Formální jazyky

Hierarchie jazyků, regulární jazyky, bezkontextové jazyky, uzávěrové vlastnosti jazyků. Reprezentace jazyků (gramatiky, regulární výrazy, automaty), převody mezi jednotlivými reprezentacemi.

Základné pojmy

Abeceda je libovolná konečná množina znaků (písmen, symbolů).

Slovo nad abecedou Σ je libovolná konečná posloupnost znaků této abecedy.

Jazyk nad abecedou Σ je libovolná množina slov nad Σ.

Hierarchie jazyků a gramatik

	Tvar pravidiel
typ 0 – frázové gramatiky	na tvar pravidel se nekladou žádné omezující požadavky
typ 1 – kontextové gramatiky (context-sensitive grammar, CSG)	každé její pravidlo je tvaru $\alpha \rightarrow \beta$, kde $\lvert\alpha\rvert \leq \lvert\beta\rvert$ s eventuální výjimkou pravidla $S \rightarrow \varepsilon$, pokud se S nevyskytuje na pravé straně žádného pravidla
typ 2 – bezkontextové gramatiky (context-free grammar, CFG)	každé její pravidlo je tvaru $A \rightarrow \alpha$, kde $\lvert\alpha\rvert \geq 1$ s eventuální výjimkou pravidla $S \rightarrow \varepsilon$, pokud se S nevyskytuje na pravé straně žádného pravidla
typ 3 – regulární gramatiky	Každé její pravidlo je tvaru $A \rightarrow aB$ nebo $A \rightarrow a$ (A,B jsou liboolné neterminály, a je libovolný terminál) s eventuální výjimkou pravidla $S \rightarrow \varepsilon$, pokud se S nevyskytuje na pravé straně žádného pravidla

Třídy jazyků

Jazyky	Gramatiky (typ)	Automaty
rekursivně spočetné	frázové (typ 0)	Turingovy stroje
rekursivní	-	úplné Turingovy stroje
kontextové	kontextové (typ 1)	lineárně ohraničené TM
bezkontextové	bezkontextové (typ 2)	zásobníkové automaty
deterministické CFL	-	deterministické PDA
regulární	regulární (typ 3)	Konečné automaty

Třída na nižším řádku je vždy vlastní podtřídou třídy na vyšším řádku.

Příklad: Každý regulární jazyk je zároveň bezkontextový, ALE každý bezkontextový není zároveň regulární (resp. existují bezkontextové jazyky co nejsou regulární).

Regulární jazyky

Jazyk L je regulární, právě když:

- může být vygenerován **regulární gramatikou** (tzn. existuje regulární gramatika G taková, že $L(G) = L$),
- je akceptovaný nějakým **deterministickým konečným automatem** (tzn. existuje deterministický konečný automat M takový, že $L(M) = L$),
- je akceptovaný nějakým **nedeterministickým konečným automatem** (tzn. existuje nedeterministický konečný automat M takový, že $L(M) = L$),
- může být popsán **regulárním výrazem** (tzn. existuje regulární výraz RE takový, že $L(RE) = L$)

Lemma o vkládání (pumping lemma)

Nechť L je regulární jazyk. Pak existuje $n \in \mathbb{N}$ takové, že libovolné slovo $w \in L$, délky alespoň n, lze psát ve tvaru $w = xyz$, kde $|xy| \leq n$, $y \neq \varepsilon$ a $xy^i z \in L$ pro každé $i \in \mathbb{N}_0$.

Lemma o vkládání je **nutnou** (nikoliv postačující) podmínkou pro regularitu jazyka a lze jej použít pro důkaz toho, že nějaký jazyk **není** regulární (NE pro důkaz toho, že jazyk je regulární!!!). **Obměnou** (v téhle části by určitě měla zaznít obměna a ne negace, uvědomte si, že PL je vlastně akorát taková větší implikace) lemmatu o vkládání lze ukázat, že jazyk není regulární:

- pro libovolné $n \in \mathbb{N}$ (pumpovací konstanta), dále pevné
- existuje takové $w \in L$, délky alespoň n, pro které platí, že
- při libovolném rozdělení slova w, splňujícím $w = xyz$, $|xy| \leq n$ a $y \neq \varepsilon$
- existuje alespoň jedno $i \in \mathbb{N}_0$ takové, že $xy^i z \notin L$

Potom z lemmatu o vkládání plyne, že L není regulární.

Myhill-Nerodova věta

Myhill-Nerodova věta představuje nutnou a postačující podmínku pro regularitu jazyka.

Pro formulaci M-N věty potřebujeme několik pomocných pojmů:

Definice: Nechť Σ je abeceda a nechť $\sim$ je ekvivalence na Σ^*. Řekněme, že $\sim$ je **zprava invariantní (pravá kongruence)**, pokud pro každé $u, v, w \in \Sigma^*$ platí $u \sim v \Longrightarrow uw \sim vw$. Index $\sim$ je počet tříd rozkladu $\Sigma^*/\sim$ (pokud je těchto tříd nekonečně mnoho, klademe index $\sim$ roven ∞).

Definice: Nechť L je libovolný (ne nutně regulární) jazyk nad abecedou Σ. Na množině Σ^* definujeme relaci $\sim_L$ zvanou **prefixová ekvivalence** pro L takto:

$u \sim_L v \Leftrightarrow \forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L.$

Tedy $\sim_L$ obsahuje právě ty dvojice (u, v) které mají tu vlastnost, že po připojení libovolného w vzniklá slova uw, vw budou do jazyka L patřit buď obě, nebo ani jedno z nich.

Myhill-Nerodova věta: Nechť L je jazyk nad Σ, pak tato tvrzení jsou ekvivalentní:

1. L je rozpoznatelný konečným automatem.
2. L je sjednocením některých tříd rozkladu určeného pravou kongruencí na Σ^* s konečným indexem.
3. Relace $\sim_L$ má konečný index.

