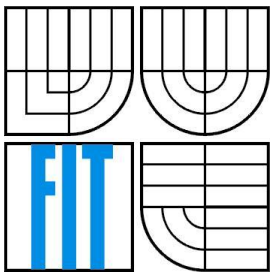
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



TIN

(TEMATICKÉ OKRUHY KE STÁTNICÍM 2009)

Vladimír Veselý – xvesel38

OBSAH

0 ZÁKLADY

Nechť **Σ je abeceda**. Označme Σ^* množinu všech konečných posloupností ***w* řetězcem nad abecedou**, kde *w* má tvar:

$$w = a_1 a_2 a_3 \dots a_n, \text{kde } a_i \in \Sigma \text{ pro } i = 1, \dots, n$$

ε značí prázdný řetězec, přičemž $|\varepsilon| = 0$.

Množina *L*, pro kterou platí $L \subseteq \Sigma^*$ nazýváme **formálním jazykem nad abecedou Σ**.

Nechť L_1 je jazyk nad abecedou Σ_1 a L_2 je jazyk nad abecedou Σ_2, pak **součinem (konkatenací) jazyků** $L_1 . L_2$ nad $\Sigma_1 \cup \Sigma_2$:

$$L_1 . L_2 = \{xy | x \in L_1 \wedge y \in L_2\}$$

1 CHOMSKÉHO KLASIFIKACE GRAMATIK A FORMÁLNÍCH JAZYKŮ

1.1 GRAMATIKA

Gramatika G = (N, Σ, P, S) je uspořádaná čtveřice, kde:

- N je konečná množina nonterminálních symbolů;
- Σ je konečná množina terminálních symbolů;
- P je podmnožina kartézského součinu:

$$(N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

- $S \in N$ se nazývá startovací symbol gramatiky.

Nechť G je gramatika a $\lambda, \mu \in (N \cup \Sigma)^*$. Mezi λ a μplatí binární relace $\underset{\mathbf{G}}{\Rightarrow}$ **přímé derivace** $\lambda \Rightarrow \mu$, můžeme-li λ a μ vyjádřit ve tvaru:

$$\lambda = \gamma\alpha\delta, \mu = \gamma\beta\delta, \text{ kde } \gamma, \delta \in (N \cup \Sigma)^* \text{ a } \alpha \to \beta \in P$$

$\overset{+}{\Rightarrow}$ označuje tranzitivní uzávěr relace ⇒ a nazývá se **relací derivace**:

$$\lambda \overset{+}{\Rightarrow} \mu \Leftrightarrow \lambda = v_0 \Rightarrow v_1 \Rightarrow v_2 \ldots \Rightarrow v_n = \mu$$

$\overset{*}{\Rightarrow}$ označuje reflexivní tranzitivní uzávěr relace ⇒, příčemž platí:

$$\lambda \overset{*}{\Rightarrow} \mu \Leftrightarrow \lambda \overset{+}{\Rightarrow} \mu \vee \lambda = \mu$$

Nechť G je gramatika, pak $\alpha \in (N \cup \Sigma)^*$ nazýváme **větnou formou**, platí-li $S \overset{*}{\Rightarrow} \alpha$. Větná forma, která obsahuje pouze terminální symboly, se nazývá **věta**.

Jazyk generovaný gramatikou G: $L(G) = \left\{ w | S \overset{*}{\Rightarrow} w \wedge w \in \Sigma^* \right\}$

1.2 CHOMSKÉHO KLASIFIKACE

Typ	Gramatika/tvar pravidel	Jazyk
Typ 0	Obecné (neomezené) $\boldsymbol{\alpha \to \beta}; \alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*, \beta \in (N \cup \Sigma)^*$	Rekurzivně vyčíslitelné
Typ 1	Kontextové $\boldsymbol{\alpha A \beta \to \alpha \gamma \beta}; A \in N;\ \alpha, \beta \in (N \cup \Sigma)^*; \gamma \in (N \cup \Sigma)^+$ nebo $S \to \varepsilon$	Kontextové
Typ 2	Bezkontextové $\boldsymbol{A \to \alpha}; A \in N;\ \alpha \in (N \cup \Sigma)^*$	Bezkontextové
Typ 3	Pravé/Levé lineární $\boldsymbol{A \to xB | x}; A, B \in N;\ x \in \Sigma^*$	Regulární

2 KONEČNÝ AUTOMAT (KA) A JAZYK PŘIJÍMANÝ KA

2.1 KONEČNÝ AUTOMAT

Nedeterministickým konečným automatem NKA rozumíme jednocestný iniciální automat M=(Q, Σ, δ, q_0, F), kde:

- Q je konečná množina stavů;
- Σ je konečná vstupní abeceda;
- δ je přechodová funkce tvaru:

$$\delta: Q \times \Sigma \to 2^Q$$

- $q_0 \in Q$ je počáteční stav;
- $F \subseteq Q$ je množina koncových stavů.

Nechť M=(Q, Σ, δ, q_0, F) je KA, pak **konfigurací *C*** rozumíme: $C = (q, w);\ q \in Q$ a $w \in \Sigma^*$

2.2 KONFIGURACE A PŘECHOD

Počáteční konfigurace $= (q_0, a_1 a_2 a_3 \dots a_n)$

Koncová konfigurace $= (q_F, \varepsilon)$

Přechodem KA rozumíme binární relaci $\vdash$ v množině konfigurací:

$\vdash \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*)$, definována $(q, aw) \vdash (q', w) \Leftrightarrow q' \in \delta(q, a)$, pro $q, q' \in Q; a \in \Sigma; w \in \Sigma^*$

2.3 PŘIJÍMÁNO KA

Řetězec *w* přijímaný KA: $(q_0, w) \overset{*}{\vdash} (q_F, \varepsilon)$

Jazyk přijímaný KA M: $L(\mathrm{M}) = \{w | (q_0, w) \overset{*}{\vdash} (q_F, \varepsilon) \wedge q_F \in F\}$

3 VARIANTY KA, MINIMALIZACE KA

3.1 VARIANTY

Deterministický konečný automat DKA má oproti NKA jinou přechodovou funkci:

$$\delta: Q \times \Sigma \to Q \cup \{\text{undef}\}, |\delta(q,a)| = 1 \text{ pro } \forall q \in Q \text{ a } \forall a \in \Sigma$$

Rozšířený konečný automat RKA se liší od ostatních KA tvarem přechodové funkce:

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$$

3.2 MINIMALIZACE

Minimalizaci KA provádíme odstraněním ε-pravidel, zdeterminizováním (sjednocení stejných stavů) a následnou redukcí.

