Poznámky - automaty a gramatiky

Petr Chmel, LS 2018/19

Definice 1 (Deterministický konečný automat (DFA))**.** Deterministický konečný automat je $A = (Q, \Sigma, \delta, q_0, F)$, kde

- Q je konečná množina stavů
- Σ je konečná neprázdná množina určující abecedu
- $\delta : Q \times \Sigma \to Q$ je přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $F \subseteq Q$ je neprázdná množina koncových stavů

Definice 2 (Slovo (prázdné), množina všech (neprázdných) slov)**.** Slovo je konečná (i prázdná) posloupnost symbolů.
Prázdné slovo značíme λ nebo ε.
Množinu všech slov nad Σ značíme Σ^*.
Množinu všech neprázdných slov nad Σ značíme Σ^+.

Definice 3 (Jazyk)**.** Jazyk $L \subseteq \Sigma^*$ je množina slov nad abecedou Σ.

Definice 4 (Zřetězení slov, mocnina, délka slova, počet výskytů)**.** Zřetězení slov $u, v \in \Sigma^*$ je $u.v = uv$.
Mocnina slova $a \in \Sigma^*$ je $a^0 = \lambda, a^i = a.a^{i-1} \forall i \in \mathbb{N}$.
Délka slova w je $|w|$ - počet znaků abecedy.
Počet výskytů symbolu $s \in \Sigma$ ve slove $w \in \Sigma^*$ je $|w|_s$.

Definice 5 (Rozšířená přechodová funkce)**.** Buď $\delta : Q \times \Sigma \to Q$ přechodová funkce. Pak rozšířenou přechodovou funkcí nazveme její tranzitivní uzávěr $\delta^* : Q \times \Sigma^* \to Q$: $\delta^*(q, \lambda) = q, \delta^*(q, wx) = \delta(\delta^*(q, w), x)$ pro $w \in \Sigma^*, x \in \Sigma$.

Definice 6 (Rozpoznávaný jazyk (konečným automatem))**.** Jazyk rozpoznávaný konečným automatem $A = (Q, \Sigma, \delta, Q_0, F)$ je jazyk $L(A) = \{w : w \in \Sigma^*, \delta^*(q_0, w) \in F\}$.

Definice 7 (Přijímané slovo)**.** Slovo w je přijímáno automatem A, právě když $w \in L(A)$.

Definice 8 (Rozpoznatelný jazyk)**.** Jazyk L je rozpoznatelný konečným automatem, jestliže existuje konečný automat A takový, že $L(A) = L$.

Definice 9 (Regulární jazyky)**.** Třídu jazyků rozpoznatelných konečnými automaty nazveme regulární jazyky a značíme $\mathcal{F}$.

Věta 1 (Pumping lemma pro regulární jazyky)**.** Nechť L je regulární jazyk. Pak existuje konstanta $n \in \mathbb{N}$ (závislá na L) taková, že pro každé $w \in L$ splňující $|w| \geq n$ můžeme w rozdělit na tři části $w = xyz$ splňující

- $|xy| \leq n$
- $y \neq \lambda$
- $\forall k \in \mathbb{N}_0 : xy^k z \in L$

Důkaz. Buď L regulární jazyk. Pak existuje DFA A s n stavy takový, že $L = L(A)$.
Vezměme libovolný řetězec délky $m \geq n$: $w = a_1 \dots a_m, a_i \in \Sigma$.
Zadefinujeme $p_0 = q_0, p_i = \delta^*(q_0, a_1 \dots a_i)$. Jelikož máme n stavů automatu a alespoň $n+1$ mezistavů slova, z Dirichletova principu dochází k alespoň jednomu opakování. Vezměme první takový stav: $\exists i, j : 0 \leq i < j \leq n : p_i = p_j$.
Zadefinujeme $x = a_1 \dots a_i, y = a_{i+1} \dots a_j, z = a_{j+1} \dots a_n$. Pak $w = xyz, |xy| \leq n, y \neq \lambda$ a zároveň můžeme část y libovolněkrát opakovat a vstup bude také akceptovaný, tedy $\forall k \in \mathbb{N}_0 : xy^k z \in L$. ⊞

Definice 10 (Pravá kongruence (konečného indexu))**.** Řekneme, že ekvivalence $\sim$ na Σ^* je pravá kongruence, jesltiže $\forall u, v, w \in \Sigma^* : u \sim v \Rightarrow uw \sim vw$.
Dále řekneme, že je konečného indexu, má-li rozklad $\Sigma^*/\sim$ konečný počet tříd.

Věta 2 (Myhill-Nerodova)**.** Nechť L je jazyk nad konečnou abecedou Σ. Pak následující tvrzení jsou ekvivalentní

1. L je rozpoznatelný konečným automatem
2. existuje pravá kongruence $\sim$ konečného indexu nad Σ^* taková, že L je sjednocením jistých tříd rozkladu $\Sigma^*/\sim$.

Důkaz. Automat $\Rightarrow$ kongruence: kongruence odpovídá stavům DFA v průběhu výpočtu, sjednocení tříd jsou přijímací stavy.
Kongruence $\Rightarrow$ automat: L má konečně mnoho tříd rozkladu - pak tyto můžeme namapovat na automat. $\boxplus$

Věta 3 (Konečný postup zjištění nekonečnosti jazyka)**.** Regulární jazyk L je nekonečný, právě když $\exists u \in L : n \leq |u| \leq 2n$ pro n z pumping lemmatu.

Důkaz. Takové slovo existuje a můžeme jej pumpovat.
Jinak kdyby neexistovalo a existovalo pouze nejkratší slovo delší než $2n$ a L by byl nekonečný, pak lze toto slovo „odpumpovat“ a zmenšíme ho tím (z pumping lemmatu) na $|w| - n$, a tedy je kratší, což je spor. $\boxplus$

Definice 11 (Dosažitelný stav)**.** Nechť $A = (Q, \Sigma, \delta, q_0, F)$ DFA, $q \in Q$. Řekneme, že q je dosažitelný, jestliže existuje $w \in \Sigma^*$ takový, že $\delta^*(q_0, w) = q$.

Algoritmus 1 (Nalezení dosažitelných stavů)**.** BFS

Důkaz. Korektnost: na nedosažitelný se nemůžu dostat
Úplnost: na dosažitelný se z definice dostaneme. $\boxplus$

Definice 12 (Ekvivalence konečných automatů)**.** Konečné automaty A, B nad Σ jsou ekvivalentní, jestliže $L(A) = L(B)$.

Definice 13 (Homomorfismus, isomorfismus DFA)**.** Nechť A_1, A_2 jsou DFA. Pak $h : Q_1 \to Q_2$ je homomorfismus, jestliže $h(q_{0_1}) = q_{0_2}, g(\delta_1(q, x) = \delta_2(h(q), x), q_1 \in F_1 \Leftrightarrow h(q_1) \in F_2$
Homomorfismus je isomorfismus, je-li prostý a na.