Bezkontextové jazyky

Jazyk L je bezkontextový, právě když:

- existuje **bezkontextová gramatika** G taková, že $L(G) = L$ (neboli která ho generuje)

- je akceptován nějakým **zásobníkovým automatem** (existuje zásobníkový automat M, takový, že $L(M) = L$)

Lemma o vkládání pro bezkontextové jazyky

Nechť L je CFL. Pak existují $p, q \in N$ (závisející na L) taková, že každé slovo $z \in L$ delší než p lze psát ve tvaru $z = uvwxy$, kde

- alespoň jedno ze slov v, x je neprázdne (tj. $vx \neq \varepsilon$),
- $|vwx| \leq q$ a
- $uv^i wx^i y \in L$ pro každé $i \in \mathbb{N}_0$.

Použití lemmatu

- pro libovolné $n \in \mathbb{N}$ (pumpovací konstanta), dále pevné
- existuje takové $z \in L$, délky alespoň n, pro které platí, že
- při libovolném rozdělení slova z splňující $z = uvwxy$, $|vwx| \leq n$ a $vx \neq \varepsilon$
- existuje alespoň jedno $i \in \mathbb{N}_0$ takové, že $uv^i wx^i y \notin L$

Potom z lemmatu o vkládání plyne, že L není bezkontextový.

Vlastnosti jazyků

Uzávěrové vlastnosti jazyků

Třída jazyků $\mathcal{L}$ (množina jazyků s určitými vlastnostmi) je uzavřená na n-ární operaci o, pokud pro libovolné jazyky $L_1, \ldots, L_n$ patřící do $\mathcal{L}$ platí, že také jazyk $o(L_1, \ldots, L_n)$ patří do $\mathcal{L}$.

Třída regulárních jazyků je uzavřena na:

- **sjednocení** ($L_1 \cup L_2$),
- **průnik** ($L_1 \cap L_2$),
- **rozdíl** ($L_1 \setminus L_2$),
- **komplement** ($co - L$),
- **zřetězení** ($L_1.L_2$),
- **iteraci** (L^*),
- **pozitivní iteraci** (L^+),
- **zrcadlový obraz** (L^R).

Třída bezkontextových jazyků je uzavřena vzhledem k operacím:

- **sjednocení**
- **zřetězení**
- **iterace**
- **pozitivní iterace**
- **průnik s regulárním jazykem**

Třída bezkontextových jazyků není uzavřena vzhledem k operacím:

- **průnik**
- **doplněk**
- **rozdíl**

Rozhodnutelné problémy

Rozhodnutelné problémy pro třídu **regulárních jazyků**:

- **ekvivalence**: jsou M a M' ekvivalentní? (platí $L(M) = L(M')$?)
- **inkluze** (jazyků): platí $L(M) \subseteq L(M')$?
- **příslušnost** (slova k jazyku): je-li dáno $w \in \Sigma^*$, platí $w \in L(M)$?
- **prázdnost** (jazyka): je $L(M) = \emptyset$?
- **univerzalita** (jazyka): je $L(M) = \Sigma^*$?
- **konečnost** (jazyka): je $L(M)$ konečný jazyk?

Rozhodnutelné problémy pro třídu **bezkontextových jazyků**:

- **příslušnost**: Ke každé CFG je možné sestrojit PDA a můžeme tak ověřit jestli daný PDA přijme nebo zamítne slovo w.
- **prázdnost**: Stačí ověřit zda $S \in N_e$. (N_e je množina použitelných symbolů vzhledem k typu I.)
- **konečnost**: Ke každé CFG lze sestrojit čísla m, n taková, že $L(G)$ je nekonečný, právě když existuje slovo $z \in L(G)$ takové, že $m < |z| < n$.

Reprezentace jazyků

Gramatiky

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

- N je neprázdná konečná množina neterminálních symbolů (**neterminálů**),
- Σ je konečná množina terminálních symbolů (**terminálů**) taková, že $N \cap \Sigma = \emptyset$; $N \cup \Sigma = V$ je množina **všech symbolů** gramatiky,
- $P \subseteq V^*NV^* \times V^*$ je konečná množina **pravidel**. Pravidlo obvykle zapisujeme ve tvaru $\alpha \to \beta$ (čteme „alfa přepiš na beta“),
- $S \in N$ je **počáteční neterminál**, neboli kořen gramatiky.

Bezkontextová gramatika (context-free grammar, CFG) G je čtveřice (N, Σ, P, S), kde:

- N je neprázdná konečná množina **neterminálních symbolů**,
- Σ je konečná množina **terminálních symbolů** taková, že $N \cap \Sigma = \emptyset$; (značení: $N \cup \Sigma = V$)
- $P \subseteq N \times V^*$ je konečná množina **pravidel**.
- $S \in N$ je **počáteční neterminál**.

Konstrukce bezkontextové gramatiky

Každou bezkontextovou gramatiku, která reprezentuje neprázdný bezkontextový jazyk, lze dále upravit do tzv. kanonických tvarů - lépe se s nimi pracuje, jsou přehlednější, vyžadují je některé algoritmy atd. Kanonické tvary bezkontextových gramatik jsou následující:

- redukované CFG
- gramatiky bez ε-pravidel
- gramatiky bez jednoduchých pravidel
- gramatiky bez levé rekurze
- normální formy CFG (např. GNF,CNF, Backusova-Naurova forma)

Symbol $x \in (N \cup \Sigma)$ je **nepoužitelný** v CFG $G = (N, \Sigma, P, S)$, právě když v G neexistuje derivace tvaru $S \Rightarrow^* wXy \Rightarrow^* wxy$ pro nějaká $w, x, y \in \Sigma^*$.