Funkce ε-uzávěr$(q) = \{p | \exists w \in \Sigma^*: (q,w) \overset{*}{\vdash} (p,w)\}$ pro stav.

Pro množinu ε-uzávěr$(T) = \bigcup_{s \in T}$ ε-uzávěr(s)

Mějme KA, stav $q \in Q$ je **dosažitelný**, pokud existuje $w \in \Sigma^*$ takové, že $(q_0,w) \overset{*}{\vdash} (q,\varepsilon)$, jinak je stav **nedosažitelný**.

M je úplně definovaný DKA. Říkáme, že řetězec $w \in \Sigma^*$ rozlišuje stavy q_1 a q_2, jestliže $(q_1,w) \overset{*}{\vdash} (q_3,\varepsilon) \wedge (q_2,w) \overset{*}{\vdash} (q_4,\varepsilon)$, kde buď $q_3 \in F$, nebo $q_4 \in F$. Říkáme, že **stavy $q_1, q_2 \in Q$ jsou k-nerozlišitelné** a píšeme $q_1 \overset{k}{\equiv} q_2$, právě když neexistuje žádný $w \in \Sigma^*, |w| \leq k$, který by q_1 a q_2 rozlišoval.

DKA **je redukovaný**, jestliže žádný stav z Q není nedostupný a žádné dva stavy nejsou nerozlišitelné.

4 REGULÁRNÍ MNOŽINY A REGULÁRNÍ VÝRAZY

4.1 REGULÁRNÍ MNOŽINY

Regulární množinu (RM) nad Σ definujeme rekurzivně takto:

1) Ø je RM nad Σ;
2) {ε} je RM nad Σ;
3) $\{a\}$ je RM nad Σ pro všechna $a \in \Sigma$;
4) Jsou-li P a Q RM nad Σ, pak taky:
   - $P \cup Q$
   - $P \cap Q$
   - P^*

   jsou RM nad Σ;
5) Žádné další množiny, než ty, které lze získat pomocí výše uvedených pravidel, nejsou RM.

4.2 REGULÁRNÍ VÝRAZY

Regulární výrazy (RV) nad Σ a RM, které označují, jsou rekurzivně definovány takto:

1) Ø je RV označující RM Ø;
2) ε je RV označující RM {ε};
3) a je RV označující RM {a} pro všechna $a \in \Sigma$;
4) Jsou-li p a q RV označující RM P a Q, pak také:
   - $p + q$ je RV označující RM $P \cup Q$;
   - $p.q$ je RV označující RM $P \cap Q$;
   - p^* je RV označující RM P^*;
5) Žádné jiné RV nad Σ neexistují.

4.3 ROVNICE NAD RV

Rovnice, jejímiž složkami jsou koeficienty a neznámé, které reprezentují (dané a hledané) regulární výrazy, nazýváme **rovnice nad regulárními výrazy**.

Soustava rovnic nad RV je ve standardním tvaru vzhledem k neznámým $\Delta = \{X_1, X_2, \dots, X_n\}$, má-li tvar:

$$\bigwedge_{i=1,\dots,n} X_i = \alpha_{i0} + \alpha_{i1}X_1 + \alpha_{i2}X_2 + \cdots + \alpha_{in}X_n$$

Nejmenším pevným bode rovnice $X = pX + q$ je $X = p^*q$.

5 VLASTNOSTI REGULÁRNÍCH JAZYKŮ

Pummping lemma – nechť L je nekonečný regulární jazyk, pak existuje celočíselná konstanta k takové, že platí:

$$w \in L \wedge |w| \geq k \Rightarrow w = xyz \wedge 0 < |y| \leq k \wedge xy^i z \in L \text{ pro } i \geq 0$$

Strukturální vlastnosti: Konečnost (každý konečný jazyk je regulární).

Uzávěrové vlastnosti: Třída RJ je uzavřena vůči sjednocení, konkatenaci a iteraci (z definice RM a RV).

Rozhodnutelné problémy: Neprázdnost, náležitost, ekvivalence (přes KA).

6 TRANSFORMACE A NORMÁLNÍ FORMY BKG

6.1 STRUKTURA BKG

Nechť δ je věta nebo větná forma v gramatice G=(N, Σ, P, S) a nechť $S = v_0 \Rightarrow v_1 \Rightarrow v_2 \ldots \Rightarrow v_n = \delta$ je její derivace v G. **Derivační strom** příslušející této derivaci je vrcholově ohodnocený strom s následujícími vlastnostmi:

1) Vrcholy derivačního stromu jsou ohodnoceny symboly z množiny $(N \cup \Sigma)$, kořen stromu pak symbolem S;
2) Přičemž přímé derivaci $v_{i-1} \Rightarrow v_i$ $(i = 1, \ldots, k$, kde $v_{i-1} = \lambda A \mu$ a $v_i = \lambda \alpha \mu; A \in N;$ $\lambda, \mu \in (N \cup \Sigma)^*$ a kde $A \rightarrow \alpha; \alpha = X_1 X_2 \ldots X_n$ je pravidlo z P) odpovídá právě n hran (A,X_j), kde j=1,..., n vycházejících s uzlu A;
3) Ohodnocení koncových uzlů-listů vytváří větnou formu či dokonce větu δ.

Nechť $S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \ldots \Rightarrow \alpha_n = \delta$ je derivace větné formy δ. Jestliže byl v každém řetězci α_i, kde $i = 1, \ldots, n$ přepsán vždy nejlevější/nejpravější nonterminál, pak tuto derivaci nazýváme **levou/pravou derivací větné formy δ**.

Nechť G gramatika a nechť $\lambda = \alpha\beta\gamma$ je větná forma. Podřetězec β se nazývá **frází větné formy vzhledem k nonterminálu** $A \in N$, jestliže:

$$S \overset{*}{\Rightarrow} \alpha A \gamma$$
$$A \overset{*}{\Rightarrow} \beta$$

Podřetězec β je **jednoduchou frází**, jestliže $A \Rightarrow \beta$. Nejlevější jednoduchá fráze se nazývá **l-fráze**.

