ q' in ascending order do
12:                 for all (q, β) ∈ δ(p, a, α), q ∈ Q, β ∈ G*, α ∈ G do
13:                     q'' ← q'' ∪ {q}
14:                     α' ← α, β' ← β
15:                     position[q] ← i
16:                 end for
17:                 i ← i + 1
18:             end for
19:             if q'' ≠ ∅ then
20:                 i ← 1
21:                 for all q ∈ q'' in ascending order do
22:                     k_i ← position[q]
23:                     i ← i + 1
24:                 end for
25:                 if not q'' ∈ Q' then
26:                     Q' ← Q' ∪ {q''}, D ← D ∪ {|q''|}
27:                     if arity(a) > 0 then
28:                         ENQUEUE(queuea, q'')
29:                     else
30:                         ENQUEUE(queueb, q'')
31:                     end if
32:                 end if
33:                 Vytvořit přechod δ'(q', a, α') = (q'', β', (k_1, k_2, ..., k_|q''|)).
34:             end if
35:         end for
36:     end while
```

Algoritmus 4.3.3 Transformace nedeterministického atributovaného zásobníkového automatu přijímajícího podstromy na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 3/4 – vlastní algoritmus, pokračování).

37: **while not** Empty($queueb$) **do**
38: $q' \leftarrow$ Dequeue($queueb$)
39: $q \in Q$, $\alpha \in G$, $\beta \in G^*$
40: $V_{base} \leftarrow \{(p, s) : p \in q' \wedge \delta_T(p, s,]) \neq \emptyset\}$
41: $|\{a : (p, \beta) \in \delta(q, a, \alpha)\}| = 1$, $p \in q'$
42: **if** $arity(a) > 1$ **or** $|V_{base}| > |q'|$ **then**
43: $V \leftarrow \mathcal{P}(V_{base}) \setminus \{\emptyset\}$
44: **else**
45: $V \leftarrow \{V_{base}\}$
46: **end if**
47: **for all** $V_{sub} \in V$ **do**
48: $q'' \leftarrow \emptyset$, $i \leftarrow 1$
49: **for all** $p \in q'$ **in ascending order do**
50: **if** $(p, s) \in V_{sub}$, $s \in seq(N)$ **then**
51: $|\{q : (q, l, \varepsilon) \in \delta_T(p, s,])\}| = 1$, $l \in N$
52: $q'' \leftarrow q'' \cup \{q\}$
53: $position[q] \leftarrow i$
54: **end if**
55: $i \leftarrow i + 1$
56: **end for**

jednoduché konstrukce množiny B jako vedlejšího efektu tohoto cyklu. Zde by ovšem použití této množiny nemělo velký význam, protože zde vznikají d-podmnožiny, kde jeden ze stavů této d-podmnožiny odpovídá uzlu, který je součástí podstromu, jehož kořenu odpovídá jiný stav této d-podmnožiny, v mnohem větší míře. Takové d-podmnožiny zde nejsou vždy podmnožiny množiny B, jak byla definována u algoritmu 3.2.1–3.2.4.

Rozhodnutí, jak vytvořit množinu V množin dvojic $(p, s) \in Q \times seq(N)$ v cyklu obsluhujícím frontu *queueb*, je zde tedy oproti algoritmu 3.2.1–3.2.4 omezeno a opět dochází k vytváření potenčních množin přechodů pro splnění třetí podmínky z definice deterministického rozšířeného atributovaného zásobníkového automatu. Je možné, že určité předzpracování stromu (např. výpočet repetic podstromů) a vytvoření vhodných pomocných datových struktur by umožnilo identifikovat potřebné přechody a stavy a vyhnout se tak vytváření potenční množiny.

Algoritmus 4.3.4 Transformace nedeterministického atributovaného zásobníkového automatu přijímajícího podstromy na odpovídající deterministický rozšířený atributovaný zásobníkový automat (část 4/4 – vlastní algoritmus, pokračování).

57: $P \leftarrow \emptyset$, $R \leftarrow \emptyset$, $i \leftarrow 1$
58: **for all** $q \in q''$ **in ascending order do**
59: **for all** $(p, s) \in V_{sub}$ **do**
60: **if** $(q, l, \varepsilon) \in \delta_T(p, s,])$, $l \in N$ **then**
61: $P \leftarrow P \cup \{(p, i, s)\}$
62: $R \leftarrow R \cup \{(q, i, l)\}$
63: **end if**
64: **end for**
65: $k_i \leftarrow position[q]$
66: $i \leftarrow i + 1$
67: **end for**
68: $p \in Q$, $\alpha \in G$, $\beta \in G^*$
69: **if** $1 \in q''$ **or** $|\{a : (q, \beta) \in \delta(p, a, \alpha)\}| > 1$, $q \in q''$ **then**
70: $Q' \leftarrow Q' \cup \{q''\}$
71: **else**
72: **if not** $q'' \in Q'$ **then**
73: $Q' \leftarrow Q' \cup \{q''\}$
74: ENQUEUE$(queuea, q'')$
75: **end if**
76: **if not** $q'' \in Q'_{enqb}$ **then**
77: $Q'_{enqb} \leftarrow Q'_{enqb} \cup \{q''\}$
78: ENQUEUE$(queueb, q'')$
79: **end if**
80: **end if**
81: Vytvořit přechod $\delta'_T(q', P,]) = (q'', R, (k_1, k_2, \ldots, k_{|q''|}), \varepsilon)$.
