Automaty a gramatiky

TIN071

Marta Vomlelová

marta@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~marta

February 29, 2024

Organizační záležitosti

- Přednáška:
  - moodle https://dl1.cuni.cz/course/index.php?categoryid=337
    - login jako do SIS
  - video nahrávky přednášek z roku 2019 https://is.mff.cuni.cz/prednasky/prednaska/NTIN071/1
    - login jako do SIS
    - poslední dvě přednášky jsou nové, nejsou tam.
- Cvičení:
  - vyzkoušíte si prakticky sestrojit automaty a gramatiky
  - zažijete příklady, což je něco jiného, než je přečíst,
  - potřebujete zápočet, který udělují **výhradně** cvičící.
- Zkouška:
  - **Zápočet je nutnou podmínkou účasti na zkoušce** (kromě předtermínů).
  - Písemná příprava a ústní část
  - Porozumění látce + schopnost formalizace
    - Orientace v Chomského hierarchii, automatech, gramatikách, (ne)determinizmu,
    - Napište definici, formulujte větu, popište ideu důkazu, algoritmus,
    - zařaďte jazyk do Chomského hierarchie a svou odpověď dokažte.
- Komunikace v konzultačních hodinách: Úterý 15:00 v S303.
  - Jindy bez záruky. Na maily se snažím odpovídat, na dotazy po přednášce také.

Požadavky ke zkoušce

- **Zápočet je nutnou podmínkou účasti na zkoušce**.
- Zkouška sestává z písemného testu a ústní části.
- **Písemný test** bude sestávat ze dvou až tří otázek, které korespondují sylabu přednášky, ověřují schopnosti získané na cvičení a znalost definic, vět a algoritmů z přednášky.

! Po dobu písemného testu musí být veškeré přinesené poznámky, přípravy, mobily, počítače apod. uloženy v uzavřeném batohu. V případě opomenutí poznámek na židli, stole, otevřeném batohu apod. je zkouška okamžitě hodnocena 'nevyhověl' a student v ní dále nepokračuje.

- **Požadavky ústní části** Ústní část bude vycházet z písemného testu, zpravidla budete dotázáni na vysvětlení-zdůvodnění-příklady k tvrzením v písemné části. Ústní část může být doplněna otázkou v rozsahu sylabu přednášky s písemným testem nesouvisející.

Zdroje a literatura

- J.E. Hopcroft, R. Motwani, J.D. Ullman: *Introduction to Automata Theory, Languages, and Computations*, Addison–Wesley
- M. Sipser: Introduction to the Theory of Computation, Cengage Learning, 2013
- M. Chytil: *Automaty a gramatiky, SNTL Praha, 1984*

⇒ moodle https://dl1.cuni.cz/course/index.php?categoryid=337

  - kde jsou tyto slajdy
  - moodle testy (které ale netestují zdůvodnění a důkazy).

⇒ cvičení.

Pohled do historie

- Počátky
  - první formalizace pojmu algoritmus Ada, Countess of Lovelace 1852
  - intenzivněji až s rozvojem počítačů ve druhé čtvrtině 20. století
  - co stroje umí a co ne?
  - Church, Turing, Kleene, Post, Markov
- Polovina 20. století
  - neuronové sítě (1943)
  - konečné automaty (Finite Automata) (Kleene 1956 neuronové sítě $\approx$ FA)
- 60. léta 20. století
  - gramatiky (Chomsky)
  - zásobníkové automaty
  - formální teorie konečných automatů.
- 80. léta 20. století
  - popularita tříd časové a prostorové složitosti.

Cíl přednášky

- Osvojit si abstraktní model výpočetních zařízení,
- vnímat, jak drobné změny v definici vedou k velmi odlišným třídám,
- zažít skutečnost algoritmicky nerozhodnutelných problémů,
- rychlý úvod do složitosti $P \subseteq NP \subseteq PSPACE \subseteq NPSPACE \subseteq EXPTIME$.

Automaty a gramatiky – dva způsoby popisu

Turingovy stroje Vícepáskové, nedeterministické	$\xleftrightarrow{\mathcal{L}_0}$	gramatiky Typu 0
lineárně omezené automaty	$\xleftrightarrow{\mathcal{L}_1}$	kontextové gramatiky monotónní gramatiky
zásobníkové automaty	$\xleftrightarrow{\mathcal{L}_2}$	bezkontextové gramatiky
konečné automaty DFA, NFA, ϵNFA	$\xleftrightarrow{\mathcal{L}_3}$	regulární gramatiky

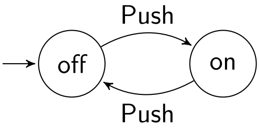
Praktické využití

- Zamyšlení nad korektností programu, algoritmu, překladače,
- zpracování přirozeného jazyka,
- překladače:
  - lexikální analýza,
  - syntaktická analýza,
- návrh, popis, verifikace hardware
  - integrované obvody
  - stroje
  - automaty
- realizace pomocí software
  - hledání výskytu slova v textu (grep)
  - verifikace systémů s konečně stavy.

Jednoduché příklady konečných automatů

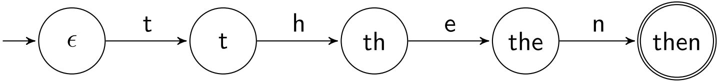
- Návrh a verifikace integrovaných obvodů.

Konečný automat modelující spínač on/off .



- Lexikální analýza

Konečný automat rozpoznávající slovo `then`.



Definition 1.1 (Deterministický konečný automat)

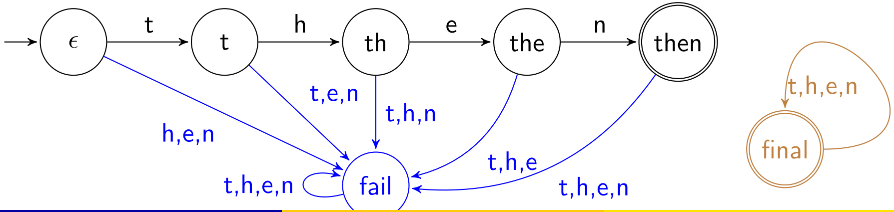
Deterministický konečný automat (DFA) $A = (Q, \Sigma, \delta, q_0, F)$ sestává z:

1. konečné množiny **stavů**, zpravidla značíme Q
2. konečné neprázdné množiny **vstupních symbolů (abecedy)**, značíme Σ
3. **přechodové funkce**, zobrazení $Q \times \Sigma \to Q$, značíme δ, která bude reprezentovaná hranami grafu nebo tabulkou
4. **počátečního stavu** $q_0 \in Q$, vede do něj šipka 'odnikud',
5. a **množiny koncových (přijímajících) stavů** (final states) $F \subseteq Q$, označených dvojitým kruhem či šipkou 'ven'.

Úmluva: Pokud pro některou dvojici stavu a písmene není definovaný přechod, přidáme nový stav *fail* a přechodovou funkci doplníme na totální přidáním šipek do *fail*.

Pokud je množina F prázdná, a je vyžadovaná neprázdná, přidáme do ní i Q

nový stav *final* do kterého vedou jen přechody z něj samého $\forall s \in \Sigma$: $\delta(final, s) = final$.

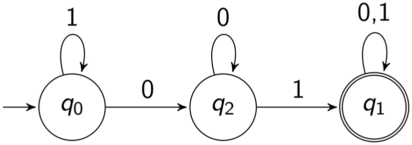


Popis konečného automatu

Example 1.1

Automat A přijímající $L = \{x01y : x, y \in \{0,1\}^*\}$.

- Stavový diagram (graf) Automat $A = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$.



- tabulka
  - řádky: stavy + přechody
  - sloupce: písmena vstupní abecedy

δ	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Abeceda, slova, jazyky

Definition 1.2 (Slovo, ϵ,λ,Σ^*,Σ^+,jazyk)

Mějme neprázdnou množinu symbolů Σ.

- **Slovo** je konečná (i prázdná) posloupnost symbolů $s \in \Sigma$, **prázdné slovo** se značí ϵ nebo λ.
- **Množinu všech slov v abecedě Σ** značíme Σ^*,
- množinu všech neprázdných slov v značíme Σ^+.
- **jazyk** $L \subseteq \Sigma^*$ je množina slov v abecedě Σ.

Definition 1.3 (operace zřetězení, mocnina, délka slova)

Nad slovy Σ^* definujeme operace:

- **zřetězení slov** $u.v$ nebo uv
- **mocnina** (počet opakování) u^n ($u^0 = \epsilon$, $u^1 = u$, $u^{n+1} = u^n.u$)
- **délka slova** $|u|$ ($|\epsilon| = 0$, $|auto| = 4$).
- **počet výskytů** $s \in \Sigma$ ve slově u značíme $|u|_s$ ($|zmrzlina|_z = 2$).

Rozšířená přechodová funkce

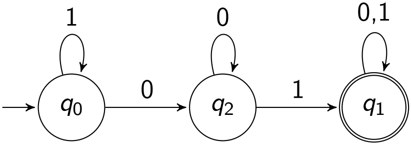
Definition 1.4 (rozšířená přechodová funkce)

Mějme přechodovou funkci $\delta : Q \times \Sigma \rightarrow Q$.
Rozšířenou přechodovou funkci δ^*: $Q \times \Sigma^* \rightarrow Q$ (tranzitivní uzávěr δ) definujeme induktivně:

- $\delta^*(q, \epsilon) = q$
- $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$ pro $x \in \Sigma, w \in \Sigma^*$.

Pozn. Pokud se v textu objeví δ aplikované na slova, míní se tím δ^*.

$\delta^*(q_0, 1100) = q_2$, $\delta^*(q_0, 1100111111111001) = q_1$



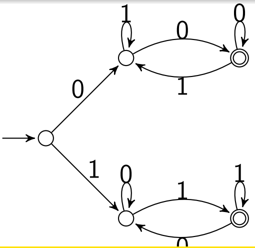
Jazyky rozpoznatelné konečnými automaty

Definition 1.5 (jazyky rozpoznatelné konečnými automaty, regulární jazyky)

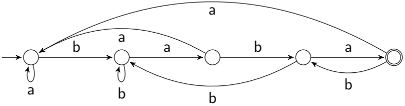
- **Jazykem rozpoznávaným (akceptovaným, přijímaným)** deterministickým konečným automatem $A = (Q, \Sigma, \delta, q_0, F)$ nazveme jazyk $L(A) = \{w \mid w \in \Sigma^* \;\&\; \delta^*(q_0, w) \in F\}$.
- Slovo w je **přijímáno** automatem A, právě když $w \in L(A)$.
- Jazyk L je **rozpoznatelný** konečným automatem, jestliže existuje konečný automat A takový, že $L = L(A)$.
- Třídu jazyků rozpoznatelných konečnými automaty označíme $\mathcal{F}$, nazveme **regulární jazyky**.

Example 1.2 (regulární jazyky)

- $L = \{w \mid w = xux, w \in \{0,1\}^*, x \in \{0,1\}, u \in \{0,1\}^*\}$.



Příklady regulárních jazyků

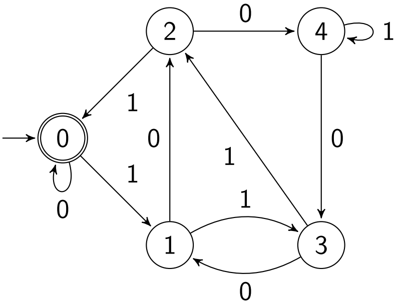


Example 1.3 (regulární jazyk)

- $L = \{w \mid w = ubaba, w \in \{a,b\}^*, u \in \{a,b\}^*\}$.

Example 1.4 (regulární jazyk)

- $L = \{w | w \in \{0,1\}^* \& w$ je binární zápis čísla dělitelného $5\}$.



Example 1.5 (!NEregulární jazyk)

- $L = \{0^n 1^n | w \in \{0,1\}^*, n \in \mathbb{N}\}$ NENÍ regulání jazyk.

Iterační (pumping) lemma pro regulární jazyky

Theorem 1.1 (!Iterační (pumping) lemma pro regulární jazyky)

Mějme regulární jazyk L. Pak existuje konstanta $n \in \mathbb{N}$ (závislá na L) tak že každé $w \in L; |w| \geq n$ můžeme rozdělit na tři části, $w = xyz$, že:

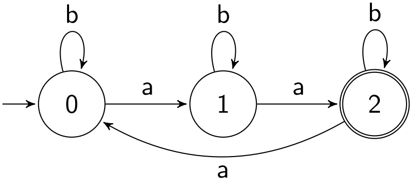
- $y \neq \epsilon$
- $|xy| \leq n$
- $\forall k \in \mathbb{N}_0$, *slovo* xy^kz *je také v* L.

Example 1.6

- Lemma řeklo: $n = 3$.
- $abbbba = a(b)bbba$;
  $\forall i \geq 0; a(b)^i bbba \in L(A)$.
- $aaaaba = (aaa)aba$;
  $\forall i \geq 0; (aaa)^i aba \in L(A)$.
- aa nelze pumpovat,

  ale $|aa| < n$.

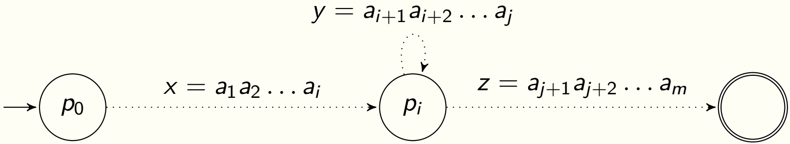
Automat A



Důkaz iteračního lematu pro regulární jazyky

Proof: iteračního lematu pro regulární jazyky

- Mějme regulární jazyk L, pak existuje DFA A s n stavy, že $L = L(A)$.
- Vezměme libovolné slovo $a_1 a_2 \dots a_m = w \in L$ délky $m \geq n$, $a_i \in \Sigma$.
- Definujme: $\forall i \; p_i = \delta^*(q_0, a_1 a_2 \dots a_i)$. Platí $p_0 = q_0$.
- Máme $n + 1$ p_i a n stavů, některý se opakuje, vezměme první takový, tj. $(\exists i, j)(0 \leq i < j \leq n \;\&\; p_i = p_j)$.
- Definujme: $x = a_1 a_2 \dots a_i$, $y = a_{i+1} a_{i+2} \dots a_j$, $z = a_{j+1} a_{j+2} \dots a_m$, tj. $w = xyz$, $y \neq \epsilon$, $|xy| \leq n$.



- Smyčka nad p_i se může opakovat libovolně krát a vstup je také akceptovaný. □

Použití pumping lemmatu

Example 1.7 (Pumping lemma jako hra s oponentem)

Jazyk $L_{eq} = \{w; |w|_0 = |w|_1\}$ slov se stejným počtem 0 a 1 není regulární.

Proof: Jazyk L_{eq} není regulární.

- Předpokládejme že L_{eq} je regulární. Vezměme n z pumping lemmatu.
- Zvolme $w = 0^n 1^n \in L_{eq}$.
- Rozdělme $w = xyz$ dle pumping lemmatu, $y \neq \epsilon$, $|xy| \leq n$.
- Protože $|xy| \leq n$ je na začátku w, obsahuje jen 0.
- Z pumping lemmatu: $xz \in L_{eq}$ (pro $k = 0$). To má ale méně 0 než 1, takže nemůže být v L_{eq}. □

Example 1.8

Jazyk $L = \{0^i 1^i; i \geq 0\}$ není regulární.

Aplikace pumping lemmatu 2

Example 1.9

Jazyk L_{pr} slov 1^p kde p je prvočíslo není regulární.

Proof: L_{pr} slov 1^p kde p je prvočíslo není regulární.

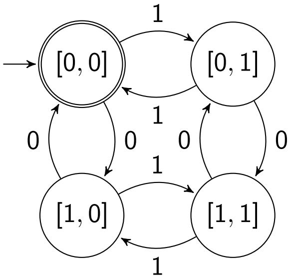
- Předpokládejme že L_{pr} je regulární. Vezměme n z pumping lemmatu. Zvolme prvočíslo $p \geq n+2$, označme $w = 1^p$.
- Rozložme $w = xyz$ dle pumping lemmatu, nechť $|y| = m$. Pak $|xz| = p - m$.
- $xy^{p-m}z \in L_{pr}$ z pumping lemmatu, ale $|xy^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m)m = (m+1)(p-m)$ není prvočíslo (žádný z činitelů není 1). $\square$

Příklad - 'součin' automatů

Example 1.10

$L = \{w \mid w \in \{0,1\}^*, |w|_0 = 2k \& |w|_1 = 2\ell, k, \ell \in \mathbb{N}_0\}$, tj.

- sudý počet 0
- a zároveň sudý počet 1.



δ	0	1
$* \rightarrow [0,0]$	$[1,0]$	$[0,1]$
$[0,1]$	$[1,1]$	$[0,0]$
$[1,0]$	$[0,0]$	$[1,1]$
$[1,1]$	$[0,1]$	$[1,1]$

Příklad (špatného) protokolu pro elektronický převod peněz

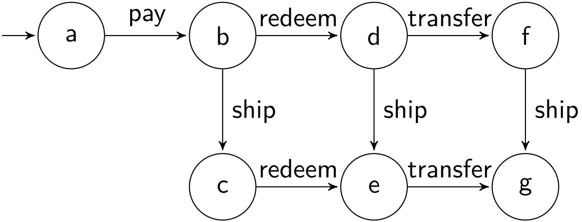
- Tři zúčastnění: zákazník, obchod, banka.
- Pro jednoduchost jen jedna platba (soubor 'money').

Example 1.11

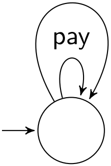
Zákazník poskytne obchodu číslo kreditní karty, obchod si vyžádá peníze od banky a pošle zboží zákazníkovi. Zákazník má možnost zablokovat kartu a žádat zrušení transakce.
Pět událostí:

- Zákazník může zadat číslo karty **pay**.
- Zákazník může kartu zablokovat **cancel**.
- Obchod může poslat **ship** zboží zákazníkovi.
- Obchod může vyžádat **redeem** peníze od banky.
- Banka může převést **transfer** peníze obchodu.

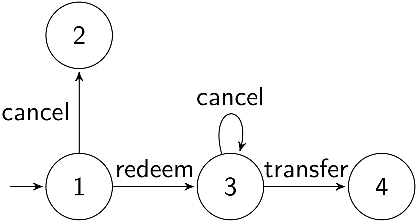
(Neúplný) konečný automat pro bankovní příklad



Obchod



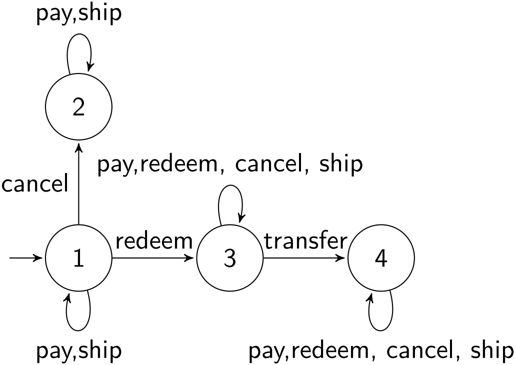
Zákazník



Banka

Hrana pro každý vstup

- Můžeme vyžadovat, aby automat provedl akci pro každý vstup. Obchod přidá hranu pro každý stav do sebe samého označenou *cancel*.
- Zákazník by neměl shodit bankovní automat opětovným zaplacením *pay*, proto přidáme smyčku *pay*. Podobně s ostatními akcemi.

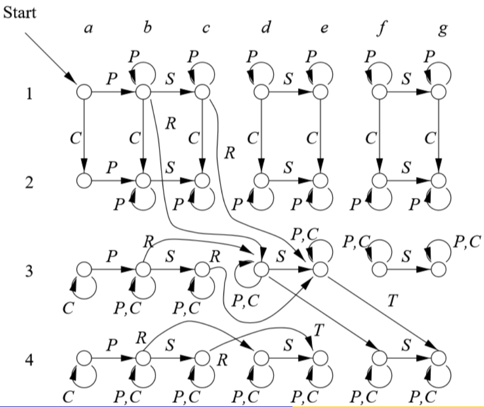


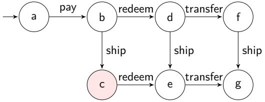
Úplnější automat pro banku.

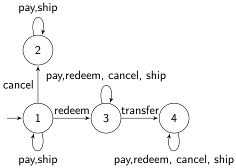
Součin automatů

- Součin automatů pro banku a obchod má stavy dvojice $B \times O$.
- Hrana v součinu automatů provádí paralelně akce v bance a obchodě. Pokud jednomu chybí akce, bude chybět i součinu automatů.

J.E. Hopcroft, R. Motwani, J.D. Ullman: *Introduction to Automata Theory, Languages, and Computations*, Addison–Wesley



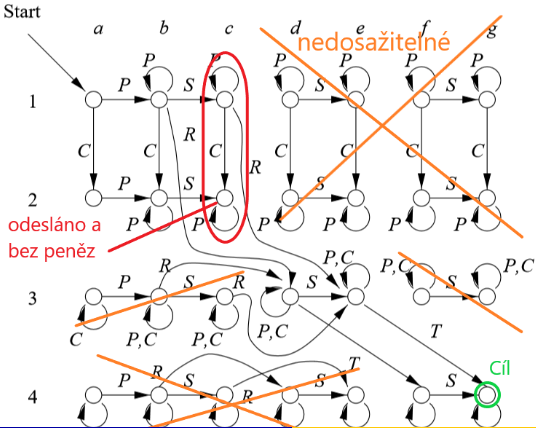


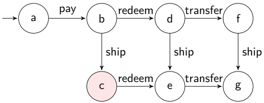


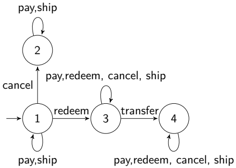
Součin automatů

- Součin automatů pro banku a obchod má stavy dvojice $B \times O$.
- Hrana v součinu automatů provádí paralelně akce v bance a obchodě. Pokud jednomu chybí akce, bude chybět i součinu automatů.

J.E. Hopcroft, R. Motwani, J.D. Ullman: *Introduction to Automata Theory, Languages, and Computations*, Addison–Wesley







Definition 1.6 (Dosažitelné stavy)

Mějme DFA $A = (Q, \Sigma, \delta, q_0, F)$ a $q \in Q$. Řekneme, že stav q je **dosažitelný**, jestliže existuje $w \in \Sigma^*$ takové, že $\delta^*(q_0, w) = q$.

Algorithm: Hledání dosažitelných stavů

Dosažitelné stavy hledáme iterativně.

- Začátek: $M_0 = \{q_0\}$.
- Opakuj: $M_{i+1} = M_i \cup \{q | q \in Q, (\exists p \in M_i,\ \exists x \in \Sigma)\ \delta(p, x) = q\}$
- opakuj dokud $M_{i+1} \neq M_i$.

Proof: Korektnost a úplnost

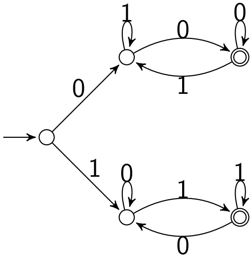
- Korektnost: $M_0 \subseteq M_1 \subseteq \ldots \subseteq Q$ a každé M_i obsahuje pouze dosažitelné stavy.
- Úplnost:
  - nechť q je dosažitelný, tj. $(\exists w \in \Sigma^*)\delta^*(q_0, w) = q$
  - vezměme nejkratší takové $w = x_1 \ldots x_n$ tž. $\delta^*(q_0, x_1 \ldots x_n) = q$
  - zřejmě $\delta^*(q_0, x_1 \ldots x_i) \in M_i$ (dokonce $M_i \setminus M_{i-1}$)
  - tedy $\delta^*(q_0, x_1 \ldots x_n) \in M_n$, tedy $q \in M_n$. $\square$

Dnes jsme probrali

- Definice
  - deterministického konečného automatu $A = (Q, \Sigma, \delta, q_0, F)$
  - jazyka $L \subseteq \Sigma^*$
  - jazyka rozpoznávaného konečným automatem
    $L(A) = \{w \mid w \in \Sigma^* \ \& \ \delta^*(q_0, w) \in F\}$
- iterační (pumping) lemma pro regulární jazyky
- příklad důkazu ne–regulárnosti jazyka $0^i 1^i$
- příklady regulárních jazyků
- způsob nalezení nedosažitelných stavů.

Konečné automaty, Regulární jazyky

- **Deterministický konečný automat (DFA)** $A = (Q, \Sigma, \delta, q_0, F)$.
- **Jazykem rozpoznávaným (akceptovaným, přijímaným)** konečným automatem $A = (Q, \Sigma, \delta, q_0, F)$ nazveme jazyk $L(A) = \{w | w \in \Sigma^* \& \delta^*(q_0, w) \in F\}$.
- Jazyk L je **rozpoznatelný** konečným automatem, jestliže existuje konečný automat A takový, že $L = L(A)$.
- Třídu jazyků rozpoznatelných deterministickými konečnými automaty označíme $\mathcal{F}$, nazveme **regulární jazyky**.



- Typická otázka na cvičeních, občas i u zkoušky: Je daný jazyk regulární (později i CFL, ...)?

ANO Setrojíte automat (deterministický či nedeterministický).

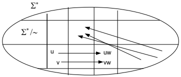
NE Najdete spor s Myhill–Nerodovou větou nebo s Pumping lemmatem.

Kongruence, Myhill–Nerodova věta

Definition 2.1 (kongruence)

Mějme konečnou abecedu Σ a relaci ekvivalence $\sim$ na Σ^* (reflexivní, symetrická, tranzitivní). Potom:

- $\sim$ je **pravá kongruence**, jestliže $(\forall u, v, w \in \Sigma^*) u \sim v \Rightarrow uw \sim vw$.
- je **konečného indexu**, jestliže rozklad $\Sigma^* / \sim$ má konečný počet tříd.
- Třídu kongruence $\sim$ obsahující slovo u značíme $[u]_\sim$, resp. $[u]$.



Example 2.1

- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence,
  - pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'končí stejně jako začíná' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{||}$ 'mají stejný počet znaků' není konečného indexu.

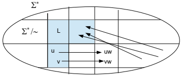
Myhill–Nerodova věta

Theorem 2.1 (Myhill–Nerodova věta)

Nechť L je jazyk nad konečnou abecedou Σ. Potom následující tvrzení jsou ekvivalentní:

a) *L je rozpoznatelný konečným automatem,*

b) *existuje pravá kongruence $\sim$ konečného indexu nad Σ^* tak, že L je sjednocením jistých tříd rozkladu $\Sigma^*/\sim$.*



Proof: Důkaz Myhill–Nerodovy věty ⇒

a)⇒b); tj. automat ⇒ pravá kongruence konečného indexu

- definujeme $u \sim v \equiv \delta^*(q_0, u) = \delta^*(q_0, v)$.
- je to ekvivalence (reflexivní, symetrická, transitivní)
- je to pravá kongruence (z definice δ^*)
- má konečný index (konečně mnoho stavů)

$$L = \{w|\delta^*(q_0, w) \in F\} = \bigcup_{q\in F}\{w|\delta^*(q_0, w) = q\} = \bigcup_{q\in F}[w|\delta^*(q_0, w) = q]_\sim.$$

Theorem (Myhill–Nerodova věta - 'kopie')

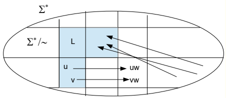
Nechť L je jazyk nad konečnou abecedou Σ. Potom následující tvrzení jsou ekvivalentní:

a) *L je rozpoznatelný konečným automatem,*
b) *existuje pravá kongruence $\sim$ konečného indexu nad Σ^* tak, že L je sjednocením jistých tříd rozkladu $\Sigma^*/\sim$.*

Proof: Důkaz Myhill–Nerodovy věty $\Leftarrow$

b)$\Rightarrow$a); tj. pravá kongruence konečného indexu $\Rightarrow$ automat

- abeceda automatu vezmeme Σ
- za stavy Q volíme třídy rozkladu $\Sigma^*/\sim$
- počáteční stav $q_0 \equiv [\epsilon]$
- koncové stavy $F = \{c_1, \ldots, c_n\}$, kde $L = \bigcup_{i=1,\ldots,n} c_i$
- přechodová funkce $\delta([u], x) = [ux]$ (je korektní z def. pravé kongruence).
- $L(A) = L$



$w \in L \Leftrightarrow w \in \bigcup_{i=1,\ldots,n} c_i \Leftrightarrow w \in c_1 \vee \ldots w \in c_n \Leftrightarrow [w] = c_1 \vee \ldots [w] = c_n \Leftrightarrow [w] \in F \Leftrightarrow w \in L(A)$

$\delta^*([\epsilon], w) = [w]$

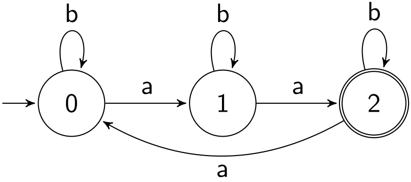
$\square$

Použití Myhill–Nerodovy věty: Konstrukce automatů

Example 2.2

Sestrojte automat přijímající jazyk
$L = \{w | w \in \{a, b\}^* \& \ |w|_a = 3k + 2\}$, tj. obsahuje $3k + 2$ symbolů a.

- $|u|_x$ značí počet symbolů x ve slově u
- definujme $u \sim v \equiv (|u|_a mod\ 3 = |v|_a mod\ 3)$
- třídy ekvivalence 0,1,2
- L odpovídá třídě 2
- a – přechody do následující třídy
- b – přechody zachovávají třídu.



'Pumpovatelný' ne-regulární jazyk

Example 2.3 (Ne–regulární jazyk, který lze pumpovat)

Jazyk $L = \{u | u = a^+ b^i c^i \vee u = b^i c^j, + \in N_{>0}, i, j \in N\}$ není regulární (Myhill–Nerodova věta), ale vždy lze pumpovat první písmeno.

- Předpokládejme, že L je regulární

$\Rightarrow$ pak existuje pravá kongruence $\sim_L$ konečného indexu m, L je sjednocení některých tříd $\Sigma^*/\sim_L$

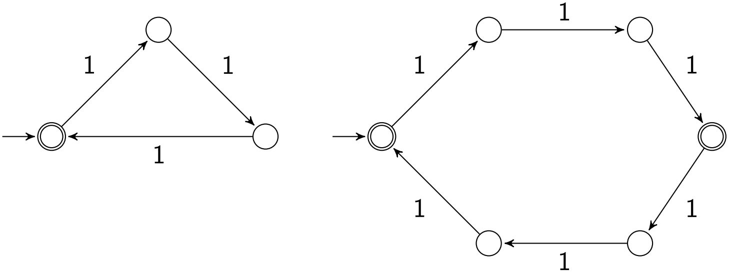
- vezmeme množinu řetězců $S = \{ab, abb, abbb, \ldots, ab^{m+1}\}$
- existují dvě slova $i \neq j$, která padnou do stejné třídy

$i \neq j$	$ab^i \sim ab^j$	
přidáme c^i	$ab^i c^i \sim ab^j c^i$	$\sim$ je kongruence
spor	$ab^i c^i \in L$ & $ab^j c^i \notin L$	s 'L je sjednocení některých tříd $\Sigma^*/\sim_L$

Nejednoznačnost

Automat přijímající daný jazyk není určen jednoznačně.

- Jazyk $L = \{w | w \in \{1\}^* \& |w| = 3k\}$.



Definition 2.2 (automatový homomorfismus)

Nechť A_1, A_2 jsou DFA. Řekneme, že zobrazení $h : Q_1 \to Q_2$ Q_1 na Q_2 je **(automatovým) homomorfismem**, jestliže:

$h(q_{0_1}) = q_{0_2}$	'stejné' počáteční stavy
$h(\delta_1(q,x)) = \delta_2(h(q),x)$	'stejné' přechodové funkce
$q \in F_1 \Leftrightarrow h(q) \in F_2$	'stejné' koncové stavy.

Homomorfismus prostý a na nazýváme **isomorfismus**.

Ekvivalence automatů a homomorfismus

Definition 2.3 (Ekvivalence automatů!)

Dva konečné automaty A, B nad stejnou abecedou Σ jsou **ekvivalentní**, jestli že rozpoznávají stejný jazyk, tj. $L(A) = L(B)$.

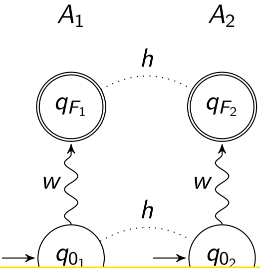
Theorem 2.2 (Věta o ekvivalenci automatů)

Existuje-li homomorfismus konečných automatů A_1 do A_2, pak jsou A_1 a A_2 ekvivalentní.

Proof:

- Pro libovolný řetězec $w \in \Sigma^*$ konečnou iterací
  - $h(\delta_1^*(q, w)) = \delta_2^*(h(q), w)$
- dále: $w \in L(A_1) \Leftrightarrow \delta_1^*(q_{0_1}, w) \in F_1$

$$\Leftrightarrow h(\delta_1^*(q_{0_1}, w)) \in F_2$$
$$\Leftrightarrow \delta_2^*(h(q_{0_1}), w) \in F_2$$
$$\Leftrightarrow \delta_2^*(q_{0_2}, w) \in F_2$$



Redukce a ekvivalence automatů, Tranzitivita

Definition 2.4 (Ekvivalence stavů)

Říkáme, že stavy $p, q \in Q$ konečného automatu A jsou **ekvivalentní** pokud:

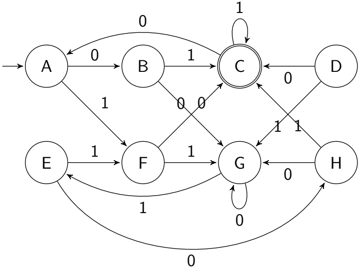
- Pro všechny řetězce $w \in \Sigma^*$; $\delta^*(p, w) \in F$ iff $\delta^*(q, w) \in F$.

Pokud dva stavy nejsou ekvivalentní, říkáme, že jsou **rozlišitelné**.

Example 2.4

Automat na obrázku:

- C a G nejsou ekvivalentní, $\delta^*(C, \epsilon) \in F$ a $\delta^*(G, \epsilon) \notin F$.
- A,G: $\delta^*(A, 01) = C$ je přijímající, $\delta^*(G, 01) = E$ není.
- A,E jsou ekvivalentní – ϵ, $1*$ zřejmé, 0 vede do ne–přijímajících stavů, 01 a 00 se sejdou ve stejném stavu.



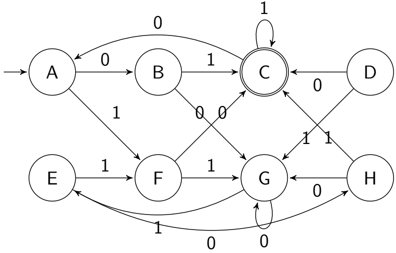
Lemma

Ekvivalence na stavech je tranzitivní.

Algorithm: !Algoritmus hledání rozlišitelných stavů v DFA

Následující algoritmus nalezne rozlišitelné stavy:

- Základ: Pokud $p \in F$ (přijímající) a $q \notin F$, pak je dvojice $\{p, q\}$ rozlišitelná.
- Indukce: Nechť $p, q \in Q$, $a \in \Sigma$ a o dvojici $r, s; r = \delta(p, a)$ a $s = \delta(q, a)$ víme, že jsou rozlišitelné. Pak i $\{p, q\}$ jsou rozlišitelné.
    - opakuj dokud existuje nová trojice $p, q \in Q$, $a \in \Sigma$.



B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

Křížek značí rozlišitelné dvojice. C je rozlišitelné hned, ostatní kromě $\{A, G\}, \{E, G\}$ také. Vidíme tři ekvivalentní dvojice stavů.