6.2 VLASTNOSTI BKG PRO TRANSFORMACI

Říkáme, že **věta *w* generovaná gramatikou je víceznačná**, existují-li alespoň 2 různé derivační stromy s koncovými uzly tvořícími větu w. **Gramatika je víceznačná**, pokud generuje aspoň jednu víceznačnou větu – v opačném případě je **jednoznačná**. Jazyky, které lze generovat víceznačnou gramatikou, ale žádnou jednoznačnou nazýváme **jazyky s inherentní víceznačností**.

Nechť G je gramatika a $X \in (N \cup \Sigma)$ je symbol. Říkáme, že ***X* je nedostupný** v G, neexistuje-li derivace $S \overset{*}{\Rightarrow} \alpha X \beta$, pro $\alpha, \beta \in (N \cup \Sigma)^*$. Symbol ***X* nazýváme zbytečným**, jestliže v G neexistuje derivace tvaru $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, pro $\alpha, \beta \in (N \cup \Sigma)^*; w \in \Sigma^*$.

G je **gramatikou bez ε-pravidel**, když neobsahuje žádné ε-pravidlo (tvar $A \rightarrow \varepsilon$), nebo pokud $\varepsilon \in L(G)$, pak obsahuje jediné takovéto pravidlo, a to $S \rightarrow \varepsilon$, přičemž se S nevyskytuje na pravé straně žádného pravidla.

Pravidlo tvaru $A \rightarrow B$, kde $A, B \in N$ nazýváme **jednoduché pravidlo**.

Gramatika **obsahuje cyklus**, jestliže $A \overset{+}{\Rightarrow} A$.

Gramatika bez zbytečných pravidel, ε-pravidel a bez cyklů se nazývá **vlastní gramatika**.

6.3 NORMÁLNÍ FORMY

Bezkontextová gramatika je v **Chomského normální formě (CNF)**, má-li každé pravidlo z P jeden z těchto tvarů:

- $A \rightarrow BC$, kde $A, B, C \in N$;
- $A \rightarrow a$, kde $A \in N, a \in \Sigma$;
- Je-li $\varepsilon \in L(G)$, pak obsahuje jediné ε-pravidlo, a to $S \rightarrow \varepsilon$, přičemž se S nevyskytuje na pravé straně žádného pravidla.

BKG je v **Greibachové normální formě (GNF)**, má-li každé pravidlo z P (vyjma případného $S \rightarrow \varepsilon$) tvar:

$$A \rightarrow a\alpha, \text{ kde } \alpha \in N^*, a \in \Sigma$$

7 ZÁSOBNÍKOVÝ AUTOMAT (ZA) A JAZYK PŘIJÍMANÝ ZA

7.1 ZÁSOBNÍKOVÝ AUTOMAT

Zásobníkový automat P=(Q, Σ, Γ, δ, q_0, Z_0, F), přičemž:

- Q je konečná množina vnitřních stavů;
- Σ je konečná vstupní abeceda;
- Γ je konečná zásobníková abeceda;
- δ je přechodová funkce tvaru:

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$$

- $q_0 \in Q$ je počáteční stav;
- $Z_0 \in \Gamma$ je počáteční symbol na zásobníku;
- $F \subseteq Q$ je množina koncových stavů.

7.2 KONFIGURACE A PŘECHOD

Nechť P je ZA, pak **konfigurací P,** nazveme trojici $(q, w, \alpha) \in (Q \times \Sigma^* \times \Gamma^*)$, kde:

- q je aktuální stav vnitřního zařízení;
- w je dosud nezpracovaná část vstupního řetězce;
- α je obsah zásobníku.

Přechod ZA je definován jako binární relace $\vdash$ na množině konfigurací:

$(q, aw, Z\gamma) \vdash (q', w, \beta\gamma) \Leftrightarrow (q', \beta) \in \delta(q, a, Z)$, kde $q, q' \in Q; a \in \Sigma^* \cup \{\varepsilon\}; w \in \Sigma^*; Z \in \Gamma; \alpha, \beta \in \Gamma^*$

Je-li $a = \varepsilon$, pak přechod nazýváme **ε-přechodem**.

7.3 PŘIJÍMÁNO ZA

Platí-li pro řetězec $w \in \Sigma^*$ relace $(q_0, w, Z_0) \vdash^* (q_F, \varepsilon, \gamma)$, kde $q_F \in F, \gamma \in \Gamma^*$, pak říkáme, že ***w* je přijímán zásobníkovým automatem**.

Jazyk přijímaný ZA je definován takto:

$$L(\mathrm{P}) = \{w | w \text{ je přijímán ZA}\}$$

ZA **přijímá s vyprázdněním zásobníku** pokud:

$$L(\mathrm{P}) = \{w | (q_0, w, Z_0) \vdash^* (q_F, \varepsilon, \varepsilon) \wedge q_F \in F\}$$

8 VARIANTY ZA

Rozšířený zásobníkový automat RZA má jiný tvar přechodové funkce:

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \to 2^{Q \times \Gamma^*}$$

P je **deterministický zásobníkový automat DZA**, jestliže pro každé $q \in Q, z \in \Gamma$:

$$\forall a \in \Sigma: |\delta(q, a, z)| \leq 1 \wedge \delta(q, \varepsilon, z) = \emptyset$$
$$\vee$$
$$\forall a \in \Sigma: \delta(q, a, z) = \emptyset \wedge |\delta(q, \varepsilon, z)| \leq 1$$

Nechť $L = L(\mathrm{P})$, kde P je DZA. Jazyk L se pak nazývá **deterministický bezkontextový jazyk**.

DZA přijímající s vyprázdněním zásobníku jsou oproti klasickým ZA rozšířeny o determinismus a vyprázdnění, mají však striktně menší vyjadřovací sílu než DZA.

Deterministické RZA musí splňovat podmínku pro každé $q \in Q, a \in (\Sigma \cup \{\varepsilon\}), \gamma \in \Gamma^*$:

1) $|\delta(q, a, z)| \leq 1$;
2) Je-li $\delta(q, a, \alpha) \wedge \delta(q, a, \beta) \wedge \alpha \neq \beta \Rightarrow \alpha$ není předponou β a naopak;
3) Je-li $\delta(q, a, \alpha) \wedge \delta(q, \varepsilon, \beta) \wedge \alpha \neq \beta \Rightarrow \alpha$ není předponou β a naopak

9 VLASTNOSTI BEZKONTEXTOVÝCH JAZYKŮ

Bezkontextová gramatika G=(N, Σ, P, S) má **vlastnost sebevložení**, jestliže existuje $A \in N; u, v \in \Sigma^+$ taková, že $A \Rightarrow uAv$ a A není zbytečný nonterminál.