Řekneme, že G je redukovaná, jestliže neobsahuje žádné nepoužitelné symboly.

Nepoužitelnost typu I.: Neexistuje terminální řetěz w takový, že $A \Rightarrow^* w$.

Nepoužitelnost typu II.: Neexistují řetězy x, y takové, že $S \Rightarrow^* xAy$.

Řekneme, že CFG $G = (N, \Sigma, P, S)$ je **bez ε-pravidel** právě když buď

1. neobsahuje žádné ε-pravidlo (tj. pravidlo tvaru $A \to \varepsilon$) nebo
2. v P existuje právě jedno ε-pravidlo $S \to \varepsilon$ a S se nevyskytuje na pravé straně žádného pravidla z P.

Jednoduchým pravidlem nazýváme každé pravidlo tvaru $A \to B$, kde $A, B \in N$.

CFG $G = (N, \Sigma, P, S)$ se nazývá **necyklická**, právě když neexistuje $A \in N$ takový, že $A \Rightarrow^+ A$.

G se nazývá **vlastní**, právě když je bez nepoužitelných symbolů, bez ε-pravidel a necyklická.

Ke každému neprázdnému bezkontextovému jazyku existuje vlastní bezkontextová gramatika, která jej generuje.

Neterminál A v CFG $G = (N, \Sigma, P, S)$ se nazývá rekursivní jestliže v G existuje derivace $A \Rightarrow^+ \alpha A \beta$. Je-li $\alpha = \varepsilon$ resp. $\beta = \varepsilon$, pak A se nazývá levorekursivní resp. pravorekursivní. CFG bez levorekursivních terminálů se nazývá nelevorekursivní.

Normální formy bezkontextových gramatik

Chomského normální forma

Bezkontextová gramatika $G = (N, \Sigma, P, S)$ je v Chomského normální formě (CNF) $\Leftrightarrow$ je bez ε-pravidel a každé pravidlo z P má jeden z těchto tvarů:

1. $A \to BC, \ B, C \in N$
2. $A \to a, \ a \in \Sigma$

3. $S \to \varepsilon$

Každý bezkontextový jazyk lze generovat bezkontextovou gramatikou v Chomského normální formě.

Greibachové normální forma

Bezkontextová gramatika $G = (N, \Sigma, P, S)$ je v Greibachové normální formě (GNF) právě když:

- G je bez ε-pravidel
- každé pravidlo z P je tvaru $A \to a\alpha$, kde $a \in \Sigma$ a $\alpha \in N^*$.

Každý bezkontextový jazyk lze generovat bezkontextovou gramatikou v Greibachové normální formě.

Regulární výrazy

Množina **regulárních výrazů** nad abecedou Σ , označovaná $RE(\Sigma)$, je definována induktivně takto:

1. ε, $\emptyset$ a a pro každé $a \in \Sigma$ jsou regulární výrazy nad Σ (tzv. základní regulární výrazy).
2. Jsou-li E, F regulární výrazy nad Σ, jsou také $(E.F)$, $(E + F)$ a (E^*) regulární výrazy nad Σ.
3. Každý regulární výraz vznikne po konečném počtu aplikací kroků 1–2.

Každý regulární výraz E nad abecedou Σ popisuje (jednoznačně určuje) jazyk $L(E)$ nad abecedou Σ podle těchto pravidel:

- $L(\varepsilon) \stackrel{\text{def}}{=} \{\varepsilon\}$
- $L(\emptyset) \stackrel{\text{def}}{=} \emptyset$
- $L(a) \stackrel{\text{def}}{=} \{a\}$ pro každé $a \in \Sigma$
- $L(E.F) \stackrel{\text{def}}{=} L(E).L(F)$
- $L(E + F) \stackrel{\text{def}}{=} L(E) \cup L(F)$
- $L(E^*) \stackrel{\text{def}}{=} L(E)^*$

Ekvivalenci mezi regulárními výrazy a konečnými automaty shrnuje Kleeneho věta:

Libovolný jazyk je popsatelný regulárním výrazem, **právě když** je rozpoznatelný konečným automatem.

Automaty

Konečný automat (Finite Automaton, FA) M je pětice $(Q, \Sigma, \delta, q_0, F)$, kde

- Q je neprázdná konečná množina **stavů**,
- Σ je konečná množina **vstupní abeceda,**
- $\delta : Q \times \Sigma \to Q$ je parciální **přechodová funkce**,
- $q_0 \in Q$ je **počáteční (iniciální) stav**,
- $F \subseteq Q$ je množina **koncových (akceptujících) stavů.**

Nedeterministický konečný automat (NFA) je $M = (Q, \Sigma, \delta, q_0, F)$, kde význam všech složek je stejný jako v definici FA s výjimkou přechodové funkce δ. Ta je definována jako (totální) zobrazení $\delta : Q \times \Sigma \to 2^Q$.

Pre každý NFA $M = (Q, \Sigma, \delta, q_0, F)$ existuje ekvivalentní deterministický FA (DFA).

Nedeterministický zásobníkový automat (PushDown Automaton, PDA) je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde

- Q je konečná množina, jejíž prvky nazýváme **stavy**,
- Σ je konečná množina, tzv. **vstupní abeceda**,
- Γ je konečná množina, tzv. **zásobníková abeceda**,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to P_{Fin}(Q \times \Gamma^*)$, tzv. (parciální) **přechodová funkce** (zápis $P_{Fin}(Q \times \Gamma^*)$ značí množinu všech konečných podmnožin množiny $Q \times \Gamma^*$).
- $q_0 \in Q$ je **počáteční stav**,
- $Z_0 \in \Gamma$ je **počáteční symbol v zásobníku**,
- $F \subseteq Q$ je množina **koncových stavů.**

Rozšířený zásobníkový automat je sedmice $R = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde

- všechny symboly až na δ mají stejný význam jako v definici PDA,
- δ je zobrazením **z konečné podmnožiny** množiny $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*$ **do konečné podmnožiny** množiny $Q \times \Gamma^*$.

