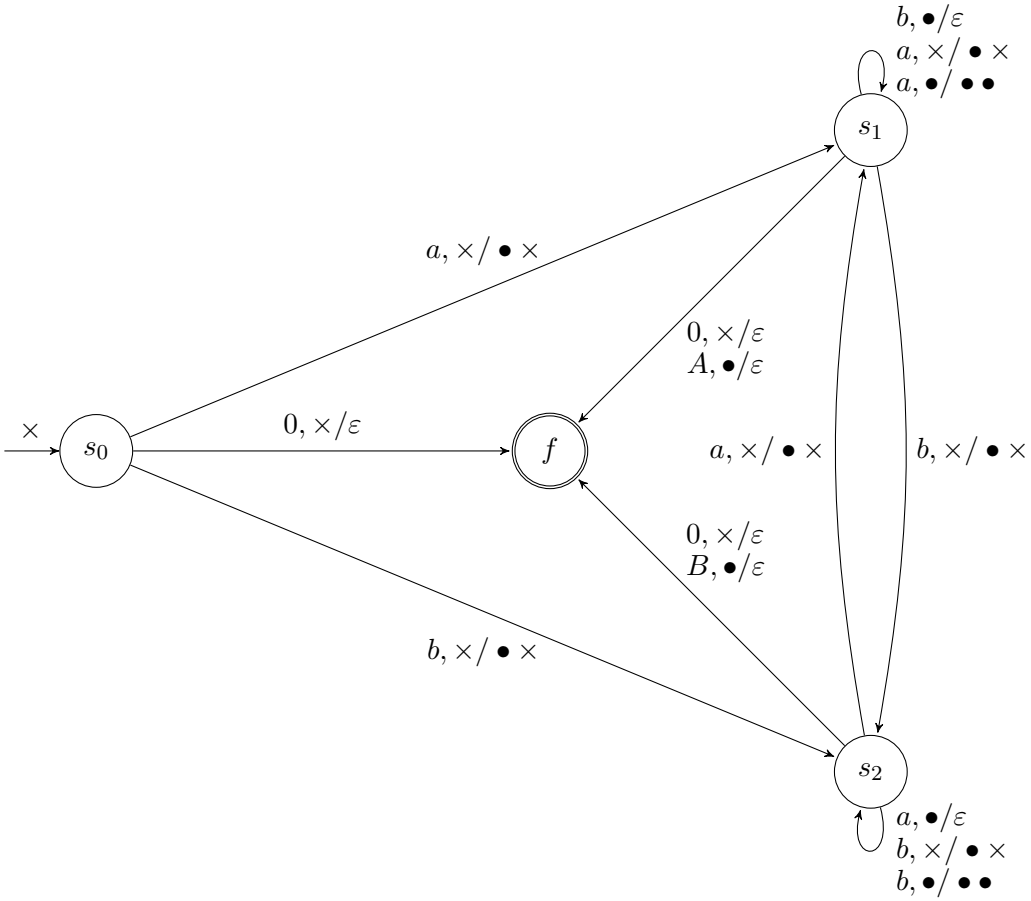
Teoretická informatika – 2. domácí úloha

Ondřej Ondryáš (xondry02), 9. listopadu 2022

Příklad 1



Automat přijímá přechodem do cílového stavu f. Bylo by možné v tomto stavu vyprázdnit zásobník, tento krok jsem ale pro přehlednost vynechal. U přechodů jsou zásobníkové symboly uvedeny tak, že nejlevější symbol bude po provedení přechodu na vrcholu zásobníku. V zápisu konfigurací níže je zápis obdobný, tedy dno zásobníku je nejpravější symbol, vrchol je nejlevější.

$$(s_0, abaabA, \times) \vdash (s_1, baabA, \bullet\times) \vdash (s_1, aabA, \times) \vdash (s_1, abA, \bullet\times) \vdash (s_1, bA, \bullet\bullet\times) \vdash (s_1, A, \bullet\times) \vdash (f, \varepsilon, \times)$$

Příklad 2

Uvažovaný jazyk L je bezkontextový, neboť je zadaný pomocí bezkontextové gramatiky. Bezkontextové jazyky nejsou uzavřeny vůči doplňku, tedy $\exists L \in \mathcal{L}_2 : \overline{L} \notin \mathcal{L}_2$.