Bezkontextový jazyk má **vlastnost sebevložení**, jestliže každá gramatika, která jej generuje má vlastnost sebevložení. Typickým příkladem je Dyckův jazyk.

Pummping lemma – Nechť $L \in \mathcal{L}_2$, pak existuje konstanta k taková, že je-li $z \in L$ a $|z| \geq k$, pak lze z zapsat ve tvaru $z = uvwxy$, přičemž platí:

$$vx \neq \varepsilon \wedge |vwx| \leq k \wedge \text{pro } i \geq 0: uvw^i xy \in L$$

Uzávěrové vlastnosti: UZAVŘENY vůči substituci, morfismu, inverznímu morfismu, sjednocení, konkatenaci. NEUZAVŘENY vůči průniku a doplňku.

Rozhodnutelné problémy: ROZHODNUTELNÉ jsou neprázdnost, náležitost, konečnost. NEROZHODNUTELNÉ ekvivalence jazyků a inkluze jazyků gramatik.

10 TURINGŮV STROJ (TS)

10.1 TURINGŮV STROJ

Church-Turingova teze – TS (a jim ekvivalentní systémy) definují svou výpočetní silou to, co považujeme za efektivně vyčíslitelné.

Deterministický Turingův stroj TS je šestice tvaru M=(Q, Σ, Γ, δ, q_0, q_F), kde:

- Q je konečná množina vnitřních řídících stavů;
- Σ je konečná množina symbolů, nazývaných vstupní abeceda;
- Γ je konečná množina symbolů, nazývaných pásková abeceda, kde Σ ⊂ Γ a Δ ∈ Γ;
- δ je parciální přechodová funkce tvaru:

$$\delta: (Q \backslash \{q_F\}) \times \Gamma \rightarrow Q \times (\Gamma \cup \{L, R\}) \text{ přičemž } L, R \notin \Gamma$$

- $q_0 \in Q$ je počáteční stav;
- $q_F \in Q$ je koncový stav;

Symbol Δ nazýváme **blank** a vyskytuje se na nevyužitých či smazaných místech pásky.

10.2 KONFIGURACE, PŘECHOD, VÝPOČET

Konfigurace pásky je dvojice sestávající se z nekonečného řetězce reprezentujícího obsah pásky a ppozice hlavy v tomto řetězci, přesněji jde o prvek množiny $\{\gamma\Delta^{\omega} | \gamma \in \Gamma^*\} \times \mathbb{N}$.

Konfigurace stroje je pak dána stavem řízení a konfigurací pásky: $Q \times \{\gamma\Delta^{\omega} | \gamma \in \Gamma^*\} \times \mathbb{N}$

Krok výpočtu (přechod) definujeme jako binární relaci ⊢ takovou, že $\forall q_1, q_2 \in Q, \forall \gamma \in \Gamma^*, \forall n \in \mathbb{N}, \forall b \in \Gamma$:

- Pro libovolný řetězec $\gamma \in \Gamma^{\omega}$ a číslo n označme $\boldsymbol{s_b^n(\gamma)}$ řetězec, který vznikne z γ záměnou n-tého symbolu γ_n symbolem b;
- $(q_1, \gamma, n) \vdash (q_2, \gamma, n+1)$ pro $\delta(q_1, \gamma_n) = (q_2, R)$ posun doprava při γ_n pod hlavou;
- $(q_1, \gamma, n) \vdash (q_2, \gamma, n-1)$ pro $\delta(q_1, \gamma_n) = (q_2, L)$ posun doleva při γ_n pod hlavou;
- $(q_1, \gamma, n) \vdash (q_2, s_b^n(\gamma), n)$ pro $\delta(q_1, \gamma_n) = (q_2, b)$ zápis b při γ_n pod hlavou;

Výpočet TS je posloupnost konfigurací K_0, K_1, K_2, … K_n, ve které je $K_i \vdash K_{i+1}$ pro všechna $i \geq 0$ taková, že K_{i+1} je v dané posloupnosti, která je buď:

- **Nekonečná**;
- **Konečná** s koncovou konfigurací (q, γ, n), přičemž rozlišujeme následující typy zastavení:
  - **Normální** ($q = q_F$)
  - **Abnormální** ($q \neq q_F$):
    - z daného stavu není definováno δ;
    - hlava je na nejlevější pozici pásky a dojde k posunu doleva;

10.3 PŘIJÍMÁNO TS

Řetězec $w \in \Sigma^*$ **je přijat**, jestliže při aktivaci z počáteční konfigurace pásky $\underline{\Delta}w\Delta$ a počátečního stavu q_0 zastaví ve stavu q_F:

$$(q_0, \Delta w \Delta^\omega, 0) \vdash (q_F, \gamma, n), \text{kde } \gamma \in \Gamma^*, n \in \mathbb{N}$$

Množinu $L(\mathrm{M}) = \{w | w \text{ je přijat TS M}\}$ nazýváme **jazykem přijímaným TS**.

11 VARIANTY TS

Obecně **k-páskový TS** s abecedami Γ_1, Γ_2, Γ_3,… Γ_k a s k odpovídajícími hlavami má přechodovou funkci definovanou jako:

$$\delta: (Q \backslash q_F) \times \Gamma_1 \times \ldots \times \Gamma_k \to Q \times \bigcup_{i=1,\ldots,k} \{i\} \times (\Gamma_i \cup \{L, R\})$$

Nedeterministický TS se od deterministického liší v přechodové funkci:

$$\delta: (Q \backslash \{q_F\}) \times \Gamma \to 2^{Q \times (\Gamma \cup \{L,R\})}$$

Jazyk $L(\mathrm{M})$ **NTS** M je množina řetězců $w \in \Sigma^*$ takových, že M při aktivaci z q_0 při počátečním konfigurace pásky $\underline{\Delta}w\Delta$ MŮŽE zastavit přechodem do q_F.