Více o automatech v otázce Automaty.

Převody mezi jednotlivými reprezentacemi

Vztah mezi konečnými automaty a regulárními gramatikami

Třídy jazyků, které lze generovat regulárními gramatikami, resp. rozpoznat konečnými automaty, jsou si rovny. To znamená, že k dané regulární gramatice lze sestrojit ekvivalentní deterministický konečný automat a naopak.

Regulární gramatika → konečný automat

Ke každé regulární gramatice $G = (N, \Sigma, P, S)$, existuje nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(G) = L(M)$.

Myšlenka důkazu:

Stavy automatu budou odpovídat neterminálům gramatiky, tj. pro každý neterminál A bude existovat stav $\overline{A}$. Pro každé pravidlo $A \to aB$ přidáme do $\delta(\overline{A}, a)$ stav $\overline{B}$. Abychom se mohli vypořádat také s pravidly tvaru $C \to a$, zavedeme speciální koncový stav q_f, který přidáme do $\delta(\overline{C}, a)$. Počáteční stav bude $\overline{S}$, koncový stav q_f a případně také $\overline{S}$, pokud gramatika obsahuje pravidlo $S \to \varepsilon$.

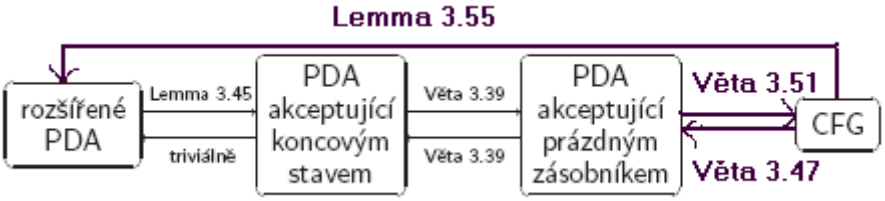
Konečný automat → regulární gramatika

Pro každý konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ existuje regulární gramatika $G = (N, \Sigma, P, S)$ taková, že $L(M) = L(G)$.

Myšlenka důkazu:

Neterminály budou odpovídat stavům, pravidla budou simulovat přechodovou funkci. Je tu však jeden problém – pokud automat přijímá prázdné slovo (tj. počáteční stav je koncovým stavem), musí každá ekvivalentní gramatika nutně obsahovat pravidlo $S \to \varepsilon$, kde S je kořen. Pak se ale S nesmí vyskytovat na pravé straně žádného pravidla. Přitom je ale možné, že některé přechody automatu končí v počátečním stavu a mají být simulovány pravidly, které mají na pravé straně S, což by vedlo ke konfliktu s požadavkem pravostranných výskytů S. Tento problém vyřešíme tak, že k původnímu automatu přidáme nový iniciální stav (původní zachováme, akorát už není iniciální) a **Z NĚJ** (do něj žádný přechod nevede a nepovede) uděláme přechody stejně jako je měl původní iniciální. Potom už můžeme použít intuitivní přístup, kdy neterminály odpovídají stavům a přechody terminálům.

Zásobníkové automaty a bezkontextové jazyky

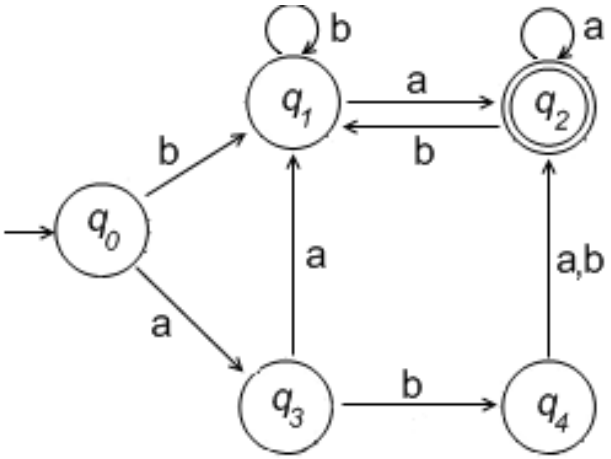


Automaty

Konečný automat, zásobníkový automat. Varianty automatů, metody akceptování. Nedeterminismus, determinizace automatů. Syntaktická analýza shora dolů a zdola nahoru.

Konečný automat

Abstraktním modelem konečně stavových systémů (i nekonečně stavových, což ale není jejich hlavní poslání) jsou tzv. ***konečné automaty***. Konečný automat je vybaven konečně stavovou řídící jednotkou (tj. konečnou pamětí), čtecí hlavou a páskou, na které je zapsané vstupní slovo – viz obrázek. Na začátku výpočtu je hlava umístěna na nejlevějším políčku pásky. Automat na základě přečteného symbolu a momentálního stavu svůj stav změní a posune čtecí hlavu o jedno políčko vpravo. Výpočet končí, pokud se automat „zablokuje“, nebo přečte celé vstupní slovo. Slovo zapsané na pásce je automatem *akceptováno*, pokud je celé přečteno a výsledný stav je některý z předem určených *koncových* stavů. Množina všech slov, která daný konečný automat M akceptuje, tvoří jazyk akceptovaný automatem M.