Uvažme, že L je právě takový jazyk, pro který $\overline{L} \notin \mathcal{L}_2$, a pak předpokládejme, že L' je bezkontextový, a tedy je možné zkonstruovat gramatiku $G' = (N, \Sigma \cup \{0,1\}, P, S)$ takovou, že $L(G') = L'$. Bez újmy na obecnosti dále předpokládejme, že G' je v Chomského normální formě, neobsahuje nedostupné ani zbytečné symboly.

Všechny řetězce v jazyce L' končí buď symbolem 0, nebo 1, zároveň se ani jeden z těchto symbolů nemůže v řetězci z L' vyskytovat na jiné než poslední pozici. Zřejmě musí v G' existovat pravidla generující symboly 0 a 1, pojmenujme neterminály na jejich levých stranách X a Y: $(X \to 0) \in P$, $(Y \to 1) \in P$.

Pro každý řetězec terminálů, který G' generuje, je tak možné zapsat derivaci, která jej generuje, jedním ze způsobů:

$$S \Rightarrow^* \alpha X \Rightarrow \alpha 0 \qquad (\alpha \in \Sigma^*) \tag{1}$$
$$S \Rightarrow^* \beta Y \Rightarrow \beta 1 \qquad (\beta \in \Sigma^*) \tag{2}$$

Existence jedné z těchto derivací zároveň z definice jazyka L' vylučuje existenci druhé:

$$\forall \alpha \in \Sigma^* : (S \Rightarrow^* \alpha X \Rightarrow \alpha 0) \to \neg(S \Rightarrow^* \alpha 1)$$
$$\forall \beta \in \Sigma^* : (S \Rightarrow^* \beta Y \Rightarrow \beta 1) \to \neg(S \Rightarrow^* \beta 0)$$

Dále G' nemůže vygenerovat větnou formu $\gamma_1 X \gamma_2$ nebo $\gamma_1 Y \gamma_2$, neboť by při G' v CNF v obou případech platilo, že $\gamma_2 \Rightarrow^* \delta$, kde $|\delta| > 0$, ale pak by se symbol 0 nebo 1 vyskytnul na jiné než poslední pozici řetězce. Pro případ (1) zjevně platí $\alpha \in \overline{L}$, pro případ (2) zjevně platí $\beta \in L$.

Sestrojme nyní gramatiku $G'' = (N, \Sigma \cup \{0,1\}, P', S)$, kde P' vznikne z P nahrazením pravidla $Y \to 1$ pravidlem $Y \to Y$. Pokud byla G' bezkontextová, G'' bude také bezkontextová, neboť do ní nebylo přidáno žádné pravidlo, které by tuto vlastnost porušovalo.

Nahlédněme, že všechny derivace generující řetězce terminálů v G'' nyní odpovídají pouze tvaru (1). Gramatika G'' tedy generuje jazyk $\overline{L} \cdot \{0\}$. Evidentně $\overline{L} \notin \mathcal{L}_2 \Rightarrow (\overline{L} \cdot \{0\}) \notin \mathcal{L}_2$.

Došli jsme tedy ke sporu: G'' má být bezkontextová, ale jazyk, který generuje, bezkontextový není. Nemůže tedy platit původní předpoklad, že L' je bezkontextový, a tedy *nelze* vytvořit algoritmus, který by sestavil bezkontextovou gramatiku G' takovou, že $L(G') = L'$. □

Příklad 3

Zavedeme pomocné množiny neterminálů generujících řetězec terminálů, řetězec terminálů začínající symbolem a a řetězec terminálů končící symbolem a:

$$N_t = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ w\}$$
$$N_{a*} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ aw\}$$
$$N_{*a} = \{A \in N \mid \exists w \in \Sigma^* : A \Rightarrow_G^+ wa\}$$