Algoritmus hledání rozlišitelných stavů

Přijímající vs. nepřijímající stavy

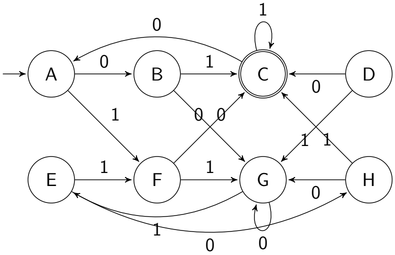
	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

1.krok1: $\delta(q,1) \in F$ pro $q \in \{B, C, H\}$

	A	B	C	D	E	F	G
B	x						
C	x	x					
D		x	x				
E		x	x				
F		x	x				
G		x	x				
H	x		x	x	x	x	x

1.krok0: $\delta(q,0) \in F$ pro $q \in \{D, F\}$

	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G		x	x	x		x	
H	x		x	x	x	x	x



B a G jsou rozlišitelné, $\delta(A,0) = B$, $\delta(G,0) = G$, tj. A,G jsou rozlišitelné.
Obdobně pro E,G vedoucí $\delta(*,0)$ do rozlišitelných stavů H,G.

	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Zůstávají tři ekvivalentní dvojice stavů.

Theorem 2.3

Pokud dva stavy nejsou odlišeny předchozím algoritmem, pak jsou tyto stavy ekvivalentní.

Proof: Koreknost algoritmu

- Uvažujme špatné páry stavů, které jsou rozlišitelné a algoritmus je nerozlišil.
- Vezměme z nich pár p, q rozlišitelný nejkratším slovem $w = a_1 \ldots a_n$.
- Stavy $r = \delta(p, a_1)$ a $s = \delta(q, a_1)$ jsou rozlišitelné kratším slovem $a_2 \ldots a_n$ takže pár není mezi špatnými.
  Tedy jsou 'vykřížkované' algoritmem.
- Tedy v příštím kroku algoritmus rozliší i p, q. $\square$

Čas výpočtu je polynomiální vzhledem k počtu stavů.

- V jednom kole uvažujeme všechny páry, tj. $O(n^2)$.
- Kol je maximálně $O(n^2)$, protože pokud nepřidáme křížek, končíme.
- Dohromady $O(n^4)$.

Algoritmus lze zrychlit na $O(n^2)$ pamatováním stavů, které závisí na páru $\{r, s\}$ a následováním těchto seznamů 'zpátky'.

Testování ekvivalence regulárních jazyků

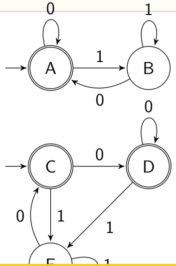
Algorithm: !Testování ekvivalence regulárních jazyků

Ekvivalenci regulárních jazyků L, M testujeme následovně:

- Najdeme DFA A_L, A_M rozpoznávající $L(A_L) = L, L(A_M) = M$, $Q_L \cap Q_M = \emptyset$.
- Vytvoříme DFA sjednocením stavů a přechodů $(Q_L \cup Q_M, \Sigma, \delta_L \cup \delta_M, q_L, F_L \cup F_M)$; zvolíme jeden z počátečních stavů.
- Jazyky jsou ekvivalentní právě když počáteční stavy původních DFA jsou ekvivalentní.

Example 2.5

Uvažujme jazyk $\{\epsilon\} \cup \{0,1\}^* 0$ přijímající prázdné slovo a slova končící 0. Vpravo obrázek dvou DFA a tabulku rozlišitelných stavů.



B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

Minimalizace DFA

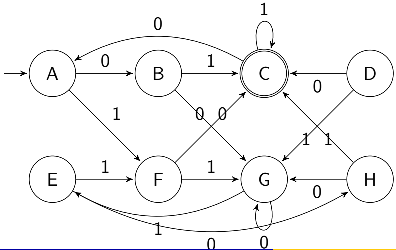
Definition 2.5 (redukovaný DFA, redukt)

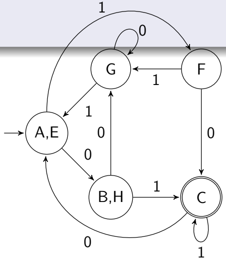
Deterministický konečný automat je **redukovaný**, pokud

- nemá nedosažitelné stavy a
- žádné dva stavy nejsou ekvivalentní
- δ je totální.

Konečný automat B je **reduktem** automatu A, jestliže:

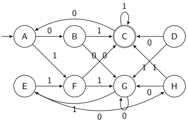
- B je redukovaný a
- A a B jsou ekvivalentní.

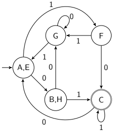




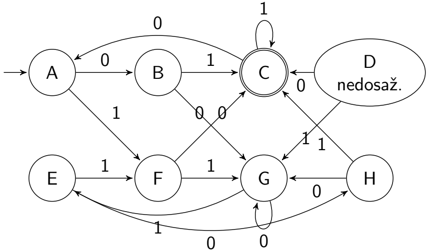
Algorithm: !Algoritmus nalezení reduktu DFA A

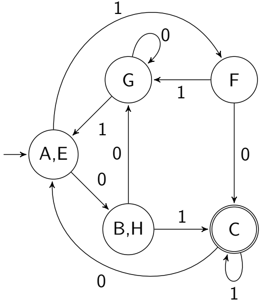
- Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
- Najdeme rozklad zbylých stavů na třídy ekvivalence.
- Konstruujeme DFA B na třídách ekvivalence jakožto stavech. Přechodovou funkci B označíme γ, mějme $S \in Q_B$. Pro libovolné $q \in S$, označíme T třídu ekvivalence $\delta(q, a)$ a definujeme $\gamma(S, a) = T$. Tato třída musí být stejná pro všechny $q \in S$.
- Počáteční stav B je třída obsahující počáteční stav A.
- Množina přijímajících stavů B jsou bloky odpovídající přijímajícím stavům A.





Příklad redukovaného DFA





B	x					
C	x	x				
E		x	x			
F	x	x	x	x		
G	x	x	x	x	x	
H	x		x	x	x	x
	A	B	C	E	F	G

Třídy ekvivalence:

$$\{A, E\}, \{B, H\}, \{C\}, \{F\}, \{G\}$$

Redukty a jejich ekvivalence

Lemma

Každé dva ekvivalentní redukované automaty jsou izomorfní.

Proof.

- Každý stav $q \in Q_1$ je dosažitelný. Najdeme pro něj slovo $q = \delta_1^*(q_{0_1}, w)$
- a definujeme $h(q) = \delta_2^*(q_{0_2}, w)$.
- Lze dokázat, že je h korektně definovaná funkce, zachovává vlastnosti homomorfizmu (q_0, F, δ) a jde o bijekci, tj. je to isomorfizmus.

□

Lemma

Pro každý deterministický konečný automat A, který přijímá alespoň jedno slovo, existuje redukovaný DFA, který je s ním ekvivalentní.

Definition 2.6 (Redukt)

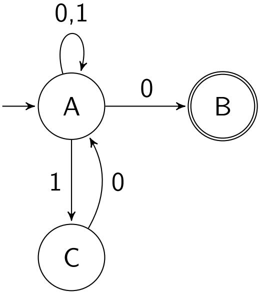
Reduktem DFA A nazýváme redukovaný automat s A ekvivalentní.

Z předchozí věty plyne, že všechny redukty automatu A jsou izomorfní.

Pro nedeterministické FA to tak snadné není

Example 2.6

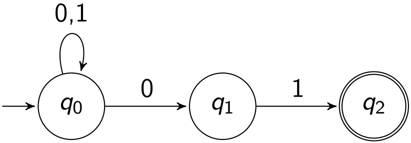
Nedeterministický FA na obrázku můžeme redukovat vypuštěním stavu C.
Stavy $\{A, C\}$ jsou rozlišitelné vstupem 0, takže algoritmus pro DFA redukci nenajde.
Mohli bychom hledat exhauzivním výpočtem.



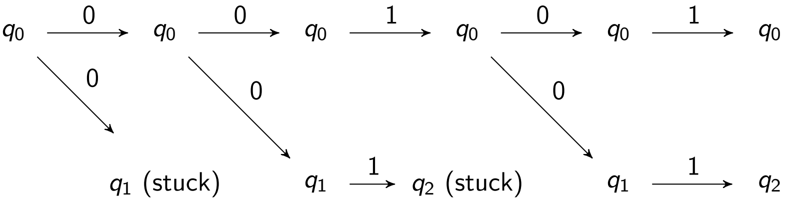
Nedeterministické konečné automaty s ϵ přechody (ϵ-NFA)

Nedeterministický automat může být ve více stavech paralelně. Má schopnost 'uhodnout' něco o vstupu.

NFA přijímající všechna slova končící 01.



NFA zpracovává vstup 00101.



Definition 3.1 (Nedeterministický konečný automat s ϵ přechody (ϵ-NFA))

Nedeterministický konečný automat s ϵ přechody (ϵ-NFA)
$A = (Q, \Sigma, \delta, q_0, F)$ sestává z:

1. konečné množiny **stavů**, zpravidla značíme Q
2. konečné množiny **vstupních symbolů**, značíme Σ
3. **přechodové funkce**, zobrazení $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ vracející podmnožinu Q.
4. **počáteční ho stavu**[a] $q_0 \in Q$,
5. a **množiny koncových (přijímajících) stavů** $F \subseteq Q$.

[a] alternativa: množiny počátečních stavů $S_0 \subseteq Q$

Example 3.1

Tabulka pro ϵ-NFA z předchozího slajdu $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$ je:

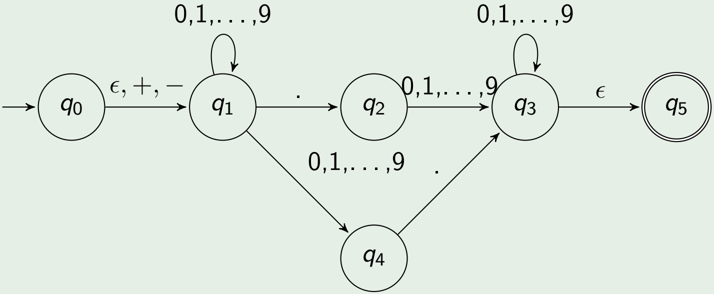
δ	ϵ	0	1
$\rightarrow q_0$	$\emptyset$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$	$\emptyset$

Konečné automaty s ϵ přechody

- Nově dovolíme přechody na ϵ, prázdné slovo, tj. bez přečtení vstupního symbolu.

Example 3.2 (NFA s ϵ přechody)

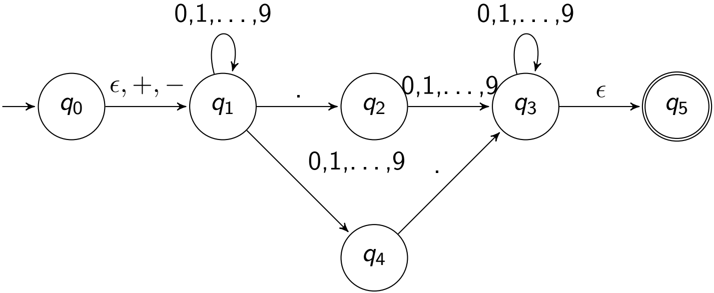
(1) Volitelně znaménko + nebo - ,
(2) řetězec číslic,
(3) desetinná tečka a
(4) další řetězec číslic. Nejméně jeden z řetězců (2) a (4) musí být neprázdný.



Example 3.3 (Přechodová funkce v tabulce)

Předešlý ϵ-NFA je: $E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\}, \delta, q_0, \{q_5\})$ s δ:

δ	ϵ	+,-	.	0,1,...,9
q_0	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
$*q_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$



ϵ–uzávěr

Definition 3.2 (ϵ–uzávěr)

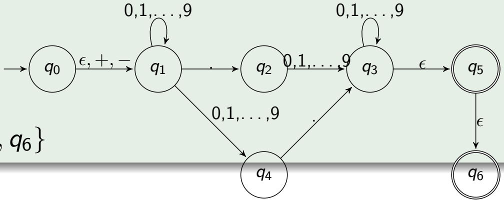
Pro $q \in Q$ definujeme **ϵ–uzávěr** $\epsilon closure(q)$ rekurzivně:

- Stav q je v $\epsilon closure(q)$.
- Je–li $p \in \epsilon closure(q)$ a $r \in \delta(p, \epsilon)$ pak i $r \in \epsilon closure(q)$.

Pro množinu stavů $S \subseteq Q$ definujeme $\epsilon closure(S) = \bigcup_{q \in S} \epsilon closure(q)$.

Example 3.4 (ϵ uzávěr)

- $\epsilon closure(q_0) = \{q_0, q_1\}$
- $\epsilon closure(q_1) = \{q_1\}$
- $\epsilon closure(q_3) = \{q_3, q_5, q_6\}$
- $\epsilon closure(\{q_3, q_4\}) = \{q_3, q_4, q_5, q_6\}$



Rozšířená přechodová funkce a jazyk přijímaný ϵ-NFA

Definition 3.3

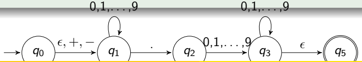
Nechť $E = (Q, \Sigma, \delta, q_0, F)$ je ϵ–NFA. Rozšířenou přechodovou funkci δ^* definujeme následovně:

- $\delta^*(q, \epsilon) = \epsilon closure(q)$.
- Indukční krok: $v = wa$, kde $w \in \Sigma^*$, $a \in \Sigma$.

$$\delta^*(q, wa) = \epsilon closure\left(\bigcup_{p \in \delta^*(q,w)} \delta(p, a)\right).$$

Example 3.5

$$\begin{array}{lclcl}
\delta^*(q_0, \epsilon) & = & \epsilon closure(q_0) & = & \{q_0, q_1\} \\
\delta^*(q_0, 5) & = & \epsilon closure(\bigcup_{q \in \delta^*(q_0,\epsilon)} \delta(q, 5)) = \epsilon closure(\delta(q_0, 5) \cup \delta(q_1, 5)) & = & \{q_1, q_4\} \\
\delta^*(q_0, 5.) & = & \epsilon closure(\delta(q_1, .) \cup \delta(q_4, .)) & = & \{q_2, q_3, q_5, q_6\} \\
\delta^*(q_0, 5.6) & = & \epsilon closure(\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6)) & = & \{q_3, q_5, q_6\}
\end{array}$$



Jazyk přijímaný ϵNFA

Definition 3.4 (Jazyk přijímaný nedeterministickým konečným automatem s přechody)

Mějme NFA $A = (Q, \Sigma, \delta, q_0, F)$, pak

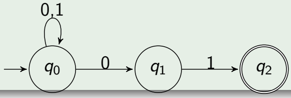
$$L(A) = \{w| \ \delta^*(q_0, w) \cap F \neq \emptyset\}$$

je jazyk přijímaný automatem A.
Tedy $L(A)$ je množina slov $w \in \Sigma^*$ takových, že $\delta^*(q_0, w)$ obsahuje alespoň jeden přijímající stav.

Example 3.6

Automat z předchozího slajdu přijímá jazyk $L = \{w|w \text{ končí na } 01, w \in \{0, 1\}^*\}$.
Důkaz indukcí konjunkce tvrzení:

- $\delta^*(q_0, w)$ obsahuje q_0 pro každé slovo w.
- $\delta^*(q_0, w)$ obsahuje q_1 iff w končí 0.
- $\delta^*(q_0, w)$ obsahuje q_2 iff w končí 01.



Konfigurace automatu, Výpočetní graf

Definition 3.5 (Konfigurace *DFA*, ϵ*NFA*)

Mějme ϵNFA $A = (Q, \Sigma, \delta, q_0, F)$, $q \in Q$, $v \in \Sigma^*$, pak dvojice (q, v) označuje **konfiguraci** konečného automatu, nacházejícího se ve stavu q s nepřečteným vstupem v.

Definition 3.6 (Výpočetní strom, graf ϵ*NFA*)

Mějme NFA $A = (Q, \Sigma, \delta, q_0, F)$ a vstupní slovo $w \in \Sigma^*$. Uzly **výpočetního grafu** jsou konfigurace A nad slovem w, orientované hrany značí možný přechod mezi konfiguracemi, tj. z (p, au) vede hrana do (q, u) právě když $q \in \delta(p, a)$.

$$
\begin{tikzcd}
(q_0, 00101) \arrow[r, "0"] \arrow[dd, "0"] & (q_0, 0101) \arrow[r, "0"] \arrow[d, "0"] & (q_0, 101) \arrow[r, "1"] & (q_0, 01) \arrow[r, "0"] \arrow[dr, "0"] & (q_0, 1) \arrow[r, "1"] & (q_0, \epsilon) \\
 & (q_1, 101) \arrow[dr, "1"] & & & (q_1, 1) \arrow[r, "1"] & (q_2, \epsilon) \\
(q_1, 0101)\ \text{(stuck)} & & (q_2, 01)\ \text{(stuck)} & & &
\end{tikzcd}
$$

Podmnožinová konstrukce (s ϵ–přechody)

Theorem 3.1 (Podmnožinová konstrukce (s ϵ–přechody))

Jazyk L je rozpoznatelný ϵ-NFA právě když je L regulární.

Algorithm: !Podmnožinová konstrukce (s ϵ–přechody)

Pro libovolný ϵ-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ zkonstruujeme DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ přijímající stejný jazyk jako N.

- Nové stavy jsou ϵuzavřené podmnožiny Q_N.

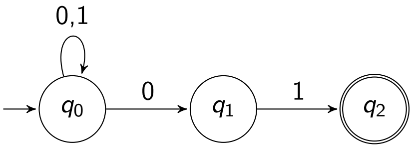
  $Q_D \subseteq \mathcal{P}(Q_N)$, $\forall S \subseteq Q_E : \epsilon closure(S) \in Q_D$. V Q_D může být i $\emptyset$.
- Počáteční stav je ϵuzávěr q_0.

  $q_D = \epsilon closure(q_0)$.
- Přijímací stavy jsou všechny množiny obsahující nějaký přijímající stav.

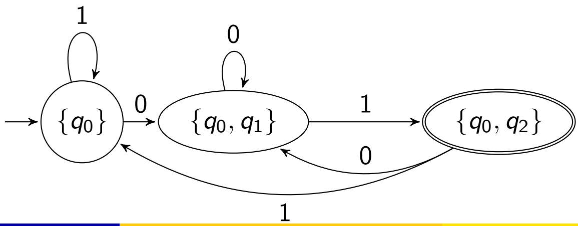
  $F_D = \{S | S \in Q_D \ \& \ S \cap F_N \neq \emptyset\}$.
- Přechodová funkce sjednotí předchody z jednotlivých stavů a uzavře ϵclosure.

  Pro $S \in Q_D, a \in \Sigma$ definujeme $\delta_D(S, a) = \epsilon closure(\bigcup_{p \in S} \delta(p, a))$.

Příklad podmnožinové konstrukce pro $\{w.01 | w \in \{0,1\}\}$



	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



Theorem 3.2 (Převod NFA na DFA)

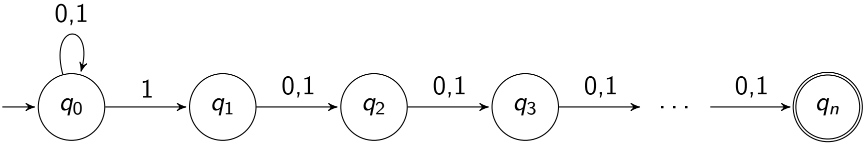
Pro DFA $D = (Q_D, \Sigma, \delta_D, q_{D_0}, F_D)$ vytvořený podmnožinovou konstrukcí z NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ platí $L(N) = L(D)$.

Proof.

Indukcí dokážeme: $\delta_D^*(q_0, w) = \delta_N^*(q_{D_0}, w)$. $\square$

Example 3.7 ('Těžký' případ pro podmnožinovou konstrukci)

Jazyk $L(N)$ slov 0's a 1's takových, že n–tý symbol od konce je 1. Intuitivně si DFA musí pamatovat n posledních přečtených symbolů.



- Aplikace hledání v textu

Regulární jazyky

- **Iterační (pumping) lemma** pro regulární jazyky,
- **Mihyll–Nerodova věta**,
  - užití pro důkaz ne–regulárnosti jazyka
  - příklad ne–regulárního jazyka, který lze pumpovat
- **dosažitelné stavy**, algoritmus nalezení,
- **ekvivalentní automaty, stavy**,
- **rozlišitelné stavy**, algoritmus nalezení,
- **redukovaný DFA**, redukt, **algoritmus nalezení reduktu**.
- **Nedeterministický FA**, **podmnožinová konstrukce**.
- ϵ **nedeterministický FA**, ϵ **uzávěr**.

Množinové operace nad jazyky

Definition 3.7 (Množinové operace nad jazyky)

Mějme dva jazyky L, M. Definujme následující operace:

(1) binární **sjednocení** $L \cup M = \{w | w \in L \vee w \in M\}$
- Příklad: jazyk obsahuje slova začínající a^i nebo tvaru $b^j c^j$.

(2) **průnik** $L \cap M = \{w | w \in L \;\&\; w \in M\}$
- Příklad: jazyk obsahuje slova sudé délky končící na 'baa'.

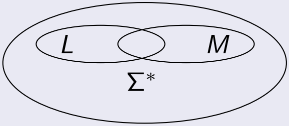
(3) **rozdíl** $L - M = \{w | w \in L \;\&\; w \notin M\}$
- Příklad: jazyk obsahuje slova sudé délky nekončící na 'baa'.

(4) **doplněk (komplement)** $\overline{L} = -L = \{w | w \notin L\} = \Sigma^* - L$
- Příklad: jazyk obsahuje slova nekončící na 'a'.

Theorem (de Morganova pravidla)

Platí:
$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$
$$L \cup M = \overline{\overline{L} \cap \overline{M}}$$
$$L - M = L \cap \overline{M}.$$

Důkaz ze vztahů $\&, \vee, \neg$.



Uzávěrové vlastnosti regulárních jazyků

Theorem 3.3 (Uzavřenost na množinové operace)

Mějme regulární jazyky L, M. Pak jsou následující jazyky také regulární:

(1) *sjednocení* $L \cup M$

(2) *průnik* $L \cap M$

(3) *rozdíl* $L - M$

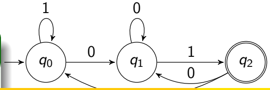
(4) *doplněk* $\overline{L} = \Sigma^* - L$.

Proof: Uzavřenost RJ na doplněk

- Pokud δ není pro některé dvojice q, a definovaná, přidáme nový nepřijímající stav q_{fail} a do něj přechod pro vše dříve nedefinované plus $\forall a \in \Sigma$: $\delta(q_{fail}, a) = q_{fail}$.
- Pak stačí prohodit koncové a nekoncové stavy přijímajícího deterministického FA $F = Q_A - F_A$. □

Example 3.8

Jazyk $\{w; w \in \{0,1\}^*01\}$

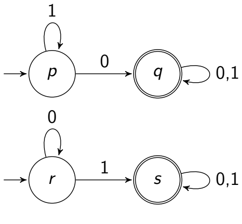


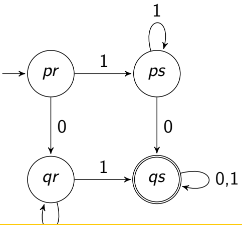
Konstrukce součinu automatů

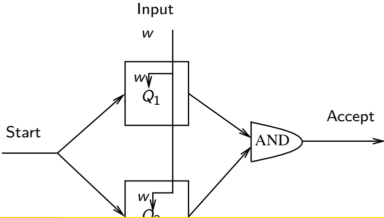
Proof: Průnik, sjednocení, rozdíl

- Pro sjednocení a rozdíl doplníme funkci δ na totální.
- Zkonstruujeme součinový automat,
  $Q = (Q_1 \times Q_2, \Sigma, \delta((p_1, p_2), x) = (\delta_1(p_1, x), \delta_2(p_2, x)), (q_{0_1}, q_{0_2}), F)$
- průnik: $F = F_1 \times F_2$
- sjednocení: $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- rozdíl: $F = F_1 \times (Q_2 - F_2)$. $\square$

Příklad součinu automatů (průnik jazyků). Slova obsahující 0,1, oboje.







Příklady na uzávěrové vlastnosti

Example 3.9

Konstruujeme konečný automat přijímající slova, která obsahují $3k+2$ symbolů 1 a neobsahují posloupnost 11.

- Přímá konstrukce je komplikovaná.
- $L_1 = \{w | w \in \{0,1\}^* \& |w|_1 = 3k+2\}$
- $L_2 = \{w | u, v \in \{0,1\}^* \& w = u11v\}$
- $L = L_1 - L_2$.

Example 3.10

Jazyk M slov s různým počtem 0 a 1 není regulární.

- Je–li M regulární, $\overline{M} = \Sigma^* - M$ je také regulární.
- O $\overline{M}$ víme, že regulární není (pumping lemma).

Ještě jeden příklad

Example 3.11

Jazyk $L_{0\neq 1} = \{0^i 1^j : i \neq j, i, j \in \mathbb{N}_0\}$ není regulární.

- Jazyk $L_{01} = \{0^i 1^j ; i, j \in \mathbb{N}_0\}$ je regulární, umíme sestrojit konečný automat.
- $L_{01} - L_{0\neq 1} = \{0^i 1^i : i \in \mathbb{N}_0\}$
- Z uzávěrových vlastností víme, že rozdíl regulárních jazyků je regulární.
- Jazyk L_{01} regulární je.
- Předpokládejme, že $L_{0\neq 1}$ je regulární. Pak by i $\{0^i 1^i : i \in \mathbb{N}_0\}$ musel být regulární, což není - SPOR.

Řetězcové operace nad jazyky

Definition 3.8 (Řetězcové operace nad jazyky)

Nad jazyky L, M definujeme následující operace:

zřetězení jazyků	$L.M = \{uv \mid u \in L \ \&\ v \in M\}$
	$L.x = L.\{x\}$ a $x.L = \{x\}.L$ pro $x \in \Sigma$
mocniny jazyka	$L^0 = \{\epsilon\}$
	$L^{i+1} = L^i.L$
pozitivní iterace	$L^+ = L^1 \cup L^2 \ldots = \bigcup_{i \geq 1} L^i$
obecná iterace	$L^* = L^0 \cup L^1 \cup L^2 \ldots = \bigcup_{i \geq 0} L^i$
	tedy $L^* = L^+ \cup \{\epsilon\}$
otočení jazyka	$L^R = \{u^R \mid u \in L\}$
(=zrcadlový obraz,reverze)	$(x_1 x_2 \ldots x_n)^R = x_n x \ldots x_2 x_1$
levý kvocient L podle M	$M \setminus L = \{v \mid uv \in L \ \&\ u \in M\}$
levá derivace L podle w	$\partial_w L = \{w\} \setminus L$ (pozn. derivace bude i v jiném významu
pravý kvocient L podle M	$L/M = \{u \mid uv \in L \ \&\ v \in M\}$
pravá derivace L podle w	$\partial_w^R L = L/\{w\}$.

Theorem 3.4 (Uzavřenost reg. jazyků na řetězcové operace)

Jsou–li L, M regulární jazyky, je regulární i $L.M$, L^*, L^+, L^R, $M \setminus L$ a L/M.

Lemma ($L.M$)

Jsou–li L, M regulární jazyky, je regulární i $L.M$.

Proof:

Vezmeme DFA $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, pak $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ tak že $L = L(A_1)$ a $M = L(A_2)$.

Definujeme nedeterministický automat $B = (Q \cup \{q_0\}, \Sigma, \delta, \{q_0\}, F_2)$ kde:

$Q = Q_1 \cup Q_2$ předpokládáme různá jména stavů, jinak přejmenujeme
končíme až po přečtení slova z L_2

$\delta(q_0, \epsilon)$	$= \{q_1, q_2\}$	pro $q_1 \in F_1$	tj. $\epsilon \in L(A_1)$
$\delta(q_0, \epsilon)$	$= \{q_1\}$	pro $q_1 \notin F_1$	tj. $\epsilon \notin L(A_1)$
$\delta(q_0, x)$	$= \emptyset$	pro $x \in \Sigma$	
$\delta(q, x)$	$= \{\delta_1(q, x)\}$	pro $q \in Q_1$ & $\delta_1(q, x) \notin F_1$	počítáme v A_1
	$= \{\delta_1(q, x), q_2\}$	pro $q \in Q_1$ & $\delta_1(q, x) \in F_1$	nedet. přechod z A_1
	$= \{\delta_2(q, x)\}$	pro $q \in Q_2$	počítáme v A_2.

Pak $L(B) = L(A_1).L(A_2)$. □

Uzavřenost iterace

Lemma (L^*, L^+)

Je–li L regulární jazyk, je regulární i L^*, L^+.

- Idea: opakovaný výpočet automatu $A = (Q, \Sigma, \delta, q_0, F)$
- realizace: nedeterministické rozhodnutí, zda pokračovat nebo restart
- speciální stav pro příjem $\epsilon \in L^0$ (pro L^+ vynecháme či $\notin F$).

Proof: Důkaz pro L^*

Vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$, tak že $L = L(A)$.
Definujeme NFA automat $B = (Q \cup \{q_B\}, \Sigma, \delta_B, \{q_B\}, F \cup \{q_B\})$ kde:

$\delta_B(q_B, \epsilon) = \{q_0\}$ nový stav q_B pro příjem ϵ, přejdeme do q_0
$\delta_B(q_B, x) = \emptyset$ pro $x \in \Sigma$
$\delta_B(q, x) = \{\delta(q, x)\}$ pokud $q \in Q$ & $\delta(q, x) \notin F$ uvnitř A
$= \{\delta(q, x), q_0\}$ pokud $q \in Q$ & $\delta(q, x) \in F$ možný restart

Pak $L(B) = L(A)^*$ ($q_B \in F_B$), $L(B) = L(A)^+$ ($q_B \notin F_B$). □

Uzavřenost reverze

Lemma (L^R)

Je–li L regulární jazyk, je regulární i L^R.

- Zřejmě $(L^R)^R = L$ a tedy stačí ukázat jeden směr.
- idea: obrátíme šipky ve stavovém diagramu; nederministický FA

Proof:

Vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$, tak že $L = L(A)$.
Definujeme nedeterministický automat $B = (Q \cup \{q_B\}, \Sigma, \delta_B, \{q_B\}, \{q_0\})$ kde:

- $\delta_B(q, x) = \{p | \delta(p, x) = q\}$ pro $q \in Q$
- $\delta_B(q_B, \epsilon) = F$, $\delta_B(q_B, x) = \emptyset$.
- Pro libovolné slovo $w = x_1 x_2 \dots x_n$
  - $q_0, q_1, q_2, \dots, q_n$ je přijímající výpočet pro w v A

  $\Leftrightarrow$

  - $q_B, q_n, q_{n-1}, \dots, , q_2, q_1, q_0$ je přijímající výpočet pro w^R v B.

$\square$

Uzavřenost kvocientu

Lemma ($M \setminus L$ a L/M)

Jsou–li L, M regulární jazyky, je regulární i $M \setminus L$ a L/M.

- Idea: A_L, budeme startovat ve stavech, do kterých se lze dostat slovem z M

Proof:

- $v \in M \setminus L$
  $\Leftrightarrow$ $(\exists u \in M)\ uv \in L$
  $\Leftrightarrow$ $(\exists u \in M, \exists q \in Q)\ \delta(q_0, u) = q\ \&\ \delta(q, v) \in F$
  $\Leftrightarrow$ $\exists q \in S_0\ \&\ \delta(q, v) \in F$
  $\Leftrightarrow$ $v \in L(B)$

  Vezmeme DFA $A = (Q, \Sigma, \delta, q_0, F)$, tak že $L = L(A)$.
  Definujeme nedeterministický NFA $B = (Q, \Sigma, \delta, S_0, F)$ kde:
- definujeme $S_0 = \{q | q \in Q\ \&\ (\exists u \in M)\ q = \delta(q_0, u)\}$
  - lze nalézt algoritmicky
    $(\{q;\ L(A_q) \cap M \neq \emptyset$ kde $A_q = (Q, \Sigma, \delta, q_0, \{q\})\})$

$L/M = (M^R \setminus L^R)^R$ □

Regulární výrazy

Regulární výrazy (RV) jsou

- algebraickým popisem jazyků
- deklarativním způsobem, jak vyjádřit slova, která chceme přijímat.
- Schopné definovat všechny a pouze regulární jazyky.
- Můžeme je brát jako programovací jazyk, uživatelsky přívětivý popis konečného automatu.

Example 4.1

- Základní UNIX grep příkaz.
- Lexikální analyzátory jako Lex a Flex (popis pomocí 'token'ů je vzásadě regulární výraz).
- Python knihovna re .

- Syntaktická analýza potřebuje silnější nástroj, bezkontextové gramatiky, budou následovat.

Regulární výrazy (RegE)

Definition 4.1 (Regulární výrazy (Regular Expression) (RegE), hodnota RegE $L(\alpha)$)

Regulární výrazy $\alpha, \beta \in RegE(\Sigma)$ nad konečnou neprázdnou abecedou $\Sigma = \{x_1, x_2, \ldots, x_n\}$ a jejich hodnota $L(\alpha)$ jsou definovány induktivně:

- Základ:

výraz α	pro	hodnota $L(\alpha) \equiv [\alpha]$
ϵ	prázdný řetězec	$L(\epsilon) = \{\epsilon\}$
$\emptyset$	prázdný výraz	$L(\emptyset) = \{\} \equiv \emptyset$
$\mathbf{a}$	$a \in \Sigma$	$L(\mathbf{a}) = \{a\}$.

- Indukce:

výraz	hodnota	poznámka
$\alpha + \beta$	$L(\alpha + \beta) = L(\alpha) \cup L(\beta)$	v grep, re |
$\alpha\beta$	$L(\alpha\beta) = L(\alpha)L(\beta)$	můžeme značit ., ale plete se s UNIX grep.
α^*	$L(\alpha^*) = L(\alpha)^*$	
(α)	$L((\alpha)) = L(\alpha)$	závorky nemění hodnotu.

Každý regulární výraz dostaneme indukcí výše, tj. třída $RegE(\Sigma)$ je nejmenší třída uzavřená na uvedené operace.

Lemma 4.1

Příklady reguláních výrazů, priorita

Example 4.2 (Regulární výrazy)

Jazyk střídajících se nul a jedniček lze zapsat:

- $(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{1}(\mathbf{01})^* + \mathbf{0}(\mathbf{10})^*$
- $(\epsilon + \mathbf{1})(\mathbf{01})^*(\epsilon + \mathbf{0})$.

Jazyk $L((\mathbf{0}^*\mathbf{1}\mathbf{0}^*\mathbf{1}\mathbf{0}^*\mathbf{1})^*\mathbf{0}^*) = \{w | w \in \{0,1\}^*, |w|_1 = 3k, k \geq 0\}$.

Definition 4.2 (priorita)

Nejvyšší prioritu má iterace $*$, nižší konkatenace (zřetězení), nejnižší sjednocení $+$.

Theorem 4.1 (!varianta Kleeneho věty)

1. *Každý jazyk reprezentovaný konečným automatem lze zapsat jako regulární výraz.*
2. *Každý jazyk popsaný regulárním výrazem můžeme zapsat jako ϵ-NFA (a tedy i DFA).*

Převod RegE výrazu na ϵ–NFA automat

Převod RegE výrazu na ϵ–NFA automat.