Věta 4 (Ekvivalence a homomorfismus DFA)**.** Existuje-li homomorfismus A_1 do A_2, pak jsou A_1 a A_2 ekvivalentní.

Důkaz. $w \in L(A_1) \Leftrightarrow \delta_1^*(q_{0_1}, w) \in F_1 \Leftrightarrow h(\delta_1^*(q_{0_1}, w)) \in F_2 \Leftrightarrow \delta_2^*(h(q_{0_1}), w) \in F_2 \Leftrightarrow \delta_2^*(q_{0_2}, w) \in F_2 \Leftrightarrow w \in L(A_2)$. $\boxplus$

Definice 14 (Ekvivalence a rozlišitelnost stavů)**.** Řekneme, že stavy $p, q \in Q$ jsou ekvivalentní, pokud pro všechna slova $w \in \Sigma^* : \delta^*(p, w) \in F \Leftrightarrow \delta^*(q, w) \in F$.
Nejsou-li stavy ekvivalentní, řekneme, že jsou rozlišitelné.

Lemma 1 (Tranzitivita ekvivalence na stavech)**.** Ekvivalence na stavech je tranzitivní.

Algoritmus 2 (Hledání rozlišitelných stavů v DFA)**.** Rozlišíme přijímající a nepřijímající stavy
Dále buď $p, q \in Q$ a existuje $a \in \Sigma : \delta(p, a)$ a $\delta(q, a)$ jsou rozlišitelné. Pak i p, q jsou rozlišitelné (děláme dolní trojúhelníkovou maticí).

Důkaz. Sporem: ať si existuje nerozlišený pár, ze všech vezmeme ten, rozlišený nejkratším slovem $a_1 \dots a_n$ - $p, q \in Q$. Pak stavy $\delta(p, a_1), \delta(q, a_1)$ jsou rozlišené kratším slovem, tedy v dalším kroku měl algoritmus rozlišit i p, q. $\boxplus$

Algoritmus 3 (Testování ekvivalence regulární jazyků)**.** Najdeme DFA rozpoznávající oba jazyky, sjednotíme stavy a přechody. Jazyky jsou ekvivalentní, právě když počáteční stavy jsou ekvivalentní.

Definice 15 (Redukovaný DFA, redukt)**.** DFA je redukovaný, pokud nemá nedosažitelné stavy a žádné dva stavy nejsou ekvivalentní.
DFA B je reduktem A, je-li B redukovaný a $L(A) = L(B)$.

Algoritmus 4 (Nalezení reduktu DFA)**.**

1. Eliminujeme nedosažitelné stavy
2. Najdeme rozklad zbylých stavů na třídy ekvivalence
3. Konstruujeme DFA B na třídách ekvivalence jakožto stavech.
4. Přechody podle přechodů přes třídy ekvivalence
5. Jinak používáme třídy přechodů

Definice 16 (Nedeterministický konečný automat)**.** NFA je $A = (Q, \Sigma, \delta, S_0, F)$ taková, že:

- Q je konečná monžina stavů
- Σ je konečná abeceda
- $\delta Q \times \Sigma \to 2^Q$ je přechodová funkce
- $S_0 \subseteq Q$ je množina počátečních stavů
- $F \subseteq Q$ je množina koncových stavů

Definice 17 (Rozšířená přechodová funkce pro NFA)**.** TODO - ale tak, jak by se dalo čekat

Definice 18 (Jazyk přijímaný NFA)**.** Buď A NFA. Pak $L(A) = \{w \in \Sigma^* : (\exists q_0 \in S_0) : \delta^*(q_0, w) \cap F \neq \emptyset\}$ je jazyk přijímaný automatem A.

Algoritmus 5 (Podmnožinová konstrukce)**.** Z NFA vytvoříme DFA na množině podmnožin, stavem jsou všechny stavy, zbytek jak se dá čekat.

Věta 5 (Převod NFA na DFA)**.** Pro DFA D vzniklý z NFA N podmnožinovou konstrukcí platí $L(D) = L(N)$.

Důkaz. Indukcí lze dokázat $\delta^*_D(\{q_0\}, w) = \delta^*_N(q_0, w)$. ⊞

Definice 19 (λ-NFA)**.** λ-NFA je $E = (Q, \Sigma, \delta, N_0, F)$ takový, že splňuje podmínky pro NFA vyjma $\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$, navíc $\lambda \notin \Sigma$.

Definice 20 (λ-uzávěr)**.** Pro $q \in Q$ definujeme λ-uzávěr $\lambda\text{CLOSE}(q)$ rekurzivně:

- $q \in \lambda\text{CLOSE}(q)$
- $p \in \lambda\text{CLOSE}(q) \wedge r \in \delta(p, \lambda) \Rightarrow r \in \lambda\text{CLOSE}(q)$

Pro $S \subset Q : \lambda\text{CLOSE}(S) = \bigcup_{q \in S} \lambda\text{CLOSE}(q)$

Definice 21 (Rozšířená přechodová funkce λ-NFA)**.** TODO: To, co bychom čekali, jenom obalené v λ-uzávěrech

Věta 6 (Eliminace λ-přechodů)**.** Jazyk L je rozpoznatelný λ-NFA $\Leftrightarrow L \in \mathcal{F}$

Algoritmus 6 (Převod λ-NFA na DFA)**.** Jako převod NFA, ale obalíme v λ-uzávěrech.

Důkaz. Rozepsat. ⊞

Definice 22 (Množinové operace na jazycích)**.** Buďte L, M jazyky. Pak

1. $L \cup M = \{w : w \in L \vee w \in M\}$
2. $L \cap M = \{w : w \in L \wedge w \in M\}$

3. $L - M = \{w : w \in L \wedge w \notin M\}$
4. $\overline{L} = -L = \{w : w \notin L\} = \sigma^* - L$

Věta 7 (De Morganova pravidla)**.** $L \cap M = \overline{\overline{L} \cup \overline{M}}$
$L \cup M = \overline{\overline{L} \cap \overline{M}}$
$L - M = L \cap \overline{M}$

Věta 8 (Uzavřenost regulárních jazyků na množinové operace)**.** Nechť L, M jsou regulární jazyky. Pak $L \cup M, L \cap M, L - M, \overline{L}$ jsou také regulární.