Definujeme algoritmy pro výpočet uvedených množin:

Algoritmus 1: Výpočet množiny neterminálů generujících řetězec terminálů.
Vstup: Gramatika $G = (N, \Sigma, P, S)$.
Výstup: Množina N_t.
Metoda:

1. $N_0 = \emptyset$, $i = 1$
2. $N_i = \{A \in N \mid \exists \alpha : (A \to \alpha) \in P \ \wedge \ \alpha \in (N_{i-1} \cup \Sigma)^*\}$
3. Pokud $N_i \neq N_{i-1}$, $i := i + 1$ a vrať se k 2., jinak $N_t := N_i$ a skonči.

Algoritmus 2: Výpočet množiny neterminálů generujících řetězec terminálů začínající symbolem a.
Vstup: Gramatika $G = (N, \Sigma, P, S)$.
Výstup: Množina N_{a*}.
Metoda:

1. Vypočítej množinu N_t pro G pomocí algoritmu 1 a množinu N_ϵ pro G.
2. $N_0 := \emptyset$, $i = 1$
3. $N_i := \{A \in N_t \mid \exists \alpha : (A \to \alpha) \in P \ \wedge \ \alpha \in N_\epsilon^* \cdot (\{a\} \cup N_{i-1}) \cdot (N_t \cup \Sigma)^*\}$
4. Pokud $N_i \neq N_{i-1}$, $i := i + 1$ a vrať se k 3., jinak $N_{a*} := N_i$ a skonči.

Algoritmus 3: Výpočet množiny neterminálů generujících řetězec terminálů končící symbolem a.
Vstup: Gramatika $G = (N, \Sigma, P, S)$.
Výstup: Množina N_{*a}.
Metoda:

1. Vypočítej množinu N_t pro G pomocí algoritmu 1 a množinu N_ϵ pro G.
2. $N_0 := \emptyset$, $i = 1$
3. $N_i := \{A \in N_t \mid \exists \alpha : (A \to \alpha) \in P \ \wedge \ \alpha \in (N_t \cup \Sigma)^* \cdot (\{a\} \cup N_{i-1}) \cdot N_\epsilon^*\}$
4. Pokud $N_i \neq N_{i-1}$, $i := i + 1$ a vrať se k 3., jinak $N_{*a} := N_i$ a skonči.

S využitím uvedených algoritmů vypočteme množinu N_{aa}:

Algoritmus 4: Výpočet množiny neterminálů generujících řetězec terminálů, který obsahuje podřetězec aa:
Vstup: Gramatika $G = (N, \Sigma, P, S)$.
Výstup: Množina N_{aa}.
Metoda:

1. Vypočítej množiny N_t, N_{a*}, N_{*a} pro G pomocí algoritmů 1, 2, 3 a množinu N_ϵ pro G.
2. $N_0 := \emptyset$, $i = 1$
3. $N_i := \{A \in N_t \mid \exists \alpha : (A \to \alpha) \in P \ \wedge \ (\alpha \in (N_t \cup \Sigma)^* \cdot N_{i-1} \cdot (N_t \cup \Sigma)^* \vee \alpha \in (N_t \cup \Sigma)^* \cdot (N_{*a} \cup \{a\}) \cdot N_\epsilon^* \cdot (N_{a*} \cup \{a\}) \cdot (N_t \cup \Sigma)^*)\}$
4. Pokud $N_i \neq N_{i-1}$, $i := i + 1$ a vrať se k 3., jinak $N_{aa} := N_i$ a skonči.

Demonstrace použití algoritmu:
$N_\epsilon = \{X, Y, S\}$.