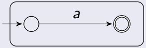
Důkaz indukcí dle struktury R. V každém kroku zkonstruujeme ϵ-NFA E rozpoznávající stejný jazyk $L(R) = L(E)$ se třemi dalšími vlastnostmi:

1. Právě jeden přijímající stav.
2. Žádné hrany do počátečního stavu.
3. Žádné hrany z koncového stavu.

Základ:

Prázdný řetězec ϵ

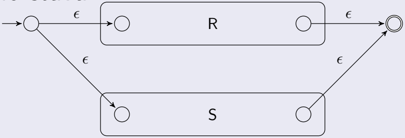
Prázdná množina $\emptyset$



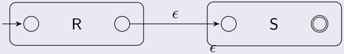
$a \in \Sigma$: výraz **a**

INDUKCE:

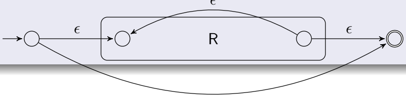
Součet $R + S$:



Zřetězení RS:



Iterace R^*:



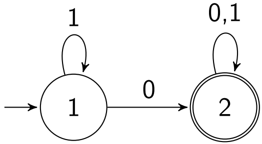
□

Od DFA k regulárním výrazům

Regulární výraz z DFA

- Mějme DFA A, $Q_A = \{1, \ldots, n\}$ o n stavech, stavy očíslujeme od 1.
- Nechť $R_{ij}^{(k)}$ je regulární výraz, $L(R_{ij}^{(k)}) = \{w | \delta^*_{\leq k}(i, w) = j\}$ množina slov převádějících stav i do stavu j v A cestou, která neobsahuje stav s vyšším indexem než k.
- Budeme rekruzivně konstruovat $R_{ij}^{(k)}$ pro $k = 0, \ldots, n$.
- $k = 0$, $i \neq j$: $R_{ij}^{(0)} = \mathbf{a_1} + \mathbf{a_2} + \ldots + \mathbf{a_m}$ kde $a_1, a_2, \ldots, a_m$ jsou symboly označující hrany i do j (nebo $R_{ij}^{(0)} = \emptyset$ nebo $R_{ij}^{(0)} = \mathbf{a}$ pro $m = 0, 1$).
- $k = 0$, $i = j$: smyčky, $R_{ii}^{(0)} = \epsilon + \mathbf{a_1} + \mathbf{a_2} + \ldots + \mathbf{a_m}$ kde $a_1, a_2, \ldots, a_m$ jsou symboly na smyčkách v i.

Příklad: Od konečného automatu k RegE

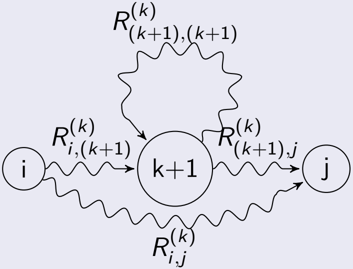


$$R_{12}^{(2)} = \mathbf{1}^*\mathbf{0}(\mathbf{0}+\mathbf{1})^*$$

$$R_{ij}^{(k+1)} = R_{ij}^{(k)} + R_{i(k+1)}^{(k)}(R_{(k+1)(k+1)}^{(k)})^* R_{(k+1)j}^{(k)}$$

$R_{11}^{(0)}$	$\epsilon+\mathbf{1}$	$=$
$R_{12}^{(0)}$	$\mathbf{0}$	$=$
$R_{21}^{(0)}$	$\emptyset$	$=$
$R_{22}^{(0)}$	$(\epsilon+\mathbf{0}+\mathbf{1})$	$=$
$R_{11}^{(1)}$	$(\epsilon+\mathbf{1})+(\epsilon+\mathbf{1})(\epsilon+\mathbf{1})^*(\epsilon+\mathbf{1})$	$=\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0}+(\epsilon+\mathbf{1})(\epsilon+\mathbf{1})^*\mathbf{0}$	$=\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset+\emptyset(\epsilon+\mathbf{1})^*(\epsilon+\mathbf{1})$	$=\emptyset$
$R_{22}^{(1)}$	$(\epsilon+\mathbf{0}+\mathbf{1})+\emptyset(\epsilon+\mathbf{1})^*\mathbf{0}$	$=\epsilon+\mathbf{0}+\mathbf{1}$
$R_{11}^{(2)}$	$\mathbf{1}^*+\mathbf{1}^*\mathbf{0}(\epsilon+\mathbf{0}+\mathbf{1})^*\emptyset$	$=\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0}+\mathbf{1}^*\mathbf{0}(\epsilon+\mathbf{0}+\mathbf{1})^*(\epsilon+\mathbf{0}+\mathbf{1})$	$=\mathbf{1}^*\mathbf{0}(\mathbf{0}+\mathbf{1})^*$
$R_{21}^{(2)}$	$\emptyset+(\epsilon+\mathbf{0}+\mathbf{1})(\epsilon+\mathbf{0}+\mathbf{1})^*\emptyset$	$=\emptyset$
$R_{22}^{(2)}$	$(\epsilon+\mathbf{0}+\mathbf{1})+(\epsilon+\mathbf{0}+\mathbf{1})(\epsilon+\mathbf{0}+\mathbf{1})^*(\epsilon+\mathbf{0}+\mathbf{1})$	$=(\mathbf{0}+\mathbf{1})^*$

INDUKCE. Mějme $\forall i, j \in Q$ $R_{ij}^{(k)}$. Konstruujeme $R_{ij}^{(k+1)}$.



1. $R_{ij}^{(k+1)} = R_{ij}^{(k)} + R_{i(k+1)}^{(k)}(R_{(k+1)(k+1)}^{(k)})^* R_{(k+1)j}^{(k)}$
   - Cesty z i do j neprocházející uzlem $(k+1)$ jsou již v $R_{ij}^{(k)}$.
   - Cesty z i do j přes $(k+1)$ s případnými smyčkami můžeme zapsat $R_{i(k+1)}^{(k)}(R_{(k+1)(k+1)}^{(k)})^* R_{(k+1)j}^{(k)}$.
   - regulární výrazy jsou uzavřené na sčítání (sjednocení), zřetězení i iteraci, tj. $R_{ij}^{k+1} \in RegE(\Sigma)$
2. Nakonec, $RegE = \oplus_{j \in F_A} R_{1j}^{(n)}$ sjedncení přes přijímající stavy j.

$\square$

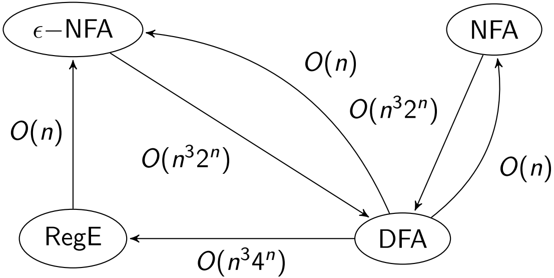
Zjednodušení regulárních výrazů (netřeba znát)

Lemma (Další vlastnosti bez důkazu)

- *Zjednodušení návrhu automatů*

$$
\begin{aligned}
L.\emptyset = \emptyset.L &= \emptyset \\
\{\epsilon\}.L = L.\{\epsilon\} &= L \\
(L^*)^* &= L^* \\
(L_1 \cup L_2)^* &= L_1^*(L_2.L_1^*)^* = L_2^*(L_1.L_2^*)^* \\
(L_1.L_2)^R &= L_2^R.L_1^R \\
\partial_w(L_1 \cup L_2) &= \partial_w(L_1) \cup \partial_w(L_2) \\
\partial_w(\Sigma^* - L) &= \Sigma^* - \partial_w L.
\end{aligned}
$$

Shrnutí převodů mezi reprezentacemi regulárních jazyků



Převod NFA na DFA

- ϵ uzávěr v $O(n^3)$ – prohledává n stavů násobeno n^2 hran pro ϵ přechody.
- Podmnožinová konstrukce, DFA s až 2^n stavy. Pro každý stav, $O(n^3)$ času na výpočet přechodové funkce.

Převod DFA na NFA

- Přidat množinové závorky k přechodové funkci a přechody pro ϵ u $\epsilon-$NFA.

Převod automatu DFA an RegE regulární výraz

- $O(n^3 4^n)$

RegE výraz na automat

- V čase $O(n)$ vytvoříme $\epsilon-$NFA.

Shrnutí

- Regulární výrazy
- Kleeneho věta
  - Jazyk je přijímaný konečným automatem právě když lze napsat jako regulární výraz,
  - tj. z $\emptyset$ a $\{\mathbf{a}\}$ pro $a \in \Sigma$
  - a konečného počtu aplikací iterace, zřetězení a sjednocení.
- Uzávěrové vlastnosti
  - dnes jen 'regulární' sloupec.

jazyk	regulární (RL)	bezkontextové	deterministické CFL
sjednocení	ANO	ANO	NE
průnik	ANO	NE	NE
$\cap$ s RL	ANO	ANO	ANO
doplněk	ANO	NE	ANO
homomorfizmus	ANO	ANO	NE
inverzní hom.	ANO	ANO	ANO

Substituce jazyků

Definition 4.3 (Substituce jazyků)

Mějme konečnou abecedu Σ. Pro každé $x \in \Sigma$ budiž $\sigma(x)$ jazyk v nějaké abecedě Y_x. Dále položme

$$\sigma(\epsilon) = \{\epsilon\}$$

$$\sigma(u.v) = \sigma(u).\sigma(v)$$

- Zobrazení $\sigma : \Sigma^* \to P(Y^*)$, kde $Y = \bigcup_{x \in \Sigma} Y_x$ se nazývá **substituce**.
- Pro jazyk L definujeme: $\sigma(L) = \bigcup_{w \in L} \sigma(w)$, podobně sjednocení.
- **nevypouštějící substituce** je substituce, kde žádné $\sigma(x)$ neobstahuje ϵ.

Example 4.3 (substituce)

1) $\Sigma = \{k, p, m, c, t\}$, $L = (kmp)(ckmp)^* t$,
   k slovník křestních jmen, p slovník příjmení, m mezera, c čárka, t tečka.
2) Pokud $\sigma(0) = \{a^i b^j, i, j \geq 0\}, \sigma(1) = \{cd\}$
   tak $\sigma(010) = \{a^i b^j c d a^k b^l, i, j, k, l \geq 0\}$.

Homomorfizmus a inverzní homomorfizmus jazyků

Definition 4.4 (homomorfizmus (jazyků), inverzní homomorfizmus)

Homomorfizmus h je speciální případ substituce, kde obraz je vždy jen jednoslovný jazyk (vynecháváme u něj závorky), tj. $(\forall x \in \Sigma)\ h(x) = w_x$.
Pokud $\forall x : w_x \neq \epsilon$, jde o **nevypouštějící homomorfizmus**.
Inverzní homomorfizmus $h^{-1}(L) = \{w | h(w) \in L\}$.

Example 4.4 (homomorfizmus)

- Znaky nahradíme TEXzápisem, $h(\mu) = \backslash mu$ a podobně.
- Homomorfizmus h definujeme: $h(0) = ab$, a $h(1) = \epsilon$. Pak $h(0011) = abab$. Pro $L = \mathbf{10^*1}$ je $h(L) = (ab)^*$.

Theorem 4.2 (uzavřenost na homomorfizmus)

Je–li jazyk L i $\forall x \in \Sigma$ jazyk $\sigma(x), h(x)$ regulární, pak je regulární i $\sigma(L), h(L)$.

Uzavřenost na substituci, homomorfizmus.

Strukturální indukcí 'probubláváním' algebraickým popisem jazyka základních, sjednocení, zřetězení a iterace. Pro sjednocení a zřetězení z definice substituce a uzavřenosti regulárních jazyků na sjednocení a zřetězení.

$$\begin{aligned} \underline{\sigma}(\alpha+\beta) &= \sigma(L(\alpha)) \cup \sigma(L(\beta)) \\ \underline{\sigma}(\alpha\beta) &= \{w | \exists u \in L(\alpha) \exists v \in L(\beta) : \sigma(u)\sigma(v) = w\} \end{aligned}$$

Pro iteraci rozložíme na nekonečné sjednocení, pro každý konkrétní počet iterací σ aplikované na konečné zřetězení.

$$\begin{aligned} \sigma(L(\alpha)^*) &= \sigma(L(\alpha)^0) \cup \sigma(L(\alpha)^1) \cup \ldots \cup \sigma(L(\alpha)^n) \cup \ldots \\ &= \underline{\sigma}(\alpha)^0 \cup \underline{\sigma}(\alpha)^1 \cup \ldots \cup \underline{\sigma}(\alpha)^n \cup \ldots \\ &= L(\underline{\sigma}(\alpha)^*). \end{aligned}$$

□

Inverzní homomorfizmus

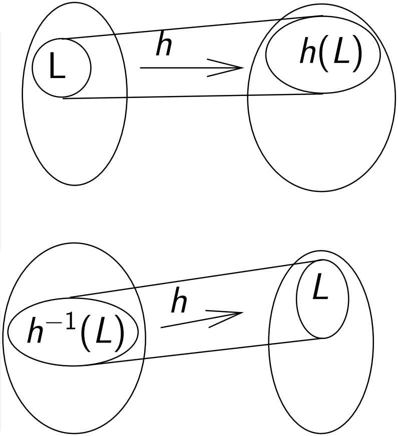
Definition ((4.4) Inverzní homomorfizmus)

Nechť h je homomorfizmus abecedy T do slov nad abecedou Σ. Pak $h^{-1}(L)$ *'h inverze L'* je množina řetězců
$h^{-1}(L) = \{w | w \in T^*;\ h(w) \in L\}$.

Example 4.5

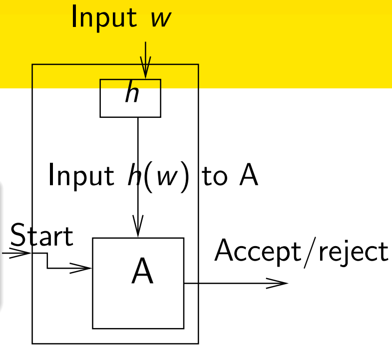
Nechť $L = (\{00\} \cup \{1\})^*$, $h(a) = 01$ a $h(b) = 10$.
Pak $h^{-1}(L) = (\{ba\})^*$.

Důkaz: $h((\{ba\})^*) \in L$ snadno. Ostatní w generují izolované 0 (rozbor případů).



Homomorfizmus aplikovaný dopředně a zpětně.

Inverzní homomorfizmus DFA



Theorem 4.3

Je–li h homomorfizmus abecedy T do abecedy Σ a L je regulární jazyk abecedy Σ, pak $h^{-1}(L)$ je také regulární jazyk.

Proof:

- pro L máme DFA $A = (Q, \Sigma, \delta, q_0, F)$
- $h : T \rightarrow \Sigma^*$
- definujeme ϵ-NFA $B = (Q', T, \delta', [q_0, \epsilon], F \times \{\epsilon\})$ kde

$$\begin{array}{lll} Q' = \{[q,u] \,|\, q \in Q, u \in \Sigma^*, \exists(a \in T)\exists(v \in \Sigma^*) h(a) = vu\} & & u \text{ je buffer} \\ \delta'([q,\epsilon], a) & = [q, h(a)] & \text{naplňuje buffer} \\ \delta'([q, bv], \epsilon) & = [p, v] \text{ kde } \delta(q,b) = p & \text{čte buffer.} \end{array}$$

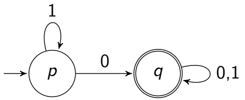
$\square$

Příklad: Navštiv všechny stavy

Example 4.6

Nechť $A = (Q, \Sigma, \delta, q_0, F)$ je DFA. Definujme jazyk $L = \{w \in \Sigma^*;\ \delta^*(q_0, w) \in F$ a pro každý stav $q \in Q$ existuje prefix x_q slova w tak, že $\delta^*(q_0, x_q) = q\}$. Tento jazyk L je regulání.

- M Označme $M = L(A)$.
- T Definujeme novou abecedu T trojic $\{[paq]; p, q \in Q, a \in \Sigma, \delta(p, a) = q\}$.
- h Definujeme homomorfizmus $(\forall\ p, q, a)\ h([paq]) = a$.
- L_1 Jazyk $L_1 = h^{-1}(M)$ je regulární, protože M je regulární (DFA inverzní homomorfizmus).
  - $h^{-1}(101)$ obsahuje $2^3 = 8$ řetězců, např. $[p1p][q0q][p1p] \in \{[p1p], [q1q]\}\{[p0q], [q0q]\}\{[p1p], [q1q]\}$.
  - Dále zkonstruujeme L z L_1 (další slide).



- L_2 Vynutíme začátek q_0. Definujeme $E_1 = \bigcup_{a\in\Sigma,q\in Q}\{[q_0 a q]\} =$ $E_1 = \{[q_0 a_1 q_0], [q_0 a_2 q_1], \ldots, [q_0 a_m q_n]\}$. Pak $L_2 = L_1 \cap L(E_1.T^*)$.
- L_3 Vynutíme stejné sousedící stavy. Definujeme ne–odpovídající dvojice $E_2 = \bigcup_{q\neq r,p,q,r,s\in Q,a,b\in\Sigma}\{[paq][rbs]\}$. Definujeme $L_3 = L_2 - L(T^*.E_2.T^*)$,
  - Končí v přijímajícím stavu, protože jsme začali z jazyku M přijímaném DFA A.
- L_4 Všechny stavy. $\forall\ q \in Q$ definujeme E_q jako regulární výraz sjednocení všech symbolů T takových, že q není ani na první, ani na poslední pozici. Odečteme $L(E_q^*)$ od L_3. $L_4 = L_3 - \bigcup_{q\in Q}\{E_q^*\}$.
- L Odstraníme stavy, necháme symboly. $L = h(L_4)$. Tedy L je regulární.

Přehled:

- M $= L(A)$
  Inverzní homom.
- L_1 $h^{-1}(M) \subseteq \{[qap]\}^*$
  průnik RJ
- L_2 $+\ q_0$
  rozdíl RJ
- L_3 + sousední stavy rovny
  rozdíl RJ
- L_4 + všechny stavy
  homomorfismus
- L $h([qap]) = a$

Rozhodovací problémy pro regulární jazyky

Lemma (Test ne–prázdnosti regulárního jazyka)

Lze algoritmicky rozhodnout, zda jazyk přijímaný DFA, NFA, ϵ-NFA je prázdný.

Jazyk je prázdný právě když žádný z koncových stavů není dosažitelný. Dosažitelnost lze testovat $O(n^2)$.

Lemma (Test náležení do regulárního jazyka)

Pro daný řetězec w; $|w| = n$ a regulární jazyk L. Lze algoritmicky rozhodnout, zda je $w \in L$.

- DFA: Spusť automat; pokud $|w| = n$, při dobré reprezentaci a konstatním čase přechodu $O(n)$.
- NFA o s stavech: čas $O(ns^2)$. Každý vstupní symbol aplikujeme na všechny stavy předchozího kroku, kterých je nejvýš s.
- $\epsilon-$NFA - nejdříve určíme ϵ–uzávěr. Pak aplikujeme přechodovou funkci a ϵ–uzávěr na výsledek.

Shrnutí 4

Definition (4.1 RJ – algebraický popis jazyků)

Pro konečnou neprázdnou abecedu Σ označme $RJ(\Sigma)$ nejmenší třídu jazyků, která:

- obsahuje prázdný jazyk $\emptyset$
- pro každé písmeno $x \in \Sigma$ obsahuje jazyk $\{x\}$
- je uzavřená na sjednocení $A, B \in RJ(\Sigma) \Rightarrow A \cup B \in RJ(\Sigma)$
- je uzavřená na zřetězení $A, B \in RJ(\Sigma) \Rightarrow A.B \in RJ(\Sigma)$
- je uzavřená na iteraci $A \in RJ(\Sigma) \Rightarrow A^* \in RJ(\Sigma)$.

Theorem (4.1 Kleene)

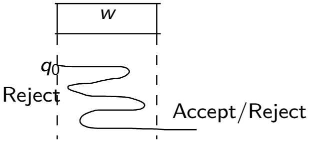
Libovolný jazyk je rozpoznatelný konečným automatem právě když je ve třídě RJ.

Třída regulárních jazyků je uzavřená na

- sjednocení, průnik, doplněk
- zřetězení, iteraci, substituci, homomorfizmus, inverzní homomorfizmus
- reverzi, levý i pravý kvocient.

Dvousměrné (dvoucestné) konečné automaty

- Konečný automat provádí následující činnosti:
    - přečte písmeno
    - změní stav vnitřní jednotky
    - posune čtecí hlavu doprava
- Čtecí hlava se nesmí vracet.



Definition 5.1 (Deterministické Dvousměrné konečné automaty)

Deterministickým dvousměrným konečným automatem nazýváme pětici $A = (Q, \Sigma, \delta, q_0, F)$, kde

1. Q je konečná množina stavů,
2. Σ je konečná množina vstupních symbolů
3. přechodové funkce δ je zobrazení $Q \times \Sigma \rightarrow Q \times \{-1, 1\}$ rozšířené o pohyb hlavy
4. $q_0 \in Q$ počáteční stav
5. množina přijímajících stavů $F \subseteq Q$.

Pozn.: Je deterministický, nedeterministický $\delta_N : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{-1, 1\})$.
Pozn.2: Nulový pohyb hlavy lze, jen trochu zkomplikuje důkaz dále.

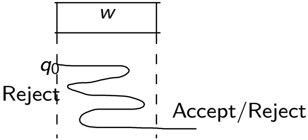
- Reprezentujeme opět stavovým diagramem, tabulkou

Výpočet dvousměrného automatu

Definition 5.2 (Výpočet dvousměrného automatu)

Slovo w je **přijato dvousměrným konečným automatem**, pokud:

- výpočet začal na prvním písmenu slova w vlevo v počátečním stavu
- čtecí hlava poprvé opustila slovo w vpravo v některém přijímajícím stavu
- mimo čtené slovo není výpočet definován (výpočet zde končí a slovo není přijato).



- Ke slovům si můžeme přidat speciální koncové znaky $\# \notin \Sigma$
- funkce $\partial_\#$ odstraní $\#$ zleva, $\partial_\#^R$ zprava.

Lemma

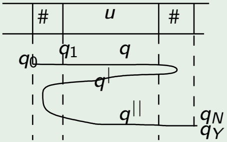
Je–li $L(A) = \{\#w\# | w \in L \subseteq \Sigma^*\}$ *regulární, potom je regulární i* $L = \partial_\# \partial_\#^R (L(A) \cap \#\Sigma^*\#)$.

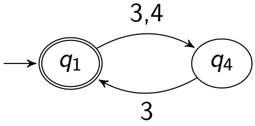
Příklad dvousměrného automatu

Example 5.1 (Příklad dvousměrného automatu)

Nechť $A = (Q, \Sigma, \delta, q_1, F)$. Dvousměrný konečný automat $B = (Q \cup Q^{|} \cup Q^{||} \cup \{q_0, q_N, q_Y\}, \Sigma \cup \{\#\}, \delta^{|}, q_0, \{q_Y\})$ přijímající jazyk $L(B) = \{\#u\# | uu \in L(A)\}$ (toto NENÍ levý ani pravý kvocient!) definujeme následovně:

$\delta^{|}$	$x \in \Sigma$	$\#$	poznámka
q_0	$q_N, -1$	$q_1, +1$	q_1 je počátek A
q	$p, +1$	$q^{|}, -1$	$p = \delta(q, x)$
$q^{|}$	$q^{|}, -1$	$q^{||}, +1$	
$q^{||}$	$p^{||}, +1$	$q_Y, +1$	$q \in F, p = \delta(q, x)$
$q^{||}$	$p^{||}, +1$	$q_N, +1$	$q \notin F, p = \delta(q, x)$
q_N	$q_N, +1$	$q_N, +1$	
q_Y	$q_N, +1$	$q_N, +1$	





Dvousměrné a jednosměrné konečné automaty

Theorem 5.1

Jazyky přijímané dvousměrnými konečnými automaty jsou právě regulární jazyky.

Proof: konečný automat $\rightarrow$ dvousměrný automat

- Konečný automat předeveme na dvousměrný přidáním posunu hlavy vpravo
- $A = (Q, \Sigma, \delta, q_0, F) \rightarrow 2A = (Q, \Sigma, \delta^{|}, q_0, F)$, kde $\delta^{|}(q, x) = (\delta(q, x), +1)$. $\square$

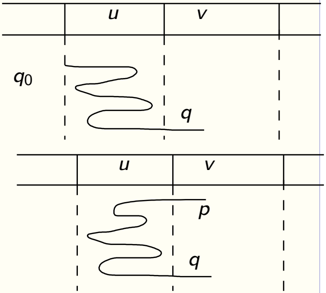
- Možnost pohybovat čtecí hlavou po pásce nezvětšila sílu konečného automatu (dokud na pásku nic nepíšeme!).
- Pro důkaz potřebujeme přípravu.

Funkce f_u popisující výpočet 2DFA nad slovem u

Algorithm: Funkce f_u popisující výpočet 2DFA nad slovem u

Definujeme funkci $f_u : Q \cup \{q_0^{|}\} \to Q \cup \{0\}$

- $f_u(q_0^{|})$ popisuje v jakém stavu poprvé odejdeme vpravo, pokud začneme výpočet vlevo v počátečním stavu q_0,
- symbol 0 značí, že daná situace nenastane (odejdeme vlevo nebo cyklus),
- $f_u(p); p \in Q$ v jakém stavu opět odejdeme vpravo, pokud začneme výpočet vpravo v p



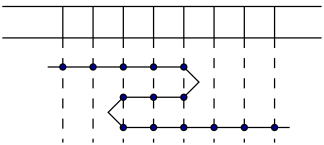
- Definujeme ekvivalenci slov následovně: $u \sim w \Leftrightarrow_{def} f_u = f_w$,
  - tj. slova jsou ekvivalentní pokud mají stejné 'výpočtové' funkce.

Regulárnost 2DFA

Ekvivalence $\sim$ je ekvivalence, má konečný index, je to pravá kongruence, jazyk 2DFA odpovídá sjednocení tříd $f_w(q_0^{|}) \in F$.

Konstruktivní důkaz věty o 2DFA

- Potřebujeme převést návraty na lineární výpočet.
- Zajímají nás jen přijímající výpočty
- Díváme se na řezy mezi symboly (v jakém stavu přechází na další políčko)



Pozorování:

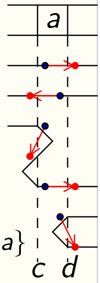
- stavy se v přechodu řezu střídají (doprava, doleva)
- první stav jde doprava, poslední také doprava
- v deterministických přijímajících výpočtech nejsou cykly
- první a poslední řez obsahují jediný stav.

Algorithm: 2DFA → NFA

- Najdeme všechny možné řezy – posloupnosti stavů (je jich konečně mnoho).
- Mezi řezy definujeme nedeterministické přechody podle čteného symbolu.
- Rekonstruujeme výpočet skládáním řezů jako puzzle.

Algorithm: Formální převod 2DFA na NFA

Nechť $A = (Q, \Sigma, \delta, q_0, F)$ je dvousměrný (deterministický) konečný automat. Definujeme ekvivalentní nedeterministický automat $B = (Q^{|}, \Sigma, \delta^{|}, (q_0), F^{|})$ kde:



- $Q^{|}$ jsou všechny korektní přechodové posloupnosti
  - posloupnosti stavů $(q^1, \dots, q^k)$; $q^i \in Q$
  - délka posloupnosti je lichá ($k = 2m + 1$)
  - žádný stav se neopakuje na liché ani na sudé pozici $(\forall i \neq j)\ (q^{2i} \neq q^{2j})\ \&\ (\forall i \neq j)(q^{2i+1} \neq q^{2j+1})$
- $F^{|} = \{(q) | q \in F\}$ posloupnosti délky 1
- $\delta^{|}(c, a) = \{d | d \in Q^{|} \& c \xrightarrow{a} d$ je lokálně konzistentní přechod pro $a\}$
  - existuje bijekce: $h : c_{odd} \cup d_{even} \rightarrow c_{even} \cup d_{odd}$, tak, že:
  - zachovává uspořádání
  - pro $h(q) \in c_{even}$ je $(h(q), -1) = \delta(q, a)$
  - pro $h(q) \in d_{odd}$ je $(h(q), +1) = \delta(q, a)$.

$L(A) = L(B)$

Trajektorie 2DFA A odpovídá řezům v FA B, odtud $L(A) = L(B)$.

Příklad převodu 2DFA na NKA

Možné řezy a jejich přechody

- Mějme následující dvousměrný konečný automat:

	a	b
$\rightarrow p$	p,+1	q,+1
$*q$	q,+1	r,-1
r	p,+1	r,-1

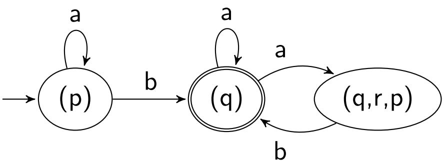
- Doleva jedině r – všechny sudé pozice r, tj. jediná sudá
- možné řezy: $(p), (q), (p, r, q), (q, r, p)$.

	a	b
$\rightarrow (p)$	(p)	(q)
$*(q)$	(q),(q,r,p)	
(p,r,q)		
(q,r,p)		(q)

Ukázka (zacykleného, nepřijímajícího) výpočtu:

	a		a		b		a		a		b		a		a		b		b	
p		p		p		q		q		q										
									r											
										p		q		q		q				
															r					
																p		q		
															r		r			

Výsledný NFA:



Automaty s výstupem (motivace)

- Dosud jediná zpráva z automatu: 'Jsme v přijímajícím stavu'.
- Můžeme z FA získat více informací? Můžeme zaznamenat trasu výpočtu?

Moore: indikace stavů (všech, nejen koncových)

- v každé chvíli víme, kde se automat nachází
- Příklad: různé (regulární) čítače

Mealy: indikace přechodů

- po přečtení každého symbolu víme, co automat dělal
- Příklad: regulární překlad slov

Automat už není tak docela černá skříňka.

Mooreův stroj

Definition 5.3 (Mooreův stroj)

Mooreovým (sekvenčním) strojem nazýváme šestici $A = (Q, \Sigma, Y, \delta, \mu, q_0)$ resp. pětici $A = (Q, \Sigma, Y, \delta, \mu)$, kde

Q je konečná neprázdná množina stavů

Σ je konečná neprázdná množina symbolů (vstupní abeceda)

Y je konečná neprázdná množina symbolů (**výstupní abeceda**)

δ je zobrazení $Q \times \Sigma \to Q$ (přechodová funkce)

μ je zobrazení $Q \to Y$ (**značkovací funkce**)

$q_0 \in Q$ (počáteční stav)

- Někdy nás nezajímá počáteční stav, ale jen práce automatu
- značkovací funkce umožňuje suplovat roli koncových stavů
  - $F \subseteq Q$ nahradíme značkovací funkcí $\mu : Q \to \{0, 1\}$ takto:
    $\mu(q) = 0$ pokud $q \notin F$,
    $\mu(q) = 1$ pokud $q \in F$.

Příklad Mooreova stroje

Example 5.2 (Mooreův stroj pro tenis)

Mooreův stroj pro počítání tenisového skóre.

- Vstupní abeceda: ID hráče, který uhrál bod
- Výstupní abeceda & stavy: skóre (tj. $Q = Y$ a $\mu(q) = q$)

Stav/výstup	A	B
00:00	15:00	00:15
15:00	30:00	15:15
15:15	30:15	15:30
00:15	15:15	00:30
30:00	40:00	30:15
30:15	40:15	30:30
30:30	40:30	30:40
15:30	30:30	15:40
00:30	15:30	00:40
40:00	A	40:15
40:15	A	40:30
40:30	A	shoda
30:40	shoda	B
15:40	30:40	B
00:40	15:00	B
shoda	A:40	40:B
A:40	A	shoda
40:B	shoda	B
A	15:00	00:15
B	15:00	00:15

Mealyho stroj

Definition 5.4 (Mealyho stroj)

Mealyho (sekvenčním) strojem nazýváme šestici $A = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$ resp. pětici $A = (Q, \Sigma, Y, \delta, \lambda_M)$, kde

Q je konečná neprázdná množina stavů

Σ je konečná neprázdná množina symbolů (vstupní abeceda)

Y je konečná neprázdná množina symbolů (výstupní abeceda)

δ je zobrazení $Q \times \Sigma \to Q$ (přechodová funkce)

λ_M je zobrazení $Q \times \Sigma \to Y$ (**výstupní funkce**)

$q_0 \in Q$ (počáteční stav)

- Výstup je určen stavem a vstupním symbolem
  - Mealyho stroj je obecnějším prostředkem než stroj Mooreův
  - Značkovací funkci $\mu : Q \to Y$ lze nahradit výstupní funkcí $\lambda_M : Q \times \Sigma \to Y$ například takto:

$$\forall x \in \Sigma \ \lambda_M(q, x) = \mu(q)$$

nebo $\forall x \in \Sigma \ \lambda_M(q, x) = \mu(\delta(q, x))$

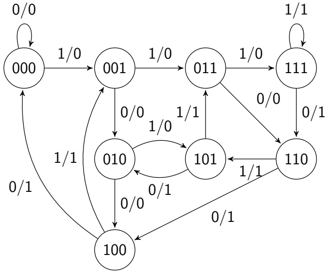
Příklad Mealyho stroje

Example 5.3 (Mealyho stroj)

Automat, který dělí vstupní slovo v bináním tvaru číslem 8 (celočíselně).

- Posun o tři bity doprava
- potřebujeme si pamatovat poslední trojici bitů
- vlastně tříbitová dynamická paměť

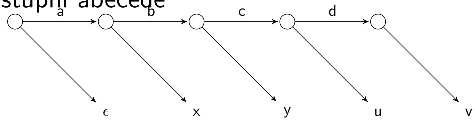
Stav\symbol	0	1
000	000/0	001/0
001	010/0	011/0
010	100/0	101/0
011	110/0	111/0
100	000/1	001/1
101	010/1	011/1
110	100/1	101/1
111	110/1	111/1



- I když nevíme, kde automat startuje, po třech symbolech začne počítat správně.

Výstup sekvenčních strojů

slovo ve vstupní abecedě $\rightarrow$ slovo ve výstupní abecedě



Mooreův stroj

značkovací funkce $\mu : Q \rightarrow Y$

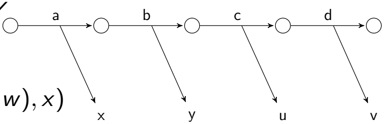
$\mu^* : Q \times \Sigma^* \rightarrow Y^*$

$\mu^*(q, \epsilon) = \epsilon$ (někdy $\mu^*(q, \epsilon) = q$)

$\mu^*(q, wx) = \mu^*(q, w).\mu(\delta^*(q, wx))$

Příklad: μ^*(00:00,AABA)=(00:00 .) 15:00 . 30:00 . 30:15 . 40:15

Mealyho stroj



výstupní funkce $\lambda_M : Q \times \Sigma \rightarrow Y$

$\lambda_M^* : Q \times \Sigma^* \rightarrow Y^*$

$\lambda_M^*(q, \epsilon) = \epsilon$

$\lambda_M^*(q, wx) = \lambda_M^*(q, w).\lambda_M(\delta^*(q, w), x)$

Příklad: λ_M^*(000,1101010)=0001101

Lemma (Převod Mooreova stroje na Mealyho)

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

Proof.

- Nechť $A = (Q, \Sigma, Y, \delta, \mu, q_0)$ je Mooreův stroj.
- Definujeme Mealyho stroj $B = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$, kde $\lambda_M(q, x) = \mu(\delta(q, x))$

  tj. λ_M vrací značku stavu, do kterého přejdeme.