Důkaz. Sestavíme automaty: sjednocení - lambda předchod do obou automatů, sjednocení - kartézský součin, rozdíl - kartézský součin a úprava stavů, aby to dávalo smysl, doplněk - prohození přijímajících a nepřijímajících stavů v DFA. ⊞

Definice 23 (Řetězcové operace na jazycích)**.** Zřetězení jazyků: $L.M = \{u.v : u \in L, v \in M\}$
Mocniny jazyka: $L^0 = \{\lambda\}, L^{i+1} = L^i.L$
Pozitivní iterace: $L^+ = L^1 \cup L^2 \cup \ldots$
Iterace: $L^* = L^0 \cup L^+$
Otočení jazyka: $L^R = \{w^R : w \in L\}$
Levý kvocient L podle M: $M \backslash L = \{v : uv \in L, i \in M\}$
Pravý kvocient L podle M: $L/M = \{u : uv \in L, i \in M\}$

Věta 9 (Uzavřenost regulárních jazyků na řetězcové operace)**.** Jsou-li L, M regulární jazyky, pak i $L.M, L^*, L^+, L^R, M \backslash L, L/M$ jsou regulární.

Důkaz. Sestavení λ-NFA, který se hezky chová. ⊞

Definice 24 (Třída RJ)**.** Pro každou neprázdnou Σ bude $RJ(\Sigma)$ nejmenší třída jazyků taková, že:

- $\emptyset \in RJ(\Sigma)$
- $x \in \Sigma : \{x\} \in RJ(\Sigma)$
- $RJ(\Sigma)$ je uzavřená na konečná sjednocení, zřetězení a iteraci.

Definice 25 (Regulární výrazy)**.** Regulární výrazy $\alpha, \beta \in RV(\Sigma)$ nad konečnou neprázdnou Σ s jejich hodnotou $L(\alpha)$ jsou definovány induktivně:

- $\alpha = \lambda, L(\lambda) = \{\lambda\}$
- $\alpha = \emptyset, L(\emptyset) = \emptyset$
- $\alpha = a \in \Sigma, L(a) = \{a\}$
- $\alpha + \beta : L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $\alpha\beta : L(\alpha\beta) = L(\alpha)L(\beta)$
- $\alpha^* : L(\alpha^*) = L(\alpha)^*$

Priority: nejvyšší má iterace, nižší konkatenace, nejnižší sjednocení.

Věta 10 (Kleeneova)**.** Libovolný jazyk je rozpoznatelný konečným automatem, právě když je ve třídě RJ.

Definice 26 (Dvousměrný konečný automat)**.** Dvousměrný (dvoucestný) konečným automatem nazýváme pětici $A(Q, \Sigma, \delta, q_0, F)$, kde Q je konečná množina stavů, Σ je konečná množina vstupních symbolů, $\delta : Q \times \Sigma \to Q \times \{1, -1\}$, $q_0 \in Q$ počáteční stav, $F \subseteq Q$.

Definice 27 (Přijetí dvousměrným konečným automatem)**.** Slovo w je přijato dvousměrným konečným automatem, pokud výpočet začal na prvním písmenu w vlevo v počátečním stavu, čtecí hlava poprvé opustila w vpravo v nějakém přijímacím stavu. (Mimo čtené slovo není výpočet definován.)

Věta 11. Jazyky přijímané dvousměrnými konečnými automaty jsou právě regulární jazyky

Důkaz. Konečný automat na dvousměrný: přidám pohyb doprava.
Dvousměrný automat na konečný: sestrojím NFA pomocí metody řezů. ⊞

Definice 28 (Mooreův stroj)**.** Mooreovým sekvenčním strojem nazýváme šestici $A = (Q, \Sigma, Y, \delta, \mu, q_0)$, kde Q je konečná neprázdná množina stavů, Σ je konečná neprázdná množina symbolů (vstupní abeceda), Y je výstupní abeceda, $\delta : Q \times \Sigma \to Q$ je přechodová funkce, $\mu : Q \to Y$ je značkovací funkce, $q_0 \in Q$ je počáteční stav.

Definice 29 (Mealyho stroj)**.** Mealyho sekvenčním strojem nazýváme šestici $A = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$, kde Q je konečná neprázdná množina stavů, Σ je konečná neprázdná množina symbolů (vstupní abeceda), Y je výstupní abeceda, $\delta : Q \times \Sigma \to Q$ je přechodová funkce, $\lambda_M : Q \times \Sigma \to Y$ je značkovací funkce, $q_0 \in Q$ je počáteční stav.

Lemma 2. Pro každý Moorův stroj existuje Mealyho stroj převádějící každé vstupní slovo na stejné výstupní slovo.

Lemma 3. Pro každý Mealyho stroj existuje Moorův stroj převádějící každé vstupní slovo na stejné výstupní slovo.

Definice 30 (Formální (generativní) gramatika)**.** Formální (generativní) gramatika je čtveřice $G = (V, T, P, S)$ složená z V, konečné množiny neterminálů, neprázdné konečné množiny terminálů T, počátečního symbolu $S \in V$ a konečné množiny pravidel P reprezentující rekurzivní definici jazyka s pravidly ve tvaru $\beta A \gamma \to \omega : A \in V, \beta, \gamma, \omega \in (V \cup T)^*$

Definice 31 (Klasifikace gramatik)**.** Typu 0: rekurzivně spočetné jazyky, libovolná pravidla
Typu 1: kontextové gramatiky, pravidla typu $\gamma A \beta \to \gamma \omega \beta : A \in V, \gamma, \beta \in (V \cup T)^*, \omega \in (V \cup T)^+$
Typu 2: bezkontextové gramatiky, v pravidlech nalevo jen neterminál
Typu 3: pravé lineární gramatiky, regulární jazyky, pravidla typu $A \to \omega B, A \to \omega$.

Definice 32 (Derivace)**.** Mějme gramatiku $G = (V, T, P, S)$.
Říkáme, že α se přímo přepíše na ω (píšeme $\alpha \Rightarrow \omega$ nebo $\alpha \Rightarrow_G \omega$), jestliže $\exists \beta, \gamma, \nu, \upsilon \in (V \cup T)^* : \alpha = \nu\beta\upsilon, \omega = \nu\delta\upsilon \wedge (\beta \to \gamma) \in P$.
Dále řekneme, že α se přepíše na ω (píšeme $\alpha \Rightarrow^* \omega$), jestliže existuje posloupnost $(\beta_i)_i = 1^n : \alpha \Rightarrow \beta_1 \Rightarrow \ldots \Rightarrow \omega$. Tuto posloupnost napíšeme derivací. Pokud zároveň $i \neq j \Rightarrow \beta_i \neq \beta_j$, řekneme, že je odvození (derivace) minimální.

Definice 33 (Jazyk generovaný gramatikou)**.** Mějme gramatiku $G = (V, T, P, S)$.
Jazyk $L(G)$ generovaný gramatikou $G = (V, T, P, S)$ je množina terminálních řetězců, pro něž existuje derivace ze startovního symbolu, tedy $L(G) = \{w \in T^* : S \Rightarrow^* w\}$.

Věta 12 (Regulární jazyky a gramatika typu 3)**.** Pro každý jazyk rozpoznávaný konečným automatem existuje gramatika typu 3, která ho generuje.