Formální definice:
(Deterministický) konečný automat (Finite Automaton, (D)FA) M je pětice M = (Q, Σ, δ, q_0, F), kde:

- Q je neprázdná konečná množina stavů
- Σ je konečná množina vstupních symbolů, nazývaná také vstupní abeceda
- $\delta: Q \times \Sigma \rightarrow Q$ je parciální přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $F \subseteq Q$ je množina koncových stavů

Dále je třeba zavést ***rozšířenou přechodovou funkci*** $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$, definovanou induktivně vzhledem k délce slova ze Σ*:

$\hat{\delta}(q, \varepsilon) = q$ pro každý stav q ∈ Q

$$\hat{\delta}(q, wa) = \begin{cases} \delta\left(\hat{\delta}(q,w), a\right) - \ pokud\ je\ \hat{\delta}(q,w)\ i\ \delta\left(\hat{\delta}(q,w), a\right)\ definováno \\ \perp\ - jinak \end{cases}$$

Značení:

- $\perp$ znamená, že funkce není definována
- $\hat{\delta}(q, w) = p$ znamená, že automat M přejde ze stavu q „pod slovem“ w (postupným přečtením zleva doprava) do stavu

Jazyk L(M) přijímaný konečným automatem M, je tvořen právě všemi slovy, pod kterými automat přejde z počátečního stavu do některého z koncových stavů:

- $L(M) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$

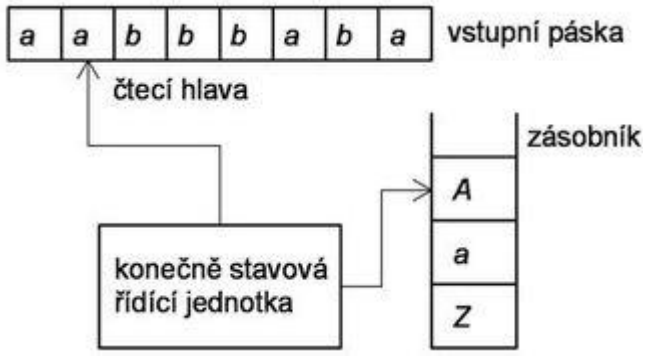
Jazyk, který je rozpoznatelný konečným automatem, se nazývá ***regulární jazyk.***

Konečné automaty M a M‘ jsou **ekvivalentní**, pokud L(M) = L(M‘).

Reprezentace konečných automatů **– tabulkou, přechodovým grafem (viz nahoře)**

Zásobníkový automat

Tak jako k regulárním gramatikám existují konečné automaty, které rozpoznávají právě jazyky generované těmito gramatikami, tak i ke gramatikám bezkontextovým existují ve výše uvedeném smyslu ekvivalentní automaty – tzv. zásobníkové automaty (push-down automata – PDA). PDA si lze představit jako (nedeterministický – nedeterminismus zaveden níže) konečný automat s tím, že navíc obsahuje (pomocnou) paměť (a díky ní i další zdroj nedeterminismu), která pracuje jako zásobník (push-down store) a jejíž velikost není shora omezena – je tzv. potenciálně nekonečná v následujícím smyslu: v každém okamžiku (konečného) výpočtu je sice konečná (tj. v zásobníku je uloženo jen konečně mnoho symbolů), ale kdykoli ji můžeme rozšířit o další konečný počet paměťových míst (přidat další symboly na zásobník).



Ze vstupní pásky, na níž je zapsáno slovo nad jistou vstupní abecedou, lze pouze číst a čtecí hlava se pohybuje jen vpravo. Automat může na vrchol (vrchol zásobníku vlevo) zásobníku ukládat symboly (opět z jisté abecedy) a takto uložené symboly může následně číst s tím, že smí číst pouze z vrcholu zásobníku. Přečtený symbol je z vrcholu odstraněn (tj. systém LIFO). Jinými slovy, nelze číst do hloubi zásobníku, aniž by přečtené symboly nebyly odstraněny.

Formální definice:
(Nedeterministický) zásobníkový automat M je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde:

- Q je konečná množina stavů automatu
- Σ je konečná množina vstupních symbolů, tzv. vstupní abeceda
- Γ je konečná množina zásobníkových symbolů, tzv. zásobníková abeceda
- $\delta: Q \times (\Sigma \cup \{\varepsilon\} \times \Gamma) \rightarrow P_{fin}(Q \times \Gamma^*)$, tzv. (parciální) přechodová funkce, kde P_{fin} značí množinu všech konečných podmnožin $Q \times \Gamma^*$
- $q_0 \in Q$ je počáteční stav

- $Z_0 \in \Gamma$ je počáteční zásobníkový symbol
- $F \subseteq Q$ je množina koncových stavů

Vnitřní konfigurací PDA *M* nazveme libovolný prvek $(q, \gamma) \in Q \times \Gamma^*$, kde q je momentální stav PDA M a γ je celý obsah zásobníku s vrcholem psaným vlevo). Konfigurací nazveme libovolný prvek $(p, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$ (navíc i ω – dosud nepřečtená část vstupního řetězu).

Krok výpočtu: binární relace $\vdash_M$ definovaná jako:

$(p, aw, Z\alpha) \vdash_M (q, w, \gamma\alpha) \overset{def}{\iff} \exists (q, \gamma) \in \delta(p, a, Z)\ pro\ a \in \Sigma \cup \{\varepsilon\}$
- (po přečtení slova *a* přejdeme ze stavu *p* do stavu *q* a obsah zásobníku *Zα* nahradíme *γα*)

Rozšířený zásobníkový automat – sedmice $R = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde všechny symboly až na δ mají stejný význam jako v definici PDA. (vrchol stejný jako PDA)
δ je zobrazením z konečné podmnožiny množiny $(Q\ x\ (\Sigma \cup \{\varepsilon\}) \times \Gamma^*)$ do konečných podmnožin množiny $Q \times \Gamma^*$.
Pojmy konfigurace a akceptování zůstávají beze změn. Krok výpočtu: $(p, aw, \gamma\alpha) \vdash_M (q, w, \gamma'\alpha)$ $\overset{def}{\iff} \exists (q, \gamma') \in \delta(p, a, \gamma)\ pro\ a \in \Sigma \cup \{\varepsilon\}$.