$\square$

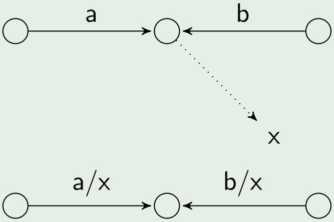
Example 5.4

Mooreův stroj

stav	0	1	výstup
a	a	b	0
b	b	c	1
c	c	a	2

Mealyho stroj
se stejným výstupem

stav	0	1
a	a/0	b/1
b	b/1	c/2
c	c/2	a/0



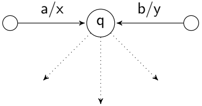
Převod Mealyho stroje na Mooreův

Lemma (Převod Mealyho stroje na Mooreův)

Pro každý Mealyho stroj existuje Mooreův stroj převádějící každé vstupní slovo na stejné výstupní slovo.

Nechť $A = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$ je Mealyho stroj.
Sestrojme Mooreův stroj B tak, aby $\forall q, w\ \lambda_M^*(q, w) = \mu^*(q, w)$.

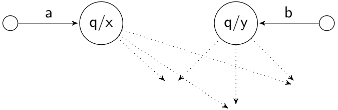
! Rozdělíme stav na více stavů, podle počtu výstupních symbolů.
$B = (Q \times (Y \cup \{\_\}), \Sigma, Y, \delta^{|}, \mu, (q_0, \_))$, kde
$\delta^{|}((q, y), x) = (\delta(q, x), \lambda_M(q, x))$ a
$\mu((q, y)) = y$



Příklad:

stav	0	1
a	a/0	b/0
b	a/1	b/1

stav	0	1	výstup
(a,0)	(a,0)	(b,0)	0
(a,1)	(a,0)	(b,0)	1
(b,0)	(a,1)	(b,1)	0
(b,1)	(a,1)	(b,1)	1



Konečné automaty – shrnutí

Konečný automat

- redukovaný deterministický automat (lze definovat i jednoznačný)
- nedeterminismus ϵ–NFA, 2^n, (dvousměrný FA n^n)

Regulární výrazy

Automaty a jazyky

- regulární jazyky
- uzavřenost na množinové operace
- uzavřenost na řetězcové operace
- uzavřenost na substituci, homomorfizmus a inverzní homomorfizmus,
- automaty výše i regulární výrazy popisují stejnou třídu jazyků.

Charakteristika regulárních jazyků

- Mihyll–Nerodova věta (kongruence)
- Kleeneova věta (elementární jazyky a operace)
- Iterační (pumping) lemma (iterace podslov, jen nutná podmínka).

(Automaty s výstupem)

- (Mooreův stroj)
- (Mealyho stroj)

Palindromy

Definition (palindrom)

Palindrom je řetězec w stejný při čtení zepředu i zedadu, tj. $w = w^R$.

- Příklady: `'otto'`; `'Madam, I'm Adam'`.

Lemma

Jazyk $L_{pal} = \{w|w = w^R, w \in \Sigma^*\}$ *není regulární.*

Example 6.1 (Bezkontextová gramatika pro palindromy)

$G = (\{S\}, \{0,1\}, P, S)$, $P =$

1. $S \rightarrow \epsilon$
2. $S \rightarrow 0$
3. $S \rightarrow 1$
4. $S \rightarrow 0S0$
5. $S \rightarrow 1S1$

Proof:

- Důkaz sporem. Předpokládejme L_{pal} je regulární, nechť n je konstanta z pumping lemma, uvažujme slovo: $w = 0^n 1 0^n$.
- z pumping lemmatu lze rozložit na $w = xyz$, y obsahuje jednu nebo více z prvních n nul. Tedy xz má být v L_{pal} ale není, tj. L_{pal} není regulární. $\square$

Formální (generativní) gramatiky, Bezkontextové gramatiky

Definition 6.1 (Formální (generativní) gramatika)

Formální (generativní) gramatika je $G = (V, T, P, S)$ složena z

- konečné množiny **neterminálů** (variables) V
- neprázdné konečné množiny **terminálních symbolů** (**terminálů**) T
- **počáteční symbol** $S \in V$.
- konečné množiny **pravidel** (**produkcí**) P reprezentující rekurzivní definici jazyka. Každé pravidlo má tvar:
  - $\beta A \gamma \rightarrow \omega$, $A \in V, \beta, \gamma, \omega \in (V \cup T)^*$

    tj. levá strana obsahuje aspoň jeden neterminální symbol.

Definition (Bezkontextová gramatika CFG)

Bezkontextová gramatika (CFG) je $G = (V, T, P, S)$ gramatika, obsahující pouze pravidla tvaru

$$A \rightarrow \omega,\ A \in V, \omega \in (V \cup T)^*.$$

Chomského hierarchie

Definition 6.2 (Klasifikace gramatik podle tvaru přepisovacích pravidel)

- **gramatiky typu 0** (**rekurzivně spočetné jazyky** $\mathcal{L}_0$)

  pravidla v obecné formě $\alpha \to \omega$, $\alpha, \omega \in (V \cup T)^*, \alpha$ obsahuje neterminál
- **gramatiky typu 1** (**kontextové gramatiky**, jazyky $\mathcal{L}_1$)
  - pouze pravidla ve tvaru $\gamma A \beta \to \gamma \omega \beta$

    $A \in V, \gamma, \beta \in (V \cup T)^*, \omega \in (V \cup T)^+$!
  - jedinou výjimkou je pravidlo $S \to \epsilon$, potom se ale S nevyskytuje na pravé straně žádného pravidla
- **gramatiky typu 2** (**bezkontextové gramatiky**, jazyky $\mathcal{L}_2$)

  pouze pravidla ve tvaru $A \to \omega, A \in V, \omega \in (V \cup T)^*$
- **gramatiky typu 3** (regulární/**pravé lineární gramatiky**, regulární jazyky $\mathcal{L}_3$)

  pouze pravidla ve tvaru $A \to \omega B, A \to \omega, A, B \in V, \omega \in T^*$.

Uspořádanost Chomského hierarchie

- Chomského hierarchie definuje uspořádání tříd jazyků

$$\mathcal{L}_0 \supseteq \mathcal{L}_1 \supseteq \mathcal{L}_2 \supseteq \mathcal{L}_3$$

- dokonce vlastní podmnožiny (později)

$\mathcal{L}_0 \supseteq \mathcal{L}_1$ rekurzivně spočetné jazyky zahrnují kontextové jazyky
  pravidla $\gamma A \beta \to \gamma \omega \beta$ obsahují vlevo neterminál A

$\mathcal{L}_2 \supseteq \mathcal{L}_3$ bezkontextové jazyky zahrnují regulární jazyky
  pravidla $A \to \omega B, A \to \omega$ obsahují vpravo řetězec $(V \cup T)^*$

$\mathcal{L}_1 \supseteq \mathcal{L}_2$ kontextové jazyky zahrnují bezkontextové jazyky
  problém je s pravidly typu $A \to \epsilon$, ale ta umíme eliminovat.

Example 6.2 (Notace)

$a, b, c, 1, *, ($	terminály
A, B, C	neterminály, proměnné
w, z	řetězec terminálů
X, Y	buď terminál nebo neterminál
α, β, γ	řetězec $(T \cup V)^*$
$A \to \alpha \vert \beta$	$\{A \to \alpha, A \to \beta\}$, OR, kompaktní zápis více pravidel.

Definition 6.3 (Derivace $\Rightarrow^*$)

Mějme gramatiku $G = (V, T, P, S)$.

- Říkáme, že α se **přímo přepíše** na ω (píšeme $\alpha \Rightarrow_G \omega$ nebo $\alpha \Rightarrow \omega$) jestliže
  $\exists \beta, \gamma, \eta, \nu \in (V \cup T)^* : \alpha = \eta\beta\nu,\ \omega = \eta\gamma\nu$ a $(\beta \rightarrow \gamma) \in P$.
- Říkáme, že α se **přepíše** na ω (píšeme $\alpha \Rightarrow^* \omega$) jestliže
  $\exists \beta_1, \ldots, \beta_n \in (V \cup T)^* : \alpha = \beta_1 \Rightarrow \beta_2 \Rightarrow \ldots \Rightarrow \beta_n = \omega$,
  tj. také $\alpha \Rightarrow^* \alpha$.
- Posloupnost $\beta_1, \ldots, \beta_n$ nazýváme **derivací (odvozením)**.
- Pokud $\forall i \neq j : \beta_i \neq \beta_j$, hovoříme o **minimálním odvození**.
- Libovolný řetězec $\omega \in (T \cup V)^*$ odvoditelný z počátečního symbolu nazýváme **sentenciální forma**.

Definition 6.4 (Jazyk generovaný gramatikou G)

Jazyk $L(G)$ generovaný gramatikou $G = (V, T, P, S)$ je množina terminálních řetězců, pro které existuje derivace ze startovního symbolu

$$L(G) = \{w \in T^* | S \Rightarrow_G^* w\}.$$

Jazyk neterminálu $A \in V$ definujeme $L(A) = \{w \in T^* | A \Rightarrow_G^* w\}$.

Gramatiky typu 3 a regulární jazyky

Definition (Gramatika typu 3, pravá lineární)

Gramatika G je **pravá lineární, tj. regulární, Typu 3**, pokud obsahuje pouze pravidla tvaru $A \to wB, A \to w, A, B \in V, w \in T^*$.

Example 6.3 (Příklad derivace gramatiky typu 3)

$$P = \{S \to 0S|1A|\epsilon,\ A \to 0A|1B,\ B \to 0B|1S\}$$

$$S \Rightarrow 0S \Rightarrow 01A \Rightarrow 011B \Rightarrow 0110B \Rightarrow 01101S \Rightarrow 01101$$

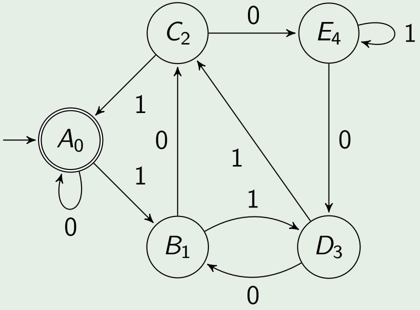
- Pozorování:
  - každá sentenciální forma derivace obsahuje právě jeden neterminál
  - tento neterminál je vždy umístěn zcela vpravo
  - aplikací pravidla $A \to w$ se derivace uzavírá
  - krok derivace generuje symboly a změní neterminál
- Idea vztahu gramatiky a konečného automatu
- neterminál = stav konečného automatu
- pravidla = přechodová funkce.

Příklad převodu FA na gramatiku

Example 6.4 (G, FA binární zápis čísla dělitelného 5)

$L = \{w | w \in \{0,1\}^* \& w$ je binární zápis čísla dělitelného 5$\}$

$$\begin{aligned} A &\rightarrow 1B|0A|\epsilon \\ B &\rightarrow 0C|1D \\ C &\rightarrow 0E|1A \\ D &\rightarrow 0B|1C \\ E &\rightarrow 0D|1E \end{aligned}$$



Příklady derivací

$$\begin{aligned} &A \Rightarrow 0A \Rightarrow 0 && (0) \\ &A \Rightarrow 1B \Rightarrow 10C \Rightarrow 101A \Rightarrow 101 && (5) \\ &A \Rightarrow 1B \Rightarrow 10C \Rightarrow 101A \Rightarrow 1010A \Rightarrow 1010 && (10) \\ &A \Rightarrow 1B \Rightarrow 11D \Rightarrow 111C \Rightarrow 1111A \Rightarrow 1111 && (15) \end{aligned}$$

Převod konečného automatu na gramatiku typu 3

Theorem 6.1 ($L \in RE \Rightarrow L \in \mathcal{L}_3$)

Pro každý jazyk rozpoznávaný konečným automatem existuje gramatika typu 3, která ho generuje.

Proof: Převod konečného automatu na gramatiku typu 3

- $L = L(A)$ pro deterministický konečný automat $A = (Q, \Sigma, \delta, q_0, F)$.
- definujme gramatiku $G = (Q, \Sigma, P, q_0)$, kde pravidla P mají tvar
  $p \to aq$, když $\delta(p, a) = q$
  $p \to \epsilon$, když $p \in F$
- je $L(A) = L(G)$?
  - $\epsilon \in L(A) \Leftrightarrow q_0 \in F \Leftrightarrow (q_0 \to \epsilon) \in P \Leftrightarrow \epsilon \in L(G)$
  - $a_1 \dots a_n \in L(A) \Leftrightarrow \exists q_0, \dots, q_n \in Q$ tž. $\delta(q_i, a_{i+1}) = q_{i+1}, q_n \in F$
    $\Leftrightarrow (q_0 \Rightarrow a_1 q_1 \Rightarrow \dots a_1 \dots a_n q_n \Rightarrow a_1 \dots a_n)$ je derivace pro $a_1 \dots a_n$
    $\Leftrightarrow a_1 \dots a_n \in L(G)$ $\square$

Příprava převodu gramatiky typu 3 na FA

- Opačný směr
  - pravidla $A \to aB$ kódujeme do přechodové funkce
  - pravidla $A \to \epsilon$ určují koncové stavy
  - pravidla $A \to a_1 \dots a_n B, A \to a_1 \dots a_n$ s více neterminály rozepíšeme
    - zavedeme nové neterminály $Y_2, \dots, Y_n, Z_1, \dots, Z_n$
    - vytvoříme pravidla $A \to a_1 Y_2, Y_2 \to a_2 Y_3, \dots, Y_n \to a_n B$
    - resp. $Z \to a_1 Z_1, Z_1 \to a_2 Z_2, \dots, Z_{n-1} \to a_n Z_n, Z_n \to \epsilon$
  - pravidla $A \to B$ odpovídají ϵ přechodům
    - zbavíme se jich tranzitivním uzávěrem
    - nebo musíme tranzitivně uzavřít $S \to B$ pro hledání $S \to \epsilon$.

Lemma

Ke každé gramatice typu 3 existuje gramatika typu 3, která generuje stejný jazyk a obsahuje pouze pravidla ve tvaru: $A \to aB, A \to \epsilon$*,* $A, B \in V, a \in T$*.*

Standardizace gramatiky typu 3

Lemma

Ke každé gramatice typu 3 existuje gramatika typu 3, která generuje stejný jazyk a obsahuje pouze pravidla ve tvaru: $A \to aB, A \to \epsilon$, $A, B \in V, a \in T$.

Proof.

Pro gramatiku $G = (V, T, S, P)$ definujeme $G^{|} = (V^{|}, T, S, P^{|})$, kde pro každé pravidlo zavedeme dostatečný počet nových neterminálů $Y_2, \ldots, Y_n, Z_1, \ldots, Z_n$ a definujeme

P	$P^{\vert}$
$A \to aB$ $A \to \epsilon$ $A \to a_1 \ldots a_n B$ $Z \to a_1 \ldots a_n$	$A \to aB$ $A \to \epsilon$ $A \to a_1 Y_2, Y_2 \to a_2 Y_3, \ldots Y_n \to a_n B$ $Z \to a_1 Z_1, Z_1 \to a_2 Z_2, \ldots, Z_{n-1} \to a_n Z_n, Z_n \to \epsilon$
odstraníme i pravidla: $A \to B$	tranzitivní uzávěr $U(A) = \{B \vert B \in V \& A \Rightarrow^* B\}$ $A \to \gamma$ pro všechna $Z \in U(A)$ a $(Z \to \gamma) \in P^{\vert}$

$\square$

Pouze pravidla $A \to aB, A \to \epsilon$

Example 6.5

P	P'
$B \to a_1$	$B \to a_1 H_1, H_1 \to \epsilon$
	$U(A) = \{A, B\}$, proto
$A \to B$ $A \to a_2$	$A \to a_1 H_2, H_2 \to \epsilon$ $A \to a_2 H_3, H_3 \to \epsilon$

Převod gramatiky typu 3 na konečný automat

Theorem 6.2 (ϵ–NFA pro gramatiku typu 3 rozpoznávající stejný jazyk)

Pro každý jazyk L generovaný gramatikou typu 3 existuje ϵ–NFA rozpoznávající L.

Proof: Převod gramatiky typu 3 na konečný automat

- Vezmeme $G = (V, T, P, S)$ obsahující jen pravidla tvaru $A \rightarrow aB$, $A \rightarrow \epsilon$, $A, B \in V, a \in T$ generující L (předchozí lemma)
- definujeme nedeterministický ϵ–NFA $A = (V, T, \delta, S, F)$, kde:

$$F = \{A | (A \rightarrow \epsilon) \in P\}$$
$$\delta(A, a) = \{B | (A \rightarrow aB) \in P\}$$

- $L(G) = L(A)$
  - $\epsilon \in L(G) \Leftrightarrow (S \rightarrow \epsilon) \in P \Leftrightarrow S \in F \Leftrightarrow \epsilon \in L(A)$
  - $a_1 \dots a_n \in L(G) \Leftrightarrow$ existuje derivace $(S \Rightarrow a_1 H_1 \Rightarrow \dots \Rightarrow a_1 \dots a_n H_n \Rightarrow a_1 \dots a_n)$

$\Leftrightarrow$ $\exists H_0, \dots, H_n \in V$ tak že $H_0 = S, H_n \in F$

$H_{i+1} \in \delta(H_i, a_k)$ pro krok $a_1 \dots a_{k-1} H_i \Rightarrow a_1 \dots a_{k-1} a_k H_{i+1}$

$\Leftrightarrow$ $a_1 \dots a_n \in L(A)$ □

Levé (a pravé) lineání gramatiky

Definition 6.5 (Levé (a pravé) lineání gramatiky)

Gramatiky typu 3 nazýváme také **pravé lineární** (neterminál je vždy vpravo).
Gramatika G je **levá lineání**, jestliže má pouze pravidla tvaru
$A \to Bw, A \to w, A, B \in V, w \in T^*$.

Lemma

Jazyky generované levou lineání gramatikou jsou právě regulární jazyky.

Proof:

$\Rightarrow$ 'otočením' pravidel levé lineární gramatiky dostaneme pravou lineární
$A \to Bw, A \to w$ převedeme na $A \to w^R B, A \to w^R$

- získaná gramatika generuje jazyk L^R, najdeme automat
- víme, že regulární jazyky jsou uzavřené na reverzi,
  L^R je regulární, tudíž i $L = (L^R)^R$ je regulární

$\Leftarrow$ takto lze získat všechny regulární jazyky

- (FA$\Rightarrow$reverze$\Rightarrow$ pravá lineární gramatika$\Rightarrow$ levá lineární gramatika). $\square$

Lineární gramatiky (a jazyky)

- Levá a pravá lineární pravidla dohromady jsou už silnější.

Definition 6.6 (lineární gramatika, jazyk)

Gramatika je lineární, jestliže má pouze pravidla tvaru $A \to uBw, A \to w, A, B \in V, u, w \in T^*$ (na pravé straně vždy maximálně jeden neterminál).
Lineární jazyky jsou právě jazyky generované lineáními gramatikami.

- Zřejmě platí: regulární jazyky $\subseteq$ lineární jazyky.
- Jde o vlastní podmnožinu $\subsetneq$.

Example 6.6 (lineární, neregulární jazyk)

Jazyk $L = \{0^i 1^i | i \geq 1\}$ není regulární jazyk, ale je lineární, generovaný gramatikou s pravidly $S \to 0S1|01$.

Pozorování:

- lineární pravidla lze rozložit na levě a pravě lineání pravidla: $S \to 0A, A \to S1$.

Bezkontextová gramatika pro jednoduché výrazy

Definition (Bezkontextová gramatika)

Bezkontextová gramatika je gramatika, kde všechna pravidla jsou tvaru $A \to \omega, \omega \in (V \cup T)^*$.

Example 6.7 (CFG pro jednoduché výrazy)

Gramatika pro jednoduché výrazy $G = (\{E, I\}, \{+, *, (,), a, b, 0, 1\}, P, E)$, P jsou pravidla vypsaná vpravo.

- Pravidla 1–4 definují výraz.
- Pravidla 5–10 definují identifikátor I, odpovídající regulárnímu výrazu $(\mathbf{a}+\mathbf{b})(\mathbf{a}+\mathbf{b}+\mathbf{0}+\mathbf{1})^*$.

CFG pro jednoduché výrazy

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

Derivační strom

Definition 6.7 (Derivační strom)

Mějme gramatiku $G = (V, T, P, S)$. **Derivační strom** pro G je strom, kde:

- Kořen (kreslíme nahoře) je označen startovním symbolem S,
- každý vnitřní uzel je ohodnocen neterminálem V.
- Každý uzel je ohodnocen prvkem $\in V \cup T \cup \{\epsilon\}$.
- Je–li uzel ohodnocen ϵ, je jediným dítětem svého rodiče.
- Je–li A ohodnocení vrcholu a jeho děti zleva pořadě jsou ohodnoceny $X_1, \ldots, X_k$, pak $(A \rightarrow X_1 \ldots X_k) \in P$ je pravidlo gramatiky.

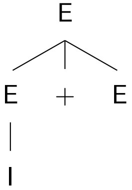
Notation 1 (Terminologie stromů)

Uzly, rodiče, děti, kořen, vnitřní uzly, listy, následníci, předci.

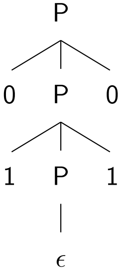
- Stromová struktura reprezentuje zdrojový program v překladači. Struktura usnadňuje překlad do strojového kódu.

Příklady stromů, Strom dává slovo (yield)

Derivační strom $E \Rightarrow^* I + E$.



Derivační strom $P \Rightarrow^* 0110$.



Definition 6.8 (Strom dává slovo (yield))

Říkáme, že **derivanční strom dává slovo w (yield)**, jestliže w je slovo složené z ohodnocení listů bráno zleva doprava.

Levá a pravá derivace

Definition 6.9 (Levá a pravá derivace)

Levá derivace (leftmost) $\Rightarrow_{lm}$, $\Rightarrow_{lm}^*$ v každém kroku přepisuje nejlevnější neterminál.
Pravá derivace (rightmost) $\Rightarrow_{rm}$, $\Rightarrow_{rm}^*$ v každém kroku přepisuje nejpravější neterminál.

Example 6.8 (levá derivace)

$E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E \Rightarrow_{lm} a * (E) \Rightarrow_{lm} a * (E + E) \Rightarrow_{lm} a * (I + E) \Rightarrow_{lm}$
$\Rightarrow_{lm} a * (a + E) \Rightarrow_{lm} a * (a + I) \Rightarrow_{lm} a * (a + I0) \Rightarrow_{lm} a * (a + I00) \Rightarrow_{lm} a * (a + b00)$

Pravá derivace používá stejné přepisy, jen je provádí v jiném pořadí.

Example 6.9 (rightmost derivation)

$E \Rightarrow_{rm} E * E \Rightarrow_{rm} E * (E) \Rightarrow_{rm} E * (E + E) \Rightarrow_{rm} E * (E + I) \Rightarrow_{rm}$
$\Rightarrow_{rm} E * (E + I0) \Rightarrow_{rm} E * (E + I00) \Rightarrow_{rm} E * (E + b00) \Rightarrow_{rm}$
$\Rightarrow_{rm} E * (I + b00) \Rightarrow_{rm} E * (a + b00) \Rightarrow_{rm} I * (a + b00) \Rightarrow_{rm} a * (a + b00)$

Derivace a derivační stromy

Theorem 6.3

Pro danou gramatiku $G = (V, T, P, S)$ a $w \in T^*$ jsou následující tvrzení ekvivalentní:

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

Od stromů k derivaci

Lemma

Mějme CFG $G = (V, T, P, S)$ *a derivační strom s kořenem* A *dávající slovo* $w \in T^*$.
Pak existuje levá derivace $A \Rightarrow^*_{lm} w$ *v* G.

Příprava důkazu: 'obalení derivace'

Mějme následující derivaci:

$$E \Rightarrow I \Rightarrow Ib \rightarrow ab.$$

Pro libovolná slova $\alpha, \beta \in (V \cup T)^*$ je také derivace:

$$\alpha E \beta \Rightarrow \alpha I \beta \Rightarrow \alpha Ib \beta \Rightarrow \alpha ab \beta.$$

Proof: $\exists$ derivační strom pak existuje levá derivace $\Rightarrow_{lm}$

Indukcí podle výšky stomu.

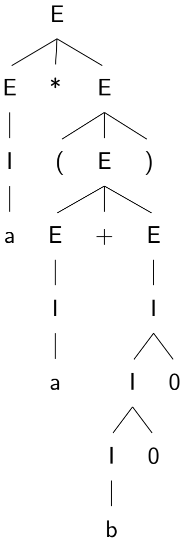
- Základ: výška 1: Kořen A s dětmi dávajícími w. Je to derivační strom, proto, $A \rightarrow w$ je pravidlo $\in P$, tedy $A \Rightarrow_{lm} w$ v jednom kroku.
- Indukce: výška $n > 1$. Kořen A s dětmi $X_1, X_2, \ldots, X_k$.
  - Je–li $X_i \in T$, definujeme $w_i \equiv X_i$.
  - Je–li $X_i \in V$, z indukčního předpokladu $X_i \Rightarrow_{lm}^* w_i$.

  Levou derivaci konstruujeme induktivně pro $i = 1, \ldots, k$ složíme $A \Rightarrow_{lm}^* w_1 w_2 \ldots w_i X_{i+1} X_{i+2} \ldots X_k$.
  - Pro $X_i \in T$ jen zvedneme čítač $i++$.
  - Pro $X_i \in V$ přepíšeme derivaci: $X_i \Rightarrow_{lm} \alpha_1 \Rightarrow_{lm} \alpha_2 \ldots \Rightarrow_{lm} w_i$ na

$$\begin{aligned} w_1 w_2 \ldots w_{i-1} X_i X_{i+1} X_{i+2} \ldots X_k &\Rightarrow_{lm} \\ w_1 w_2 \ldots w_{i-1} \alpha_1 X_{i+1} X_{i+2} \ldots X_k &\Rightarrow_{lm} \\ &\ldots \\ \Rightarrow_{lm} w_1 w_2 \ldots w_{i-1} w_i X_{i+1} X_{i+2} \ldots X_k&. \end{aligned}$$

Pro $i = k$ dostaneme levou derivaci w z A. $\square$

Příklad levé derivace z derivačního stromu



Je příjemnější zachytit derivaci stromem.

- Kořen: $E \Rightarrow_{lm} E * E$
- Levé dítě kořene: $E \Rightarrow_{lm} I \Rightarrow_{lm} a$
- Kořen a levé dítě: $E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E$
- Pravé dítě kořene:
  $E \Rightarrow_{lm} (E) \Rightarrow_{lm} (E + E) \Rightarrow_{lm} (I + E) \Rightarrow_{lm} (a + E)$
  $\Rightarrow_{lm} (a+I) \Rightarrow_{lm} (a+I0) \Rightarrow_{lm} (a+I00) \Rightarrow_{lm} (a+b00)$
- Plná derivace:
  $E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E \Rightarrow_{lm}$
  $\Rightarrow_{lm} a * (E) \Rightarrow_{lm} a * (E + E) \Rightarrow_{lm} a * (I + E) \Rightarrow_{lm}$
  $\Rightarrow_{lm} a * (a + E) \Rightarrow_{lm} a * (a + I) \Rightarrow_{lm} a * (a + I0) \Rightarrow_{lm}$
  $\Rightarrow_{lm} a * (a + I00) \Rightarrow_{lm} a * (a + b00)$.

Ekvivalence gramatik

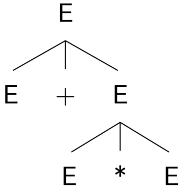
Definition 6.10 (ekvivalence gramatik)

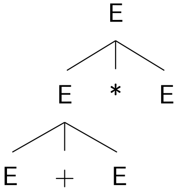
Gramatiky G_1, G_2 jsou **ekvivalentní**, jestliže $L(G_1) = L(G_2)$, tj. generují stejný jazyk.

Víceznačnost gramatik

Dvě derivace téhož výrazu:

$E \Rightarrow E + E \Rightarrow E + E * E \quad E \Rightarrow E * E \Rightarrow E + E * E$





- Rozdíl je důležitý, vlevo $1 + (2 * 3) = 7$, vpravo $(1 + 2) * 3 = 9$.
- Tato gramatika může být modifikovaná na jednoznačnou.

Example 6.10

Různé derivace mohou reprezentovat stejný derivační strom, pak není problém.

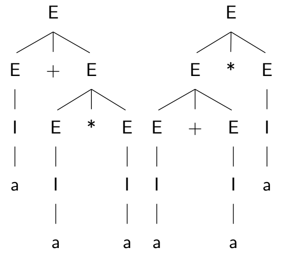
1. $E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$
2. $E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$.

Definition 6.11 (Jednoznačnost a víceznačnost CFG)

- Bezkontextová gramatika $G = (V, T, P, S)$ je **víceznačná** pokud existuje aspoň jeden řetězec $w \in T^*$ pro který můžeme najít dva různé derivační stromy, oba s kořenem S dávající slovo w.
- V opačném případě nazýváme gramatiku **jednoznačnou**.
- Bezkontextový jazyk L je **jednoznačný**, jestliže existuje jednoznačná CFG G tak, že $L = L(G)$.
- **Bezkontextový jazyk L je (podstatně) nejednoznačný**, jestliže každá CFG G taková, že $L = L(G)$, je nejednoznačná. Takovému jazyku říkáme i **víceznačný**.

Example 6.11 (nejednoznačnost CFG)

Dva derivační stromy dávající $a + a * a$ ukazující víceznačnost gramatiky.



Příklad víceznačného jazyka

Example 6.12 (Víceznačný jazyk)

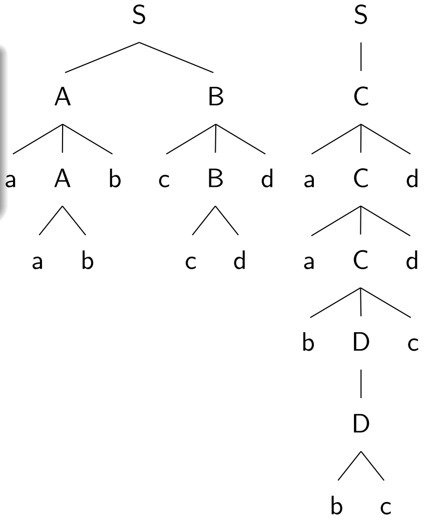
Příklad víceznačného jazyka:

$$L = \{a^n b^n c^m d^m | n \geq 1, m \geq 1\} \cup \\ \cup \{a^n b^m c^m d^n | n \geq 1, m \geq 1\}.$$

1. $S \to AB|C$
2. $A \to aAb|ab$
3. $B \to cBd|cd$
4. $C \to aCd|aDd$
5. $D \to bDc|bc$.

Jakákoli gramatika pro daný jazyk bude generovat pro některá slova typu $a^n b^n c^n d^n$ dva různé derivační stromy.

Dva derivační stromy pro *aabbccdd*.



Odstanění víceznačnosti gramatiky

- Neexistuje algoritmus, který nám řekne, zda je daná gramatika víceznačná.
- Existují bezkontextové jazyky, pro které neexistuje jednoznačná bezkontextová gramatika, pouze víceznačné CFG.
- Existují určitá doporučení pro odstranění víceznačnosti.

Víceznačnost má různé příčiny:

- Není respektovaná priorita operátorů.
- Posloupnost identických operátorů lze shlukovat zleva i zprava.
- $S \rightarrow$ if then S else $S|$ if then $S|\epsilon$

  slovo 'if then if then else' má dva významy

  'if then (if then else)' nebo 'if then (if then) else'

  Řešení:

  - syntaktická chyba (Algol 60)
  - else patří k bližšímu if (preference pořadí pavidel)
  - závorky begin-end,odsazení v Python (asi nejčistší řešení).

Vynucení priority

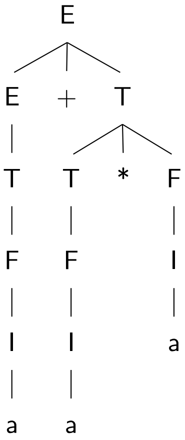
Řešením je zavést více různých proměnných, každou pro jednu úroveň 'priority'.
Konkrétně:

- **Faktor** je výraz který nesmí rozdělit žádný operátor.
    - identifikátory
    - výraz v závorkách
- **Term** je výraz, který nemůže rozdělit operátor $+$.
- **Výraz** může být rozdělen $*$ i $+$.

Jednoznačná gramatika pro výrazy:

1. $I \to a|b|Ia|Ib|I0|I1$
2. $F \to I|(E)$
3. $T \to F|T * F$
4. $E \to T|E + T$.

Jediný derivační strom pro $a + a * a$.



Jednoznačnost a kompilátory

Kompilace výrazu (zásobník na mezivýsledky + dva registry):

(1)	$E \to E + T$	... pop r1; pop r2; add r1,r2; push r2
(2)	$E \to T$	
(3)	$T \to T * F$	... pop r1; pop r2; mul r1,r2; push r2
(4)	$T \to F$	
(5)	$F \to (E)$	
(6)	$F \to a$	... push a

- 'a+a*a' získáme postupnou aplikací pravidel 1,2,4,6,3,4,6,6
- posloupnost obrátíme a vybereme pouze pravidla generující kód

  6,6,3,6,1
- nyní nahradíme pravidla příslušným kódem

  push a; push a; pop r1; pop r2; mul r1,r2; push r2; push a; pop r1; pop r2; add r1,r2; push r2

Shrnutí

- Gramatiky
  - obecné
  - kontextové
  - bezkontextové
  - regulární, pravé lineární
- jazyk gramatiky, derivace, derivace dává slovo, derivační strom (pro bezkontextové gramatiky), ekvivalentní gramatiky
- ne každá lineární gramatika má ekvivalentní pravou lineární
- bezkontextové gramatiky
- jednoznačné a (podstatně) víceznačné gramatiky.

Chomského normální forma

- Chomského normální forma bezkontextové gramatiky:
  - neobsahuje zbytečné symboly
  - všechna pravidla tvaru $A \to BC$ nebo $A \to a$, A, B, C jsou neterminály, a terminál.
  - Pro každý bezkontextový jazyk L, kde $L \setminus \{\epsilon\} \neq \emptyset$ existuje gramatika v Chomského normálním tvaru, která generuje $L \setminus \{\epsilon\}$.

Postupně provedeme zjednodušení gramatiky, nejdřív:

- Eliminace *zbytečných symbolů*
- eliminace ϵ*–pravidel* $A \to \epsilon$; $A \in V$
- eliminace *jednotkových pravidel* $A \to B$ pro $A, B \in V$.

- To vše potřebujeme k formulaci iteračního lemmatu pro bezkontextové jazyky.

Eliminace zbytečných symbolů

Definition 7.1 (zbytečný, užitečný, generující, dosažitelný symbol)

- Symbol X je **užitečný** v gramatice $G = (V, T, P, S)$ pokud existuje derivace tvaru $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ kde $w \in T^*, X \in (V \cup T)$, $\alpha, \beta \in (V \cup T)^*$.
- Pokud X není užitečný, říkáme, že je **zbytečný**.
- X je **generující** pokud $X \Rightarrow^* w$ pro nějaké slovo $w \in T^*$. Vždy $w \Rightarrow^* w$ v nula krocích.
- X je **dosažitelný** pokud $S \Rightarrow^* \alpha X \beta$ pro nějaká $\alpha, \beta \in (V \cup T)^*$.

Chceme eliminovat ne–generující a ne–dosažitelné symboly.

Example 7.1

Uvažujme gramatiku:
$S \rightarrow AB|a$
$A \rightarrow b$

Eliminujeme B (ne–generující):
$S \rightarrow a$
$A \rightarrow b$.

Eliminujeme A (nedosažitelný):
$S \rightarrow a$.