Důkaz. Sestavíme gramatiku z automatu (tak, jak bychom čekali). ⊞

Lemma 4 (Speciální forma gramatik typu 3)**.** Pro každou gramatiku typu 3 existuje gramatiku typu 3 generující stejný jazyk a navíc mající pravidla pouze ve tvarech $A \to aB, A \to B, A \to \lambda, A, B \in V, a \in T$.

Důkaz. Pravidla s více terminály rozsypeme a tím budeme mít více pravidel. ⊞

Věta 13. Pro každý jazyk L generovaný gramatikou třetího typu existuje λ-NFA rozpoznávající L.

Důkaz. Vezmeme gramatiku z předchozího lemmatu. Pak sestavíme λ-NFA tak, jak bychom čekali. ⊞

Definice 34 (Levá a pravá lineární gramatika)**.** Gramatiky typu 3 nazýváme pravé lineární.
Gramatiky s pravidly pouze ve tvaru $A \to Bw, A \to w, A, B \in V, w \in T^*$ nazveme levé lineární.

Definice 35 (Lineární gramatika a jazyk)**.** Gramatika je lineární, jestliže má pravidla pouze typu $A \to uBw, A \to w : A, B \in V, u, w \in T^*$.
Lineární jazyky jsou právě jazyky generované lineárními gramatikami.

Definice 36 (Derivační strom)**.** Derivační strom pro gramatiku $G = (V, T, P, S)$ je strom, kde každý vnitřní uzel je ohodnocen neterminálem V, každý uzel je ohodnocem prvkem $V \cup T \cup \{\lambda\}$.
Navíc, je-li uzel ohodnocen λ, je jediným synem svého rodiče a je-li A ohodnocení vrcholu a jeho děti jsou zleva pořadě ohodnoceny $A_1 \dots A_n$, pak $A \to A_1 \dots A_n$ je pravidlo gramatiky.

Definice 37 (Strom dává (yields) slovo)**.** Řekneme, že derivační strom dává slovo w, jestliže w je slovo složené z ohodnocení listů (bráno zleva doprava).

Definice 38 (Levá a pravá derivace)**.** Levá derivace (značeno $\Rightarrow_{lm}$) v každém kroku přepisuje nejlevější terminál.
Pravá derivace (značeno $\Rightarrow_{rm}$) v každém kroku přepisuje nejpravější terminál.

Věta 14 (Derivace a derivační strom)**.** Pro danou gramatiku $G = (V, T, P, S), w \in T^*$ jsou následující tvrzení ekvivalentní

- $A \Rightarrow^* w$
- $A \Rightarrow^*_{lm} w$
- $A \Rightarrow^*_{rm} w$
- Existuje derivační strom s kořenem A dávající slovo w.

Definice 39 (Ekvivalence gramatik)**.** Gramatiky jsou ekvivalentní, jestliže generují stejný jazyk.

Definice 40 (Jednoznačnost a víceznačnost CFG)**.** Bezkontextová gramatika je víceznačná, jestliže existuje slovo dávané dvěma různými derivačními stromy.
V opačném případě je jazyk jednoznačný.

Definice 41 (Zásobníkový automat)**.** Zásobníkový automat (PDA) je $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde

- Q je konečná množina stavů
- Σ je neprázdná konečná množina vstupních symbolů
- Γ je neprázdná konečná zásobníková abeceda
- $\delta : Q \times (Q \times \{\lambda\}) \times \Gamma \Rightarrow P_{FIN}(Q \times \Gamma^*)$ je přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $Z_O \in \Gamma$ je počáteční symbol na zásobníku
- F je množina přijímajících stavů

Definice 42 (Situace zásobníkového automatu)**.** Situace zásobníkového automatu je trojice (q, w, γ), kde $q \in Q$ je aktuální stav, $w \in \Sigma^*$ je zbývající vstup a $\gamma \in \Gamma^*$ je obsah zásobníku.

Definice 43 (Posloupnost situací)**.** Nechť P je PDA. Pak $\vdash_P$ definujeme jako $(p, aw, X\beta) \vdash_P (q, w, \alpha\beta)$ iff $(q, \alpha) \in \delta(p, a, X)$.

Definice 44 (Jazyk přijímaný koncovým stavem nebo prázdným zásobníkem)**.** Jazyk přijímaný koncovým stavem je $L(P) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^* (q, \lambda, \alpha), q \in F, \alpha \in \Gamma^*\}$
Jazyk přijímaný prázdným zásobníkem je $N(P) = \{w \in \Sigma^* : (q_0, w, Z_O) \vdash^* (q, \lambda, \lambda), q \in Q\}$

Lemma 5. Mějme $L = L(P_F)$ pro nějaký PDA P_F. Pak existuje PDA P_N takový, že $L = N(P_N)$.

Důkaz. Obalím začátek (2 stavy na zásobníku) a z koncových stavů přidám λ-přechod do vyprazdňovacího stavu. ⊞

Lemma 6. Mějme $L = N(P_N)$ pro nějaký PDA P_N. Pak existuje PDA P_F takový, že $L = L(P_F)$.

Důkaz. Přidám nový znak na dno zásobníku a když na něj narazím, přejdu do přijímacího stavu. ⊞

Věta 15 (Bezkontextovost a PDA)**.** Následující tvrzení jsou ekvivalentní:

- Jazyk L je bezkontextový
- Jazyk L je přijímaný nějakým zásobníkovým automatem koncovým stavem
- Jazyk L je přijímaný nějakým zásobníkovým automatem prázdným zásobníkem

Algoritmus 7 (PDA z CFG)**.** Buď $G = (V, T, P, S)$, pak $PDAD = (\{q\}, T, V \cup T, \delta, q, S)$.
Pro neterminály $A \in V : \delta(q, \lambda, A) = \{(q, \beta) : A \to \beta \in P\}$ Pro terminály $a \in T : \delta(q, a, a) = \{(q, \lambda)\}$.

Lemma 7 (Přijímání prázdným zásobníkem z PDA)**.** Pro PDA P konstruovaný algoritmem z CFG G výše je $N(P) = L(G)$.

Důkaz. Přes levou derivaci. ⊞

Lemma 8 (Gramatika pro PDA)**.** Buď PDA P. Pak existuje bezkontextová gramatika G splňující $L(G) = N(P)$.

Důkaz. Zkonstruujeme gramatiku s neterminály $[pXq]$ - PDA vyšel z p, vzal ze zásobníku X a přešel do q + nový počáteční symbol S.
$\forall p \in Q : S \to [q_0 Z p]$, pro všechny $(r, Y_1 \dots Y_n) \in \delta(q, s, X), \forall r_1 \dots r_{k-1} \in Q : [qXr_n] \to s[rY_1r_1][r_1Y_2r_2] \dots [r_{k-1}Y_kr_n]$.
⊞

Definice 45 (Zbytečný, užitečný, generující, dosažitelný symbol)**.** Symbol v gramatice je užitečný, pokud existuje derivace, v níž se vyskytuje.
Pokud X není užitečný, je zbytečný.
X je generující, pokud $X \Rightarrow^* w$ pro nějaké $w \in T^*$. Vždy $w \to^* w$ v nula krocích.
X je dosažitelný, pokud $S \Rightarrow^* \alpha X \beta$ pro nějaká $\alpha, \beta \in (V \cup T)^*$.