Ke každému rozšířenému PDA existuje ekvivalentní PDA.

Metody akceptování:

- Jazyk akceptovaný PDA *M* ***koncovým stavem*** definujeme jako:
  L(M) = $\{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \varepsilon, \alpha), kde\ q_f \in F, \alpha \in \Gamma^* \}$

- Jazyk akceptovaný PDA *M* ***prázdným zásobníkem*** definujeme jako:
  L(M) = $\{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon), kde\ q \in Q \}$

Každý výpočet pro vstupní slovo ω tedy začíná v konfiguraci (q_0, ω, Z_0), tj. ve vnitřní konfiguraci (q_0,Z_0) a dosud nečteným vstupem. Způsobů akceptování je obecně více: každá akceptující (finální) konfigurace je charakterizována zcela přečteným vstupním slovem, tj. (q, ε, α), vzájemně se však tyto způsoby liší tím, co prohlásíme za akceptující vnitřní konfiguraci. Ve výše uvedené definici jsou to po řadě totální stavy z $F \times \Gamma^*$, resp. $Q \times \{\varepsilon\}$. Další způsoby akceptování lze definovat tak, že za akceptující vnitřní konfigurace prohlásíme např. prvky z $F \times \{\varepsilon\}$ - akceptování koncovým stavem a prázdným zásobníkem.

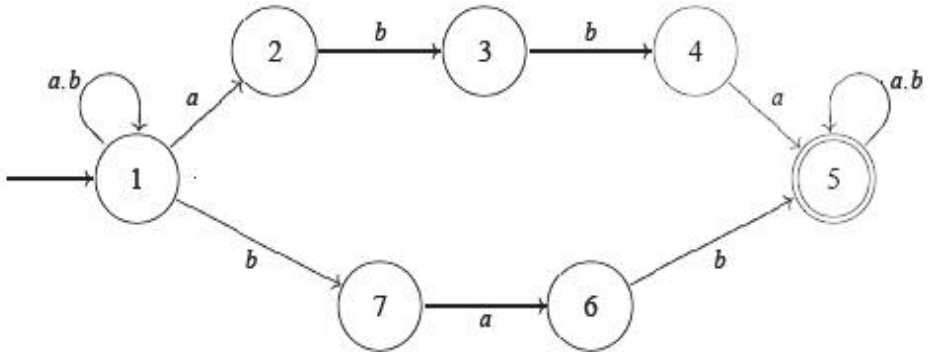
Ekvivalence dvou způsobů akceptování: Pro každý jazyk L platí: L = L(N) pro nějaký PDA N $\Leftrightarrow L = L_e(M)$ pro nějaký PDA M

Nedeterminismus

Nedeterministický konečný automat (NFA) je zařízení, které je velimi podobné konečnému automatu, avšak rozdíl je v tom, že NFA nemusí mít pro daný stav a vstupní symbol určen následující stav jednoznačně (lze si to představit tak, že z jednoho uzlu může vycházet více hran se stejným návěštím). Není tedy předem jasné, do jakého stavu se automat dostane po zpracování daného slova *w*, neboť automat si během výpočtu může „vybírat“ jeden z možných následujících stavů. Slovo *w* bude akceptováno, pokud *alespoň jeden* z možných výpočtů nad slovem *w* skončí v koncovém stavu.

Nedeterminismus je velmi silný popisný aparát, který často umožňuje zachytit strukturu jazyka elegantním a přirozeným způsobem.

Navrhnout DFA, který rozpoznává L = $\{w \in \{a,b\}^* \mid w$ obsahuje podslovo $abba$ nebo $bab\}$, není zcela triviální. Naopak NFA lze zkonstruovat poměrně snadno (viz obrázek níže).



Nedeterminismus lze dobře využít jako popisný prostředek, nemá však vliv na výpočetní sílu konečných automatů.
Ke každému NFA totiž ve skutečnosti existuje ekvivalentní DFA, který lze algoritmicky zkonstruovat (viz. v sekci *determinizace*)

Formální definice:
NFA je pětice $M = (Q, \Sigma, \delta, q_0, F)$, kde význam všech složek je stejný jako u FA, s výjimkou přechodové funkce – ta je definovaná jako (totální) zobrazení: $\delta: Q \times \Sigma \rightarrow 2^Q$

Na deterministické automaty tedy můžeme pohlížet jako na speciální případ NFA, kdy pro každé $q \in Q$ a $a \in \Sigma$ je množina $\delta(q,a)$ nejvýše jednoprvková.

Zavedeme ***rozšířenou přechodovou funkci*** $\hat{\delta}: Q \times \Sigma^* \rightarrow 2^Q$:

- $\hat{\delta}(q, \varepsilon) = \{q\}$
- $\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q,w)} \delta(p,a)$

Jazyk akceptovaný NFA *M* je definovaný jako: L(*M*) = $\{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

Další varianty automatů

Automaty s ε-kroky:

Model NFA je možné rozšířit o tzv. ε-kroky. Automat pak může svůj stav za určitých okolností změnit samovolně, tj. bez přečtení vstupního symbolu. Tato schopnost je formálně popsána pomocí ε-kroků, které si lze na přechodových grafech představit jako hrany, jejíž návěštím je prázdné slovo. Přes tyto hrany může automat během výpočtu na slově *w* měnit svůj stav bez toho, aby ze vstupu cokoliv přečetl – mezi přečtením dvou po sobě jdoucích symbolů z *w* může provést libovolné konečné množství ε-přechodů. Ke každému takovému automatu existuje ekvivalentní NFA bez ε-kroků.