Lemma 7.1 (Eliminace zbytečných symbolů)

Nechť $G = (V, T, P, S)$ je CFG, předpokládejme $L(G) \neq \emptyset$. Zkonstruujeme $G_1 = (V_1, T_1, P_1, S)$ následovně:

- *Eliminujeme ne–generující symboly a pravidla je obsahující*
- *poté eliminujeme všechny nedosažitelné symboly*

Pak G_1 nemá zbytečné symboly a $L(G_1) = L(G)$.

Algorithm: Generující symboly

Základ Každý $a \in T$ je generující.

Indukce Pro každé pravidlo $A \to \alpha$, kde každý symbol v α je generující. Pak i A je generující. (Včetně $A \to \epsilon$).

Algorithm: Dosažitelné symboly

Základ S je dosažitelný.

Indukce Je–li A dosažitelný, pro všechna pravidla $A \to \alpha$ jsou všechny symboly v α dosažitelné.

Lemma 7.2 (generující/dosažitelné symboly)

Výše uvedené algoritmy najdou právě všechny generující / dosažitelné symboly.

Eliminace ϵ pravidel

Definition 7.2 (nulovatelný neterminál)

Neterminál A je **nulovatelný** pokud $A \Rightarrow^* \epsilon$.

Pro nulovatelné neterminály na pravé straně pravidla $B \rightarrow CAD$, vytvoříme dvě verze pravidla – s a bez nulovatelného neterminálu.

Algorithm: Nalezení nulovatelných symbolů v G

Základ Pokud $A \rightarrow \epsilon$ je pravidlo G, pak A je nulovatelné.

Indukce Pokud $B \rightarrow C_1 \dots C_k$, kde jsou všechna C_i nulovatelná, je i B nulovatelné (terminály nejsou nulovatelné nikdy).

Algorithm: Konstrukce gramatiky bez ϵ–pravidel z G

- Najdi nulovatelné symboly
- Pro každé pravidlo $A \rightarrow X_1 \dots X_k \in P, k \geq 1$, nechť m z X_i je nulovatelných. Nová gramatika G_1 bude mít 2^m verzí tohoto pravidla s/bez každého nulovatelného symbolu kromě ϵ v případě $m = k$.

Příklad eliminace ϵ–pravidel

Example 7.2

Mějme gramatiku:

$S \rightarrow AB$
$A \rightarrow aAA|\epsilon$
$B \rightarrow bBB|\epsilon$

$S \rightarrow AB|A|B$
$A \rightarrow aAA|aA|aA|a$
$B \rightarrow bBB|bB|bB|b$

Výsledná gramatika:

$S \rightarrow AB|A|B$
$A \rightarrow aAA|aA|a$
$B \rightarrow bBB|bB|b$.

Eliminace jednotkových pravidel

Definition 7.3 (jednotkové pravidlo)

Jednotkové pravidlo je $A \to B \in P$ kde A, B jsou oba neterminály.

Example 7.3

$I \to a|b|Ia|Ib|I0|I1$
$F \to I|(E)$
$T \to F|T * F$
$E \to T|E + T$

Expanze T v $E \to T$
$E \to F|T * F$
Expanze $E \to F$
$E \to I|(E)$

Expanze $E \to I$
$E \to a|b|Ia|Ib|I0|I1$

Dohromady: $E \to a|b|Ia|Ib|I0|I1|(E)|T * F|E + T$.

Musíme se vyhnout možným cyklům.

Definition 7.4 (jednotkový pár)

Dvojici $A, B \in V$ takovou, že $A \Rightarrow^* B$ pouze jednotkovými pravidly nazýváme **jednotkový pár** (jednotková dvojice).

Algorithm: Nalezení jednotkových párů

Základ (A, A) pro každý $A \in V$ je jednotkový pár.

Indukce Je–li (A, B) jednotkový pár a $(B \rightarrow C) \in P$, pak (A, C) je jednotkový pár.

Example 7.4 (Jednotkové páry z předchozí gramatiky)

$(E, E), (T, T), (F, F), (I, I), (E, T), (E, F), (E, I), (T, F), (T, I), (F, I)$.

Algorithm: Eliminace jednotkových pravidel z G

- najdi všechny jednotkové páry v G
- pro každý jednotkový pár (A, B) dáme do nové gramatiky všechna pravidla $A \rightarrow \alpha$ kde $B \rightarrow \alpha \in P$ a $B \rightarrow \alpha$ není jednotkové pravidlo.

Example 7.5

$I \rightarrow a \vert b \vert Ia \vert Ib \vert I0 \vert I1$	$I \rightarrow a \vert b \vert Ia \vert Ib \vert I0 \vert I1$
$F \rightarrow I \vert (E)$	$F \rightarrow (E) \vert a \vert b \vert Ia \vert Ib \vert I0 \vert I1$

Gramatiky v normálním tvaru

Lemma 7.3 (Gramatika v normálním tvaru, redukovaná)

Mějme bezkontextovou gramatiku G, $L(G) - \{\epsilon\} \neq \emptyset$. pak existuje CFG G_1 taková že $L(G_1) = L(G) - \{\epsilon\}$ a G_1 neobsahuje ϵ–pravidla, jednotková pravidla ani zbytečné symboly. Gramatika G_1 se nazývá ***redukovaná****.*

Proof.

Idea důkazu:

- Začneme eliminací ϵ–pravidel.
- Eliminujeme jednotková pravidla. Tím nepřidáme ϵ–pravidla.
- Eliminujeme zbytečné symboly. Tím nepřidáme žádná pravidla.
  - Nejdříve negenerující,
  - pak nedosažitelné.

□

Definition 7.5 (Chomského normální tvar)

O bezkontextové gramatice $G = (V, T, P, S)$ bez zbytečných symbolů kde jsou všechna pravidla v jednom ze dvou tvarů

- $A \rightarrow BC$, $A, B, C \in V$,
- $A \rightarrow a$, $A \in V$, $a \in T$,

říkáme, že je v **Chomského normálním tvaru (ChNF)**.

Potřebujeme dva další kroky:

- pravé strany délky 2 a více předělat na samé neterminály
- rozdělit pravé strany délky 3 a více neterminálů na více pravidel

Algorithm: neterminály

- Pro každý terminál a vytvoříme nový neterminál, řekněme A,
- přidáme pravidlo $A \rightarrow a$,
- použijeme A místo a na pravé straně pravidel délky 2 a více.

Algorithm: rozdělení pravidel

- Pro pravidlo $A \rightarrow B_1 \dots B_k$ zavedeme $k-2$ neterminálů C_i
- Přidáme pravidla $A \rightarrow B_1 C_1, C_1 \rightarrow B_2 C_2, \dots, C_{k-2} \rightarrow B_{k-1} B_k$.

Theorem 7.1 (Chomského normální tvar bezkontextové gramatiky)

Mějme bezkontextovou gramatiku G, $L(G)-\{\epsilon\}\neq\emptyset$. Pak existuje CFG G_1 v Chomského normálním tvaru taková, že $L(G_1)=L(G)-\{\epsilon\}$.

Example 7.6

$I\rightarrow a|b|Ia|Ib|I0|I1$
$F\rightarrow I|(E)$
$T\rightarrow F|T*F$
$E\rightarrow T|E+T$

$I\rightarrow a|b|IA|IB|IZ|IU$
$F\rightarrow LER|a|b|IA|IB|IZ|IU$
$T\rightarrow TMF|LER|a|b|IA|IB|IZ|IU$
$E\rightarrow EPT|TMF|LER|a|b|IA|IB|IZ|IU$
$A\rightarrow a$
$B\rightarrow b$
$Z\rightarrow 0$
$U\rightarrow 1$
$P\rightarrow +$
$M\rightarrow *$
$L\rightarrow ($
$R\rightarrow)$

$F\rightarrow LC_3|a|b|IA|IB|IZ|IU$
$T\rightarrow TC_2|LC_3|a|b|IA|IB|IZ|IU$
$E\rightarrow EC_1|TC_2|LC_3|a|b|IA|IB|IZ|IU$
$C_1\rightarrow PT$
$C_2\rightarrow MF$
$C_3\rightarrow ER$
$I, A, B, Z, U, P, M, L, R$ jako vlevo

Příprava na (pumping) lemma o vkládání

Lemma (Velikost derivačního stromu gramatiky v CNF)

Mějme derivační strom podle gramatiky $G = (V, T, P, S)$ v Chomského normálním tvaru, který dává slovo w. Je–li délka nejdelší cesty n, pak $|w| \leq 2^{n-1}$.

Proof.

Indukcí podle n,

Základ $|a| = 1 = 2^0$

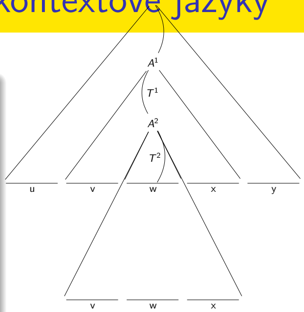
Indukce $2^{n-2} + 2^{n-2} = 2^{n-1}$.

$\square$

Lemma (Důsledek)

Mějme derivační strom podle gramatiky $G = (V, T, P, S)$ v Chomského normální formě, který dává slovo w, $|w| > p = 2^{n-1}$. Pak ve stromě existuje cesta delší než n.

Lemma o vkládání (pumping) pro bezkontextové jazyky



Theorem 7.2 (!!Lemma o vkládání (pumping) pro bezkontextové jazyky)

Mějme bezkontextový jazyk L. Pak existuje přirozené číslo $n \in \mathbb{N}$ takové, že každé $z \in L, |z| > n$ lze rozložit na $z = uvwxy$ kde:

- $|vwx| \leq n$
- $vx \neq \epsilon$
- $\forall i \geq 0,\ uv^iwx^iy \in L$.

Idea důkazu:

- vezmeme derivační strom pro z
- najdeme nejdelší cestu
- na ní dva stejné neterminály
- tyto neterminály určí dva podstromy
- podstromy definují rozklad slova
- nyní můžeme větší podstrom posunout ($i > 1$)
- nebo nahradit menším podstromem ($i = 0$)

Proof: $|z| > n : z = uvwxy, |vwx| \le n, vx \ne \epsilon, \forall i \ge 0 uv^iwx^iy \in L$

- vezmeme gramatiku v Chomského NF (pro $L = \{\epsilon\}$ a $\emptyset$ zvol $n = 1$).
- Nechť $|V| = k$. Položíme $n = 2^k$.
- Pro $z \in L, |z| > 2^k$, má v derivačním stromu z cestu délky $> k$
- vezmeme cestu maximální délky; terminál kam vede označíme t
- Aspoň dva z posledních $(k+1)$ neterminálů na cestě do t jsou stejné
- vezmeme dvojici A^1, A^2 nejblíže k t (určuje podstromy T^1, T^2)
- cesta z A^1 do t je nejdelší v podstromu T^1 a má délku maximálně $(k+1)$

  tedy slovo dané stromem T^1 není delší než 2^k (tedy $|vwx| \le n$)
- z A^1 vedou dvě cesty (ChNF), jedna do T^2 druhá do zbytku vx

  ChNF je nevypouštějící, tedy $vx \ne \epsilon$
- derivace slova ($A^1 \Rightarrow^* vA^2x, A^2 \Rightarrow^* w$)

  $S \Rightarrow^* uA^1y \Rightarrow^* uvA^2xy \Rightarrow^* uvwxy$
- posuneme–li A^2 do A^1 $(i = 0)$

  $S \Rightarrow^* uA^2y \Rightarrow^* uwy$
- posuneme–li A^1 do A^2 $(i = 2, 3, \ldots)$

  $S \Rightarrow^* uA^1y \Rightarrow^* uvA^1xy \Rightarrow^* uvvA^2xxy \Rightarrow^* uvvwxxy$. $\square$

Použití lemma o vkládání

Example 7.7 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{0^i 1^i 2^i | i \geq 1\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 2^n| > n$

\# žádné dělení nesplňuje PL neboť

- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé symboly
- $vx \neq \epsilon$, iterací se slovo změní
- poruší se rovnost počtu symbolů – SPOR.

Example 7.8 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{0^i 1^j 2^k | 0 \leq i \leq j \leq k\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 2^n| > n$

\# žádné dělení nesplňuje PL neboť

- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé symboly
  - pokud 0 (nebo 1), pumpujeme nahoru – SPOR $i \leq j$ (nebo $j \leq k$)
  - pokud 2 (nebo 1), pumpujeme dolů – SPOR $j \leq k$ (nebo $i \leq j$)

Použití lemma o vkládání

Example 7.9 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{0^i 1^j 2^i 3^j | i, j \geq 1\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 2^n 3^n| > n$
- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé sousední symboly
- poruší se rovnost počtu symbolů 0 a 2 nebo 1 a 3 – SPOR.

Example 7.10 (ne–bezkontextový jazyk)

Následující jazyk není bezkontextový

- $\{ww | w \in \{0, 1\}^*\}$

- důkaz sporem: předpokládejme bezkontextovost
- z lemmatu o vkládání máme n
- zvolme $z = |0^n 1^n 0^n 1^n| > n$
- pumpovací slovo $|vwx| \leq n$
- tj. vždy lze pumpovat maximálně dva různé sousední symboly
- poruší se buď rovnost nul či jedniček
  - rozeberete 4 případy, vx obsahuje znak z prvních nul, prvních jedniček, druhých nul, druhých jedniček.

Kdy lemma o vkládání nezabere

- Lemma o vkládání je pouze implikace!

Example 7.11 (pumpovatelný, ne–bezkontextový jazyk)

$L = \{a^i b^j c^k d^l | i = 0 \vee j = k = l\}$ není bezkontextový jazyk, přesto lze pumpovat.

$i = 0 : b^j c^k d^l$ lze pumpovat v libovolném písmenu
$i > 0 : a^i b^n c^n d^n$ lze pumpovat v části obsahující a

Co s tím?

- zobecnění pumping lemmatu (Ogdenovo lemma)
  - pumpování vyznačených symbolů
- uzávěrové vlastnosti.

Příklad gramatiky

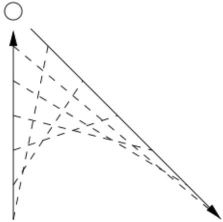
Example 8.1 (Uzávorkování v ChNF)

Mějme gramatiku $G = (\{S, L, R\}, \{(,)\}, P, S)$, kde $P = \left\{\begin{array}{lcl} S & \rightarrow & LR|SS|LA \\ A & \rightarrow & SR \\ L & \rightarrow & (\\ R & \rightarrow &) \end{array}\right\}$.

- $S \Rightarrow LR \Rightarrow (R \Rightarrow ()$
- $S \Rightarrow SS \Rightarrow LRS \Rightarrow (RS \Rightarrow ()S \Rightarrow ()LR \Rightarrow ()(R \Rightarrow ()()$
- $S \Rightarrow LA \Rightarrow (A \Rightarrow (SR \Rightarrow (LRR \Rightarrow ((RR \Rightarrow (()R \Rightarrow (())$

Příklad CYK: pro slovo najít derivační strom

- Chceme rozhodnout, zda dané slovo w patří do jazyka bezkontextové gramatiky
  - Vezmeme gramatiku v Chomského normální formě
  - exponenciální složitost: prozkoušíme všechny derivační stromy do hloubky $|n|$,
  - polynomiálně: Cocke-Younger-Kasami algoritmus.



Example 8.2 (CYK algoritmus)

Gramatika $G = (\{S, A, L, R\}, \{(,)\}, P, S)$:

$$P = \left\{ \begin{array}{lcl} S & \rightarrow & LR|SS|LA \\ A & \rightarrow & SR \\ L & \rightarrow & (\\ R & \rightarrow &) \end{array} \right\}$$

Tabulku vyplňujeme odspodu:

$\{S\}$					
$\{\}$	$\{A\}$				
$\{\}$	$\{S\}$	$\{\}$			
$\{\}$	$\{\}$	$\{\}$	$\{A\}$		
$\{\}$	$\{S\}$	$\{\}$	$\{S\}$	$\{\}$	
$\{L\}$	$\{L\}$	$\{R\}$	$\{L\}$	$\{R\}$	$\{R\}$
(	(	)	(	)	)

Cocke-Younger-Kasami algoritmus náležení slova do CFL

Algorithm: CYK algoritmus, v čase $O(n^3)$

- Mějme gramatiku v ChNF $G = (V, T, P, S)$ pro jazyk L a slovo $w = a_1 a_2 \dots a_n \in T^*$.
- Vytvořme trohúhelníkovou tabulku (vpravo),
  - horizontální osa je w
  - X_{ij} jsou množiny neterminálů A takových, že $A \Rightarrow^* a_i a_{i+1} \dots a_j$.

  Základ: $X_{ii} = \{A; A \rightarrow a_i \in P\}$

  Indukce: $X_{ij} = \{A \rightarrow BC; B \in X_{ik}, C \in X_{k+1,j}\}$
- Vyplňujeme tabulku zdola nahoru.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

Theorem 8.1 (CYK)

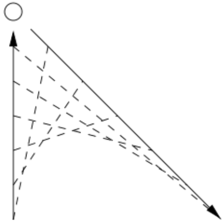
Slovo w patří do jazyka gramatiky G právě když je v CYK algoritmu $S \in X_{1,n}$.

Obecný příklad CYK

Example 8.3 (CYK algoritmus)

Uvažujme gramatiku
$G = (\{S, A, B, C\}, \{a, b\}, P, S)$:

$$P = \left\{ \begin{array}{lcl} S & \rightarrow & AB|BC \\ A & \rightarrow & BA|a \\ B & \rightarrow & CC|b \\ C & \rightarrow & AB|a \end{array} \right\}.$$



Pravidla pozpátku:

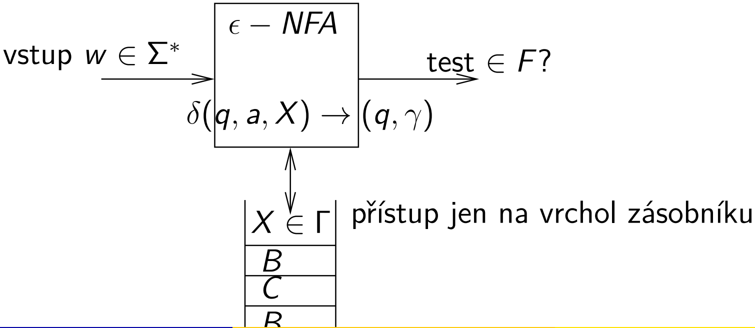
$$\begin{array}{rcl} AB & \rightarrow & \{S, C\} \\ BA & \rightarrow & \{A\} \\ BC & \rightarrow & \{S\} \\ CC & \rightarrow & \{B\} \\ b & \rightarrow & \{B\} \\ a & \rightarrow & \{A, C\} \end{array}$$

Tabulku vyplňujeme odspodu:

$\{S, A, C\}$				
-	$\{S, A, C\}$			
-	$\{B\}$	$\{B\}$		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

Zásobníkové automaty

- Zásobníkové automaty jsou rozšířením $\epsilon-$NFA nedeterministických konečných automatů s ϵ přechody.
- Přidanou věcí je **zásobník**. Má vlastní abecedu Γ.
- V každém kroku vidíme horní písmeno zásobníku (zde X), můžeme dát navrch libovolný konečný počet znaků $\gamma \in \Gamma^*$.
- Může si pamatovat neomezené množství informace.
- Deterministické zásobníkové automaty přijímají jen vlastní podmnožinu bezkontextových jazyků.



Zásobníkový automat (PDA)

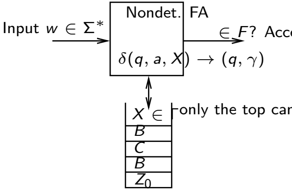
Definition 9.1 (Zásobníkový automat (PDA))

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

- Q konečná množina stavů
- Σ neprázdná konečná množina vstupních symbolů
- Γ neprázdná konečná zásobníková abeceda
- δ přechodová funkce $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to P_{FIN}(Q \times \Gamma^*)$, $\delta(p, a, X) \ni (q, \gamma)$
  - kde q je nový stav a γ je řetězec zásobníkových symbolů, který nahradí X na vrcholu zásobníku
- $q_0 \in Q$ počáteční stav
- $Z_0 \in \Gamma$ Počáteční zásobníkový symbol. Víc na začátku na zásobníku není.
- F Množina přijímajících (koncových) stavů; může být nedefinovaná.

V jednom časovém kroku zásobníkový automat:

- Přečte na vstupu žádný nebo jeden symbol. (ϵ přechody pro prázdný vstup.)
- Přejde do nového stavu.
- Nahradí symbol na vrchu zásobníku libovolným řetězcem (ϵ odpovídá samotnému pop, jinak následuje push jednoho nebo více symbolů).



Example 9.1

Zásobníkový automat pro jazyk: $L_{wwr} = \{ww^R | w \in (\mathbf{0} + \mathbf{1})^*\}$.

PDA přijímající L_{wwr}:

- Start q_0 reprezentuje odhad, že ještě nejsme uprostřed.
- V každém kroku nedeterministicky hádáme;
  - Zůstat q_0 (ještě nejsme uprostřed).
  - Přejít ϵ přechodem do q_1 (už jsme viděli střed).
- V q_0, přečte vstupní symbol a dá (push) ho na zásobník
- V q_1, srovná vstupní symbol s vrcholem zásobníku
  pokud se shodují, přečte vstupní symbol a umaže (pop) vrchol zásobníku
- Když vyprázdníme zásobník, přijmeme vstup, který jsme doteď přečetli.

PDA pro L_{wwr}

Example 9.2 (PDA pro L_{wwr})

PDA pro L_{wwr} můžeme popsat $P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$ kde δ je definovaná:

$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$	Ulož vstup na zásobník, startovní symbol tam nech
$\delta(q_0, 0, 0) = \{(q_0, 00)\}$ $\delta(q_0, 0, 1) = \{(q_0, 01)\}$ $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ $\delta(q_0, 1, 1) = \{(q_0, 11)\}$	Zůstaň v q_0, přečti vstup a dej ho na zásobník
$\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0)\}$ $\delta(q_0, \epsilon, 0) = \{(q_1, 0)\}$ $\delta(q_0, \epsilon, 1) = \{(q_1, 1)\}$	ϵ přechod q_1 bez změny zásobníku (a vstupu)
$\delta(q_1, 0, 0) = \{(q_1, \epsilon)\}$ $\delta(q_1, 1, 1) = \{(q_1, \epsilon)\}$	stav q_1 srovná vstupní symbol a vrchol zásobníku
$\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$	našli jsme ww^R a jdeme do přijímajícího stavu

Grafická notace PDA's

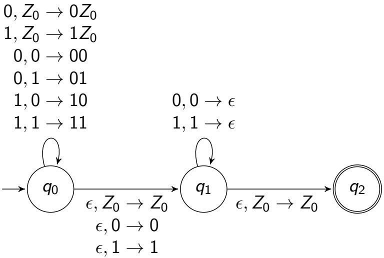
Definition 9.2 (Přechodový diagram pro zásobníkový automat)

Přechodový diagram pro zásobníkový automat obsahuje:

- Uzly, které odpovídají stavům PDA.
- Šipka 'odnikud' ukazuje počáteční stav, dvojité kruhy označují přijímající stavy.
- hrana odpovídá přechodu PDA. Hrana označená $a, X \rightarrow \alpha$ ze stavu p do q znamená $\delta(p, a, X) \ni (q, \alpha)$
- Konvence je, že počáteční symbol zásobníku značíme Z_0.

Anotace hrany:

vstupní_znak, zásobníkový_znak → push_řetězec



Notace zásobníkových automatů

Example 9.3 (Notace)

$a, b, c, *, +, 1, (,)$	symboly vstupní abecedy
p, q, r	stavy
u, v, w, x, y, z	řetězce vstupní abecedy
X, Y, E, I, S	zásobníkové symboly
α, β, γ	řetězce zásobníkových symbolů

- Narozdíl od gramatik může vstupní a zásobníková abeceda obsahovat stejné symboly.
- Vyhýbáme se stejným názvům stavů jako jsou písmena kterékoli z abeced.

Definition 9.3 (Konfigurace zásobníkového automatu)

Konfiguraci zásobníkového automatu reprezentujeme trojicí (q, w, γ), kde

q je stav

w je zbývající vstup a

γ je obsah zásobníku (vrch zásobníku je vlevo).

konfiguraci značíme zkratkou **(ID)** z anglického **instantaneous description (ID)**.

Definition 9.4 ($\vdash, \vdash^*$ posloupnosti konfigurací)

Mějme PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Mějme stavy $p, q \in Q$, $a \in (\Sigma \cup \{\epsilon\}), X \in \Gamma, \alpha \in \Gamma^*$ a instrukci $\delta(p, a, X) \ni (q, \alpha)$. Pak říkáme, že

- konfigurace $(p, aw, X\beta)$ **bezprostředně vede** na konfiguraci $(q, w, \alpha\beta)$,
  - Značíme $(p, aw, X\beta) \vdash (q, w, \alpha\beta)$.
- **konfigurace I vede na konfiguraci J** $I \vdash_P^* J$ a $I \vdash^* J$ používáme na označení nuly a více kroků zásobníkového automatu, t.j.
  - $I \vdash^* I$ pro každou konfiguraci I
  - $I \vdash^* J$ pokud existuje konfigurace K tak že $I \vdash K$ a $K \vdash^* J$.

konfigurace zásobníkového automatu na vstup 1111

$$
\begin{tikzcd}[column sep=small]
 & & & & (q_0, 1111, Z_0) \arrow[dl] \arrow[dr] & \\
 & & & (q_0, 111, 1Z_0) \arrow[dl] \arrow[dr] & & (q_1, 1111, Z_0) \arrow[d] \\
 & & (q_0, 11, 11Z_0) \arrow[dl] \arrow[dr] & & (q_1, 111, 1Z_0) \arrow[d] & (q_2, 1111, Z_0) \arrow[d] \\
 & (q_0, 1, 111Z_0) \arrow[dl] \arrow[dr] & & (q_1, 11, 11Z_0) \arrow[d] & (q_1, 11, Z_0) \arrow[d] & \text{nedočtené slovo} \\
(q_0, \epsilon, 1111Z_0) \arrow[d] & & (q_1, 1, 111Z_0) \arrow[d] & (q_1, 1, 1Z_0) \arrow[d] & (q_2, 11, Z_0) \arrow[d] & \\
(q_1, \epsilon, 1111Z_0) \arrow[d] & & (q_1, \epsilon, 11Z_0) \arrow[d] & (q_1, \epsilon, Z_0) \arrow[d] & \text{nedočtené slovo} & \\
\text{nepřijímající stav} & & \text{nepřijímající stav} & \boxed{(q_2, \epsilon, Z_0)} & &
\end{tikzcd}
$$

Jazyky zásobníkových automatů

Definition 9.5 (Jazyk přijímaný koncovým stavem, prázdným zásobníkem)

Mějme zásobníkový automat $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Pak $L(P)$, **jazyk přijímaný (akceptovaný) koncovým stavem** je

$L(P) = \{w | (q_0, w, Z_0) \vdash_P^* (q, \epsilon, \alpha)$ pro nějaké $q \in F$ a libovolný řetězec $\alpha \in \Gamma^*; w \in \Sigma^*\}$.

jazyk přijímaný prázdným zásobníkem $N(P)$ definujeme

$N(P) = \{w | (q_0, w, Z_0) \vdash_P^* (q, \epsilon, \epsilon)$ pro libovolné $q \in Q; w \in \Sigma^*\}$.

- Protože je množina přijímajících stavů F nerelevantní, může se vynechat a PDA je šestice $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$.

Example 9.4

Zásobníkový automat z předchozího příkladu přijímá L_{wwr} koncovným stavem.

Example 9.5

$P' \equiv P$ z předchozího příkladu, jen změníme instrukci, aby umazala poslední symbol
$\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$ nahradíme
$\delta(q_1, \epsilon, Z_0) = \{(q_2, \epsilon)\}$
Nyní $L(P') = N(P') = L_{wwr}$.

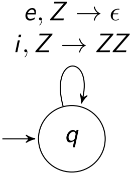
Příklad If-Else

Example 9.6 (If-else příjímané prázdným zásobníkem)

Následující zásobníkový automat zastaví při první chybě na if (i) a else (e), máme–li více else než if.

$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z)$ kde

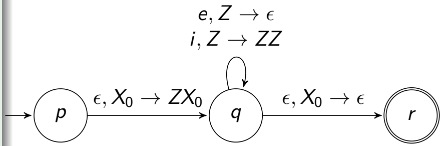
- $\delta_N(q, i, Z) = \{(q, ZZ)\}$ push
- $\delta_N(q, e, Z) = \{(q, \epsilon)\}$ pop



Example 9.7 (Přijímání koncovým stavem)

$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\})$ kde

- $\delta_F(p, \epsilon, X_0) = \{(q, ZX_0)\}$ start
- $\delta_F(q, i, Z) = \{(q, ZZ)\}$ push
- $\delta_F(q, e, Z) = \{(q, \epsilon)\}$ pop
- $\delta_F(q, \epsilon, X_0) = \{(r, \epsilon)\}$ přijmi



Nečtený vstup a dno zásobníku P neovlivní výpočet

Lemma 9.1

Mějme PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ *a* $(p, u, \alpha) \vdash_P^* (q, v, \beta)$. *Potom pro libovolné slovo* $w \in \Sigma^*$ *and* $\gamma \in \Gamma^*$ *platí:* $(p, uw, \alpha\gamma) \vdash_P^* (q, vw, \beta\gamma)$. *Specielně pro* $\gamma = \epsilon$ *a/nebo* $w = \epsilon$.

Proof.

Indukcí podle počtu konfigurací mezi $(p, uw, \alpha\gamma)$ a $(q, vw, \beta\gamma)$. Každý krok $(p, u, \alpha) \vdash_P^* (q, v, \beta)$ je určen bez w a/nebo γ. Proto je možný i se symboly na konci vstupu / dně zásobníku. □

Lemma 9.2

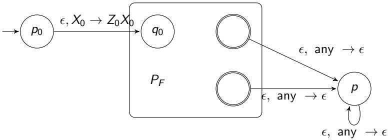
Pro PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ *a* $(p, uw, \alpha) \vdash_P^* (q, vw, \beta)$ *platí* $(p, u, \alpha) \vdash_P^* (q, v, \beta)$.

Remark Pro zásobník ale obdoba neplatí. PDA může zásobníkové symboly γ použít a zase je tam naskládat (push). $L = \{0^i 1^i 0^j 1^j\}$, konfigurace $(p, 0^{i-j} 1^i 0^j 1^j, 0^j Z_0) \vdash^* (q, 1^j, 0^j Z_0)$, mezitím vyčištíme zásobník k Z_0.

Od přijímajícího stavu k prázdnému zásobníku

Lemma 9.3 (Od přijímajícího stavu k prázdnému zásobníku)

Mějme $L = L(P_F)$ pro nějaký PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Pak existuje PDA P_N takový, že $L = N(P_N)$.



Proof:

Nechť $P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$, kde

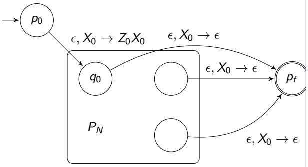
- $\delta_N(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$ start
- $\forall (q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma)$ $\delta_N(q, a, Y) = \delta_F(q, a, Y)$ simulujeme
- $\forall (q \in F, Y \in \Gamma \cup \{X_0\})$, $\delta_N(q, \epsilon, Y) \ni (p, \epsilon)$ přijmout pokud P_F přijímá,
- $\forall (Y \in \Gamma \cup \{X_0\})$, $\delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$ vyprázdnit zásobník.

Pak $w \in N(P_N)$ iff $w \in L(P_F)$. $\square$

Od prázdného zásobníku ke koncovému stavu

Lemma 9.4 (Od prázdného zásobníku ke koncovému stavu)

Pokud $L = N(P_N)$ pro nějaký PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, pak existuje PDA P_F takový, že $L = L(P_F)$.



Proof:

$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

kde δ_F je

- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$ (start).
- $\forall (q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma)$, $\delta_F(q, a, Y) = \delta_N(q, a, Y)$.
- Navíc, $\delta_F(q, \epsilon, X_0) \ni (p_f, \epsilon)$ pro každý $q \in Q$.

Chceme ukázat $w \in N(P_N)$ iff $w \in L(P_F)$.

- (If) P_F přijímá následovně: $(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \vdash^*_{P_F = N_F} (q, \epsilon, X_0) \vdash_{P_F} (p_f, \epsilon, \epsilon)$.
- (Only if) Do p_f nelze dojít jinak než předchozím bodem.

□

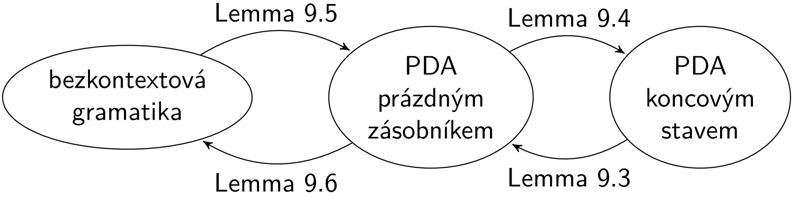
Ekvivalence jazyků rozpoznávaných zásobníkovými automaty a bezkontextových jazyků

Theorem 9.1 (!L(CFG), L(PDA), N(PDA))

Následující tvrzení jsou ekvivalentní:

- *Jazyk L je bezkontextový, tj. generovaný bezkontextovou gramatikou.*
- *Jazyk L je přijímaný nějakým zásobníkovým automatem koncovým stavem.*
- *Jazyk L je přijímaný nějakým zásobníkovým automatem prázdným zásobníkem.*

Důkaz bude veden směry dle následujícího obrázku.



Od bezkontextové gramatiky k zásobníkovému automatu

Algorithm: Konstrukce PDA z CFG G

Mějme CFG gramatiku $G = (V, T, P, S)$.
Konstruujeme PDA $P = (\{q\}, T, V \cup T, \delta, q, S)$.

(1) Pro neterminály $A \in V$, $\delta(q, \epsilon, A) = \{(q, \beta) | A \to \beta$ je pravidlo $G\}$.

(2) pro každý terminál $a \in T$, $\delta(q, a, a) = \{(q, \epsilon)\}$.

Example 9.8

Konvertujme gramatiku: $G = (\{E, I\}, \{a, b, 0, 1, (,), +, *\}, \{I \to a|b|Ia|Ib|I0|I1, E \to I|E * E|E + E|(E)\}, E)$.

Množina vstupních symbolů PDA je $\Sigma = \{a, b, 0, 1, (,), +, *\}$, $\Gamma = \Sigma \cup \{I, E\}$, přechodová funkce δ:

- $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$.
- $\delta(q, \epsilon, E) = \{(q, I), (q, E * E), (q, E + E), (q, (E))\}$.
- $\forall s \in \Sigma$ je $\delta(q, s, s) = \{(q, \epsilon)\}$, např. $\delta(q, +, +) = \{(q, \epsilon)\}$.

CFG a PDA

Lemma 9.5 (Přijímání prázdným zásobníkem ze CFG)

Pro PDA P konstruovaný z CFG G algoritmem výše je $N(P) = L(G)$.

(1) Pro neterminály $A \in V$, $\delta(q, \epsilon, A) = \{(q, \beta) | A \to \beta$ je pravidlo $G\}$.

(2) pro každý terminál $a \in T$, $\delta(q, a, a) = \{(q, \epsilon)\}$.

- Levá derivace: $E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * I \Rightarrow a * b$
- Posloupnost konfigurací:
  $(q, a * b, E) \vdash (q, a * b, E * E) \vdash (q, a * b, I * E) \vdash (q, a * b, a * E)$
  $\vdash (q, *b, *E) \vdash (q, b, E) \vdash (q, b, I) \vdash (q, b, b) \vdash (q, \epsilon, \epsilon)$

Pozorování:

- Kroky derivace simuluje PDA ϵ přepisy zásobníku
- odmazávaný vstup u PDA v derivaci zůstává až do konce
- až PDA vymaže terminály, pokračuje v přepisech.