12 LINEÁRNĚ OMEZENÉ AUTOMATY

Lineárně omezený automat LOA je TS, který nikdy neopustí tu část pásky, na které je zapsán jeho vstup (formálněji je zaveden páskový symbol, který nejde překročit ani přepsat).

Není známo, zda deterministické LOA má slabší vyjadřovací sílu než nedeterministické LOA.

Třída jazyků, které lze generovat kontextovými gramatikami, odpovídá třídě jazyků, které lze přijímat LOA.

Uzávěrové vlastnosti KJ: průnik, sjednoceni, iterace, konkatenace, doplněk.

Rozhodnutelné problémy KJ: náležitost.

Nerozhodnutelné problémy KJ: jazyková inkluze a prázdnost.

13 NEROZHODNUTELNOST

13.1 NE-ROZHODNUTELNOST A JAZYKY

TS se nazývá totální (úplný), právě když pro každý vstup zastaví.

Jazyk $L \subseteq \Sigma^*$ se nazývá:

- **Rekurzivně vyčíslitelný**, jestliže $L(\mathrm{M}) = L$ pro nějaký TS M;
- **Rekurzivní**, jestliže $L(\mathrm{M}) = L$ pro nějaký totální TS M.

Ke každému rekurzivnímu jazyku existuje TS, který ho rozhoduje, tzn., že zastaví pro každé slovo. TS přijímající rekurzivně vyčíslitelný L zastaví pro každé $w \in L$, ovšem pro $w \notin L$ může zastavit nebo cyklit.

Nechť P je problém, specifikovaný jazykem L_P nad Σ. Pak P nazveme:

- **Rozhodnutelný** – L_P je rekurzivní jazyk, tj. existuje TS, který L_P rozhoduje (příjme $\forall w \in L_P$ a nepřijme $\forall w \notin L_P$);
- **Nerozhodnutelný** – když není rozhodnutelný, ale může být…
- **Částečně rozhodnutelný** – když L_P je rekurzivně vyčíslitelný jazyk.

Každý jazyk přijímaný TS je jazykem typu 0. Jazyky mimo tuto třídu, nelze přijímat TS, přičemž pro každou abecedu existuje takový jazyk. Množina všech TS je totiž spočetná, zatímco množina 2^{Σ^*} není.

13.2 VLASTNOSTI JAZYKŮ

Uzávěrové vlastnosti rekurzivně vyčíslitelných jazyků: sjednocení, průnik, konkatenace, iterace.

Uzávěrové vlastnosti rekurzivních jazyků: viz rekurzivně vyčíslitelné + doplněk.

Riceova věta – každá netriviální vlastnost rekurzivně vyčíslitelných jazyků je nerozhodnutelná; každá netriviální nemonotónní vlastnost rekurzivně vyčíslitelných jazyků není ani částečně rozhodnutelná.

14 UNIVERZÁLNÍ TS, PROBLÉM ZASTAVENÍ TS

14.1 UNIVERZÁLNÍ TS

Zavádí koncept programového stroje, který umožňuje ve vstupním řetězci specifikovat konkrétní TS (tj. program) i jeho data (vstup), nad kterým bude pracovat.

U TS, který má být simulován, budeme speciálně kódovat program jakožto množinu $\delta(p, x) = (q, y)$ pomocí iterací 0 a oddělovače 1. Dále použijeme jiný oddělovač mezi programem a daty, např. # (sharpina). Za ním pak bude následovat binárně zakódovaný vstup.

Univerzální TS je tedy obecně třípáskový stroj, který:

- Má na 1. pásce zadání ⟨M⟩#⟨w⟩;
- 2. pásku používá k simulaci pracovní pásky původního stroje;
- Na 3. pásce má zaznamenán řídící stav simulovaného stroje a aktuální pozici hlavy.

14.2 HALTING PROBLEM

Problém zastavení TS (Halting problem – HP) je, když nás zajímá, zda daný TS M pro danou vstupní větu w zastaví.

HP není rozhodnutelný, avšak je částečně rozhodnutelný.

Problém zastavení odpovídá rozhodování jazyka:

$$HP = \{\langle M\rangle\#\langle w\rangle | M \text{ zastaví pro } w\}, \text{ kde } \langle M\rangle \text{ je kód TS a } \langle w\rangle \text{ vstup}$$

Jazyk HP je tedy rekurzivně vyčíslitelný.

Komplement problému zastavení není ani částečně rozhodnutelný a jazyk: $co\text{-}HP = \{\langle M\rangle\#\langle w\rangle | M \text{ nezastaví pro } w\}$ je příkladem jazyka, který není ani rekurzivně vyčíslitelný (je tedy v nadmnožině třídy jazyků typu 0).

15 DIAGONALIZACE

Je důkazová metoda, pomocí které se dokázala existence jazyků nad jazyky typu 0. Základ spočívá ve faktu, že pro neprázdné konečné Σ je množina 2^{Σ^*} nespočetná:

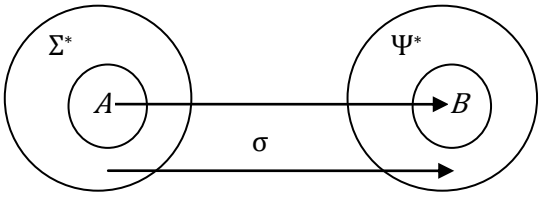
- Předpokládejme, že je 2^{Σ^*} spočetná, pak dle definice spočetnosti existuje bijektivní zobrazení $f: \mathbb{N} \leftrightarrow 2^{\Sigma^*}$
- Uspořádejme Σ^* do nějaké posloupnosti w_1, w_2,w_3, …, např. ε, *x*, *y*, *xx*, *xy*, *yx*, *yy*, *xxx*, … pro Σ={*x*, *y*}. Nyní můžeme *f* zobrazit jako nekonečnou matici:

$$\begin{array}{lccccccc} & \boldsymbol{w_0} & \boldsymbol{w_1} & \boldsymbol{w_2} & & \boldsymbol{w_i} & \\ \boldsymbol{L_0 = f(0)} & a_{00} & a_{01} & a_{02} & & a_{0i} & \\ \boldsymbol{L_1 = f(1)} & a_{10} & a_{11} & a_{21} & \dots & a_{1i} & \dots \\ \boldsymbol{L_2 = f(2)} & a_{20} & a_{11} & a_{22} & & a_{2i} & \\ \vdots & & \vdots & & & \vdots & \\ \boldsymbol{L_i = f(i)} & a_{i0} & a_{i1} & a_{i2} & & a_{ii} & \cdots \\ \vdots & & \vdots & & & \vdots & \ddots \end{array} \quad \text{kde } a_{ji} = \begin{cases} 1 \Leftrightarrow w_i \in L_j \\ 0 \Leftrightarrow w_i \notin L_j \end{cases}$$