82: **end for**
83: **end while**
84: **end while**

Příklad 4.6. Uvažujme nedeterministický atributovaný zásobníkový automat $M_{nia}(t_1)$ z příkladu 4.5, jehož přechodový diagram je na obrázku 4.6. Odpovídající deterministický rozšířený atributovaný zásobníkový automat, který byl zkonstruován algoritmem 4.3.1–4.3.4 z nedeterministického atributovaného zásobníkového automatu $M_{nia}(t_1)$, je deterministický rozšířený atributovaný zásobníkový automat $M_{dia}(t_1) = (\{[0], [1], [1,2], [2], [2,3], [3], [3,5,7], [4], [4,6,8], [5], [5,7], [6], [6,8], [7], [8]\}, \{0,1,2,3,4,5,6,7,8\}, \mathcal{A}, \{S,]\}, \{1,2\}, \{1,2,3\}, \delta, \delta_T, [0], S, \emptyset^{15}, \emptyset)$, kde zobrazení δ a δ_T jsou takovéto množiny přechodů:

$$
\begin{aligned}
\delta([0], a2, S) &= ([2,3], SS], (1,1)) \\
\delta([0], a1, S) &= ([5,7], S], (1,1)) \\
\delta([0], a0, S) &= ([4,6,8],], (1,1,1)) \\
\delta([2], a2, S) &= ([3], SS], (1)) \\
\delta([2], a1, S) &= ([7], S], (1)) \\
\delta([2,3], a2, S) &= ([3], SS], (1)) \\
\delta([2,3], a1, S) &= ([5,7], S], (2,1)) \\
\delta([2,3], a0, S) &= ([4],], (2)) \\
\delta([3], a1, S) &= ([5], S], (1)) \\
\delta([3], a0, S) &= ([4],], (1)) \\
\delta([5], a0, S) &= ([6],], (1)) \\
\delta([5,7], a0, S) &= ([6,8],], (1,1)) \\
\delta([7], a0, S) &= ([8],], (1)) \\
\delta_T([2], \{(2,1,(1,2))\},]) &= ([1], \{(1,1,1)\}, \varepsilon, (1)) \\
\delta_T([2,3], \{(2,1,(1,2))\},]) &= ([1], \{(1,1,1)\}, \varepsilon, (1)) \\
\delta_T([2,3], \{(2,1,(1,2)),(3,2,(1,2))\},]) &= ([1,2], \{(1,1,1),(2,2,1)\}, \varepsilon, (1,2)) \\
\delta_T([2,3], \{(3,2,(1,2))\},]) &= ([2], \{(2,1,1)\}, \varepsilon, (2)) \\
\delta_T([3], \{(3,1,(1,2))\},]) &= ([2], \{(2,1,1)\}, \varepsilon, (1)) \\
\delta_T([4], \{(4,1,\emptyset)\},]) &= ([3], \{(3,1,1)\}, \varepsilon, (1)) \\
\delta_T([4,6,8], \{(4,1,\emptyset),(6,2,\emptyset),(8,3,\emptyset)\},]) &= ([3,5,7], \{(3,1,1),(5,2,1),(7,3,1)\}, \\
&\qquad \varepsilon, (1,2,3)) \\
\delta_T([5], \{(5,1,(1))\},]) &= ([3], \{(3,1,2)\}, \varepsilon, (1)) \\
\delta_T([5,7], \{(5,2,(1)),(7,1,(1))\},]) &= ([2,3], \{(2,1,2),(3,2,2)\}, \varepsilon, (2,1)) \\
\delta_T([6], \{(6,1,\emptyset)\},]) &= ([5], \{(5,1,1)\}, \varepsilon, (1)) \\
\delta_T([6,8], \{(6,1,\emptyset),(8,2,\emptyset)\},]) &= ([5,7], \{(5,1,1),(7,2,1)\}, \varepsilon, (1,2))
\end{aligned}
$$

$$\delta_T([7],\{(7,1,(1))\},]) = ([2],\{(2,1,2)\},\varepsilon,(1))$$
$$\delta_T([8],\{(8,1,\emptyset)\},]) = ([7],\{(7,1,1)\},\varepsilon,(1))$$

Tabulka 4.3 je tabulka přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{dia}(t_1)$, která pouze znázorňuje množiny stavů, do kterých vedou přechody odpovídající zobrazení δ pro každý stav a každý symbol vstupní abecedy, resp. přechody odpovídající zobrazení δ_T pro každý stav automatu.