CFG a PDA

$w \in N(P) \Leftarrow w \in L(G)$.

Nechť $w \in L(G)$, w má levou derivaci $S = \gamma_1 \underset{lm}{\Rightarrow} \gamma_2 \underset{lm}{\Rightarrow} \dots \underset{lm}{\Rightarrow} \gamma_n = w$.
Indukcí podle i dokážeme $(q, w, S) \underset{P}{\vdash^*} (q, v_i, \alpha_i)$, kde $\gamma_i = u_i\alpha_i$ je levá sentenciální forma a $u_i v_i = w$.

- Pokud γ_i obsahuje pouze terminály, $\gamma_i = w = u_i, v_i = \epsilon = \alpha_i$, hotovo.
- Každá nekoncová sentenciální forma γ_i může být zapsaná $u_i A\alpha_i$,
  A nejlevější neterminál, u_i řetězec terminálů
- indukční předpoklad nás dovedl do konfigurace $(q, v_i, A\alpha_i)$, $w = u_i v_i$
- Pro $\gamma_i \underset{lm}{\Rightarrow} \gamma_{i+1}$ bylo použilo pravilo $(A \to \beta) \in P$
- PDA nahradí A na zásobníku β, přejde na konfiguraci $(q, v_i, \beta\alpha_i)$.
- odstraňme všechny terminály $v \in \Sigma^*$ zleva $\beta\alpha$ porovnáváním se vstupem
  - $v_i = v v_{i+1}$ a zároveň $\beta\alpha = v\alpha_{i+1}$
- přešli jsme do nové konfigurace $(q, v_{i+1}, \alpha_{i+1})$ a iterujeme.

$\square$

$w \in N(P) \Rightarrow w \in L(G)$.

Dokazujeme: Pokud $(q, u, A) \vdash_P^* (q, \epsilon, \epsilon)$, tak $A \overset{*}{\underset{G}{\Rightarrow}} u$.

Indukcí podle počtu kroků P.

- $n = 1$ kroků:
  - $a \in \Sigma$, přechod $\delta(q, a, a) \ni (q, \epsilon)$, v derivaci žádný krok,
  - $A \in \Gamma$, přechod $\delta(q, \epsilon, A) \ni (q, \epsilon)$ pro pravidlo gramatiky $(A \rightarrow \epsilon) \in G$.
- $n > 1$ kroků;
  - První krok typu (2) – terminály, nerozšiřujeme derivaci.
  - První krok typu (1), A nahrazeno $Y_1 Y_2 \dots Y_k$ z pravidla $A \rightarrow Y_1 Y_2 \dots Y_k$.
    Rozdělíme $u = u_1 u_2 \dots u_k$:
    - čtením symbolu Y_i skončilo slovo u_{i-1} a začíná u_i.

    Použijeme indukční hypotézu na každé $i = 1, \dots, k$:
    $(q, u_i u_{i+1} \dots u_k, Y_i) \vdash^* (q, u_{i+1} \dots u_k, \epsilon)$ a dostaneme $Y_i \Rightarrow^* u_i$.

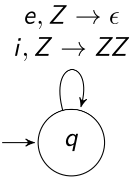
Příklad: Od zásobníkového automatu ke gramatice

Example 9.9

Převeďme PDA $P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z)$ na obrázku na gramatiku.

- Neterminály gramatiky budou $V = \{S, [qZq]\}$ nový start a jediná trojice P_N.
- Pravidla:
  - $S \rightarrow [qZq]$.
  - $[qZq] \rightarrow e$
  - $[qZq] \rightarrow i[qZq][qZq]$.

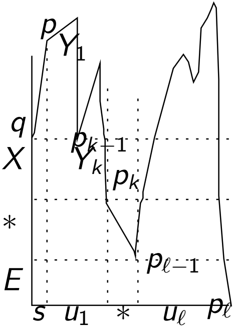
  Můžeme nahradit trojici $[qZq]$ symbolem A a dostaneme:
  $S \rightarrow A$
  $A \rightarrow iAA|e$.
  Protože A a S odvozují přesně stejné řetězce, můžeme je ztotožnit: $G = (\{S\}, \{i, e\}, \{S \rightarrow iSS|e\}, S)$.



Od zásobníkového automatu ke gramatice CFG

- Zásobní automat bere jeden symbol ze zásobníku. Stav před a po kroku může být různý.
- Neterminály gramatiky budou složené symboly $[qXp]$, PDA vyšel z q, vzal X a přešel do p;

  a zavedeme nový počáteční symbol S.



Lemma 9.6 (Gramatika pro PDA)

Mějme PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$. Pak existuje bezkontextová gramatika G taková, že $L(G) = N(P)$.

Pravidla definujeme:

- $\forall p \in Q$: $S \rightarrow [q_0 Z_0 p]$, tj. uhodni koncový stav a spusť PDA na $(q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)$.
- Pro všechny dvojice $(p, Y_1 Y_2 \ldots Y_k) \in \delta(q, s, X), s \in \Sigma \cup \{\epsilon\}, \forall p_1, \ldots, p_{k-1} \in Q$ vytvoř pravidlo

$$[qXp_k] \rightarrow s[pY_1p_1][p_1Y_2p_2] \ldots [p_{k-1}Y_kp_k]$$

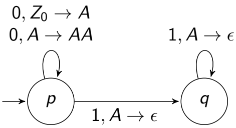
- spec. pro $(p, \epsilon) \in \delta(q, a, A)$ vytvoř $[qAp] \rightarrow a$.

Proof.

Pro $w \in \Sigma^*$ dokazujeme
$[qXp] \Rightarrow^* w$ právě když $(q, w, X) \vdash^* (p, \epsilon, \epsilon)$
indukcí v obou směrech (počet kroků PDA, počet kroků derivace.) □

Example 9.10 ($\{0^n 1^n; n > 0\}$)

δ	Pravidla	
	$S \rightarrow [pZ_0p] \| [pZ_0q]$	(1)
$\delta(p, 0, Z_0) \ni (p, A)$	$[pZ_0p] \rightarrow 0[pAp]$	(2)
	$[pZ_0q] \rightarrow 0[pAq]$	(3)
$\delta(p, 0, A) \ni (p, AA)$	$[pAp] \rightarrow 0[pAp][pAp]$	(4)
	$[pAp] \rightarrow 0[pAq][qAp]$	(5)
	$[pAq] \rightarrow 0[pAp][pAq]$	(6)
	$[pAq] \rightarrow 0[pAq][qAq]$	(7)
$\delta(p, 1, A) \ni (q, \epsilon)$	$[pAq] \rightarrow 1$	(8)
$\delta(q, 1, A) \ni (q, \epsilon)$	$[qAq] \rightarrow 1$	(9)



Derivace 0011
$S \Rightarrow^{(1)} [pZ_0q] \Rightarrow^{(3)} 0[pAq] \Rightarrow^{(7)} 00[pAq][qAq] \Rightarrow^{(8)} 001[qAq] \Rightarrow^{(9)} 0011$

Shrnutí

- Zásobníkový automat PDA je ϵ–NFA automat rozšířený o zásobník, potenciálně nekonečnou paměť
  - a zásobníkovou abecedu, počáteční zásobníkový symbol, přechodová funkce čte a píše na zásobník, píše i řetězec
- Přijímání koncovým stavem a prázdným zásobníkem, pro nedeterministiké PDA přijímají stejnou třídu jazyků
- a to bezkontextové jazyky, generované bezkontextovými gramatikami.

Deterministický zásobníkový automat (DPDA)

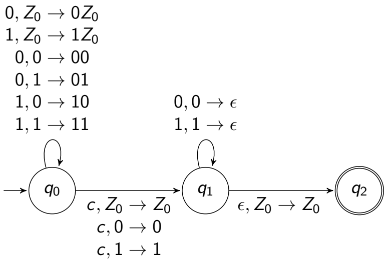
Definition 9.6 (Deterministický zásobníkový automat (DPDA))

Zásobníkový automat $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je **deterministický** PDA právě když platí zároveň:

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

Example 9.11 (Det. PDA přijímající L_{wcwr})

- Jazyk L_{wwr} palindromů je bezkontextový, ale nemá přijímající deterministický zásobníkový automat.
- Druhá podmínka zaručuje, že nebude volba mezi ϵ přechodem a čtením vstupního symbolu.
- Vložením středové značky c do $L_{wcwr} = \{wcw^R | w \in (\mathbf{0} + \mathbf{1})^*\}$ dostaneme jazyk rozpoznatelný DPDA.



Regulární jazyky, DPDA's

$$RL \subsetneq L_{DPDA} \subsetneq L_{PDA} = CFL = N_{PDA} \supsetneq N_{DPDA}.$$

Theorem 9.2

Nechť L je regulární jazyk, pak $L = L(P)$ pro nějaký DPDA P.

Proof.

DPDA může simulovat deterministický konečný automat a ignorovat zásobník. (nechat tam Z_0). $\square$

Lemma

Jazyk L_{wcwr} je přijímaný DPDA ale není regulární.

Důkaz neregularity z pumping lemmatu na slovo $0^n c 0^n$.

Example 9.12

Jazyk $L_{abb} = \{a^i b^i | i \in \mathbb{N}\} \cup \{a^i b^{2i} | i \in \mathbb{N}\}$ je bezkontextový, ale není přijímaný žádným deterministickým zásobníkovým automatem.

Proof.

- SPOREM: Předokládejme, že existuje deterministický PDA M přijímající jazyk L_{abb}.
- Vytvořme dvě kopie, M_1 a M_2, odpovídající si uzly budeme nazývat sourozenci.
- Zkonstuujeme nový automat:
  - Počátečním stavem bude počáteční stav M_1
  - koncovými stavy budou koncové stavy M_2
  - přechody z koncových stavů M_1 $\delta(p, b, X) = (q, X)$
    - přesměrujeme do sourozenců q v M_2 a přejmenujeme b na c
  - v automatu M_2 hrany označené b přeznačíme na c.
- Výsledný automat přijímá $\{a^i b^i c^i | i \in \mathbb{N}\}$ protože
  - M je deterministický, nemá jinou cestu, tj. i ve slově $a^i b^{2i}$ musel jít začátek stejně a pak číst b^i, nyní c^i,
- o $\{a^i b^i c^i | i \in \mathbb{N}\}$ víme, že není bezkontextový, tj. deterministický M nemůže existovat.

Bezprefixové jazyky

Definition 9.7 (bezprefixové jazyky)

Říkáme, že jazyk $L \subset \Sigma^*$ je **bezprefixový** pokud neexistují slova $u, v \in L$ a $z \in \Sigma^+$ tak, že $u = vz$. Tj. pro žádná slova jazyka u, v není vlastní v **prefix** u.

Example 9.13

- Jazyk L_{wcwr} je bezprefixový.
- Jazyk $L = \{0\}^*$ není bezprefixový.

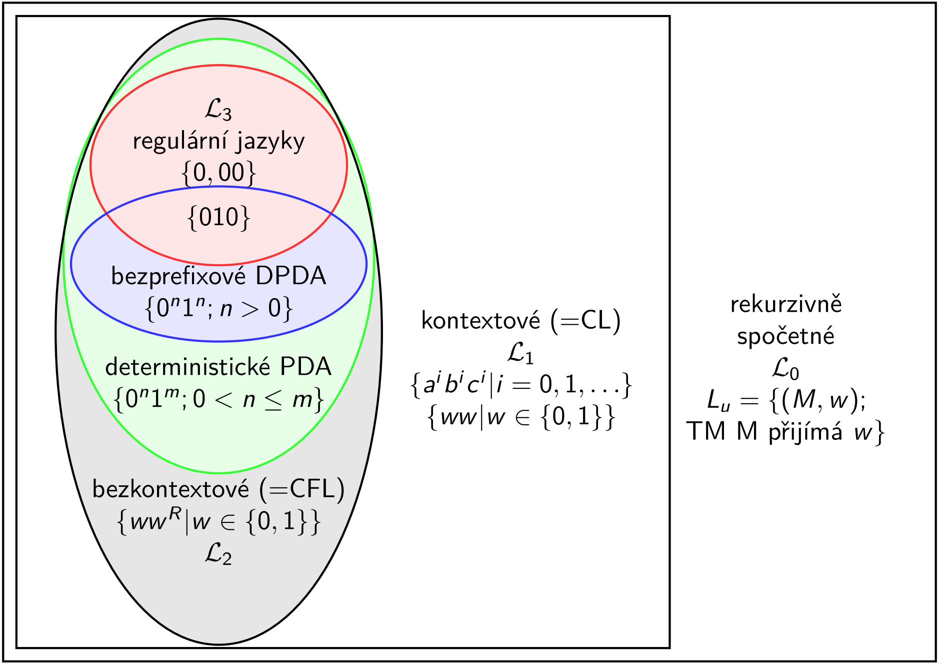
Theorem 9.3 ($L \in N(P_{DPDA})$ právě když L bezprefixový a $L \in L(P'_{DPDA})$)

Jazyk L je $N(P)$ pro nějaký DPDA P právě když L je bezprefixový a L je $L(P')$ pro nějaký DPDA P'.

Proof.

$\Rightarrow$ Prefix přijmeme prázdným zásobníkem, pro prázdný zásobník neexistuje instrukce, tj. žádné prodloužení není v $N(P)$.

$\Leftarrow$ Převod $P^|$ na P nepřidá nedeterminismus (první koncový -> smaž, přijmi).



$L_d = \{w;$ TM s kódem w nepřijímá vstup $w\}$

Uzávěrové vlastnosti

Theorem 10.1 (CFL uzavřené na sjednocení, konkatenaci, iteraci, reverzi)

CFL jsou uzavřené na sjednocení, konkatenaci, iterace $(*)$, *positivní iterace* $(+)$, *zrcadlový obraz* w^R.

Proof:

- Sjednocení:
  - pokud $V_1 \cap V_2 \neq \emptyset$ přejmenujeme neterminály,
  - přidáme nový symbol S_{new} a pravidlo $S_{new} \rightarrow S_1 | S_2$
- zřetězení (=konkatenace) $L_1.L_2$

  $S_{new} \rightarrow S_1 S_2$ (pro $V_1 \cap V_2 = \emptyset$, jinak přejmenujeme)
- iterace $L^* = \bigcup_{i \geq 0} L^i$

  $S_{new} \rightarrow S S_{new} | \epsilon$
- pozitivní iterace $L^+ = \bigcup_{i \geq 1} L^i$

  $S_{new} \rightarrow S S_{new} | S$
- zrcadlový obraz $L^R = \{w^R | w \in L\}$

  $X \rightarrow \omega^R$ obrátíme pravou stranu pravidel.

Průnik bezkontextových jazyků

Example 10.1 (!CFL nejsou uzavřené na průnik)

- Jazyk $L = \{0^n1^n2^n | n \geq 1\} = \{0^n1^n2^i | n, i \geq 1\} \cap \{0^i1^n2^n | n, i \geq 1\}$

není CFL, i když oba členy průniku jsou bezkontextové, dokonce deterministické bezkontextové.

$\{0^n1^n2^i | n, i \geq 1\}$ $\{S \to AC, A \to 0A1|01, C \to 2C|2\}$
$\{0^i1^n2^n | n, i \geq 1\}$ $\{S \to AB, A \to 0A|0, B \to 1B2|12\}$

- průnik není CFL z pumping lemmatu.

paralelní běh dvou zásobníkových automatů

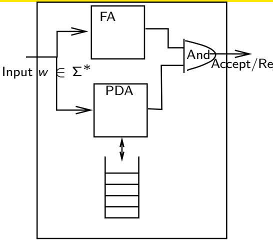
- řídící jednotky umíme spojit (viz konečné automaty)
- čtení umíme spojit (jeden automat může čekat)
- bohužel dva zásobníky nelze obecně spojit do jednoho

dva neomezené zásobníky =Turingův stroj
=rekurzivně spočetné jazyky $\mathcal{L}_0$

Průnik bezkontextového a regulárního jazyka

Theorem 10.2 (CFL i DCFL jsou uzavřené na průnik s regulárním jazykem)

- *Mějme L bezkontextový jazyk a R regulární jazyk. Pak $L \cap R$ je bezkontextový jazyk.*
- *Mějme L deterministický CFL a R regulární jazyk. Pak $L \cap R$ je deterministický CFL.*



Proof:

- zásobníkový a konečný automat můžeme spojit
  - FA $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  - PDA přijímání stavem $M_1 = (Q_2, \Sigma, \Gamma, \delta_2, q_2, Z_0, F_2)$
- nový automat $M = (Q_1 \times Q_2, \Sigma, \Gamma, \delta, (q_1, q_2), Z_0, F_1 \times F_2)$
  - $((r, s), \alpha) \in \delta((p, q), a, Z)$ právě když

$a \neq \epsilon$:	$r = \delta_1(p, a) \& (s, \alpha) \in \delta_2(q, a, Z)$	... automaty čtou vstup
$a = \epsilon$:	$(s, \alpha) \in \delta_2(q, \epsilon, Z)$	PDA mění zásobník
	$r = p$	FA stojí

- zřejmě $L(M) = L(A_1) \cap L(M_2)$

Substituce a homomorfismus

- Opakování definice:

Definition ((5.1,5.2)substituce, homomorfismus, inverzní homomorfismus)

Mějme jazyk L nad abecedou Σ.
Substituce σ; $\forall a \in \Sigma : \sigma(a) = L_a$ jazyk abecedy Σ_a, tj. $\sigma(a) \subseteq \Sigma_a^*$
převádí slova na jazyky:

- $\sigma(\epsilon) = \{\epsilon\}$,
- $\sigma(a_1 \dots a_n) = \sigma(a_1). \dots .\sigma(a_n)$ (konkatenace), tj. $\sigma : \Sigma^* \to P((\bigcup_{a \in \Sigma} \Sigma_a)^*)$
- $\sigma(L) = \bigcup_{w \in L} \sigma(w)$.

homomorfismus h, $\forall a \in \Sigma : h(a) \in \Sigma_a^*$
převádí slova na slova

- $h(\epsilon) = \epsilon$,
- $h(a_1 \dots a_n) = h(a_1). \dots .h(a_n)$ (konkatenace) tj. $h : \Sigma^* \to (\bigcup_{a \in \Sigma} \Sigma_a)^*$
- $h(L) = \{h(w) \mid w \in L\}$.

Inverzní homomorfismus převádí slova zpět

- $h^{-1}(L) = \{w \mid h(w) \in L\}$.

Příklad: Substituce

Example 10.2

Mějme gramatiku $G = (\{E\}, \{a, +, (,)\}, \{E \to E + E|(E)|a\}, E)$. Mějme substituci:

- $\sigma(a) = L(G_a)$, $G_a = (\{I\}, \{a, b, 0, 1\}, \{I \to I0|I1|Ia|Ib|a|b\}, I)$,
- $\sigma(+) = \{-, *, :, div, mod\}$,
- $\sigma(() = \{(\}$,
- $\sigma()) = \{)\}$.

- $(a + a) + a \in L(G)$
- v $\sigma(+)$ chybí $+$ pro ukázku, že $(a + a) + a \notin \sigma(L(G))$.
- $(a001 - bba) * b1 \in \sigma((a + a) + a) \subset \sigma(L(G))$

Co se stane, když změníme definici:

- $\sigma(() = \{(, [\}$,
- $\sigma()) = \{),]\}$?

Příklad: Homomorfismus

Example 10.3

Mějme gramatiku $G = (\{E\}, \{a, +, (,)\},$ $\{E \to E + E|(E)|a\}, E)$. Mějme homomorfimus:

- $h(a) = \epsilon$
- $h(+) = \epsilon,$
- $h(() = left,$
- $h()) = right.$

- $h((a + a) + a) = leftright,$
- $h^{-1}(leftright) \ni (a + +)a.$

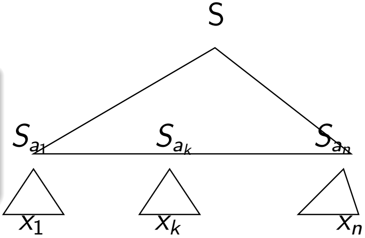
Example 10.4

Mějme gramatiku $G = (\{E\}, \{a, +, (,)\},$ $\{E \to a + E|(E)|a\}, E)$. Mějme homomorfimus:

- $h_2(a) = a$
- $h_2(+) = +,$
- $h(() = \epsilon,$
- $h()) = \epsilon.$

1. Je jazyk $L(G)$ regulární?
2. Je jazyk $h(L(G))$ regulární?
3. Je jazyk $h^{-1}(h(L(G)))$ regulární?
4. Je $h^{-1}(h(L(G))) = L(G)$?

Uzávěrové vlastnosti bezkontextových jazyků



Theorem 10.3 (CFL jsou uzavřené na substituci)

Mějme CFL jazyk L nad Σ a substituci σ na Σ takovou, že $\sigma(a)$ je CFL pro každé $a \in \Sigma$. Pak je i $\sigma(L)$ bezkontextový (CFL).

Proof:

- Idea: listy v derivačním stromu generují další stromy.
- Přejmenujeme neterminály na jednoznačné všude v $G = (V, \Sigma, P, S)$, $G_a = (V_a, T_a, P_a, S_a), a \in \Sigma$.
- Vytvoříme novou gramatiku $G = (V', T', P', S)$ pro $\sigma(L)$:
  - $V' = V \cup \bigcup_{a \in \Sigma} V_a$
  - $T' = \bigcup_{a \in \Sigma} T_a$
  - $P' = \bigcup_{a \in \Sigma} P_a \cup \{p \in P$ kde všechna $a \in \Sigma$ nahradíme $S_a\}$.

G' generuje jazyk $\sigma(L)$. $\square$

Substituce bezkontextových jazyků

Example 10.5 (substituce)

$L = \{a^i b^j \mid 0 \leq i \leq j\}$	$S \rightarrow aSb \mid Sb \mid \epsilon$
$\sigma(a) = L_1 = \{c^i d^i \mid i \geq 0\}$	$S_1 \rightarrow cS_1 d \mid \epsilon$
$\sigma(b) = L_2 = \{c^i \mid i \geq 0\}$	$S_2 \rightarrow cS_2 \mid \epsilon$
$\sigma(L)$:	$S \rightarrow S_1 S S_2 \mid S S_2 \mid \epsilon,\ S_1 \rightarrow cS_1 d \mid \epsilon,\ S_2 \rightarrow cS_2 \mid \epsilon$

Theorem 10.4 (homomorfismus)

Bezkontextové jazyky jsou uzavřeny na homomorfismus.

Proof:

- Přímý důsledek předchozí věty.
- Terminál a v derivačním stromě nahradím slovem $h(a)$. $\square$

CFL jsou uzavřené na inverzní homomorfizmus

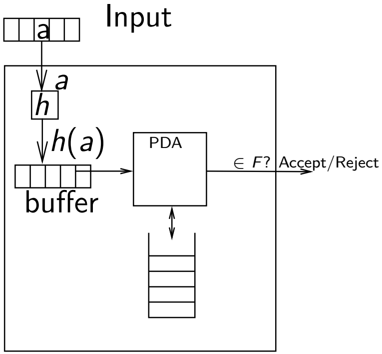
Theorem 10.5 (CFL jsou uzavřené na inverzní homomorfizmus)

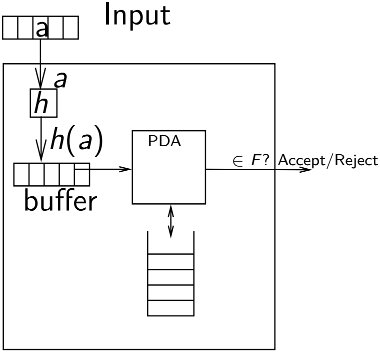
Mějme CFL jazyk L a homomorfizmus h. Pak $h^{-1}(L)$ je bezkontextový jazyk. Je–li L deterministický CFL, je i $h^{-1}(L)$ deterministický CFL.

Idea

- přečteme písmeno a a do vnitřního bufferu dáme $h(a)$
- simulujeme výpočet M, kdy vstup bereme z bufferu
- po vyprázdnění bufferu načteme další písmeno ze vstupu
- slovo je přijato, když je buffer prázdný a M je v koncovém stavu

! buffer je konečný, můžeme ho tedy modelovat ve stavu





Proof:

- pro L máme PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ (koncovým stavem)
- $h : T \to \Sigma^*$
- definujeme PDA $M' = (Q', T, \Gamma, \delta', [q_0, \epsilon], Z_0, F \times \{\epsilon\})$ kde

$$
\begin{array}{lll}
Q' = \{[q,u] \,|\, q \in Q, u \in \Sigma^*, \exists(a \in T)\exists(v \in \Sigma^*) h(a) = vu\} & & u \text{ je buffer} \\
\delta'([q,\epsilon], a, Z) & = \{([q, h(a)], Z)\} & \text{naplňuje buffer} \\
\delta'([q,u], \epsilon, Z) & = \{([p,u], \gamma) | (p,\gamma) \in \delta(q,\epsilon,Z)\} & \\
 & \cup \{([p,v], \gamma) | (p,\gamma) \in \delta(q,x,Z), u = xv\} & \text{čte buffer, } x \in \Sigma
\end{array}
$$

Použití uzavřenosti průniku CFL a RL

Example 10.6

Jazyk $L = \{0^i 1^j 2^k 3^l | i = 0 \vee j = k = l\}$ není bezkontextový.

Proof: Důkaz sporem:

- Nechť L je bezkontextový jazyk
- $L_1 = \{0 1^j 2^k 3^l | j, k, l \geq 0\}$ je regulární jazyk
  - $\{S \rightarrow 0B, B \rightarrow 1B|C, C \rightarrow 2C|D, D \rightarrow 3D|\epsilon\}$
- $L \cap L_1 = \{0 1^i 2^i 3^i | i \geq 0\}$ není bezkontextový $\Rightarrow$ SPOR s uzavřeností na průnik s regulárním jazykem. $\square$

L je kontextový jazyk

$S \rightarrow \epsilon|0|0A|B_1|C_1|D_1$
$B_1 \rightarrow 1|1B_1|C_1,\ C_1 \rightarrow 2|2C_1|D_1,\ D_1 \rightarrow 3|3D_1$
$A \rightarrow 0|0A|P$
$P \rightarrow 1PCD|1CD$
$DC \rightarrow CD$ přepíšeme $\{DC \rightarrow XC, XC \rightarrow XY, XY \rightarrow CY, CY \rightarrow CD\}$
$1C \rightarrow 12,\ 2C \rightarrow 22,\ 2D \rightarrow 23,\ 3D \rightarrow 33.$

Rozdíl a doplněk

Theorem 10.6 (Rozdíl s regulárním jazykem)

Mějme bezkontextový jazyk L a regulární jazyk R. Pak:

- *$L - R$ je CFL.*

Proof.

$L - R = L \cap \overline{R}$, $\overline{R}$ je regulární. □

Theorem 10.7 (CFL nejsou uzavřené na doplněk ani rozdíl)

Třída bezkontextových jazyků není uzavřená na doplněk ani na rozdíl.

CFL nejsou uzavřené na doplněk ani rozdíl.

Mějme bezkontextové jazyky L, L_1, L_2, regulární jazyk R. Pak:

- $\overline{L}$ nemusí být CFL. $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.
- $L_1 - L_2$ nemusí být CFL. $\Sigma^* - L$ není vždy CFL.

□

Uzávěrové vlastnosti deterministických CFL

- Rozumné programovací jazyky jsou deterministické CFL.
- Deterministické bezkontextové jazyky
  - nejsou uzavřené na průnik
  - jsou uzavřené na průnik s regulárním jazykem
  - jsou uzavřené na inverzní homomorfismus.

Lemma

Doplněk deterministického CFL je opět deterministický CFL.

Proof:

- idea: prohodíme koncové a nekoncové stavy
- nedefinované kroky ošetříme 'podložkou' na zásobníku
- cyklus odhalíme pomocí čítače
- až po přečtení slova prochází koncové a nekoncové stavy

  stačí si pamatovat, zda prošel koncovým stavem. □

Ne–uzavřenost deterministických CFL

Example 10.7 (DCFL nejsou uzavřené na sjednocení)

Jazyk $L = \{a^i b^j c^k | i \neq j \vee j \neq k \vee i \neq k\}$ je CFL, ale není DCFL.

Proof.

Vzhledem k uzavřenosti DCFL na doplněk by byl DCFL i $\overline{L} \cap a^* b^* c^* = \{a^i b^j c^k | i = j = k\}$, o kterém víme, že není CFL (pumping lemma) □

Example 10.8 (DCFL nejsou uzavřené na homomorfismus)

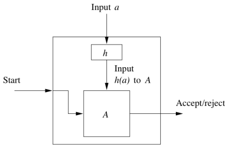
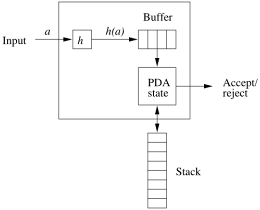
Jazyky $L_1 = \{a^i b^j c^k | i \neq j\}$, $L_2 = \{a^i b^j c^k | j \neq k\}$, $L_3 = \{a^i b^j c^k | i \neq k\}$ jsou deterministické bezkontextové.

- Jazyk $0L_1 \cup 1L_2 \cup 2L_3$ je deterministický bezkontextový
- Jazyk $L_1 \cup L_2 \cup L_3$ není deterministický bezkontextový
  položme $h(0) = \epsilon,\ h(1) = \epsilon, ,\ h(2) = \epsilon$
  $h(x) = x$ pro ostatní symboly
  - $h(0L_1 \cup 1L_2 \cup 2L_3) = L_1 \cup L_2 \cup L_3$,
  - doplněk $\overline{L_1 \cup L_2 \cup L_3} \cap a^* b^* c^* = \{a^i b^j c^k | i = j = k\}$

Uzávěrové vlastnosti v kostce

jazyk	regulární (RL)	bezkontextové	deterministické CFL
sjednocení	ANO	ANO	NE
průnik	ANO	NE	NE
$\cap$ s RL	ANO	ANO	ANO
doplněk	ANO	NE	ANO
homomorfizmus	ANO	ANO	NE
inverzní hom.	ANO	ANO	ANO

Uzávěrové vlastnosti v kostce

jazyk	regulární (RL)	bezkontextové	deterministické CFL
sjednocení	$F = F_1 \times Q_2 \cup Q_1 \times F_2$	$S \to S_1 \vert S_2$	$A \cap B = \overline{\overline{A} \cup \overline{B}}$
průnik	$F = F_1 \times F_2$	$L = \{0^n 1^n 2^n \vert n \geq 1\} = \left\{ \begin{array}{c} \{0^n 1^n 2^i \vert n, i \geq 1\} \\ \cap \{0^i 1^n 2^n \vert n, i \geq 1\} \end{array} \right\}$	
$\cap$ s RL	$F = F_1 \times F_2$	$F = F_1 \times F_2$	$F = F_1 \times F_2$
doplněk	$F = Q_1 - F_1$, δ tot.	$A \cap B = \overline{\overline{A} \cup \overline{B}}$	$F = Q_1 - F_1$, Z_0, cykly, tot.
homom.	Kleene + RegExp + uz.	a nahraď S_a	$h(0) = h(1) = 0$ cca. $\cup$
inverzní hom.			

Dyckovy jazyky

Definition 10.1 (Dyckův jazyk)

Dyckův jazyk D_n je definován nad abecedou $Z_n = \{a_1, a_1^{|}, \ldots, a_n, a_n^{|}\}$ následující gramatikou: $S \to \epsilon | SS | a_1 S a_1^{|} | \ldots | a_n S a_n^{|}$.

Úvodní pozorování:

- jedná se zřejmě o jazyk bezkontextový
- Dyckův jazyk D_n popisuje správně uzávorkované výrazy s n druhy závorek
- tímto jazykem lze popisovat výpočty libovolného zásobníkového automatu
- pomocí Dyckova jazyka lze popsat libovolný bezkontextový jazyk.

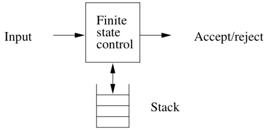
$$L = h(D \cap R)$$

regulární jazyk; popisuje všechny kroky výpočtu

homomorfismus; čistí pomocné symboly

Dyckův jazyk; vybírá pouze korektní výpočty

Jak charakterizovat bezkontextové jazyky?



- Pokud do zásobníku pouze přidáváme

  potom si stačí pamatovat poslední symbol
- stačí konečná paměť $\rightarrow$ konečný automat.
- potřebujeme ze zásobníku také odebírat (čtení symbolu)

  takový proces nelze zaznamenat v konečné struktuře
- přidávání a odebírání není zcela libovolné

  jedná se o zásobník, tj. LIFO (last in, first out) strukturu
- roztáhněme si výpočet se zásobníkem do lineární struktury

  X symbol přidán do zásobníku

  X^{-1} symbol odebrán do zásobníku
- přidávaný a odebíraný symbol tvoří pár $\underbrace{ZZ^{-1}}\underbrace{B\,\underbrace{AA^{-1}}\,\underbrace{CC^{-1}}\,B^{-1}}$

  který se v celé posloupnosti chová jako závorka

Theorem 10.8 (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.

Proof:

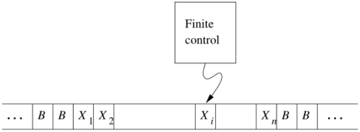
- máme PDA přijímající L prázdným zásobníkem
- převedeme na instrukce tvaru $\delta(q, a, Z) \in (p, w), |w| \leq 2$
  - delší psaní na zásobník rozdělíme zavedením nových stavů
- nechť $R^{|}$ obsahuje všechny výrazy
  - $q^{-1}aa^{-1}Z^{-1}BAp$ pro instrukci $\delta(q, a, Z) \ni (p, AB)$
  - podobně pro instrukce $\delta(q, a, Z) \in (p, A), \delta(q, a, Z) \in (p, \epsilon)$
  - je–li $a = \epsilon$, potom dvojici aa^{-1} nezařazujeme
- definujeme R takto: $Z_0 q_0 (R^{|})^* Q^{-1}$
- Dyckův jazyk je definován nad abecedou $\Sigma \cup \Sigma^{-1} \cup Q \cup Q^{-1} \cup \Gamma \cup \Gamma^{-1}$
- $D \cap Z_0 q_0 (R^{|})^* Q^{-1}$ popisuje korektní výpočty

$$\underbrace{Z_0 \overbrace{q_0 q_0^{-1}} aa^{-1} Z_0^{-1}} \underbrace{B \underbrace{A \overbrace{pp^{-1}} bb^{-1} A^{-1}} \overbrace{qq^{-1}} cc^{-1} B^{-1}} \overbrace{rr^{-1}}$$

- homomorfismus h vydělí přečtené slovo, tj.
  $h(a) = a$ pro vstupní (čtené) symboly
  $h(y) = \epsilon$ pro ostatní.

Turingovy stroje – historie a motivace

- 1931–1936 pokusy o formalizaci pojmu algoritmu
  Gödel, Kleene, Church, Turing
- **Turingův stroj**
  - zachycení práce matematika
    - nekonečná tabule
      lze z ní číst a lze na ni psát
    - mozek (řídící jednotka)
  - Formalizace TM:
    - místo tabule **nekonečná páska**
    - místo křídy **čtecí a zapisovací hlava, kterou lze posunovat v obou směrech**
    - místo mozku konečná řídící jednotka (jako u PDA)
  - další formalizace:
    - λ–kalkul, částečně rekurzivní funkce, RAM



- Snažíme se definovat problémy nerozhodnutelné jakýmkoli počítačem.