Formální definice:
NFA s *ε*-kroky je pětice $M = (Q, \Sigma, \delta, q_0, F)$, kde význam všech složek je stejný jako v definici NFA s výjimkou přechodové funkce – ta je definována jako totální zobrazení: $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$

Rozšířenou přechodovou funkci $\hat{\delta}$ ovšem musíme definovat odlišným způsobem, nejprve zavedeme ***funkci*** $\boldsymbol{D_\varepsilon}: Q \rightarrow 2^Q$, která pro daný stav *p* vrací množinu stavů, kterých může M dosáhnout z *p* bez toho, aby četl vstup. Pro dané $p \in Q$ je $D_\varepsilon(p)$ nejmenší množina $X \subseteq Q$ taková, že platí:

- $p \in X$
- Pokud $q \in X$ a $r \in \delta(q, \varepsilon)$, pak také $r \in X$

Nyní již můžeme definovat ***rozšířenou přechodovou funkci*** $\hat{\delta}$: $Q\ x\ \Sigma^*\ \rightarrow 2^Q$ takto:

- $\hat{\delta}(q, \varepsilon) = D_\varepsilon(q)$
- $\hat{\delta}(q, wa) = \bigcup_{p\ \epsilon\ \hat{\delta}(q,w)} D_\varepsilon(\delta(p, a))$

Jazyk přijímaný automatem M s ε-kroky je definován stejně jako v případě NFA, tedy:
L(*M*) = $\{\omega\ \epsilon\ \Sigma^* |\ \hat{\delta}(q_0, \omega) \cap\ F\ \neq\ \emptyset\}$

Deterministický zásobníkový automat (DPDA)

Formální definice:
Řekněme, že PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je *deterministický*, jestliže jsou splněny tyto podmínky:

1. Pro všechna $q \in Q$ a $Z\ \in\ \Gamma$ platí: kdykoliv $\delta(q, \varepsilon, Z) \neq\ \emptyset$, pak $\delta(q, a, Z) =\ \emptyset$ pro všechna $a\ \in\ \Sigma$
2. Pro žádné $q \in Q, Z\ \in\ \Gamma, a\ \in (\Sigma\ \cup \{\varepsilon\})$ neobsahuje $\delta(q, a, Z)$ více než jeden prvek.

Podmínka 1 vylučuje možnost volby mezi krokem nezávislým na vstupním symbolu (ε-krokem) a krokem, kdy se ze vstupu čte.
Podmínka 2 říká, že jak v případě čtecího kroku, tak i pro ε-krok, neexistuje více než jedna varianta, jak dále pokračovat.

Řekněme, že L je *deterministický bezkontextový jazyk (DCFL)*, právě když existuje DPDA *M* takový, že: L = L(*M*)

Determinizace

Transformace NFA na ekvivalentní DFA:

Vstup: Nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$
Výstup: Ekvivalentní deterministický konečný automat $M' = (Q', \Sigma, \delta', \{q_0\}, F')$ bez nedosažitelných stavů s totální přechodovou funkcí. (Výstupem algoritmu nemusí být minimální automat)

$Q' := \{\{q_0\}\};\ \delta' :=\ \emptyset; F' :=\ \emptyset; Done :=\ \emptyset;$
while $(Q'\ Done)\ \neq\ \emptyset$ ***do***
    $M := libovolný\ prvek\ množiny\ Q' - Done;$
    if $M\ \cap F\ \neq\ \emptyset$ ***then***
        $F' := F' \cup \{M\};$
    end if
    for all $a\ \in\ \Sigma$ ***do***
        $N := \bigcup_{p\ \epsilon M} \delta(p, a);$
        $Q' := Q' \cup \{N\};$
        $\delta' :=\ \delta' \cup \{((M, a), N)\};$
    end for
    $Done := Done\ \cup \{M\};$
end while

Příklad:

Zadaný automat:	a	b	c
→1	{2,3}	{3,4}	{1}
←2	{3}	{4}	{2}

3	{1,2,3}	{1}	{3,4}
4	{1}	{1}	{3,4}

Výsledek:	a	b	c
→1	2,3	3,4	1
←2,3	1,2,3	1,4	2,3,4
3,4	1,2,3	1	3,4
←1,2,3	1,2,3	1,3,4	1,2,3,4
1,4	1,2,3	1,3,4	1,3,4
←2,3,4	1,2,3	1,4	2,3,4
1,3,4	1,2,3	1,3,4	1,3,4
←1,2,3,4	1,2,3	1,3,4	1,2,3,4

Syntaktická analýza shora dolů

Syntaktická analýza se snaží zkonstruovat derivační strom pro dané slovo (vstup syntaktického analyzátoru, posloupnost symbolů). Metoda shora dolů (Top-Down) se nejprve zabývá nejvyšší úrovní derivačního stromu a postupně prochází derivační strom dolů s využitím formálních pravidel gramatiky. To znamená, že derivační strom věty konstruujeme od kořene (je ohodnocen startovacím symbolem gramatiky) směrem dolů k listům, zleva doprava, podle levé derivace. Syntaktickou analýzu shora dolů (její deterministickou verzi) používají pro svou práci LL syntaktické analyzátory. Analyzátorem je PDA (případně rozšířený PDA).

Příklad

Pro danou gramatiku G navrhněte zásobníkový automat, který provádí syntaktickou analýzu shora dolů a proveďte analýzu slova *abaa*.

$$G = (\{S, A, B\}, \{a, b\}), P, S)$$
$$P = \{\ S \rightarrow \varepsilon \mid abSA,$$
$$A \rightarrow AaB \mid aB \mid a,$$
$$B \rightarrow aSS \mid bA\ \}$$

Nejdříve sestrojíme zásobníkový automat A, který provádí syntaktickou analýzu.