- Uvažujme jazyk: $\bar{L} = \{w_i | a_{ii} = 0\}$, kde $\bar{L}$ se liší od každého jazyka $L_i = f(i), i \in \mathbb{N}$:
  - Je-li $a_{ii} = 0$ pak w_i patří do jazyka $\bar{L}$;
  - Je-li $a_{ii} = 1$ pak w_i nepatří do jazyka $\bar{L}$;
- Současně ale $\bar{L} \in 2^{\Sigma^*}$, tudíž *f* není surjektivní, což je SPOR;

16 PRINCIP REDUKCE

Nechť A, B jsou jazyky $A \subseteq \Sigma^*, B \subseteq \Psi^*$. **Redukce** jazyka A na jazyk B je totální, rekurzivně vyčíslitelná funkce $\sigma: \Sigma^* \to \Psi^*$ taková, že $\forall w \in \Sigma^*$ kde $w \in$ A $\Leftrightarrow \sigma(w) \in$ B



Existuje-li redukce jazyka A na B, říkáme, že ***A* je redukovatelný na *B*** a značíme $A \leq B$.

Z redukce plynou následující implikace:

1) Není-li A rekurzivně vyčíslitelné, pak ani B není rekurzivně vyčíslitelné;
2) Není-li A rekurzivní, pak ani B není rekurzivní;
3) Je-li A rekurzivně vyčíslitelné, pak i B je rekurzivně vyčíslitelné;
4) Je-li A rekurzivní, pak i B je rekurzivní.

Redukce spočívá v technice, kdy známý jazyk A (který je/není rekurzivní popř. rekurzivně vyčíslitelný) převedeme (redukujeme) pomocí úplného TS na jazyk B, který zkoumáme. Takto dokážeme, že jazyk B rovněž je/není rekurzivní popř. rekurzivně vyčíslitelný.

17 POSTŮV KORESPONDENČNÍ PROBLÉM

Postův systém nad abecedou Σ je dán neprázdným seznamem S dvojic neprázdných řetězců nad abecedou Σ:

$$S = \langle(\alpha_1, \beta_1); (\alpha_2, \beta_2); \dots; (\alpha_k, \beta_k)\rangle, \text{ kde } k \geq 1; \alpha, \beta \in \Sigma^+$$

Řešením Postova systému je každá neprázdná posloupnost přirozených čísel $I = \langle i_1, i_2, \dots, i_m\rangle$, kde $m \geq 1$ a $1 \leq i_j \leq k$:

$$\alpha_{i_1}\alpha_{i_2} \dots \alpha_{i_m} = \beta_{i_1}\beta_{i_2} \dots \beta_{i_m}\text{ , } m \text{ není omezené a indexy se mohou opakovat}$$

Postův korespondenční problém PCP zní: „Má pan Post známky v dostatečné hodnotě, aby mohl odeslat doporučenou listovní zásilku do mraveniště tamní královně?“ („Existuje pro daný Postův systém řešení?“)

O **PCP** se ví, že **je nerozhodnutelný**, proto se často používá při redukci.

18 PARCIÁLNÍ REKURZIVNÍ FUNKCE

18.1 POČÁTEČNÍ FUNKCE

Totální funkce nad množinou X je taková, jejímž definičním oborem je celé X. **Striktně parciální funkce** je zase taková, kde alespoň pro jeden prvek z X není definovaná funkční hodnota.

Hierarchie parciálně rekurzivních funkcí je založena na dostatečně elementárních tzv. **počátečních funkcích**, jež tvoří stavební kameny vyšších funkcí:

- **Nulová funkce**: $\mathbb{N}^m \to 0: \xi(\,) = 0$;
- **Funkce následníka**: $\mathbb{N} \to \mathbb{N}: \sigma(x) = x + 1$;
- **Projekce**: $\mathbb{N}^n \to \mathbb{N}: \pi_k^n$ vybírá z n-tice k-tý prvek;

18.2 PRIMITIVNĚ REKURZIVNÍ FUNKCE

Primitivně rekurzivní funkce vznikají z počátečních skládáním pomocí následujících způsobů:

1) **Kombinace**:
$$\left.\begin{matrix} f: \mathbb{N}^k \to \mathbb{N}^m \\ g: \mathbb{N}^k \to \mathbb{N}^n \end{matrix}\right\} f \times g: \mathbb{N}^k \to \mathbb{N}^{m+n}: f \times g(\bar{x}) = \big(f(\bar{x}), g(\bar{x})\big), \bar{x} \in \mathbb{N}^k$$
2) **Kompozice**:
$$\left.\begin{matrix} f: \mathbb{N}^k \to \mathbb{N}^m \\ g: \mathbb{N}^m \to \mathbb{N}^n \end{matrix}\right\} f \circ g: \mathbb{N}^k \to \mathbb{N}^n: f \circ g(\bar{x}) = f\big(g(\bar{x})\big), \bar{x} \in \mathbb{N}^k$$
3) **Primitivní rekurze**:
$$\left.\begin{matrix} f: \mathbb{N}^{k+1} \to \mathbb{N}^m \\ g: \mathbb{N}^k \to \mathbb{N}^m \\ h: \mathbb{N}^{k+m+1} \to \mathbb{N}^m \end{matrix}\right\} \begin{matrix} f(\bar{x}, 0) = g(\bar{x}) \\ f(\bar{x}, y+1) = h\big(\bar{x}, y, f(\bar{x}, y)\big), \bar{x} \in \mathbb{N}^k \end{matrix}$$

Příkladem složené funkce může být např. **konstantní funkce** κ_m^n, která libovolné n-tici $\bar{x} \in \mathbb{N}^k$ přiřadí konstantní hodnotu $m \in \mathbb{N}$:

$$\kappa_m^0 = \underbrace{\sigma \circ \sigma \circ \ldots \circ \sigma}_{m\text{-krát}} \circ \xi(\,)$$
$$\kappa_m^n(\bar{x}, 0) = \kappa_m^{n-1}(\bar{x}, 0)$$
$$\kappa_m^n(\bar{x}, y+1) = \pi_{n+1}^{n+1}\big(\bar{x}, y, \kappa_m^n(\bar{x}, y)\big)$$

Každá primitivní rekurzivní funkce je totální funkcí. **Parciálně rekurzivní funkce** jsou tvořeny z počátečních pomocí kombinace, kompozice, primitivní rekurze a minimalizace.

