AAG – BI-SPOL-01

Přehled Chomského hierarchie formálních jazyků a gramatik. Turingovy stroje. Třídy problémů P, NP, NP-těžký, NP-úplný.

Obsah

1 Přehled Chomského hierarchie formálních jazyků a gramatik

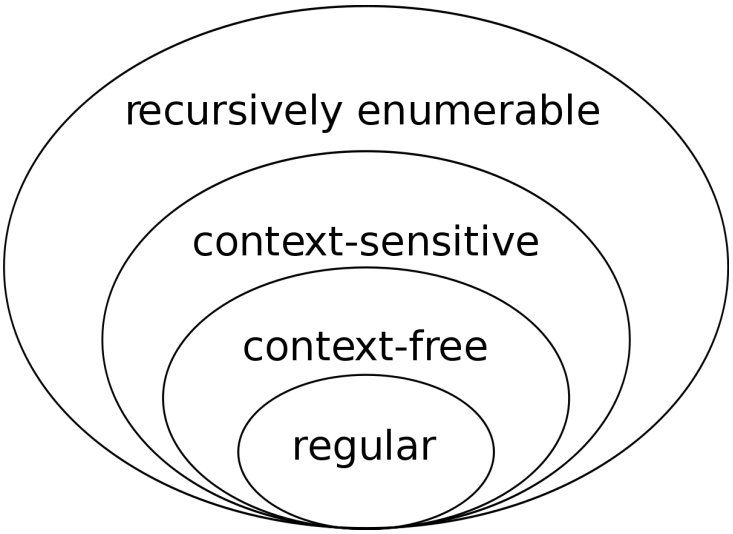

Figure 1: Chomského hierarchie formálních jazyků a gramatik

- *Regulární jazyk* vždy zároveň bezkontextový, kontextový a rekurzivně spočetný
- *Bezkontextový jazyk* vždy kontextový a rekurzivně spočetný
- *Kontextový jazyk* vždy rekurzivně spočetný

1.1 Regulární jazyk

Nejjednodušší množina formálních jazyků, formální jazyk je regulární, když lze:

- přijmout **(ne)deterministickým konečným automatem**
- generovat **regulární gramatikou**
- popsat **regulárním výrazem**

1.2 Bezkotextový jazyk

Formální jazyk je bezkontextový, když lze:

- přijmout **nedeterministickým zásobníkovým automatem**
- generovat **bezkontextovou gramatikou**

Například výraz A^nB^n není regulární, ale je kontextový a rekurzivně spočetný.

1.3 Kontextový jazyk

Formální jazyk je kontextový, když lze:

- přijmout **nedeterministickým lineárně omezeným Turingovým strojem**
- generovat **kontextovou gramatikou**
- generovat **nezkracující gramatikou**

Například výraz $A^nB^nC^n$ není bezkontextový ani regulární, ale je rekurzivně spočetný.

1.4 Rekurzivně spočetný jazyk

Formální jazyk je rekurzivně spočetný, když lze:

- přijmout **(ne)deterministickým Turingovým strojem**
- generovat **neomezenou gramatikou**

Například (problém zastavení)L = {[R, w] : Turingův stroj R se pro vstup w zastaví}.

2 Turingovy stroje

Turingův stroj se skládá z **řídící jednotky**, **neomezené čtecí pásky** rozdělené do buněk a **čtecí hlavy**. Čtecí hlava se umí pohybovat oběma směry, či posečkat na stejném místě.
Formální definice Turingova stroje je:

uspořádaná **sedmice: R = (Q, Σ, G, δ, q_0, B, F)**.

- Q je konečná neprázdná množina stavů
- Σ je konečná vstupní abeceda
- G je konečná neprázdná pracovní abeceda ($\Sigma \subset$ G)
- δ je přechodová funkce (liší se u jednotlivých TS)
- q_0 počáteční stav
- B $\in$ (G bez Σ) je prázdný symbol (BLANK)
- F je množina koncových stavů (F $\subseteq$ Q)

Na začátku vypočtu se nachází TS v počátečním symbolu q_0, páska je vyplněna BLANK znaky, na "prvních" buňkách pásky je zapsán vstup a čtecí hlava ukazuje na první buňku vstupu.
Konfigurace obecně je prvek $< q, s, n >$, kde q je aktuální stav, s je nejmenší souvislá část pásky obsahující všechny neprázdné symboly a n je pozice čtecí hlavy.

2.1 Deterministický Turingův stroj

Má následující přechodovou funkci δ:

δ je zobrazení z (Q bez F) x G do Q x G x {-1, 0, 1} ($(Q \setminus F) \times G \to Q \times G \times \{-1, 0, 1\}$)
Je-li TS v **nekoncovém** stavu a z pásky **čte** nějaký symbol z pracovní abecedy, poté **přejde** do dalšího stavu, na pásku **zapíše** nějaký symbol a čtecí hlava se **posune**.
Je-li TS v **koncovém** stavu, **zastaví se** a dál **nepokračuje**.