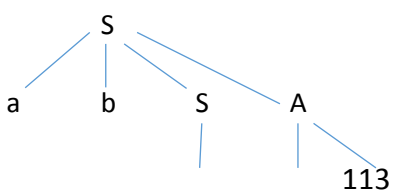
$A = (\{q\}, \{a, b\}, \{a, b, S, A, B\}, \delta, q, S, \emptyset)$ (akc. prázdným zásobníkem)

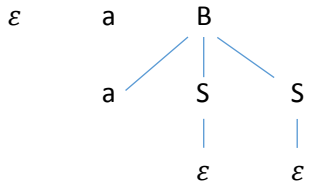
A k němu pravidla následujícím způsobem:

Čtecí pravidla
$\delta(q, a, a) = \{(q, \varepsilon)\}$
$\delta(q, b, b) = \{(q, \varepsilon)\}$

Nečtecí pravidla
$\delta(q, \varepsilon, S) = \{(q, \varepsilon), (q, abSA)\}$
$\delta(q, \varepsilon, A) = \{(q, AaB), (q, aB), (q, a)\}$
$\delta(q, \varepsilon, B) = \{(q, aSS), (q, bA)\}$

Pro lepší postup si sestrojíme derivační strom zadaného slova:





A nyní podle sestrojeného stromu aplikujeme pravidla zásobníkového automatu A:

(q, abaa, S) ⊢ (q, abaa, abSA) ⊢ (q, baa, bSA) ⊢ (q, aa, SA) ⊢ (q, aa, A) ⊢ (q, aa, aB) ⊢ (q, a, B) ⊢ (q, a, aSS) ⊢ (q, ε, SS) ⊢ (q, ε, S) ⊢ (q, ε, ε)

Zjistili jsme, že námi vytvořený PDA slovo *abaa* **akceptuje** a proto ho gramatika G **generuje**.

Syntaktická analýza zdola nahoru

Syntaktická analýza zdola nahoru je postup syntaktické analýzy, který identifikuje nejdříve základní jednotky a z nich potom odvozuje strukturu s vyšší úrovní uspořádání. Název strategie vychází z konceptu derivačního stromu, který má nejzákladnější jednotky ve spodní části, ze kterých postupně vnikají struktury složené. Na vrcholu stromu je posléze samostatná jednotka nebo symbol zahrnující všechny informace, které jsou analyzovány. V této strategii zpracování dochází k analýze obou jazyků jak „přirozeného jazyka“, tak „počítačového jazyka“, což je v kontrastu se syntaktickou analýzou shora dolů, ve které jsou komplexní jednotky děleny do jednodušších částí až do okamžiku, kdy jsou veškeré informace zpracovány.

Příklad

Pro danou gramatiku G navrhněte rozšířený zásobníkový automat, který provádí syntaktickou analýzu zdola nahoru a proveďte analýzu slova *abaa*.

$$G = (\{S, A, B\}, \{a, b\}), P, S) \text{ (zásobník se sklápí doprava)}$$
$$P = \{\ S \rightarrow \varepsilon \mid abSA,$$
$$A \rightarrow AaB \mid aB \mid a,$$
$$B \rightarrow aSS \mid bA\ \}$$

Nejdříve zase sestrojíme zásobníkový automat A, který provádí syntaktickou analýzu.

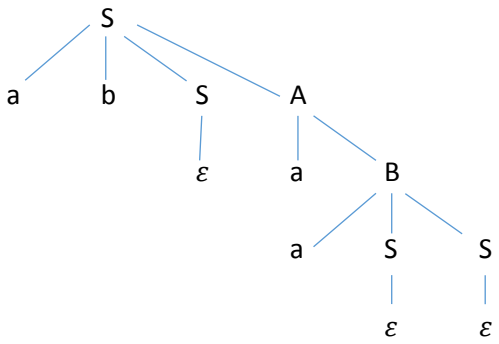
$A = (\{q\ \ , r\}, \{a, b\}, \{a, b, S, A, B, \bot\}, \delta, q, \bot, \{r\})$ (akc. prázdným zásobníkem)

A k němu pravidla následujícím způsobem:

Čtecí pravidla
$\delta(q_0, a, \varepsilon) = \{(q_0, a)\}$
$\delta(q_0, b, \varepsilon) = \{(q_0, b)\}$

Nečtecí pravidla
$\delta(q_0, \varepsilon, \varepsilon) = \{(q_0, S)\}$
$\delta(q_0, \varepsilon, abSA) = \{(q_0, S)\}$
$\delta(q_0, \varepsilon, AaB) = \{(q_0, A)\}$
$\delta(q_0, \varepsilon, aB) = \{(q_0, A)\}$
$\delta(q_0, \varepsilon, a) = \{(q_0, A)\}$
$\delta(q_0, \varepsilon, aSS) = \{(q_0, B)\}$
$\delta(q_0, \varepsilon, bA) = \{(q_0, B)\}$
$\delta(q_0, \varepsilon, \bot\ S) = \{(q_1, \varepsilon)\}$

Opět sestrojíme derivační strom zadaného slova:



A nyní podle sestrojeného stromu aplikujeme pravidla zásobníkového automatu A:

$(q_0, abaa, \perp) \vdash (q_0, baa, \perp a) \vdash (q_0, aa, \perp ab) \vdash (q_0, aa, \perp abS) \vdash (q_0, a, \perp abSa) \vdash (q_0, \varepsilon, \perp abSaa) \vdash (q_0, \varepsilon, \perp abSaaS) \vdash (q_0, \varepsilon, \perp abSaaSS) \vdash (q_0, \varepsilon, \perp abSaB) \vdash (q_0, \varepsilon, \perp abSA) \vdash (q_0, \varepsilon, \perp S) \vdash \underline{(q_1, \varepsilon, \varepsilon)}$

Zjistili jsme, že gramatika G **generuje** slovo *abaa* neboť námi vytvořený PDA A ho **akceptuje**.