Turingovsky vyčíslitelná funkce je parciální funkce, kterou může počítat nějaký TS. Každá parciálně rekurzivní funkce je turingovsky vyčíslitelná.

19 ČASOVÁ A PAMĚŤOVÁ SLOŽITOST

Důsledkem Church-Turingovy teze je, že každý algoritmus je implementovatelný jistým TS. Zavedení **TS** nám **umožňuje klasifikovat problémy** do dvou tříd:

- problémy, jež **nejsou algoritmicky ani částečně rozhodnutelné** (resp. funkce algoritmicky nevyčíslitelné);
- problémy **algoritmicky alespoň částečně rozhodnutelné** (resp. funkce algoritmicky vyčíslitelné).

Analýzu složitosti algoritmu budeme chápat jako **analýzu složitosti** výpočtů příslušného TS, jejímž cílem je vyjádřit (kvantifikovat) požadované zdroje (čas, prostor) jako funkci závisející na délce vstupního řetězce.

Analyzovat a rozlišit lze složitost nejlepšího, nejhoršího a průměrného případu běhu algoritmu. Obvykle nás zajímá ten nejhorší. **Průměrná složitost** je definována jako suma součinu nad všemi různými složitostmi jednotlivých běhů a pravděpodobností, se kterou nastávají.

Časová složitost je počet kroků (přechodů) TS provedený od počátku do konce výpočtu. **Prostorová (paměťová) složitost** je počet „buněk“ pásky TS požadovaný pro daný výpočet. Je-li časová složitost výpočtu prováděného TS rovna n, pak prostorová složitost tohoto výpočtu není větší než $n + 1$.

Řekneme, že k-páskový **DTS (resp. NTS)** M **přijímá jazyk** L nad abecedou Σ **v čase** $T_{\mathrm{M}}: \mathbb{N} \to \mathbb{N}$, jestliže $L = L(\mathrm{M})$ a M příjme (resp. může přijmout) každé $w \in L$ v nanejvýš $T_{\mathrm{M}}(|w|)$ krocích.

Řekneme, že k-páskový **DTS (resp. NTS)** M **přijímá jazyk** L nad abecedou Σ **v prostoru** $S_{\mathrm{M}}: \mathbb{N} \to \mathbb{N}$, jestliže $L = L(\mathrm{M})$ a M příjme (resp. může přijmout) každé $w \in L$ při použití nanejvýš $S_{\mathrm{M}}(|w|)$ buněk pásky.

Složitost je možné analyzovat i mimo prostředí TS, je důležité najít **cenové kritérium**. Analýza složitosti za takových předpokladů není zcela přesná, důležité ale obvykle je to, aby se zachovala informace o tom, jak rychle roste čas/prostor potřebný k výpočtu v závislosti na velikosti vstupu.

20 ASYMPTOTICKÁ OHRANIČENÍ SLOŽITOSTÍ, TŘÍDY SLOŽITOSTI

20.1 ASYMPTOTICKÉ OHRANIČENÍ SLOŽITOSTI

Při popisu složitosti algoritmů (výpočtů TS), chceme často vyloučit vliv aditivních a multiplikativních konstant, které snadno vznikají drobnými úpravami algoritmu, avšak významněji nemají vliv na rychlost nárustu složitosti. Proto se často složitost popisuje pomocí tzv. **asymptotických odhadů složitosti**.

Nechť $\mathcal{F}$ je množina funkcí $f: \mathbb{N} \to \mathbb{N}$. Pro danou funkci $f \in \mathcal{F}$ definujeme tři množiny funkcí:

- **Asymptotické horní omezení** $f(n)$ je množina:
  $$O\big(f(n)\big) = \{g(n) \in \mathcal{F} | \exists c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n \in \mathbb{N}: n \geq n_0 \Rightarrow 0 \leq g(n) \leq c.f(n)\}$$
- **Asymptotické dolní omezení** $f(n)$ je množina:
  $$\Omega\big(f(n)\big) = \{g(n) \in \mathcal{F} | \exists c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n \in \mathbb{N}: n \geq n_0 \Rightarrow 0 \leq c.f(n) \leq g(n)\}$$
- **Asymptotické oboustranné omezení** $f(n)$ je množina:

$$\Theta\big(f(n)\big) = \{g(n) \in \mathcal{F} | \exists c_1, c_2 \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n \in \mathbb{N}: n \geq n_0 \Rightarrow 0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)\}$$

20.2 TŘÍDY SLOŽITOSTI

Přejdeme nyní od složitosti konkrétních algoritmů (implementacích na Turingových strojích) ke složitosti problémů. **Třídy složitosti** zavádíme jako prostředek ke kategorizaci (vytvoření hierarchie) problémů dle jejich složitosti, tedy dle toho, jak dalece efektivní algoritmy můžeme navrhnout pro jejich rozhodování (u funkcí můžeme analogicky mluvit o složitosti jejich vyčíslování). Podobně jako u určování typu jazyka se budeme snažit zařadit problém do co nejnižší třídy složitosti – tedy určit složitost problému jako složitost jeho nejlepšího možného řešení. Zařazení různých problémů do téže třídy složitosti může odhalit různé vnitřní podobnosti těchto problémů a může umožnit řešení jednoho převodem na druhý.