- pro každý symbol má jasně dán přechod
- pro špatnou kombinaci stroj skončí s chybou a vstup nepřijme
- TS se zastaví právě tehdy, když přejde do koncového stavu

2.2 Nedeterministický Turingův stroj

Má následující přechodovou funkci δ:

δ je zobrazení z (Q bez F) x G do množiny všech podmnožin množiny Q x G x {-1, 0, 1} ($(Q \setminus F) \times G \to \mathcal{P}(Q \times G \times \{-1, 0, 1\})$)

Na rozdíl od DTS může mít v daném stavu několik přechodů pro daný symbol. NTS si tedy může vybrat, jakým způsovem bude ve výpočtu pokračovat.

2.3 Porovnání síly výkonu TS

NTS je stejně výkonný jako DTS.

2.4 Formální zápis

(DTS) δ(S, a) = (A, c, 1) = ze stavu S na symbol a se přesune do stavu A, zapíše symbol C a posune se o 1.
(NTS) δ(S, a) = {(A, c, 1), (B, r, -1)}

2.5 Přijímání řetězce

NTS **přijme** vstupní řetězec, pokud existuje aspoň jedna posloupnost přechodů, kdy TS přejde do koncového stavu a páska je na konci výpočtu prázdná.
NTS **nepřijme** vstupní řetězec, pokud každá posloupnost skončí jako u DTS.

DTS **přijme** řetězec pokud přejde do koncového stavu a páska je po skončení vyplněná prázdnými symboly BLANK.
DTS **nepřijme** řetězec pokud:

- dojde-li během výpočtu k chybě v podobě nedefinovaného přechodu
- pro daný vstup neskončí (zacyklí se)
- přejde do koncového stavu, ale páska není po ukončení výpočtu prázdná

2.6 Lineárně omezený TS

- nesmí použít neomezený počet buněk na pásce
- na začátku výpočtu si zvolí konstantu K a daný TS se během výpočtu může pohybovat pouze na K*(délka vstupu) buňkách pásky

2.7 Vícepáskový TS

- má více pásek a více čtecích hlava
- jednopáskové a více páskové TS jsou stejně výkonné

2.8 Kódovanání TS

- zakódování přechodové funkce TS do řetězce nad jeho abecedou
- nekonečná paměť TS lze zakódovat do konečného řetězce
- výsledná množina stavů je konečná, abeceda je konečná i pravidla jsou konečné

2.9 Univerzální TS

- dostane na vstupu zakódovaný TS a řetězec w
- univerzální TS pak simuluje výpočet TS nad řetězcem w
- R_u tedy vstup přijme (nepřijme) právě tehdy, když jej příjme (nepříjme) R
- Formální zápis: L(R_u) = L_n, kde L_n = {[R,w], TS R přijmá řetězec w }

2.10 Rozhodování jazyka

TS R rozhoduje jazyk L, jestli-že jej příjmá a výpočet se pro každé slovo zastaví.
Pro w∈L ⇒ přejde do koncového stavu a páska **je** prázdná.
Pro w∉L ⇒ přejde do koncového stavu a páska **není** prázdná.
Tedy pro ∀w∉L se TS **zastaví**.

3 Třídy problémů P, NP, NP-těžký, NP-úplný

3.1 Rozhodovací problém

Rozhodovací problém je takový problém, na který je odpovězeno Ano nebo Ne. Rozlišují se instance Ano-instance a Ne-instance pro použití TS se instance namapují na {1,0}*. Všechny Ano-instance tvoří jazyk L_a. TS řeší rozhodovací problém, pokud rozhodne L_a.

3.2 Optimalizační problém

Optimalizační problém je problém, který hledá v nějakém ohledu optimální řešení. Pro lepší názornost se používá rozhodovací verze problému. Optimalizační a rozhodovací verze jsou výpočetně stejně náročné.

3.3 Rozhodnutelné problémy

(Ne)Rozhodnutelné problémy jsou problémy, pro které existuje algoritmus, který je řeší. Nerozhodnutelný problém je ten, který není rozhodnutelný.
Rozhodnutelný problém odpovídá rekurzivnímu jazyku.
Nerozhodnutelný problém odpovídá nerekurzivním jazykům.

3.4 Třída P

Třída rozhodovacích problémů, které lze řešit v polynomiálně omezeném čase deterministickým Turingovým strojem.

3.5 Třída NP

Třída rozhodovacích problémů, které lze řešit v polynomiálně omezeném čase na nedeterministickým Turingovým stroji. Všechny P problémy patří do NP.

3.6 Třída NP-těžký

Problém, na který lze převést všechny problémy ze třídy NP. Sám NP-těžký problém nemusí patřit do třídy NP. Jeden takový problém lze převést na jiný pomocí polynomiální redukce.

3.7 NP-úplný

Je NP-těžký a patří do skupiny NP. Jsou to nejtěžší problémy ze třídy NP. Využívají se v kryptografii. Pokud by byl nalezen polynomiálně deterministický algoritmus pro libovolnou NP-Úplnou úlohu, všechny NP problémy by byly řešitelné.

3.8 Polynomiální redukce

$\leq_p$: proces který převádí problém A→B (A$\leq_p$B). Dostane na vstup instanci problému A (I_A) a jako výstup vrátí v polynomiálním čase instanci problému B (I_B) se stejnou pravdivostní hodnotou. Je-li splněno I_A pak je i I_B.