Lemma 9 (Eliminace zbytečných symbolů)**.** Buď CFG G s $L(G) \neq \emptyset$. Pak G_1 zkonstruovaná eliminací negenerujících symbolů a pravidel je obsahujících a poté eliminací všech nedosažitelných symbolů nemá zbytečné symboly a $L(G_1) = L(G)$.

Algoritmus 8 (Generující symboly)**.** Každý terminál je generující.
Jsou-li všechny symboly na pravé straně pravidla generující, je i neterminál na levé straně generující.

Algoritmus 9 (Dosažitelné symboly)**.** S je dosažitelný.
Je-li $A \in V$ dosažitelný, pak i všechny symboly na pravé straně všech pravidel s A na levé straně jsou dosažitelné.

Definice 46 (Nulovatelný neterminál)**.** Neterminál A je nulovatelný, jestliže $A \Rightarrow^* \lambda$.

Algoritmus 10 (Nalezení nulovatelných symbolů)**.** $A \to \lambda$ - A je nulovatelný.
Pokud $B \to C_1 \dots C_k$ a všechna C_i jsou nulovatelná, je i B nulovatelné.

Algoritmus 11 (Konstrukce gramatiky bez λ-pravidel)**.** Najdi nulovatelné symboly. Pro každé pravidlo s nulovatelnými symboly vytvoř všechna možná pravidla s/bez těchto symbolů (2^i, kde i je počet nulovatelných symbolů).

Definice 47 (Jednotkové pravidlo)**.** Jednotkové pravidlo je $A \to B \in P$, kde $A, B \in V$.

Definice 48 (Jednotkový pár)**.** Dvojice $A, B \in V$ takovou, že $A \Rightarrow^* B$ pouze jednotkovými pravidly nazveme jednotkový pár.

Algoritmus 12 (Nalezení jednotkových párů)**.** (A, A) je jednotkový pár $\forall A \in V$.
Je-li (A, B) jednotkový pár a $(B \to C) \in P$, pak (A, C) je jednotkový pár.

Algoritmus 13 (Eliminace jednotkových pravidel z G)**.** Najdi všechny jednotkové páry
Pro všechny jednotkové páry (A, B) dáme do nové gramatiky všechna pravidla $A \to \alpha : B \to \alpha \in P$ a zároveň se nejedná o jednotkové pravidlo.

Lemma 10 (Gramatika v normálním tvaru)**.** Mějme bezkontextovou gramatiku $G : L(G) \setminus \{\lambda\} \neq \emptyset$. Pak existuje CFG G_1 taková, že $L(G_1) = L(G) \setminus \{\lambda\}$ a G_1 neobsahuje λ-pravidla, jednotková pravidla ani zbytečné symboly.

Důkaz. Eliminujeme λ-pravidla.
Eliminujeme jednotková pravidla.
Eliminujeme zbytečné symboly. ⊞

Definice 49 (Chomského normální tvar)**.** Bezkontextová gramatika $G = (V, T, P, S)$ je v Chomského normálním tvaru, je-li bez zbytečných symbolů a všechna pravidla jsou ve tvarech $A \to BC, A, B, C \in V \vee A \to a, A \in V, a \in T$.

Algoritmus 14 (Neterminály)**.** Pro každý terminál vytvoříme nový neterminál, přidáme generující pravidlo a nahradíme v nutných pravidlech.

Algoritmus 15 (Rozdělení pravidel)**.** Je-li dlouhé pravidlo $A \to B_1 \dots B_k$, přidáme pravidla tak, aby $A \to B_1C_1, C_1 \to B_2C_2, \dots, C_{k-2} \to C_{k-1}C_k$.

Věta 16 (O Chomského normální formě)**.** Buď G bezkontextová gramatika splňující $L(G) \setminus \{\lambda\} \neq \emptyset$. Pak existuje CFG G_1 v Chomského normálním tvaru taková, že $L(G_1) = L(G) \setminus \{\lambda\}$.

Lemma 11 (Velikost derivačního stromu gramatiky v ChNF)**.** Mějme derivační strom podle gramatiky G v Chomského normálním tvaru dávající slovo w. Je-li délka nejdelší cesty n, pak $|w| \leq 2^{n-1}$.

Věta 17 (Pumping lemma pro bezkontextové jazyky)**.** Nechť L je bezkontextový jazyk. Pak existují dvě přirozená čísla p, q taková, že každé $z \in L : |z| \geq p$ lze rozložit na $z = uvwxy$, kde

- $|vwx| < q$
- $vx \neq \lambda$
- $\forall i \geq 0 : uv^iwx^iy \in L$.

Důkaz. Vezmeme gramatiku v Chomského normální formě. Buď $|V| = n$, položíme $p = 2^{n-1}, q = 2^n$. Je-li $z \in L : |z| > p$, má v derivačním stromě cestu délky $> n$. Vezmeme nejdelší cestu, a terminál t. Alespoň dva z posledních $n + 1$ neterminálů na cestě do t jsou z Dirichletova principu stejné.
Tedy můžeme toto jejich odvozování opakovat kolikrát chceme, a tím získáme vx. Navíc $|vwx| \leq q$ díky tomu, že vybereme poslední iteraci a navíc Chomského normální forma je nevypouštěcí, takže $vx \neq \lambda$. ⊞

Lemma 12 (Nekonečnost bezkontextového jazyka)**.** Pro každý bezkontextová jazyk L existují přirozená čísla m, n taková, že L je nekonečný, právě když $\exists z \in L : m < |z| < n$.

Důkaz. Vezmu p, q z pumping lemmatu a projdu slova pro $m = p, n = p + q$. ⊞

Definice 50 (Greibachové normální forma)**.** Řekneme, že gramatika je v Greibachové normální formě, jestliže všechna pravidla mají tvar $A \to a\beta : a \in T, A \in V, \beta \in V^*$.

Věta 18 (O Greibachové normální formě)**.** Ke každému bezkontextovému jazyku L existuje bezkontextová gramatika G v Greibachové normální formě taková, že $L(G) = L \setminus \{\lambda\}$.