Mějme dány funkce $t, s: \mathbb{N} \to \mathbb{N}$ a nechť T_M, resp. S_M, značí časovou, resp. prostorovou, složitost TS M. Definujeme následující časové a prostorové třídy složitosti deterministických a nedeterministických TS:

- $DTime[t(n)] = \{L | \exists k\text{-páskový DTS M: } L = L(\mathrm{M}) \text{ a } T_\mathrm{M} \in O\big(t(n)\big)\}$
- $NTime[t(n)] = \{L | \exists k\text{-páskový NTS M: } L = L(\mathrm{M}) \text{ a } T_\mathrm{M} \in O\big(t(n)\big)\}$
- $DSpace[t(n)] = \{L | \exists k\text{-páskový DTS M: } L = L(\mathrm{M}) \text{ a } S_\mathrm{M} \in O\big(t(n)\big)\}$
- $NSpace[t(n)] = \{L | \exists k\text{-páskový NTS M: } L = L(\mathrm{M}) \text{ a } S_\mathrm{M} \in O\big(t(n)\big)\}$

Časově zkonstruovatelná funkce je taková, pokud existuje TS, který pro libovolný vstup w zastaví přesně po $t(|w|)$ krocích. **Prostorově zkonstruovatelná funkce** je taková, pokud existuje TS, který pro libovolný vstup w zastaví s využitím přesně $s(|w|)$ buněk pásky.

21 TŘÍDA P A NP PROBLÉMŮ

Nejběžněji užívané třídy složitosti:

$$\mathbf{P} = \bigcup_{k=0}^{\infty} DTime(n^k) \qquad \mathbf{NP} = \bigcup_{k=0}^{\infty} NTime(n^k)$$

$$\mathbf{PSPACE} = \bigcup_{k=0}^{\infty} DSpace(n^k) \qquad \mathbf{NPSPACE} = \bigcup_{k=0}^{\infty} NSpace(n^k)$$

$$\mathbf{LOGSPACE} = \bigcup_{k=0}^{\infty} DSpace(k \lg n) \quad \mathbf{NLOGSPACE} = \bigcup_{k=0}^{\infty} NSpace(k \lg n)$$

$$\mathbf{EXP} = \bigcup_{k=0}^{\infty} DTime(2^{n^k}) \qquad \mathbf{NEXP} = \bigcup_{k=0}^{\infty} NTime(2^{n^k})$$

$$\mathbf{EXPSPACE} = \bigcup_{k=0}^{\infty} DSpace(2^{n^k}) \qquad \mathbf{NEXPSPACE} = \bigcup_{k=0}^{\infty} NSpace(2^{n^k})$$

Na **vrcholu hierarchie tříd složitosti** se pak hovoří o obecných třídách jazyků (funkcí), se kterými jsme se již setkali:

- **třída primitivně rekurzívních funkcí PR** (implementovatelných pomocí zanořených cyklů s pevným počtem opakování);
- **třída μ-rekurzívních funkcí** (rekurzívních jazyků) **R** (implementovatelných pomocí cyklů s předem neurčeným počtem opakování);
- třída rekurzívně vyčíslitelných funkcí (rekurzívně vyčíslitelných jazyků) **RE**.

Třídy P a EXP jsou uzavřeny vůči doplňku

22 NP-ÚPLNOST

Až doposud jsme třídy používali jako horní omezení složitosti problémů. Všimněme si nyní omezení dolního – to zavedeme pomocí redukovatelnosti třídy problémů na daný problém.

Nechť $\mathcal{R}$ je třída funkcí. Jazyk $L_1 \subseteq \Sigma_1^*$ je **$\mathcal{R}$ redukovatelný** (přesněji $\mathcal{R}$ many-to-one reducible) na jazyk $L_2 \subseteq \Sigma_2^*$, což zapisujeme $L_1 \leq_{\mathcal{R}}^m L_2$, jestliže existuje funkce f z $\mathcal{R}$ taková, že $w \in L_1 \Leftrightarrow f(w) \in L_2$.

Nechť $\mathcal{R}$ je třída funkcí a $\mathcal{C}$ třída jazyků. Jazyk L_0 je **$\mathcal{C}$-těžký ($\mathcal{C}$-hard) vzhledem k $\mathcal{R}$ redukovatelnosti**, jestliže $\forall L \in \mathcal{C}: L \leq_{\mathcal{R}}^m L_0$.

Nechť $\mathcal{R}$ je třída funkcí a $\mathcal{C}$ třída jazyků. Jazyk L_0 nazveme **$\mathcal{C}$-úplný ($\mathcal{C}$-complete) vzhledem k $\mathcal{R}$ redukovatelnosti**, jestliže $L_0 \in \mathcal{C}$ a L_0 je $\mathcal{C}$-těžký jazyk vzhledem k $\mathcal{R}$ redukovatelnosti.

Pro třídy jazyků $\mathcal{C}_1 \subseteq \mathcal{C}_2$ a jazyk L, jenž je $\mathcal{C}_2$-úplný vzhledem k $\mathcal{R}$ redukovatelnosti, platí, že buď $\mathcal{C}_2$ je celá $\mathcal{R}$ redukovatelná na $\mathcal{C}_1$, nebo $L \in \mathcal{C}_2 \backslash \mathcal{C}_1$.

Nejběžnější typy úplností:

- NP, PSPACE, EXP úplnost je definována vůči polynomiální redukovatelnosti (tj. redukovatelnosti pomocí DTS pracujících v polynomiálním čase);
- P, NLOGSPACE úplnost definujeme vůči redukovatelnosti v deterministickém logaritmickém prostoru;
- NEXP úplnost definujeme vůči exponenciální redukovatelnosti (tj. redukovatelnosti pomocí DTS pracujících v exponenciálním čase).

23 SAT PROBLÉM

Nechť $V = \{v_1, v_2, \dots, v_n\}$ je konečná množina Booleovských proměnných (prvotních formulí výrokového počtu). **Literálem** nazveme každou proměnnou v_i nebo její negaci $\overline{v_i}$. **Klausulí** nazveme výrokovou formuli obsahující pouze literály spojené výrokovou spojkou ∨ (nebo).

SAT-problém lze formulovat takto: Je dána množina proměnných *V* a množina klausulí nad *V*. Je tato množina klausulí splnitelná?

Klausule zakódujeme jako posloupnost 0 a pouze na pořadovém místě klausule bude obsahovat *p* nebo *n* podle toho zda obsahuje literál je kladný nebo záporný. Lze zkonstruovat dvoupáskový NTS, který přijímá speciálně zakódovaný jazyk L_{SAT}, který spadá do množiny NP-problémů. **SAT problém je tedy NP-úplný problém**.

Cookův teorém říká, že je-li *L* libovolný jazyk z třídy NP, pak $L \leq_P^m L_{SAT}$ ⇝ takže libovolný jazyk s třídy NP lze redukovat na jazyk problému SAT.