Turingův stroj

Definition 11.1

Turingův stroj (TM) je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ se složkami:

- Q konečná množina **stavů**
- Σ konečná neprázdná množina **vstupních symbolů**
- Γ konečná množina všech **symbolů pro pásku**. Vždy $\Gamma \supseteq \Sigma$, $Q \cap \Gamma = \emptyset$.
- δ (částečná) **přechodová funkce**. $(Q - F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$, v $\delta(q, x) = (p, Y, D)$:
  - q $\in (Q - F)$**!** aktuální stav
  - X $\in \Gamma$ aktuální symbol na pásce
  - p nový stav, $p \in Q$.
  - Y $\in \Gamma$ symbol pro zapsání do aktuální buňky, přepíše aktuální obsah.
  - D $\in \{L, R\}$ je **směr** pohybu hlavy (doleva, doprava).
- q_0 $\in Q$ **počáteční stav**.
- B $\in \Gamma - \Sigma$. Blank. Symbol pro prázdné buňky, na začátku všude kromě konečného počtu buněk se vstupem.
- F $\subseteq Q$ množina **koncových** neboli **přijímajících** stavů.

Definition 11.2 (Konfigurace Turingova stroje (Instantaneous Description ID))

Konfigurace Turingova stroje (Instantaneous Description ID) je řetězec $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n$ kde

- q je stav Turingova stroje
- čtecí hlava je vlevo od i–tého symbolu
- $X_1 \dots X_n$ je část pásky mezi nejlevějším a nejpravějším symbolem různým od prázdného (B). S výjimkou v případě, že je hlava na kraji – pak na tom kraji vkládáme jeden B navíc.

Definition 11.3 (Krok Turingova stroje)

Kroky Turingova stroje M značíme $\underset{M}{\vdash}, \underset{M}{\vdash^*}, \vdash^*$ jako u zásobníkových automatů.

Pro $\delta(q, X_i) = (p, Y, L)$

- $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \underset{M}{\vdash} X_1X_2 \dots X_{i-2}pX_{i-1}\mathbf{Y}X_{i+1} \dots X_n$

Pro $\delta(q, X_i) = (p, Y, R)$

- $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \underset{M}{\vdash} X_1X_2 \dots X_{i-1}\mathbf{Y}pX_{i+1} \dots X_n$.

Odvození v konečném počtu kroků $\underset{M}{\vdash^*}$ definuji rekurzivně pro $\mathcal{I}, \mathcal{J}, \mathcal{K}$ konfigurace

Základ: $\mathcal{I} \vdash^* \mathcal{I}$ Rekurze: Pokud $\mathcal{I} \vdash \mathcal{J}$ a $\mathcal{J} \vdash^* \mathcal{K}$, tak i $\mathcal{I} \vdash^* \mathcal{K}$.

A TM for $\{0^n 1^n; n \geq 1\}$

Definition 11.4 (TM přijímá jazyk, rekurzivně spočetný jazyk)

Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ **přijímá jazyk** $L(M) = \{w \in \Sigma^* : q_0 w \vdash_M^* \alpha p \beta, p \in F, \alpha, \beta \in \Gamma^*\}$, tj. množinu slov, po jejichž přečtení se dostane do koncového stavu. Pásku (u nás) nemusí uklízet.

Jazyk nazveme **rekurzivně spočetným**, pokud je přijímán nějakým Turingovým strojem T (tj. $L = L(T)$).

Example 11.1 (TM pro jazyk $\{0^n 1^n; n \geq 1\}$)

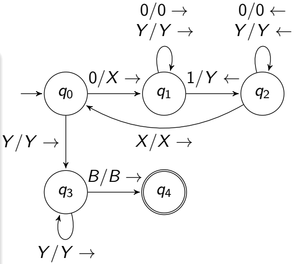
Turingův stroj $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$ s δ v tabulce přijímá jazyk $\{0^n 1^n; n \geq 1\}$.

Stav	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

Přechodový diagram pro Turingův stroj

Definition 11.5 (Přechodový diagram pro TM)

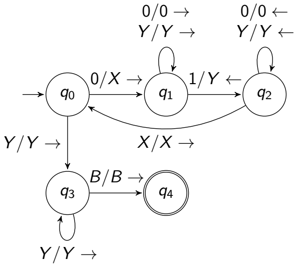
Přechodový diagram pro TM sestává z uzlů odpovídajícím stavům TM. Hrany $q \to p$ jsou označeny seznamem všech dvojic X/YD, kde $\delta(q, X) = (p, Y, D)$, $D \in \{\leftarrow, \rightarrow\}$.
Pokud neuvedeme jinak, B značí blank – prázdný symbol.



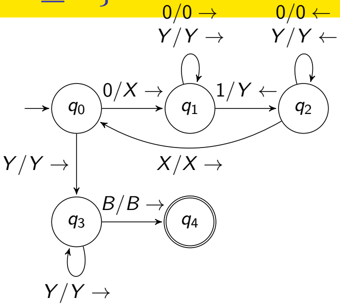
State	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

A TM for $\{0^n 1^n; n \geq 1\}$

- Na pásce vždy výraz typu $\mathbf{X}^*\mathbf{0}^*\mathbf{Y}^*\mathbf{1}^*$
  - postupně přepisujeme 0 na X a odpovídající 1 na Y
  - q_0 přepíše 0 na X a předá řízení q_1
  - q_1 najde první 1, přepíše na Y a předá řízení q_2
  - q_2 se vrátí k X, nechá ho být a předá řízení q_0
  - ...
  - pokud q_0 vidí Y, předá řízení q_3
  - q_3 dojde zkontrolovat, jestli na konci nezbyly 1
  - pokud q_3 našlo B, předá řízení q_4
  - q_4 skončí úspěchem (je přijímající)
  - ...
  - pokud q_3 narazilo na 1, tak skončí neúspěchem
    - nemá instrukci
    - není přijímající.



TM pro $\{0^n1^n; n \geq 1\}$



Slovo 0011

$$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash Xq_2 0Y1 \vdash q_2 X0Y1 \vdash Xq_0 0Y1 \vdash XXq_1 Y1 \vdash$$

$$\vdash XXYq_1 1 \vdash XXq_2 YY \vdash Xq_2 XYY \vdash XXq_0 YY \vdash XXYq_3 Y \vdash XXYYq_3 B \vdash XXYYBq_4$$

Slovo 0010

$$q_0 0010 \vdash Xq_1 010 \vdash X0q_1 10 \vdash Xq_2 0Y0 \vdash q_2 X0Y0 \vdash Xq_0 0Y0 \vdash XXq_1 Y0 \vdash$$

$\vdash XXYq_1 0 \vdash XXY0q_1 B$ a skončí neúspěchem, protože nemá instrukci.

Ještě příklad, rekurzivně spočetné jazyky

Example 11.2

Jazyk $L = \{a^{2n} | n \geq 0\}$
přijímá Turingův stroj $M = (\{q_0, q_1, q_F\}, \{a\}, \{a, B\}, \delta, q_0, B, \{q_F\})$ s δ v tabulce:

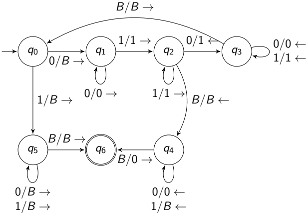
δ	komentář
$\delta(q_0, B) = (q_F, B, R)$	prázdné slovo, konec výpočtu
$\delta(q_0, a) = (q_1, a, R)$	zvětší čítač ($2k + 1$ symbolů)
$\delta(q_1, a) = (q_0, a, R)$	nuluje čítač ($2k$ symbolů).

- Regulární jazyky:
  - simulujeme konečný automat, pohyb hlavy vždy vpravo,
  - vidím-li B, tj. konec vstupu. Pokud je stav DFA přijímající, přejdu do nového přijímajícího stavu q_F.
  - (Z q_F nesmí být instrukce, z přijímajících stavů DFA potřebuji instrukce kopírovat.)
- Bezkontextové jazyky: nejsnáze s pomocnou páskou simulující zásobník, bude za chvíli.

TM s výstupem

Turingův stoj počítající **monus** $m \dot{-} n = max(m-n, 0)$.

- $M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_6\})$
- Počáteční páska $0^m 1 0^n$.
- M zastaví s páskou $0^{m \dot{-} n}$ obklopenou prázdnem B.
- Najdi nejlevější 0, přepiš na B.
- Jdi doprava a najdi 1; pokračuj, najdi 0 a přepiš na 1.
- Vrať se doleva.
- Pokud nenajdeš 0 (ukliď):
  - vpravo: přepiš všechny 1 na B.
  - vlevo: $m < n$: přepiš všechny 1 a 0 na B, nech pásku prázdnou.



Rekurzivní jazyky

Definition (TM zastaví)

TM **zastaví** pokud vstoupí do stavu q, s čteným symbolem X, a není instrukce pro tuto konfiguraci, t.j., $\delta(q, X)$ není definováno.

- Předpokládáme, že v přijímajícím stavu $q \in F$ TM zastaví,
- dokud nezastaví, nevíme, jestli přijme nebo nepřijme slovo.

Definition (Rekurzivní jazyky)

Říkáme, že TM M **rozhoduje jazyk** L, pokud $L = L(M)$ a pro každé $w \in \Sigma^*$ stroj nad w zastaví.
Jazyky rozhodnutelné TM nazýváme **rekurzivní jazyky**.

Paměť v řídící jednotce, Pomocná stopa

Example 11.3 (Příklad paměti ve stavu TM)

Pro jazyk $L(M) = (01^* + 10^*)$ si pamatujeme první znak, stav je dvojice (obecně n–tice). $M = (\{q_0, q_1\} \times \{0, 1, B\}, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], B, \{[q_1, B]\})$

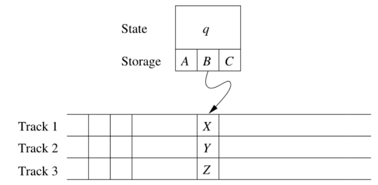
δ	0	1	B
$\rightarrow [q_0, B]$	$([q_1, 0], 0, R)$	$([q_1, 1], 1, R)$	
$[q_1, 0]$		$([q_1, 0], 1, R)$	$([q_1, B], B, R)$
$[q_1, 1]$	$([q_1, 1], 0, R)$		$([q_1, B], B, R)$
$*[q_1, B]$			

Example 11.4 (Příprava pomocné stopy)

Budeme potřebovat dvě stopy na pásce. Přepíšeme vstup na dvojice, nahoře budeme mít volno na poznámky. V definici δ používám zástupný znak pro vstupní písmeno $a \in \{0, 1\}$.

- $\delta([q_0, B], a) = ([q_0, B], [B, a], R)$
- $\delta([q_0, B], B) = ([q_{-1}, B], [B, B], L)$ jsem na konci, otáčím
- $\delta([q_{-1}, B], [B, a]) = ([q_{-1}, B], [B, a], L)$ jdi vlevo

Více stop na pásce



- $L_{wcw} = \{wcw|w \in (\mathbf{0}+\mathbf{1})^+\}$,
- $M = (\{q_1,\ldots,q_9\} \times \{0,1,B\}, \{[B,0],[B,1],[B,c]\}, \{B,*\} \times \{0,1,B,c\}, \delta, [q_1,B],[B,B],\{[q_9,B]\})$
- δ je definováno ($a,b \in \{0,1\}$):
  - $\delta([q_1,B],[B,a]) = ([q_2,a],[*,a],R)$ načti symbol a
  - $\delta([q_2,a],[B,b]) = ([q_2,a],[B,b],R)$ jdi vpravo, hledej střed c,
  - $\delta([q_2,a],[B,c]) = ([q_3,a],[B,c],R)$ pokračuj vpravo ve stavu q_3,
  - $\delta([q_3,a],[*,b]) = ([q_3,a],[*,b],R)$ pokračuj vpravo,
  - $\delta([q_3,a],[B,a]) = ([q_4,B],[*,a],L)$ zkontroluj shodu, vymaž paměť a jdi vlevo,
  - $\delta([q_4,B],[*,a]) = ([q_4,B],[*,a],L)$ jdi vlevo,
  - $\delta([q_4,B],[B,c]) = ([q_5,B],[B,c],L)$ c pokračuj za střed ve stavu q_5,
- rozeskok podle toho, jestli je ještě co kontrolovat
  - $\delta([q_5,B],[B,a]) = ([q_6,B],[B,a],L)$ ještě budeme kontolovat,
  - $\delta([q_6,B],[B,a]) = ([q_6,B],[B,a],L)$ jdi vlevo,
  - $\delta([q_6,B],[*,a]) = ([q_1,B],[*,a],R)$ znovu začni,
  - $\delta([q_5,B],[*,a]) = ([q_7,B],[*,a],R)$ už vše vlevo od c porovnáno, jdi vpravo,
  - $\delta([q_7,B],[B,c]) = ([q_8,B],[B,c],R)$ pokračuj vpravo,
  - $\delta([q_8,B],[*,a]) = ([q_8,B],[*,a],R)$ pokračuj vpravo,
  - $\delta([q_8,B],[B,B]) = ([q_9,B],[B,B],R)$ přijmi.

TM rozšíření: Vícepáskový TM

Definition 11.6 (Vícepáskový Turingův stroj)

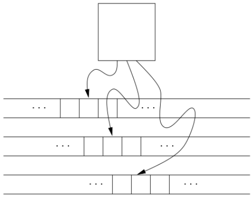
Počáteční pozice

- vstup na první pásce, ostatní zcela prázdné
- první hlava vlevo od vstupu, ostatní libovolně
- hlava v počátečním stavu

Jeden krok vícepáskového TM

- hlava přejde do nového stavu
- na každé pásce napíšeme nový symbol
- každá hlava se nezávisle posune vlevo, zůstane, vpravo.

Vícepáskový TM



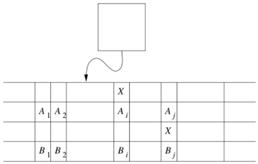
Theorem 11.1 (Vícepáskový TM)

Každý jazyk přijímaný vícepáskovým TM je přijímaný i nějakým (jednopáskovým) Turingovým strojem TM.

Proof: vícepáskový TM

- konstruujeme Turingův stroj M
- pásku si představíme, že má $2k$ stop
  - liché stopy: pozice k–té hlavy
  - sudé stopy: znak na k–té pásce
- pro simulaci jednoho kroku navštívíme všechny hlavy
- ve stavu si pamatujeme
  - počet hlav vlevo
  - $\forall k$ symbol pod k–tou hlavou
- pak už umíme provést jeden krok (znovu běhat) $\square$

Simulace 2–páskového TM na jedné pásce



- Simulaci výpočtu k–páskového stroje o n krocích lze provést v čase $O(n^2)$ (simulace jednoho kroku z prvních n trvá $4n + 2k$, hlavy nejvýš $2n$ daleko, přečíst, zapsat, posunout značky).

Rozšíření: Nedeterministické Turingovy stroje

Definition 11.7 (Nedeterministický TM)

Nedeterministickým Turingovým strojem nazýváme sedmici $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, kde $Q, \Sigma, \Gamma, q_0, B, F$ jsou jako u TM a $\delta : (Q - F) \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$.
Slovo $w \in \Sigma^*$ **je přijímáno nedeterministickým TM** M, pokud existuje nějaký výpočet $q_0 w \vdash^* \alpha p \beta$, $p \in F$.

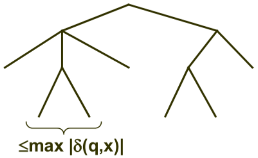
Theorem 11.2 (Nedeterministický TM)

Pro každý M_N nedeterministický TM existuje deterministický TM M_D takový, že $L(M_N) = L(M_D)$.

Velmi stručně (příprava)

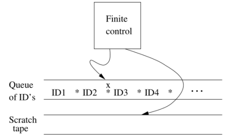
prohledáváme do šířky možné výpočty M_N

- odvozeno v k krocích
- maximálně m^k konfigurací
  - kde $m = max|\delta(q, x)|$ je max. počet voleb M_N



Proof: idea důkazu

- páska nekonečná – nelze použít podmnožinovou konstrukci
- prohledáváme do šířky všechny výpočty M_N
- modelujeme TM se dvěma páskami
  - první páska: posloupnost konfigurací
    - aktuální označena (křížkem na obrázku)
    - vlevo už prozkoumané, můžeme zapomenout
    - vpravo aktuální a pak další čekající
  - druhá páska: pomocný výpočet



- zpracování jedné konfigurace obnáší
  - přečti stav a symbol aktuální konfigurace ID
  - je–li stav přijímající $\in F$, přijmi a skonči
  - napiš konfiguraci ID na pomocnou pásku
  - pro každý možný krok δ (uložený v hlavě M_D)
    - proveď krok a napiš novou ID na konec první pásky
  - vrať se k označené ID, značku vymaž a posuň o 1 doprava
  - opakuj

□

Jednosměrná páska

X_0	X_1	X_2	...
*	X_{-1}	X_{-2}	...

Lemma (Jednosměrná páska)

Pro každý Turingův stroj M_2 existuje Turingův stroj M_1, který přijímá stejný jazyk a

- *M_1 nikdy nejde vlevo od počáteční pozice*
- *M_1 nikdy nepíše blank B.*

Shrnutí

- **Turingův stroj**: nekonečná oboustranná páska, může číst, psát, pohybovat hlavou.
- **Přijímání TM**: TM přijímá pokud vstoupí do koncového stavu.
  TM s výstupem: Hlava na prvním písmenu odpovědi, kromě odpovědi B.
- **Rekurzivně spočetné jazyky (RE)**: jazyky přijímané nějakým Turingovým strojem.
- **Konfigurace TM**: Všechny symboly pásky mezi nejlevějším a nejpravějším ne–B. Stav a pozice hlavy hned vlevo od právě čteného symbolu.
- modelovací triky
  - **Paměť v řídící jednotce**
  - **Více stop**
- Rozšíření TM bez rozšíření třídy přijímaných jazyků:
  - **Vícepáskové TM** Samostatný pohyb hlav na páskách (lze simulovat na přidaných stopách).
  - **Nedeterministický TM**: Má instrukce na výběr, na přijetí stačí jeden přijímající výpočet.
- Budou: **Lineárně omezené automaty LBA**
  - Vstupní slovo mezi levou a pravou značkou, hlava nesmí za tyto značky ani je přepsat.
  - LBA rozpoznávají právě kontextové jazyky.

Kontextové jazyky

Turingovy stroje Vícepáskové, nedeterministické	$\longleftrightarrow$	gramatiky Typu 0
lineárně omezené automaty	$\longleftrightarrow$	kontextové gramatiky monotónní gramatiky
zásobníkové automaty	$\longleftrightarrow$	bezkontextové gramatiky
konečné automaty DFA, NFA, ϵNFA	$\longleftrightarrow$	regulární gramatiky

- **gramatiky typu 1** (kontextové jazyky $\mathcal{L}_1$)
  - pouze pravidla ve tvaru $\alpha A \beta \rightarrow \alpha \omega \beta$
    $A \in V, \alpha, \beta \in (V \cup T)^*, \omega \in (V \cup T)^+$
  - jedinou výjimkou je pravidlo $S \rightarrow \epsilon$, potom se ale S nevyskytuje na pravé straně žádného pravidla

Chomského hierarchie

- **gramatiky typu 0** (rekurzivně spočetné jazyky $\mathcal{L}_0$)

  pravidla v obecné formě $\alpha \to \beta$, $\alpha, \beta \in (V \cup T)^*, \alpha$ obsahuje neterminál

gramatiky typu 1 (kontextové jazyky $\mathcal{L}_1$)

- pouze pravidla ve tvaru $\alpha A \beta \to \alpha \omega \beta$

  $A \in V, \alpha, \beta \in (V \cup T)^*, \omega \in (V \cup T)^{+}$
  - jedinou výjimkou je pravidlo $S \to \epsilon$, potom se ale S nevyskytuje na pravé straně žádného pravidla

- **gramatiky typu 2** (bezkontextové jazyky $\mathcal{L}_2$)

  pouze pravidla ve tvaru $A \to \omega, A \in V, \omega \in (V \cup T)^*$
- **gramatiky typu 3** (regulární/pravé lineární jazyky $\mathcal{L}_3$)

  pouze pravidla ve tvaru $A \to \omega B, A \to \omega, A, B \in V, \omega \in T^*$

Monotónní a kontextové gramatiky

Example 12.1 (kontextový jazyk)

$L = \{a^n b^n c^n | n \geq 1\}$ je kontextový jazyk, není bezkontextový.

Monotónní gramatika:

$$
\begin{array}{ll}
S \to aSBC | abC & \\
CB \to BC & \text{není kontextové, nutno rozepsat!} \\
\hline
bB \to bb & \\
bC \to bc & \\
cC \to cc &
\end{array}
$$

- Je snažší napsat monotónní gramatiku
  a upravit ji na kontextovou.
- Vygeneruji stejný počet a, B, C.
- Přeuspořádám - proto potřebuji B, C neterminály.
- Dovolím přepsat jen v abecedním pořadí
  - Začínám s malými a s jedním b za nimi.
  - B se může přepsat, když je na řadě - vlevo od něj b.
  - $\Rightarrow$ Především nedovolím přepsat B, pokud je před ním c.

Separované gramatiky

Definition 12.1 (Separovaná gramatika)

Gramatika je **separovaná**, pokud obsahuje pouze pravidla tvaru $\alpha \to \beta$, kde:

- buď $\alpha, \beta \in V^+$ (neprázdné posloupnosti neterminálů)
- nebo $\alpha \in V$ a $\beta \in T \cup \{\epsilon\}$.

Lemma

Ke každé gramatice G lze sestrojit ekvivalentní separovanou gramatiku G'.

Proof:

- Nechť $G = (V, T, P, S)$
- pro každý terminál $a \in T$ zavedeme nový neterminál A'.
- v pravidlech z P
  - nahradíme terminály odpovídajícími neterminály
  - přidáme pravidla $A' \to a$
- Výsledná gramatika je separovaná a zřejmě $L(G) = L(G')$. $\square$

Od monotonie ke kontextovosti

Definition 12.2 (monotónní (nevypouštějící) gramatika)

Gramatika je **monotónní (nevypouštějící)**, jestliže pro každé pravidlo $(\alpha \rightarrow \beta) \in P$ platí $|\alpha| \leq |\beta|$. Monotónní gramatiky slovo v průběhu generování nezkracují.

Lemma

Ke každé monotónní gramatice lze nalézt ekvivalentní gramatiku kontextovou.

Proof:

- nejprve převedeme gramatiku na separovanou
  - tím se monotonie neporuší (a pravidla $A' \rightarrow a$ jsou kontextová)
- zbývající pravidla $A_1 \dots A_m \rightarrow B_1 \dots B_n$, $m \leq n$ převedeme na pravidla s novými neterminály C

$$
\begin{array}{llll}
A_1 A_2 \dots A_m & \rightarrow C_1 A_2 \dots A_m & C_1 C_2 \dots C_m & \rightarrow B_1 C_2 \dots C_m \\
C_1 A_2 \dots A_m & \rightarrow C_1 C_2 \dots A_m & B_1 C_2 \dots C_m & \rightarrow B_1 B_2 \dots C_m \\
\vdots & \vdots & \vdots & \vdots \\
C_1 \dots C_{m-1} A_m & \rightarrow C_1 \dots C_{m-1} C_m & B_1 \dots B_{m-1} C_m & \rightarrow B_1 \dots B_{m-1} B_m \dots B_n
\end{array}
$$

Příklad kontextového jazyka

Example 12.2

Jazyk $L = \{a^i b^j c^k | 1 \leq i \leq j \leq k\}$ je kontextový jazyk, není bezkontextový.

Proof: (na jedničku povinné)

$S \to aSBC \vert aBC$	generování symbolů a
$B \to BBC$	množení symbolů B
$C \to CC$	množení symbolů C
$CB \to BC$	uspořádání symbolů B a C
$aB \to ab$	začátek přepisu B na b
$bB \to bb$	pokračování přepisu B na b
$bC \to bc$	začátek přepisu C na c
$cC \to cc$	pokračování přepisu C na c

$CB \to BC$ není kontextové pravidlo, nahradíme ho

$CB \to XB$, $XB \to XY$, $XY \to BY$, $BY \to BC$ $\square$

Lineárně omezené automaty

- Ještě potřebujeme ekvivalent pro kontextové gramatiky
- kontextovou gramatiku dostaneme z libovolné monotónní gramatiky

Definition 12.3 (lineárně omezený automat (LBA))

Lineárně omezený automat LBA je nedeterministický TM, kde na pásce je označen levý a pravý konec $\underline{l}, \underline{r}$. Tyto symboly nelze při výpočtu přepsat a nesmí se jít nalevo od $\underline{l}$ a napravo od $\underline{r}$.
Slovo w **je přijímáno lineárně omezeným automatem**, pokud existuje přijímající výpočet $q_0\underline{l}w\underline{r} \vdash^* \alpha p \beta$, $p \in F$.

- Prostor výpočtu je definován vstupním slovem a automat při jeho přijímání nesmí překročit jeho délku
- u monotónních (kontextových) derivací to není problém – žádné slovo v derivaci není delší než vstupní slovo.

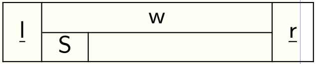
Od kontextových jazyků k LBA

Theorem 12.1

Každý kontextový jazyk lze přijímat pomocí LBA.

Proof: z kontextové gramatiky k LBA

- derivaci gramatiky budeme simulovat pomocí LBA
- použijeme pásku se dvěma stopami
- slovo *w* dáme nahoru, na začátek dolní stopy *S*



Aplikace pravidla

$\alpha X \beta \rightarrow \alpha \gamma \beta$

u	α	X	β	v

u	α	γ	β	v

- přepisujeme slovo ve druhé stopě podle pravidel G
  - nedeterministicky vybereme část k přepsání
  - provedeme přepsání dle pravidla (pravá část se odsune)
- pokud jsou ve druhé stopě samé terminály, porovnáme ji s první stopou
  - slovo přijmeme nebo zamítneme

□

Od LBA ke kontextovým jazykům

Theorem 12.2

LBA přijímají pouze kontextové jazyky.

Proof: z LBA ke kontextovým gramatikám

- potřebujeme převést LBA na monotónní gramatiku
  - tj. gramatika nesmí generovat nic navíc
- výpočet ukryjeme do 'dvoustopých' neterminálů
- generuj slovo ve tvaru $(a_0, [q_0, \underline{l}, a_0]), (a_1, a_1), \ldots, (a_n, [a_n, \underline{r}])$

w		
$q_0, \underline{l}, a_0$		$a_n, \underline{r}$

- simuluj práci LBA ve 'druhé' stopě (stejně jako u TM)
  - pro $\delta(p, x) = (q, x', R)$: $Px \rightarrow x'Q$
  - pro $\delta(p, x) = (q, x', L)$: $yPx \rightarrow Qyx'$
- pokud je stav koncový, smaž 'druhou' stop
- speciálně je třeba ošetřit přijímání prázdného slova
  - pokud LBA přijímá ϵ, přidáme speciální startovací pravidlo

Mějme lineáně omezený automat pro jazyk $L = \{a^{2n} | n \geq 1\}$, LBA $M = (\{q_0, q_1, q_2, q_F\}, \{a\}, \{a, \underline{l}, \underline{r}\}, \delta, q_0, B, \{q_F\})$ s δ v tabulce:

δ	komentář
$\delta(q_0, \underline{l}) = (q_0, \underline{l}, R)$	přeskočím prázdnou zarážku
$\delta(q_0, a) = (q_1, a, R)$	zvětší čítač ($2k+1$ symbolů)
$\delta(q_2, a) = (q_1, a, R)$	zvětší čítač ($2k+1$ symbolů)
$\delta(q_1, a) = (q_2, a, R)$	nuluje čítač ($2k$ symbolů)
$\delta(q_2, \underline{r}) = (q_F, \underline{r}, L)$	konec výpočtu, přijímám.

Example 12.3 (Gramatika z lineáně omezeného automatu)

Monotónní gramatika $G = (V, \{a\}, S, P_1 \cup P_2 \cup P_3)$

$$V = \left\{ \begin{array}{c} S, L, C, R, \begin{bmatrix} a \\ q_0\underline{l}a \end{bmatrix}, \begin{bmatrix} a \\ \underline{l}q_0 a \end{bmatrix}, \begin{bmatrix} a \\ \underline{l}a \end{bmatrix}, \\ \begin{bmatrix} a \\ q_2 a \end{bmatrix}, \begin{bmatrix} a \\ q_1 a \end{bmatrix}, \begin{bmatrix} a \\ a\underline{r} \end{bmatrix}, \begin{bmatrix} a \\ M_\leftarrow \end{bmatrix}, \\ \begin{bmatrix} a \\ a \end{bmatrix}, \begin{bmatrix} a \\ q_1 a\underline{r} \end{bmatrix}, \begin{bmatrix} a \\ aq_2\underline{r} \end{bmatrix} \end{array} \right\}$$

$$P_1 = \left\{ \begin{array}{lcl} S & \to & LR | LCR \\ C & \to & CC | \begin{bmatrix} a \\ a \end{bmatrix} \\ L & \to & \begin{bmatrix} a \\ q_0\underline{l}a \end{bmatrix} \\ R & \to & \begin{bmatrix} a \\ a\underline{r} \end{bmatrix} \end{array} \right\}$$

Pravidla pro přechodovou funkci

$$P_2 = \left\{ \begin{array}{ccc}
\begin{bmatrix} a \\ q_0\underline{l}a \end{bmatrix} & \to & \begin{bmatrix} a \\ \underline{l}q_0 a \end{bmatrix} \\
\begin{bmatrix} a \\ \underline{l}q_0 a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} & \to & \begin{bmatrix} a \\ \underline{l}a \end{bmatrix} \begin{bmatrix} a \\ q_1 a \end{bmatrix} \\
\begin{bmatrix} a \\ q_2 a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} & \to & \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ q_1 a \end{bmatrix} \\
\begin{bmatrix} a \\ q_1 a \end{bmatrix} \begin{bmatrix} a \\ a \end{bmatrix} & \to & \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ q_2 a \end{bmatrix} \\
\begin{bmatrix} a \\ q_2 a \end{bmatrix} \begin{bmatrix} a \\ a\underline{r} \end{bmatrix} & \to & \begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ q_1 a\underline{r} \end{bmatrix} \\
\begin{bmatrix} a \\ q_1 a\underline{r} \end{bmatrix} & \to & \begin{bmatrix} a \\ aq_2\underline{r} \end{bmatrix} \\
\begin{bmatrix} a \\ aq_2\underline{r} \end{bmatrix} & \to & \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix}
\end{array} \right\}$$

Pravidla mazání

$$P_3 = \left\{ \begin{array}{ccc}
\begin{bmatrix} a \\ a \end{bmatrix} \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} & \to & \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} a \\
\begin{bmatrix} a \\ \underline{l}a \end{bmatrix} \begin{bmatrix} a \\ M_{\leftarrow} \end{bmatrix} & \to & aa
\end{array} \right\}$$

Obecněji

- Při více písmenech bychom měli varianty (téměř) všech neterminálů pro každé písmeno.
- V pravidlech P_1 jen totéž písmeno nahoře i dole.
- Pokud skončíme uvnitř slova, musíme mazat do obou stran až k zarážkám.

Theorem 12.3 (Rekurzivně spočetné jsou $\mathcal{L}_0$)

Každý rekurzivně spočetný jazyk je typu 0.

Proof: Od Turingova stroje ke gramatice

pro Turingův stroj T najdeme gramatiku G, $L(T) = L(G)$

- $G = (\{S, C, D, E\} \cup \{\underline{X}\}_{x \in \Sigma \cup \Gamma} \cup \{Q_i\}_{q_i \in Q}, \Sigma, P, S)$, P je:
- gramatika nejdříve vygeneruje pásku stroje a kopii slova $wB^n\underline{W}^R Q_0 B^m$, kde B^i představují volný prostor pro výpočet
- potom simuluje výpočet (stavy jsou součástí slova)
- v koncovém stavu smažeme pásku, necháme pouze kopii slova

1)	$S \to DQ_0E$	
	$D \to xD\underline{X}\vert E$	generuje slovo a jeho revizní kopii pro výpočet
	$E \to BE\vert\epsilon$	generuje volný prostor pro výpočet
2)	$\underline{X}P \to Q\underline{X'}$	pro $\delta(p, x) = (q, x', R)$
	$\underline{X}P\underline{Y} \to \underline{X'Y}Q$	pro $\delta(p, x) = (q, x', L)$
3)	$P \to C$	pro $p \in F$
	$C\underline{A} \to C, \underline{A}C \to C$	mazání pásky
	$C \to \epsilon$	konec výpočtu

Příklad

Turingův stroj pro jazyk $L = \{a^{2n} | n \geq 0\}$, TM
$M = (\{q_0, q_1, q_2, q_F\}, \{a\}, \{a\}, \delta, q_0, B, \{q_F\})$ s δ v tabulce:

δ	komentář
$\delta(q_0, a) = (q_1, a, R)$	první symbol
$\delta(q_1, a) = (q_0, a, R)$	nuluje čítač ($2k$ symbolů)
$\delta(q_0, B) = (q_F, B, L)$	přijme a zastaví

Example 12.4

Gramatika $G = (\{S, C, D, E, Q_0, Q_1, Q_F, \underline{a}\}, \{a\}, S, P_1 \cup P_2 \cup P_3)$,

Inicializace | Přechodová funkce | Mazání

$$P_1 = \left\{ \begin{array}{ccc} S & \rightarrow & DQ_0E \\ D & \rightarrow & aD\underline{a}|E \\ E & \rightarrow & BE|\epsilon \end{array} \right\} P_2 = \left\{ \begin{array}{ccc} \underline{a}Q_0 & \rightarrow & Q_1\underline{a} \\ \underline{a}Q_1 & \rightarrow & Q_0\underline{a} \\ BQ_0\underline{a} & \rightarrow & B\underline{a}Q_F \end{array} \right\} P_3 = \left\{ \begin{array}{ccc} Q_F & \rightarrow & C \\ C\underline{a} & \rightarrow & C \\ \underline{a}C & \rightarrow & C \\ BC & \rightarrow & C \\ C & \rightarrow & \epsilon \end{array} \right\}.$$

Konkrétně pro $aa \in L(M)$ vygeneruji $aaB\underline{aa}Q_0$, mezivýsledek $aaB\underline{a}Q_F\underline{a}$ a výsledek aa.

Od Turingova stroje ke gramatice

Ještě $L(T) = L(G)$?

- $w \in L(T)$
  - existuje konečný výpočet stroje T (konečný prostor)
  - gramatika vygeneruje dostatečně velký prostor pro výpočet
  - simuluje výpočet a smaže dvojníky
- $w \in L(G)$
  - pravidla v derivaci nemusí být v pořadí, jakém chceme
  - derivaci můžeme přeuspořádat tak, že pořadí je 1),2),3).
  - podtržené symboly smazány, tj. vygenerován koncový stav.

Example 12.5

Turingův stroj $M = (\{q_0, q_1, q_F\}, \{a\}, \{a, B\}, \delta, q_0, B, \{q_F\})$

$\delta(q_0, B) = (q_F, B, R)$
$\delta(q_0, a) = (q_1, a, R)$
$\delta(q_1, a) = (q_0, a, R)$

Gramatika po zjednodušení

$G = (\{S, C, D, E, \underline{a}, Q_0, Q_1\}, \{a\}, P, S)$
$S \to DQ_0$
$D \to aD\underline{a} | B$
$BQ_0 \to C$
$\underline{a}Q_0 \to Q_1\underline{a}$
$\underline{a}Q_1 \to Q_0\underline{a}$
$C\underline{a} \to C$
$C \to \epsilon$

Od gramatik k Turingově stroji

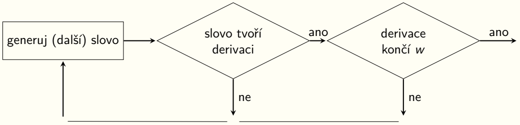
Theorem 12.4

Každý jazyk typu 0 je rekurzivně spočetný.