Tabulka 4.4 znázorňuje posloupnost přechodů provedených deterministickým rozšířeným atributovaným zásobníkovým automatem $M_{dia}(t_1)$ pro podstromy stromu t_1 v prefixovém zápisu, které byly použity v tabulce 4.2.

Přechody $\delta_T([2,3],\{(2,1,(1,2))\},]) = ([1],\{(1,1,1)\},\varepsilon,(1))$ a $\delta_T([2,3],\{(2,1,(1,2)),(3,2,(1,2))\},]) = ([1,2],\{(1,1,1),(2,2,1)\},\varepsilon,(1,2))$ jsou z hlediska funkčnosti automatu zbytečné. Navíc stav $[1,2]$ nemusí být v automatu vůbec obsažen. Bylo by možné rozšířit algoritmus 4.3.1–4.3.4 tak, že by měl k dispozici vhodné datové struktury, v tomto konkrétním příkladě uchovávající informaci, že stav 3 odpovídá uzlu, který je následníkem uzlu, jemuž odpovídá stav 2, a tuto informaci využil k vynechání vytváření těchto přechodů a stavu $[1,2]$.

	stav	$a2$	$a1$	$a0$	δ_T
$\rightarrow$	$[0]$	$[2,3]$	$[5,7]$	$[4,6,8]$	–
	$[2,3]$	$[3]$	$[5,7]$	$[4]$	$[1],[2],[1,2]$
	$[5,7]$	–	–	$[6,8]$	$[2,3]$
	$[3]$	–	$[5]$	$[4]$	$[2]$
	$[5]$	–	–	$[6]$	$[3]$
	$[4,6,8]$	–	–	–	$[3,5,7]$
	$[4]$	–	–	–	$[3]$
	$[6,8]$	–	–	–	$[5,7]$
	$[6]$	–	–	–	$[5]$
	$[2]$	$[3]$	$[7]$	–	$[1]$
	$[7]$	–	–	$[8]$	$[2]$
	$[8]$	–	–	–	$[7]$
	$[3,5,7]$	–	–	–	–
	$[1]$	–	–	–	–
	$[1,2]$	–	–	–	–

Tabulka 4.3: Tabulka přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{dia}(t_1)$ přijímajícího všechny podstromy stromu t_1 s prefixovým zápisem $pref(t_1) = a2\ a2\ a0\ a1\ a0\ a1\ a0$ s libovolným seřazením následníků z příkladu 4.6

stav	vstupní řetězec/změny atributů	zásobník
$[0]$	$a2\ a1\ a0\ a0$	S
$[2,3]$	$a1\ a0\ a0$	$S\ S\ ]$
	$s \leftarrow (s^{[1]}, s^{[1]})$	
$[5,7]$	$a0\ a0$	$S\ ]\ S\ ]$
	$s \leftarrow (s^{[2]}, s^{[1]})$	
$[6,8]$	$a0$	$]\ ]\ S\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]})$	
$[5,7]$	$a0$	$]\ S\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]})$,	
	$s_5^{[1]} \leftarrow (1)$, $s_6^{[1]} \leftarrow \emptyset$, $s_7^{[2]} \leftarrow (1)$, $s_8^{[2]} \leftarrow \emptyset$	
$[2,3]$	$a0$	$S\ ]$
	$s \leftarrow (s^{[2]}, s^{[1]})$,	
	$s_2^{[1]} \leftarrow (2)$, $s_7^{[1]} \leftarrow \emptyset$, $s_3^{[2]} \leftarrow (2)$, $s_5^{[2]} \leftarrow \emptyset$	
$[4]$	ε	$]\ ]$
	$s \leftarrow (s^{[2]})$	
$[3]$	ε	$]$
	$s \leftarrow (s^{[1]})$, $s_3^{[1]} \leftarrow (1,2)$, $s_4^{[1]} \leftarrow \emptyset$	
$[2]$	ε	ε
	$s \leftarrow (s^{[1]})$, $s_2^{[1]} \leftarrow (1)$, $s_3^{[1]} \leftarrow \emptyset$	

stav	vstupní řetězec/změny atributů	zásobník
$[0]$	$a1\ a0$	S
$[5,7]$	$a0$	$S\ ]$
	$s \leftarrow (s^{[1]}, s^{[1]})$	
$[6,8]$	ε	$]\ ]$
	$s \leftarrow (s^{[1]}, s^{[2]})$	
$[5,7]$	ε	$]$
	$s \leftarrow (s^{[1]}, s^{[2]})$,	
	$s_5^{[1]} \leftarrow (1)$, $s_6^{[1]} \leftarrow \emptyset$, $s_7^{[2]} \leftarrow (1)$, $s_8^{[2]} \leftarrow \emptyset$	
$[2,3]$	ε	ε
	$s \leftarrow (s^{[2]}, s^{[1]})$,	
	$s_2^{[1]} \leftarrow (2)$, $s_7^{[1]} \leftarrow \emptyset$, $s_3^{[2]} \leftarrow (2)$, $s_5^{[2]} \leftarrow \emptyset$	

Tabulka 4.4: Posloupnost přechodů deterministického rozšířeného atributovaného zásobníkového automatu $M_{dia}(t_1)$ z příkladu 4.6 pro podstromy stromu t_1 v prefixovém zápisu, které byly použity v tabulce 4.2

Závěr

Byla popsána konstrukce deterministického atributovaného zásobníkového automatu přijímajícího neseřazené stromy v prefixovém zápisu. Princip této konstrukce byl převzat z bakalářské práce [6], avšak byl zaveden nový formalismus atributovaného zásobníkového automatu.