Důkaz. Očíslujeme neterminály a povolíme levou rekurzi pouze ve tvaru $A_i \to A_j\alpha : i < j$. Zbytku se zbavíme pomocí spojování pravidel nebo odstranění levé rekurze skrze rozepsání pravidel (1..n).
Poté pospojujeme (n..1)
Odstraníme neterminály uvnitř pravidel. ⊞

Algoritmus 16 (Cocke-Younger-Kasami)**.** TODO, „dolní trojúhelníková matice“

Věta 19 (CFL a substituce)**.** Buď L bezkontextový jazyk nad Σ a σ substituce na Σ taková, že $\sigma(a)$ je CFL pro každé $a \in \Sigma$. Pak i $\sigma(L)$ je CFL.

Důkaz. Idea: Listy v d.s. generují další stromy.
Přejmenujeme neterminály tak, aby byly všude jednoznačné: v G a G_a pro $a \in \Sigma$.
Nová gramatika: $G' = (V \cup \bigcup_{a\in\Sigma} V_a, \bigcup_{a\in\Sigma} T_a, \bigcup_{a\in\Sigma} P_a \cup \{p \in P : \text{ s a nahrazeným za } S_a\}, S)$.
Zjevně G' generuje $\sigma(L)$. ⊞

Věta 20 (CFL a inverzní homomorfismus)**.** Nechť L je bezkontextový jazyk a homomorfismus h. Pak $h^{-1}(L)$ je bezkontextový jazyk. Navíc, inverzní homomorfismus zachovává determinismus.

Důkaz. Idea: pracujeme se slovem a na jeho znaky aplikujeme homomorfismus a ten pak kontrolujeme. ⊞

Věta 21 (CFL a sjednocení, konkatenace, Kleene star, pozitivní uzávěr a reverzi)**.** CFL jsou uzavřené na sjednocení, konkatenaci, uzávěr (*), pozitivní uzávěr (+) a zrcadlový obraz.

Důkaz. Sjednocení: přejmenujeme neterminály pokud je potřeba a začneme s výběrem jazyka $S \to S_1|S_2$.
Konkatenace: $S \to S_1S_2$
Iterace: $S \to S_1S|\lambda$
Pozitivní iterace: $S \to S_1S|S_1$
Zrcadlový obraz: otočení všech pravých stran pravidel. ⊞

Lemma 13 (CFL a kvocienty s RL)**.** Bezkontextové jazyky jsou uzavřené na kvocienty s regulárním jazykem.

Důkaz. Levý kvocient: paralelní běh DFA a PDA, s PDA se začne, když DFA je v přijímacím stavu.
Pravý kvocient: $L/M = (M^R \backslash L^R)^R$. ⊞

Věta 22 (CFL a DCFL jsou uzavřené na průnik s regulárním jazykem)**.** Buď L bezkontextový jazyk a R regulární jazyk. Pak $L \cap R$ je bezkontextový jazyk.
Buď L deterministický bezkontextový jazyk a R regulární jazyk. Pak $L \cap R$ je deterministický bezkontextový jazyk.

Důkaz. Prostě zároveň ve stavu simulujeme DFA současně s PDA. ⊞

Věta 23 (Rozdíl CFL a RL)**.** Nechť L je bezkontextový jazyk a R je regulární jazyk. Pak $L \setminus R$ je bezkontextový jazyk.

Důkaz. $L - R = L \cap \overline{R}$ a $\overline{R}$ je regulární. ⊞

Věta 24 (CFL NEjsou uzavřené na doplněk nebo rozdíl)**.** CFL nejsou uzavřené na doplněk nebo rozdíl.

Lemma 14 (Doplněk deterministického CFL)**.** Doplněk deterministického CFL je opět deterministický CFL.

Definice 51 (Dyckův jazyk)**.** Dyckův jazyk nad abecedou $Z_N = \{a_1, a'_1, \ldots, a_n, a'_n\}$ gramatikou s pravidly $S \to \lambda|SS|a_1Sa'_1|\ldots|a_nSa'_n$.

Věta 25 (Dyckovy jazyky)**.** Pro každý bezkontextový jazyk L existuje regulární jazyk R tak, že $L = h(D \cap R)$ pro vhodný Dyckův jazyk D a homomorfismus h.

Definice 52 (Deterministický zásobníkový automat)**.** Zásobníkový automat $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ je deterministický, právě když zároveň platí

- $\delta(q, a, X)$ je nejvýše jednoprvková $\forall q \in Q, a \in \Sigma \cup \{\lambda\}, X \in \Gamma$
- Je-li $\delta(q, a, X)$ neprázdná pro nějaké $a \in \Sigma$, pak $\delta(q, \lambda, X)$ musí být prázdná.

Věta 26 (DPDA a regulární jazyky)**.** Nechť L je regulární jazyk. Pak $L = L(P)$ pro nějaký DPDA P.

Definice 53 (Bezprefixový jazyk)**.** Jazyk L je bezprefixový, pokud neexistují $x, y \in L$ taková, že x je prefix y.

Věta 27 (DPDA a přijímání prázdným zásobníkem)**.** Jazyk $L \in N(P)$ pro nějaký DPDA P, právě když L je bezprefixový a $L \in L(P')$ pro nějaký DPDA P'.

Věta 28 (DPDA a jednoznačnost gramatik)**.** Nechť $L = N(P)$ pro nějaký DPDA P. Pak L má jednoznačnou gramatiku.
Nechť $L = L(P)$ pro nějaký DPDA P. Pak L má jednoznačnou gramatiku.

Definice 54 (Turingův stroj)**.** Turigův stroj je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ se složkami

- Q - neprázdná konečná množina stavů
- Σ - konečná neprázdná množina vstupních symbolů
- Γ - konečná množina všech symbolů pro pásku, $\Gamma \supset \Sigma, Q \cap \Gamma = \emptyset$
- $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ - (částečná) přechodová funkce
- $q_0 \in Q$ - počáteční stav
- $B \in \Gamma \setminus \Sigma$ - blank
- $F \subseteq Q$ - množina koncových (přijímacích) stavů

Definice 55 (Konfigurace Turingova stroje (Instantaneous Description, ID))**.** Konfigurace Turingova stroje je řetězec $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n$, kde q je stav TM, čtecí hlava je vlevo od i-tého symbolu a $X_1 \dots X_n$ je část pásky mezi nejlevějším a nejpravějším symbolem různým od blanku. Pokud je hlava na kraji, přidáme ještě jeden blank.

Definice 56 (Krok Turingova stroje)**.** Krok Turingova stroje M (značeno $\vdash, \vdash_M, \vdash_M^*$):

- Pro $\delta(q, X_i) = (p, Y, L)$: $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \vdash_M X_1X_2 \dots X_{i-2}pX_{i-1}YX_{i+1} \dots X_n$
- Pro $\delta(q, X_i) = (p, Y, R)$: $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \vdash_M X_1X_2 \dots X_{i-1}YpX_{i+1} \dots X_n$

Definice 57 (TM přijímá jazyk)**.** Řekneme, že Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ přijímá jazyk $L(M) = \{w \in \Sigma^* : q_0w \vdash_M^* \alpha p\beta, p \in F, \alpha, \beta \in \Gamma^*\}$.