Výpočet množiny N_t:

1. $N_0 = \emptyset$
2. $N_1 = \{X, Y\}$
3. $N_2 = \{X, Y, S\}$
4. $N_3 = \{X, Y, S\} = N_t$

Výpočet množiny N_{a*}:

1. $N_0 = \emptyset$
2. $N_1 = \{Y\}$
3. $N_2 = \{Y, S\}$
4. $N_3 = \{Y, S\} = N_{*a}$

Výpočet množiny N_{*a}:

1. $N_0 = \emptyset$
2. $N_1 = \{X\}$
3. $N_2 = \{X, S\}$
4. $N_3 = \{X, S\} = N_{*a}$

Výpočet množiny N_{aa}:

1. $N_0 = \emptyset$
2. $N_1 = \{S\}$
3. $N_2 = \{S\} = N_{aa}$

Příklad 4

V jazyce $L_1 \blacktriangle L_2$ jsou řetězce ze Σ^*, pro jejichž všechna možná rozdělení $w = w_1, w_2$, platí aspoň jedno z: $w_1 \in L_1$, $w_2 \in L_2$. Uzavřenost operace $\blacktriangle$ na třídě $\mathcal{L}_0$ ukážeme sestrojením algoritmu, který pracuje s TS přijímajícími jazyky $L_1, L_2 \in \mathcal{L}_0$, jehož výstupem je TS, který přijímá jazyk $L_1 \blacktriangle L_2$.

1. Zkonstruujeme TS M_1, M_2, pro které $L(M_1) = L_1$, $L(M_2) = L_2$.
2. Zkonstruujeme 3páskový TS M, který:
   - na pásce 1 obsahuje řetězec w,
   - na pásce 2 obsahuje kód TS M_1,
   - na pásce 3 obsahuje kód TS M_2.

   Předpokládejme přitom, že TS M je univerzální TS, který na páskách 2 a 3 zvládne opakovaně spouštět simulaci jiných TS s různými vstupy (tedy je schopen na požádání pásku uvést do stavu, kdy obsahuje pouze původní kód TS bez vstupu).
3. M na pásce 1 postupně enumeruje prefixy w_1 řetězce w a dělí jej tak na dvě části $w_1, w_2 : w_1 w_2 = w$. V prvním kroce zvolí $w_1 = \varepsilon, w_2 = w$, v dalších krocích postupně rozšiřuje w_1 o jeden symbol z w.
4. Pro každé zvolené w_1, w_2: M vhodně okopíruje právě zvolený w_1 na pásku 2 tak, aby byl vstupem pro simulaci M_1, podobně okopíruje právě zvolený w_2 na pásku 3 tak, aby byl vstupem pro simulaci M_2.
5. Pak M současně simuluje M_1 na pásce 2 a M_2 na pásce 3, a to tak, že jednotlivé kroky simulace provádí proloženě krok po kroku (tedy nejprve provede krok M_1, poté krok M_2, poté další krok $M_1 \ldots$).
6. Pokud M_1 *nebo* M_2 přijme, pak:
   - pokud $w_2 = \varepsilon$, pak M přijímá, tedy $w \in L_1 \blacktriangle L_2$;
   - pokud $w_2 \neq \varepsilon$, pak M zastaví obě simulace, uvede pásky 2 a 3 do počátečního stavu (z bodu 2) a pokračuje v enumeraci prefixů z bodu 3.

   Díky proloženému provádění kroků simulace může jeden ze simulovaných TS přijmout, i když druhý cyklí.
7. Pokud M_1 i M_2 odmítnou, M odmítá, tedy $w \notin L_1 \blacktriangle L_2$.
8. Pokud M_1 i M_2 cyklí, M cyklí, tedy $w \notin L_1 \blacktriangle L_2$.
9. Pokud právě jeden z M_1, M_2 cyklí a druhý z nich odmítne, M cyklí, tedy $w \notin L_1 \blacktriangle L_2$.

TS M přijme řetězec $w \in \Sigma^*$ právě tehdy, když pro každé rozdělení $w_1 w_2 = w$ alespoň jeden ze simulovaných TS přijme, a v jiných případech řetězec w odmítne nebo cyklí. $L(M) = L_1 \blacktriangle L_2$, tedy $L_1 \blacktriangle L_2 \in \mathcal{L}_0$. □