Tento automat dále posloužil jako základ pro konstrukci deterministických rozšířených atributovaných zásobníkových automatů pro vyhledávání a pro indexování neseřazených stromů. Uvedená definice deterministického rozšířeného atributovaného zásobníkového automatu vyžaduje konstrukci potenční množiny z množiny přechodů vedoucích ze stavů tvořících d-podmnožiny stavů výsledného automatu. Představené automaty pro vyhledávání a pro indexování stromů s libovolným seřazením následníků obecně nevyžadují zdaleka všechny takto vzniklé přechody a stavy a nabízí se tuto definici zpřesnit a tomuto vytváření potenční množiny při determinizaci se vyhnout. Je možné, že předchozí předzpracování stromu a vytvoření vhodných datových struktur obsahujících např. repetice podstromů, by umožnilo identifikovat potřebné přechody a stavy a vyhnout se tak vytváření této potenční množiny.

Byl také uveden alternativní postup pro vytvoření deterministického atributovaného zásobníkového automatu přijímajícího všechny podstromy určitého stromu s libovolným seřazením následníků. Výhodou tohoto postupu je, že nevyžaduje definici rozšířeného atributovaného zásobníkového automatu, naopak nevýhodou je jeho kvadratická časová a paměťová asymptotická složitost vzhledem k počtu uzlů stromu.

Předmětem navazující snahy by mohlo být dokázání správnosti a asymptotické složitosti zde uvedených algoritmů, a také správnosti výsledných automatů. Dále by mělo být možné rozšířit zde uvedené algoritmy a automaty tak, aby umožňovaly přibližné vyhledávání, resp. přibližné indexování

stromů. Je otázkou, jak by bylo potřeba zde uvedené automaty rozšířit, aby umožňovaly přijímání vzorů stromů obsahující symboly S. V případě neseřazených stromů není obecně možnost jednoznačně určit, který podstrom konkrétní symbol S ve vstupním řetězci zastupuje.

Literatura

[1] Arbology. březen 2013. Dostupné z: `http://www.arbology.org/`

[2] Christou, M.; Crochemore, M.; Flouri, T.; aj.: Computing all subtree repeats in ordered trees. *Information Processing Letters*, červen 2012.

[3] Flouri, T.; Kobert, K.; Pissis, S. P.; aj.: A simple method for computing all subtree repeats in unordered trees in linear time. *Festschrift for Bořivoj Melichar, Czech Technical University in Prague*, srpen 2012.

[4] Melichar, B.; Holub, J.; Polcar, T.: *Text Searching Algorithms, Volume I: Forward String Matching*. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science and Engineering, listopad 2005.

[5] Melichar, B.; Janoušek, J.; Flouri, T.; aj.: *Introduction to Arbology*. Czech Technical University in Prague, Faculty of Information Technology, Department of Theoretical Computer Science, říjen 2010.

[6] Sobota, V.: Zásobníkový automat pro neseřazené stromy. Bakalářská práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů, červen 2009.

Příloha **A**

Seznam použitých zkratek

DAG Directed acyclic graph

NFA Nondeterministic finite automaton

DFA Deterministic finite automaton

PDA Pushdown automaton

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `src` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Sobota_Vojtech_2013.pdf` ..... text práce ve formátu PDF