Definice 58 (Rekurzivně spočetný jazyk)**.** Jazyk nazveme rekurzivně spočetným, jestliže je přijímán nějakým Turingovým strojem.

Definice 59 (Přechodový diagram pro TM)**.** „Orientovaný graf“ přechodů ze stavu do stavu s označenými hranami čteným/psaným znakem a směrem posunu hlavy.

Věta 29 (Rekurzivně spočetné jazyky jsou typu 0)**.** Každý rekurzivně spočetný jazyk je typu 0.

Věta 30 (Jazyky typu 0 jsou rekurzivně spočetné)**.** Každý jazyk typu 0 je rekurzivně spočetný.

Definice 60 (Vícepáskový Turingův stroj)**.** TM s více páskami, vstup na první pásce, ostatní prázdné, první hlava vlevo od vstupu, ostatní libovolně, hlava je v počátečním stavu, 1 krok: hlava přejde do nového stavu, na každé pásce napíšeme nový symbol, každá hlava se nezávisle posune vlevo, vpravo, nebo zůstane.

Věta 31 (Vícepáskové a jednopáskové TM)**.** Každý jazyk přijímaný vícepáskovýn TM je přijímaný i nějakým jednopáskovým TM.

Definice 61 (Lineárně omezený automat)**.** Lineárně omezený automat (LBA) je nedeterministický Turingův stroj, který je omezen konci l, r a na pásce je na začátku napsáno lwr, kde w je slovo.
Slovo w je přijímáno LBA, pokud $q_0lwr \vdash^* \alpha p\beta, p \in F$.

Věta 32 (Kontextové jazyky a LBA)**.** Každý kontextový jazyk lze přijímat pomocí LBA.

Věta 33 (LBA a kontextové jazyky)**.** LBA přijímají pouze kontextové jazyky.

Definice 62 (TM zastaví, rozhoduje jazyk)**.** Řekneme, že TM zastaví, pokud vstoupí do stavu q se čteným symbolem X a $\delta(q, X)$ není definováno.
Dále TM M rozhoduje jazyk L, pokud $L = L(M)$ a pro každé $w \in \Sigma^*$ stroj nad w zastaví.

Definice 63 (Rekurzivní jazyky)**.** Jazyky rozhodnutelné TM nazýváme rekurzivní jazyky.

Věta 34 (Postova)**.** Jazyk L je rekurzivní, právě když L i $\overline{L}$ jsou rekurzivně spočetné.

Důkaz. Máme TM $M_1, M_2 : L = L(M_1), \overline{L} = L(M_2)$.
Pro dané w simulujeme naráz M_1, M_2 (dvě pásky).
Pokud jeden přijme, M zastaví a odpoví.
Navíc, jazyky jsou komplementární, tedy jeden z TM zastaví, tedy L je rekurzivní. ⊞

Důsledek 1. Je-li L rekurzivní jazyk, je i $\overline{L}$ rekurzivní jazyk.

Poznámka (Kódování TM)**.** Kódujeme do $\{0, 1\}$: očíslujeme od 1 stavy, symboly a směry. Pak kódujeme instrukce jako $0^i10^j10^k10^l10^m$, kde i je číslo původního stavu, j je číslo symbolu, k je číslo nového stavu, l je číslo nového symbolu a m je číslo směru.

Definice 64 (Diagonální jazyk)**.** Diagonální jazyk je $L_d = \{w : \text{TM reprezentovaný jako } w \text{ který nepřijímá slovo } w\}$.

Věta 35 (Diagonální jazyk není rekurzivně spočetný)**.** L_D není rekurzivně spočetný.

Důkaz sporem. Pro spor předpokládejme, že L_d je rekurzivně spočetný, a tedy $L_d = L(M)$ pro nějaký TM M.
Zkonstruujeme tabulku, kde řádky jsou označeny kódy TM, sloupce slovy jazyka $0, 1^*$, obě lexikograficky uspořádané.
Alespoň jeden řetězec kóduje M, tedy $code(M) = w_i$.
Je-li $w_i \in L_d$, pak M_i přijímá w_i, což je spor s definicí.
Naopak, je-li $w_i \notin L_d$, pak $w_i \in L_d$ z definice, což je opět spor.
Tedy takový M neexistuje a L_d není rekurzivně spočetný. ⊞

Definice 65 (Univerzální jazyk)**.** Definujeme univerzální jazyk L_u jakožto množinu binárních řetězců, které kódují pár (M, w), kde M je MT a $w \in L(M)$.
Pak TM rozpoznávající L_u se nazývá univerzální Turingův stroj.

Věta 36 (Nerozhodnutelnost univerzálního jazyka)**.** L_u je rekurzivně spočetný, ale není rekurzivní.

Důkaz. Máme TM přijímající L_u, tedy L_u je rekurzivně spočetný.
Nechť L_u je také rekurzivní. Pak $\overline{L_u}$ by byl také rekurzivní, a pro TM přijímající $\overline{L_u}$ můžeme zkonstruovat TM přijímající L_d. Ale L_d není rekurzivně spočetný, tedy $\overline{L_u}$ není rekurzivně spočetný a L_u není rekurzivní. ⊞

Definice 66 (Rozhodnutelný problém)**.** Problémem P myslíme matematicky/informaticky definovanou množinu otázek kódovatelnou řetězci nad abecedou Σ s odpověďmi {ano, ne}.
Problém je (algoritmicky) rozhodnutelný, pokud existuje TM takový, že pro každý vstup $w \in P$ zastaví a navíc přijme, právě když $P(w) =$ano.
Problém, který není algoritmicky rozhodnutelný nazveme nerozhodnutelný.

Definice 67 (Redukce)**.** Redukcí problému P_1 na P_2 nazýváme algoritmus R, který pro každou instanci $w \in P_1$ vydá $R(w) \in P_2$ tak, že $P_1(w) = P_2(R(w))$.

Věta 37 (Redukce, rozhodnutelnost a rekurzivní spočetnost)**.** Nechť existuje redukce z problému P_1 na P_2.
Pak je-li P_1 nerozhodnutelný, je i P_2 nerozhodnutelný.
Pak není-li P_1 rekurzivně spočetný, ani P_2 není rekurzivně spočetný.

Důkaz sporem. Je-li P_1 nerozhodnutelný a P_2 rozhodnutelný, pak lze P_1 rozhodnout za pomocí redukce, což je spor.
Není-li P_1 rekurzivně spočetný a P_2 je, pak lze za pomocí redukce přijmout P_1, což je spor. ⊞

seznam A	seznam B	
$\#$	$\#q_0w\#$	
X	X	$\forall X \in \Gamma$
$\#$	$\#$	
qX	Yp	pro $\delta(q, X) = (p, Y, R)$
ZqX	pZY	pro $\delta(q, X) = (p, Y, L), Z \in \Gamma$
$q\#$	$Yp\#$	pro $\delta(q, B) = (p, Y, R)$
$Zq\#$	$pZY\#$	pro $\delta(q, B) = (p, Y, L), Z \in \Gamma$
XqY	q	$q \in F$
Xq	q	$q \in F$
qY	q	$q \in F$
$q\#\#$	$q\#$	$q \in F$

Věta 38 (Halting problem)**.** Mějme $M, w \in \{0,1\}^*$. Hledáme algoritmus $Halt(M, w)$, který vydá 1, právě když M zastaví na w, jinak vydá 0.

Důkaz. Redukce L_d na Halt. Mějme takový algoritmus pro Halt. Pak vytvoříme $Halt_{no}(w)$: Halt(w,w)=1 - nekonečný cyklus, jinak zastavíme.
Halt($Halt_{no}, Halt_{no}$) není řešitelná, tedy algoritmus pro Halt nemůže existovat. ⊞

Definice 68 (Postův korespondenční problém)**.** Instance Postova korespondenčního problému (PCP) jsou dva seznamy slov nad Σ značené $A = w_1, \ldots, w_k, B = x_1, \ldots, x_k$ stejné délky.
Řekneme, že instance PCP má řešení, jestliže existuje posloupnost $(i_j)_{j=1}^m$ taková, že $x_{i_1}x_{i_2}\ldots x_{i_m} = w_{i_1}w_{i_2}\ldots w_{i_m}$. Tuto posloupnost navíc nazveme řešením.
Postův korespondenční problém je: Pro danou instance PCP rozhodněte, zda má řešení.

Definice 69 (Částečné řešení PCP)**.** Částečné řešení PCP je posloupnost indexů $(i_j)_{j=1}^m$ taková, že jeden z dvojice řetězců $x_{i_1}x_{i_2}\ldots x_{i_m}$ a $w_{i_1}w_{i_2}\ldots w_{i_m}$ je prefixem druhého.

Lemma 15 (O řešení a částečném řešení PCP)**.** Je-li posloupnost čísel řešením, pak každý její prefix je také řešením.

Definice 70 (Modifikovaný Postův korespondenční problém, iniciální řešení)**.** Mějme instanci PCP, tedy $A = w_1, \ldots, w_k, B = x_1, \ldots, x_k$.
Iniciálním řešením nazveme seznam 0 nebo více přirozených čísel (i_j) tak, že $x_1x_{i_1}x_{i_2}\ldots x_{i_m} = w_1w_{i_1}w_{i_2}\ldots w_{i_m}$.
Modifikovaný Postův korespondenční problém zní: má instance PCP iniciální řešení?

Lemma 16 (Redukce MPCP na PCP)**.** Vezmu všechna slova z A, za každý znak přidáme hvezdičku, zkopírujeme první slovo na začátek a před první znak také přidáme hvězdičku. Přidáme poslední slovo dolar. Vezmu všechna slova z B, před každý znak přidáme hvezdičku, zkopírujeme první slovo na začátek. Přidáme poslední slovo „*$“.

Algoritmus 17 (Redukce L_u na PCP)**.** Konstruujeme MPCP pro $TM = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, který nikdy nepíše B a nikdy nejde hlavou doleva od počáteční pozice. Nechť w je vstupní slovo.

Věta 39 (PCP je algoritmicky nerozhodnutelný)**.** PCP je algoritmicky nerozhodnutelný.

Důkaz. Máme algoritmus redukující L_u na MPCP. Chceme, M přijímá w, právě když zkonstruovaný PCP má iniciální řešení.
„⇐“: $w \in L(M)$ - začneme iniciálním párem a simulujeme výpočet.
„⇒“: Máme-li iniciální řešení PCP, odpovídá přijímajícímu výpočtu M nad w - musel „simulovat“ výpočet a uspět. ⊞

Věta 40 (Nerozhodnutelnost víceznačnosti)**.** Je algoritmicky nerozhodnutelné, zda je bezkontextová gramatika víceznačná.

Důkaz. Mějme instanci PCP $A = w_1, \ldots, w_k, B = x_1, \ldots, x_k,\ a_1, \ldots, a_k \in \mathbb{N}$.
$G_A : A \to w_1 A a_1 | w_2 A a_2 | \ldots | w_k A a_k | w_1 a_1 | \ldots | w_k a_k$
$G_B : B \to x_1 B a_1 | x_2 B a_2 | \ldots | x_k B a_k | x_1 a_1 | \ldots | x_k a_k$
$G_{AB} : \{A \to A|B\} \cup G_A \cup G_B$
G_{AB} je víceznačná, právě když instance PCP (A, B) má řešení. ⊞

Věta 41 (Nerozhodnutelné problémy)**.** Nechť G_1, G_2 jsou CFG, R je regulární výraz. Následující problémy jsou algoritmicky nerozhodnutelné:

1. $L(G_1) \cap L(G_2) = \emptyset$?
2. $L(G_1) = T^*$ pro nějakou T?
3. $L(G_1) = L(G_2)$?
4. $L(G_1) = L(R)$?
5. $L(G_1) \subseteq L(G_1)$?
6. $L(R) \subseteq L(G_1)$?

Důkaz. 1: Mějme instanci PCP $A = w_1, \ldots, w_k, B = x_1, \ldots, x_k,\ a_1, \ldots, a_k \in \mathbb{N}$.
$G_A : A \to w_1 A a_1 | w_2 A a_2 | \ldots | w_k A a_k | w_1 a_1 | \ldots | w_k a_k$
$G_B : B \to x_1 B a_1 | x_2 B a_2 | \ldots | x_k B a_k | x_1 a_1 | \ldots | x_k a_k$
PCP má řešení, právě když $L(G_A) \cap L(G_B) \neq \emptyset$.
2: Mějme instanci PCP $A = w_1, \ldots, w_k, B = x_1, \ldots, x_k,\ a_1, \ldots, a_k \in \mathbb{N}$.
$G_A : A \to w_1 A a_1 | w_2 A a_2 | \ldots | w_k A a_k | w_1 a_1 | \ldots | w_k a_k$
$G_B : B \to x_1 B a_1 | x_2 B a_2 | \ldots | x_k B a_k | x_1 a_1 | \ldots | x_k a_k$
$L(G_A), L(G_B)$ jsou deterministické, tedy i jejich doplňky jsou deterministické CFL a i jejich sjednocení je CFL.
Tedy PCP má řešení $\Leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset L(G) = \overline{L(G_1)} \cup \overline{L(G_2)} \neq \Sigma^*$.
3: ať G_1 generuje Σ^*
4: za R volíme Σ^*
5: ať G_1 generuje Σ^*
6: za R volíme Σ^*

⊞

Seznam témat