Proof:

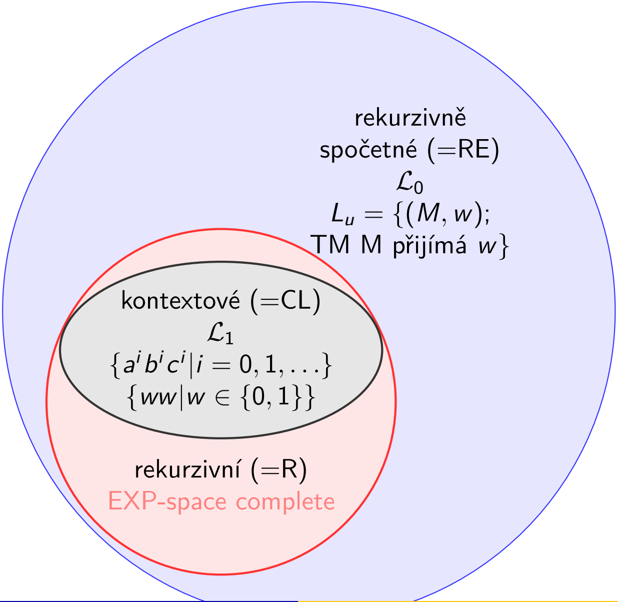
idea: TM postupně generuje všechny derivace

- derivaci $S \Rightarrow \omega_1 \Rightarrow \ldots \Rightarrow \omega_n = w$ kódujeme jako slovo $\#S\#\omega_1\#\ldots\#w\#$
- umíme udělat TM, který přijímá slova $\#\alpha\#\beta\#$, kde $\alpha \Rightarrow \beta$
- umíme udělat TM, který přijímá slova $\#\omega_1\#\ldots\#\omega_k\#$, kde $\omega_1 \Rightarrow^* \omega_k$
- umíme udělat TM postupně generující všechna slova.



□

Hierarchie jazyků (kontextové a výš)



$L_d = \{w;$ TM s kódem w nepřijímá vstup $w\}$

Rekurzivní jazyky

Definition 12.4 (TM zastaví)

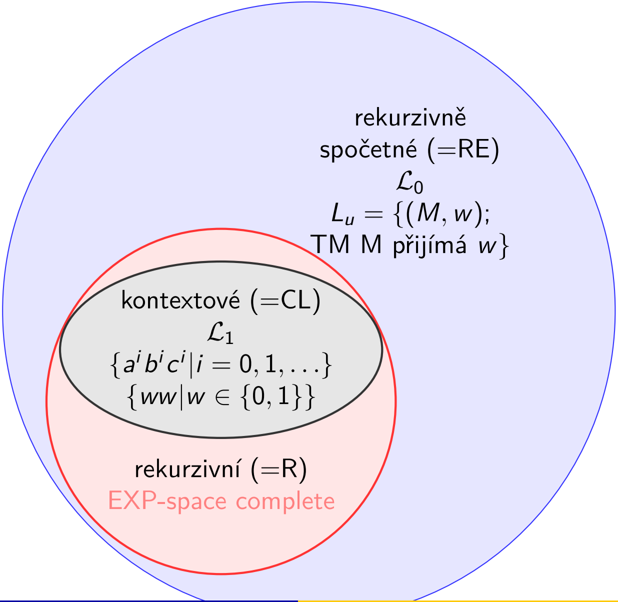
TM **zastaví** pokud vstoupí do stavu q, s čteným symbolem X, a není instrukce pro tuto konfiguraci, t.j., $\delta(q, X)$ není definováno.

- Z definice, v přijímajícím stavu $q \in F$ TM zastaví,
- dokud nezastaví, nevíme, jestli přijme nebo nepřijme slovo.

Definition 12.5 (Rekurzivní jazyky)

Říkáme, že TM M **rozhoduje jazyk** L, pokud $L = L(M)$ a pro každé $w \in \Sigma^*$ stroj nad w zastaví.
Jazyky rozhodnutelné TM nazýváme **rekurzivní jazyky**.

Hierarchie jazyků (kontextové a výš)



$L_d = \{w;$ TM s kódem w nepřijímá vstup $w\}$

Jazyk který není rekurzivně spočetný

Směřujeme k důkazu nerozhodnutelnosti jazyka dvojic (M, w) takových, že:

- M je binárně kódovaný Turingův stroj s abecedou $\{0, 1\}$,
- $w \in \{0, 1\}^*$ a
- M nepřijímá vstup w.

Postup:

- Kódování TM binárním kódem pro libovolný počet stavů TM.
- Kód TM vezmeme TM jako binární řetězec.
- Pokud kód nedává smysl, reprezentuje TM bez transakcí. Tedy každý kód reprezentuje nějaký TM.
- **Diagonální jazyk L_d**;
  $L_d = \{w;$ TM reprezentovaný jako w takový, že **nepřijímá** $w\}$.
- Neexistuje TM přijímající jazyk L_d. Spuštění takového stroje na vlastním kódu by vedlo k paradoxu.

Jazyk L_d není rekurzivně spočetný. Proto $\overline{L_d}$ není rekurzivní. Lze dokázat, že $\overline{L_d}$ je rekurzivně spočetný.

Kódování

- Pro kódování TM $M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, \{q_2\})$ očíslujeme stavy, symboly a směry L, R.
- Předpokládejme:
  - Počáteční stav je vždy q_1.
  - Stav q_2 je vždy jediný koncový stav (nepotřebujeme víc, TM zastaví).
  - První symbol je vždy 0, druhý 1, třetí B, prázdný symbol. Ostatní symboly pásky očíslujeme libovolně.
  - Směr L je 1, směr R je 2.
- Jeden krok $\delta(q_i, X_j) = (q_k, X_l, D_m)$ kódujeme: $0^i 1 0^j 1 0^k 1 0^l 1 0^m$. Všechna $i, j, k, l, m \geq 1$ takže se dvě jedničky za sebou nevyskytují.
- Celý TM se skládá z kódů všech přechodů v nějakém pořadí oddělených dvojicemi jedniček 11: $C_1 11 C_2 11 \ldots C_{n-1} 11 C_n$.

Budeme potřebovat uspořádat řetězce do posloupnosti:

- Řetězce bereme uspořádané podle délky, stejně dlouhé uspořádáme lexikograficky.
- První je ϵ, druhý 0, třetí 1, čtvrtý 00 atd.
- i–tý řetězec označujeme w_i.

Příklad kódování TM

Turingův stroj

$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$

δ	0	1	B
$\rightarrow q_1$		$(q_3, 0, R)$	
$*q_2$			
q_3	$(q_1, 1, R)$	$(q_2, 0, R)$	$(q_3, 1, L)$.

- Kód pro transakce:

C_1	C_2	C_3	C_4
0100100010100	0001010100100	00010010010100	0001000100010010

- Kód celého TM:
  0100100010100110001010100100110001001001010011000100010001001 0.

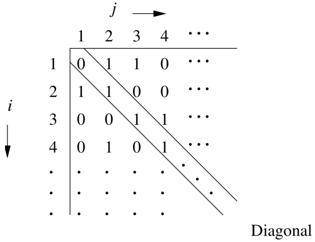
Definition 12.6 (Diagonální jazyk)

Diagonální jazyk L_d je definovaný
$L_d = \{w \in \{0,1\}^*; \text{TM reprezentovaný jako } w \text{ který nepřijímá slovo } w\}$.

$L_d = \{w; \text{na diagonále je } 0\}$.

Theorem 12.5

L_d není rekurzivně spočetný jazyk, tj. neexistuje TM přijímající L_d.



Proof.

- Předpokládejme L_d je RE, $L_d = L(M_d)$ pro nějaký TM M_d.
- Jeho jazyk je $\{0,1\}$, tedy je v seznamu na obrázku: 'Přijímá TM M_i vstupní slovo w_j?'
- Alespoň jeden řetězec ho kóduje, řekněme $code(M_d) = w_d$.
- Je $w_d \in L_d$
  - Pokud 'ano', na diagonále má být 0, tj. $w_d \notin L(M_d) = L_d$, spor.
  - Pokud 'ne', na diagonále má být 1, $w_d \in L(M_d) = L_d$, spor.

Proto takový M_d neexistuje. Tedy L_d není rekurzivně spočetný. □

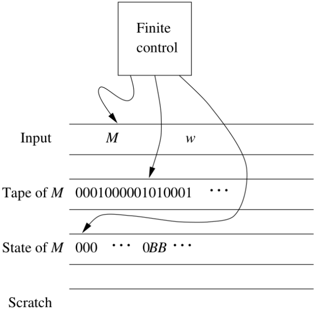
Univerzální Turingův stroj

Definition 12.7 (Univerzální jazyk)

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

Theorem 12.6 (Existence Univerzálního Turingova stroje)

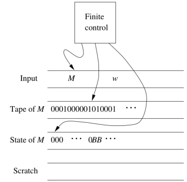
Existuje Turingův stroj U, pro který $L_u = L(U)$.



Popíšeme U jako vícepáskový Turingův stroj.

- Přechody M jsou napsány na první pásce spolu s řetězcem w.
- Na druhé pásce simulujeme výpočet M, používající formát jako kód M, tj. symboly 0^i oddělené jedničkou 1.
- Třetí páska obsahuje stav M reprezentovaný i nulami.

Operace univerzálního Turingova stroje



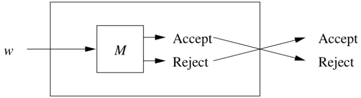
Operace U jsou následující:

- Otestuj, zda je kód M legitimní; pokud ne, U zastav bez přijetí.
- Inicializuj druhou pásku kódovaným slovem w: 10 pro 0 ve w, 100 pro 1; blank jsou nechané prázdné a nahrazeny 1000 pouze 'v případě potřeby'.
- Napiš 0, počáteční stav M, na třetí pásku. Posuň hlavu druhé pásky na první simulované políčko.
- Simuluj jednotlivé přechody M
  - Najdi na první pásce správnou transakci $0^i10^j10^k10^l10^m$, 0^i na pásce 3, 0^j na pásce 2.
  - Změň obsah pásky 3 na 0^k.
  - Nahraď 0^j na 2. pásce řetězcem 0^l. Použij čtvrtou 'scratch tape' pro správné mezery.
  - Posuň hlavu 2. pásky na pozici vedle 1 vlevo nebo vpravo, podle pohybu m.
- Pokud jsme nenašli instrukci pro M, zastavíme.
- Pokud M přejde do přijímajícího stavu, pak U také přijme. □

$L \& \overline{L} \in RE \Rightarrow L, \overline{L}$ je rekurzivní

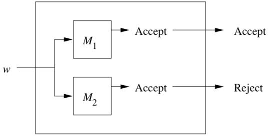
Lemma

Je–li L rekurzivní jazyk, je rekurzivní i $\overline{L}$.



Theorem 12.7 (Postova věta)

Jazyk L je rekurzivní, právě když L i $\overline{L}$ (doplněk) jsou rekurzivně spočetné.



Proof:

- Máme TM $L = L(M_1)$ a $\overline{L} = L(M_2)$.
- pro dané slovo w naráz simulujeme M_1 i M_2 (dvě pásky, stav se dvěma komponentami).
- Pokud jeden z M_i přijme, M zastaví a odpoví.
- Jazyky jsou komplementární, jeden z M_i vždy zastaví, L je rekurzivní. $\square$

Nerozhodnutelnost univerzálního jazyka

Theorem 12.8 (Nerozhodnutelnost univerzálního jazyka)

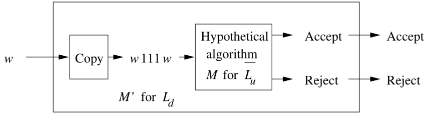
L_u je rekurzivně spočetný, ale není rekurzivní.

Proof.

- Máme TM přijímající L_u, tj. je RE.
- Předpokládejme, že je L_u rekurzivní.
- Pak $\overline{L_u}$ by byl také rekurzivní.
- Pro TM přijímající $\overline{L_u}$ můžeme zkonstruovat TM přijímající L_d (vpravo).
- Protože víme, že L_d není RE, $\overline{L_u}$ není RE a L_u není rekurzivní.

□

Modifikace TM pro $\overline{L_u}$ na TM pro L_d:



- Řetězec w přepiš na $w111w$ (2–páskový, převeď na 1–páskový).
- Simuluj M na novém vstupu. Přijmi iff M přijme.
- Zvol i tak že $w_i = w$. Předchozí krok přijímá $\overline{L_u}$, tj. případy kdy M_i nepřijímá w_i, tj. jazyk L_d.

Nerozhodnutelné problémy o automatech a gramatikách

Definition 13.1 (Rozhodnutelný problém)

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

'Rozhodnutelný' mluví o problémech, 'rekurzivní' o jazycích, jinak jde o 'totéž'.

Example 13.1 ('Problémy')

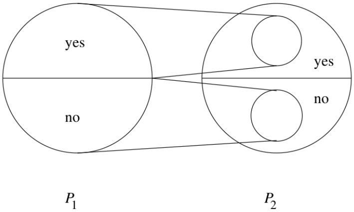
- Obsahuje vstupní slovo pět nul?
- Je vstupní slovo korektně definovaným kódem Turingova stroje v kódování výše?
- Zastaví TM kódu M nad slovem w?
- Zastaví TM kódu w nad slovem w?

Redukce

Definition 13.2 (Redukce)

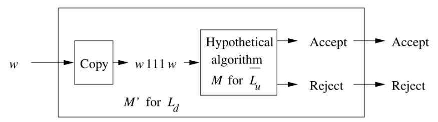
Redukcí problému P_1 na P_2, nazýváme algoritmus R, který pro každou instanci $w \in P_1$ zastaví a vydá $R(w) \in P_2$ tak, že

- $P_1(w) = \mathit{ano}$ právě když $P_2(R(w)) = \mathit{ano}$
- tj. i $P_1(w) = \mathit{ne}$ právě když $P_2(R(w)) = \mathit{ne}$.



Example 13.2

Redukce TM pro L_d na TM pro $\overline{L_u}$:



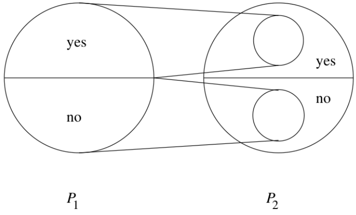
- P_1 = Nepřijímá TM reprezentovaný w vstupní slovo w?
- P_2 = Nepřijímá TM reprezentovaný M vstupní slovo w?

Věta o (ne)rozhodnutelnosti díky redukci

Theorem 13.1 (Redukce)

Pokud existuje redukce problému P_1 na P_2, pak:

- *Pokud P_1 je nerozhodnutelný, pak je nerozhodnutelný i P_2.*
- *Pokud P_1 není rekurzivně spočetný, pak není RE ani P_2.*



Proof.

- Předpokládejme P_1 je nerozhodnutelný. Je–li možné rozhodnout P_2, pak můžeme zkombinovat redukci P_1 na P_2 s algoritmem rozhodujícím P_2 pro konstrukci algoritmu rozhodujícího P_1. Proto je P_2 nerozhodnutelný.
- Předpokládejme P_1 ne–RE, ale P_2 je RE. Podobně jako výše zkombinujeme redukci a výsledek P_2 k důkazu P_1 je RE; SPOR.

□

Problém zastavení

Víme: L_u je rekurzivně spočetný, ale není rekurzivní.

Definition 13.3 (Problém zastavení)

Instancí problému zastavení je dvojice řetězců $M, w \in \{0,1\}^*$.
Problém zastavení je najít alogritmus $Halt(M, w)$, který vydá 1 právě když stroj M zastaví na vstupu w, jinak vydá 0.

Theorem (Problém zastavení)

Problém zastavení není rozhodnutelný.

Proof.

- Redukujeme L_d na $Halt$.
- Předpokládejme, že máme algoritmus (Turingův stroj) pro $Halt()$.
- Modifikujeme ho na stroj $Halt_{no}(w)$; $w \in \{0,1\}^*$:
  - Pokud $Halt(w, w)$, spustíme nekonečný cyklus
  - jinak zastavíme.
- Otázka $Halt(Halt_{no}, Halt_{no})$ není řešitelná, proto algoritmus $Halt()$ nemůže

Postův korespondenční problém

Definition 13.4 (Postův korespondenční problém)

Instance **Postova korespondenčního problému (PCP)** jsou dva seznamy slov nad abecedou Σ značené $A = w_1, w_2, \dots, w_k$ a $B = x_1, x_2, \dots, x_k$ stejné délky k. Pro každé i, dvojice (w_i, x_i) se nazývá **odpovídající** dvojice.
Instance PCP **má řešení**, pokud existuje posloupnost jednoho či více přirozených čísel $i_1, i_2, \dots, i_m$ tak že $w_{i_1} w_{i_2} \dots w_{i_m} = x_{i_1} x_{i_2} \dots x_{i_m}$ tj. dostaneme stejné slovo. V tom případě říkáme, že posloupnost $i_1, i_2, \dots, i_m$ **je řešení**.
Postův korespondenční problém je: Pro danou instanci PCP, rozhodněte, zda má řešení.

Example 13.3

	Seznam A	Seznam B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

- $\Sigma = \{0, 1\}$, seznamy A,B v tabulce.
- Řešení $2, 1, 1, 3$ vytvoří slovo 101111110.
- Jiné řešení: 2,1,1,3,2,1,1,3.

Částečná řešení

Example 13.4

$\Sigma = \{0, 1\}$. Neexistuje řešení pro seznamy:

	List A	List B
i	w_i	x_i
1	10	101
2	011	11
3	101	011.

Zdůvodnění:

- $i_1 = 1$, jinak by první symbol neodpovídal.
- Máme částečné řešení:
  A: $10\cdots$
  B: $101\cdots$

Definition 13.5 (Částečné řešení)

Částečným řešením nazýváme posloupnost indexů $i_1, i_2, \ldots, i_r$ taková že jeden z řetězců $w_{i_1}, w_{i_2}, \ldots, w_{i_r}$ a $x_{i_1}, x_{i_2}, \ldots, x_{i_r}$ je prefix druhého (i v případě, že řetězce nejsou totožné).

Lemma

Je–li posloupnost čísel řešením, pak je každý prefix částečným řešením.

- $i_2 = 1$, řetězce $\begin{array}{l}1010\\101101\end{array}$ nesouhlasí na 4.pozici.
- $i_2 = 2$, $\begin{array}{l}10011\\10111\end{array}$ nesouhlasí na 3.pozici.
- Je možné jen $i_2 = 3$.

A: $10101\cdots$
B: $101011\cdots$

- Jsme ve stejné pozici jako po volbě $i_1 = 1$.
- Nelze dostat oba řetězce na stejnou délku.

Modifikovaný Postův korespondenční probém MPCP

Definition 13.6 (Modifikovaný Postův korespondenční probém MPCP)

Mějme PCP, tj. seznamy $A = w_1, w_2, \ldots, w_k$ a $B = x_1, x_2, \ldots, x_k$. Hledáme seznam 0 nebo více přirozených čísel $i_1, i_2, \ldots, i_m$ tak že $\mathbf{w_1}, w_{i_1}, w_{i_2}, \ldots, w_{i_m} = \mathbf{x_1}, x_{i_1}, x_{i_2}, \ldots, x_{i_m}$. V tom případě říkáme, že PCP **má iniciální řešení**.

Modifikovaný Postův korespondenční problém: má PCP iniciální řešení?

Example 13.5

Tento PCP nemá iniciální řešení.

	seznam A	seznam B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Proof:

- Částečné instance $\begin{matrix}1\\111\end{matrix}$, $\begin{matrix}11\\111111\end{matrix}$ se nikdy nesrovnají na stejnou délku.
- Jiné volby vedou k různým písmenům abecedy. $\square$

MPCP redukce na PCP

Lemma 13.1 (Red. MPCP na PCP)

$w \in MPCP$ má iniciální řešení, právě když má $R(w)$ řešení.

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Example 13.6 (MPCP redukce na PCP.)

	List C	List D
i	y_i	z_i
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	$	*$

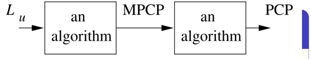
Proof:

- Vezměme nové symboly $*, \$ \notin \Sigma$.
- $\forall i = 1, \ldots, k$ definujeme y_i rozšířením w_i s $*$ za každým písmenem w_i.
- $\forall i = 1, \ldots, k$ def. z_i rozšířením x_i s $*$ **před** každým písmenem x_i.
- $y_0 = *y_1$, $z_0 = z_1$.
- $y_{k+1} = \$$, $z_{k+1} = *\$$.
- $i_1, i_2, \ldots, i_m$ je iniciální řešení, iff $0, i_1, i_2, \ldots, i_m, (k+1)$ je řešení PCP. $\square$

Nerozhodnutelnost PCP

- Chceme dokázat, že PCP je algoritmicky nerozhodnutelný.
- Redukovali jsme MPCP na PCP (minulý slajd)
- a redukujeme L_u na MPCP.

Algorithm: Redukce L_u na MPCP



Konstruujeme MPCP pro TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, který nikdy nepíše B a nejde hlavou doleva od počáteční pozice. Nechť $w \in \Sigma^*$ je vstupní slovo.

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

Example 13.7

Konvertujme TM $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_1, B, \{q_3\})$

q_i	$\delta(q_i, 0)$	$\delta(q_i, 1)$	$\delta(q_i, B)$
q_1	$(q_2, 1, R)$	$(q_2, 0, L)$	$(q_2, 1, L)$
q_2	$(q_3, 0, L)$	$(q_1, 0, R)$	$(q_2, 0, R)$
q_3	–	–	–

a vstupní slovo $w = 01$ na instanci MPCP.

seznam A	seznam B	zdroj
$q_1 0$	$1 q_2$	z $\delta(q_1, 0) = (q_2, 1, R)$
$0 q_1 1$	$q_2 00$	z $\delta(q_1, 1) = (q_2, 0, L)$
$1 q_1 1$	$q_2 10$	z $\delta(q_1, 1) = (q_2, 0, L)$
$0 q_1 \#$	$q_2 01 \#$	z $\delta(q_1, B) = (q_2, 1, L)$
$1 q_1 \#$	$q_2 11 \#$	z $\delta(q_1, B) = (q_2, 1, L)$
$0 q_2 0$	$q_3 00$	z $\delta(q_2, 0) = (q_3, 0, L)$
$1 q_2 0$	$q_3 10$	z $\delta(q_2, 0) = (q_3, 0, L)$
$q_2 1$	$0 q_1$	z $\delta(q_2, 1) = (q_1, 0, R)$
$q_2 \#$	$0 q_2 \#$	z $\delta(q_2, B) = (q_2, 0, R)$

Seznam dvojic bez B symbolu (ve dvou tabulkách)

seznam A	seznam B
$\#$	$\# q_1 01 \#$
0	0
1	1
$\#$	$\#$
$0 q_3 0$	q_3
$0 q_3 1$	q_3
$1 q_3 0$	q_3
$1 q_3 1$	q_3
$0 q_3$	q_3
$1 q_3$	q_3
$q_3 0$	q_3
$q_3 1$	q_3
$q_3 \#\#$	$\#$

MPCP simulace TM

seznam A	seznam B	zdroj
$q_1 0$	$1 q_2$	z $\delta(q_1, 0) = (q_2, 1, R)$
$0 q_1 1$	$q_2 00$	z $\delta(q_1, 1) = (q_2, 0, L)$
$1 q_1 1$	$q_2 10$	z $\delta(q_1, 1) = (q_2, 0, L)$
$0 q_1 \#$	$q_2 01\#$	z $\delta(q_1, B) = (q_2, 1, L)$
$1 q_1 \#$	$q_2 11\#$	z $\delta(q_1, B) = (q_2, 1, L)$
$0 q_2 0$	$q_3 00$	z $\delta(q_2, 0) = (q_3, 0, L)$
$1 q_2 0$	$q_3 10$	z $\delta(q_2, 0) = (q_3, 0, L)$
$q_2 1$	$0 q_1$	z $\delta(q_2, 1) = (q_1, 0, R)$
$q_2 \#$	$0 q_2 \#$	z $\delta(q_2, B) = (q_2, 0, R)$

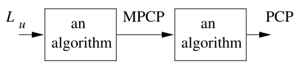
seznam A	seznam B
$\#$	$\# q_1 01 \#$
0	0
1	1
$\#$	$\#$
$0 q_3 0$	q_3
$0 q_3 1$	q_3
$1 q_3 0$	q_3
$1 q_3 1$	q_3
$0 q_3$	q_3
$1 q_3$	q_3
$q_3 0$	q_3
$q_3 1$	q_3
$q_3 \#\#$	$\#$

- M přijímá posloupností $q_1 01 \vdash 1 q_2 1 \vdash 10 q_1 \vdash 1 q_2 01 \vdash q_3 101$.

$A: \quad \# q_1 01 \# 1 q_2 1 \# 10 q_1 \# 1 q_2 01 \# q_3 101 \# q_3 01 \# q_3 1 \# q_3 \#\#$

$B: \quad \# q_1 01 \# 1 q_2 1 \# 10 q_1 \# 1 q_2 01 \# q_3 101 \# q_3 01 \# q_3 1 \# q_3 \#\#$

PCP je algoritmicky nerozhodnutelný



Theorem 13.2 (PCP je algoritmicky nerozhodnutelný)

Postův korespondenční problém PCP je algoritmicky nerozhodnutelný.

Proof.

Předchozí algoritmus redukuje L_u na MPCP. Chceme dokázat:

- M přijímá w právě když zkonstruovaný PCP má iniciální řešení.

$\Rightarrow$ Pokud $w \in L(M)$, začneme iniciálním párem a simulujeme výpočet M na w.

$\Leftarrow$ Máme–li iniciální řešení PCP, odpovídá přijímajícímu výpočtu M nad w.

- MPCP musí začít první dvojicí.
- Dokud $q \notin F$, mazací pravidla se nepoužijí.
- Pokud $q \notin F$, částečné řešení je tvaru: $\begin{array}{l} \text{A:}x \\ \text{B:}xy \end{array}$, t.j. B je delší než A
- tedy musel skončit v přijímajícím stavu.

$\square$

Algoritmická rozhodnutelnost u CFL

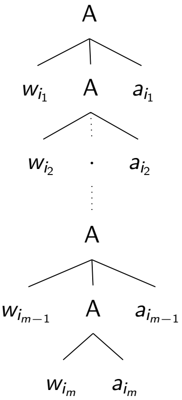
Pro bezkontextové jazyky je algoritmicky rozhodnutelné

- zda dané slovo patří či nepatří do jazyka
    - prázdné slovo zvlášť
    - pak algoritmus CYK
    - nebo otestovat všechny derivace s $2|w| - 1$ pravidly,
- zda je jazyk prázdný
    - algoritmus redukce gramatiky (ne–nenerujících a nedosažitelných), zjistíme, zda lze z S generovat terminální slovo

Nerozhodnutelnost víceznačnosti CFG

Theorem 13.3

Je algoritmicky nerozhodnutelné, zda je bezkontextová gramatika víceznačná (tj. existuje slovo jazyka gramatiky, které má dva různé derivační stomy).



Redukujeme PKP na náš problém.
Mějme instanci PCP ($A = w_1, w_2, \ldots, w_k, B = x_1, x_2, \ldots, x_k$), množinu indexů $a_1, a_2, \ldots, a_k \in N$ a tři gramatiky G_A, G_B, G_{AB}:

$$\begin{array}{lll} G_A & A \rightarrow & w_1Aa_1|w_2Aa_2|\ldots|w_kAa_k| \\ & & w_1a_1|w_2a_2|\ldots|w_ka_k \\ G_B & B \rightarrow & x_1Ba_1|x_2Ba_2|\ldots|x_kBa_k| \\ & & x_1a_1|x_2a_2|\ldots|x_ka_k \\ G_{AB} & \{S \rightarrow & A|B\} \cup G_A \cup G_B. \end{array}$$

Gramatika G_{AB} je víceznačná právě když instance (A, B) PCP má řešení.

- Každé slovo v G_A má jednoznačnou derivaci (danou a_i vpravo). Podobně pro B.

Nerozhodnutelné problémy pro bezkontextové jazyky CFG

Theorem 13.4

Mějme G_1, G_2 bezkontextové gramatiky, R regulární výraz. Následující problémy jsou algoritmicky nerozhodnutelné:

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

Průnik $L(G_1) \cap L(G_2) = \emptyset$

Proof: 1 $L(G_1) \cap L(G_2) = \emptyset$

Převedeme PKP na (1)

- zvolíme nové terminály $\{a_1, a_2, \ldots, a_m\}$ pro kódy indexů

$$
\begin{array}{lll}
G_1 & A \rightarrow & w_1 A a_1 | w_2 A a_2 | \ldots | w_k A a_k | \\
 & & w_1 a_1 | w_2 a_2 | \ldots | w_k a_k \\
G_2 & B \rightarrow & x_1 B a_1 | x_2 B a_2 | \ldots | x_k B a_k | \\
 & & x_1 a_1 | x_2 a_2 | \ldots | x_k a_k
\end{array}
$$

- PKP má řešení právě když $L(G_1) \cap L(G_2) \neq \emptyset$
- první část se musí rovnat, druhá (a_i) zajišťuje stejné pořadí. $\square$

Vše $L(G) = T^*$

Proof: 2 $L(G) = T^*$

Převedeme PKP na (2):

- zvolíme nové terminály $\{a_1, a_2, \ldots, a_m\}$ pro kódy indexů

$$\begin{array}{lll} G_1 & A \rightarrow & w_1 A a_1 | w_2 A a_2 | \ldots | w_k A a_k | \\ & & w_1 a_1 | w_2 a_2 | \ldots | w_k a_k \\ G_2 & B \rightarrow & x_1 B a_1 | x_2 B a_2 | \ldots | x_k B a_k | \\ & & x_1 a_1 | x_2 a_2 | \ldots | x_k a_k \end{array}$$

- jazyky $L(G_1), L(G_2)$ jsou deterministické,
- tedy $\overline{L(G_1)}, \overline{L(G_2)}$ jsou deterministické CFL a $\overline{L(G_1)} \cup \overline{L(G_2)}$ je CFL
- máme CFG G gramatiku s $L(G) = \overline{L(G_1)} \cup \overline{L(G_2)}$
- PKP má řešení $\Leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset \Leftrightarrow L(G) = \overline{L(G_1)} \cup \overline{L(G_2)} \neq \Sigma^*$. $\square$

- Poznámka: $L(G) = \emptyset$ je algoritmicky rozhodnutelné.
- CFL nejsou uzavřené na doplněk, pouze deterministické CFL ano.

Proof: 3-6

Zbylé algoritmicky nerozhodnutelné problémy.

3. Je $L(G_1) = L(G_2)$?
   - Důkaz: ať G_1 generuje Σ^*.
4. Je $L(G_1) = L(R)$?
   - Důkaz: za R zvolíme Σ^*.
5. Je $L(G_1) \subseteq L(G_2)$?
   - Důkaz: ať G_1 generuje Σ^*.
6. Je $L(R) \subseteq L(G_1)$?
   - Důkaz: za R zvolíme Σ^*. $\square$

- Poznámka: $L(G) \subseteq L(R)$ je algoritmicky rozhodnutelné $L(G) \subseteq L(R) \Leftrightarrow L(G) \cap \overline{L(R)} = \emptyset$ a zároveň $(L(G) \cap \overline{L(R)})$ je CFL (uzavřenost operací)

Shrnutí

Popis nekonečných objektů konečnými prostředky

- regulární jazyky
  - konečné automaty (NFA, 2FA)
  - Nerode (rozklad), Kleene (elementární operace), pumpování
- bezkontextové jazyky
  - zásobníkové automaty (DPDA$\neq$ PDA)
  - pumpování
- kontextové jazyky
  - lineárně omezené automaty
  - monotonie
- rekurzivně spočetné jazyky
  - Turingovy stroje
  - algoritmická nerozhodnutelnost

použití nejen pro práci s jazyky.

Časová složitost

Definition 14.1 (časová složitost)

Mějme Turingův stroj M, který zastaví na každém vstupu. **Časová složitost** M je funkce $f : \mathbb{N} \to \mathbb{N}$, kde $f(n)$ je maximální počet kroků výpočtu M nad vstupy délky n.

Definition 14.2 ((Asymptotická) horní hranice $O(g(n))$)

Mějme funkce $f, g : \mathbb{N} \to \mathbb{R}^+$. Říkáme, že $f(n) = O(g(n))$, pokud existují $c, n_0 \in \mathbb{N}^+$ taková, že:

$$\forall n \geq n_0 \text{ platí } f(n) \leq c \cdot g(n).$$

V takovém případě říkáme, že $g(n)$ je (asymptotická) **horní hranice** pro $f(n)$. (Slovo asymptotická vyjadřující ignorování prvních n_0 i konstanty c se zpravidla vynechává.)

Reálná čísla jsou tam kvůli logaritmu.

Definition 14.3 (třída časové složitosti)

Mějme funkci $t : \mathbb{N} \to \mathbb{R}^+$. Definujeme **třídu časové složitosti** $TIME(t(n))$ jakožto množinu všech jazyků, které jsou rozhodnutelné turingovým strojem v čase $O(t(n))$.

Lemma

Mějme funkci $t : \mathbb{N} \to \mathbb{R}^+$, $t(n) \geq n$. Každý vícepáskový Turingův stroj s časem $t(n)$ má jednopáskový ekvivalent $O(t^2(n))$.

Definition 14.4 (doba běhu nedeterministického TM)

Mějme **nedeterministický** Turingův stroj N, který zastaví na každém vstupu. **Doba běhu** N je funkce $f : \mathbb{N} \to \mathbb{N}$, kde $f(n)$ je maximální počet kroků který N potřebuje v jakékoli větvi výpočtu nad jakýmkoli vstupem délky n.

Lemma

Mějme funkci $t : \mathbb{N} \to \mathbb{R}^+$, $t(n) \geq n$. Každý nedeterministický Turingův stroj s časem $t(n)$ má deterministický ekvivalent $2^{O(t(n))}$.

Definition 14.5 (třída P)

Definujeme P **(*PTIME*) třídu jazyků rozhodnutelných v polynomiálním čase** jednopáskovým deterministickým Turingovým strojem. Tedy:

$$P = \bigcup_k TIME(n^k).$$

Theorem 14.1 ($CFL \subseteq P$)

Každý bezkontextový jazyk patří to P.

- CYK algorithmu je polynomiální.

Verifikátory, třída *NP*

Definition 14.6 (Verifikátor)

Verifikátor jazyka L je algoritmus V, kde:

$$L = \{w \mid V \text{pro nějaké } c \in \Sigma^* \text{ přijímá } \langle w, c \rangle\}.$$

Časová složitost verifikátoru se měří pouze vzhledem k délce w, **polynomiální verifikátor** rozhoduje v čase polynomiálním vzhledem k $|w|$.
Jazyk L je **polynomiálně verifikovatelný**, pokud má polynomiální verifikátor.
Třída jazyků rozhodnutelných v polynomiálním čase NP je tvořena jazyky s polynomiálním verifikátorem.

Definition 14.7 (*NP*)

Mějme funkci $t : \mathbb{N} \to \mathbb{R}^+$. Definujeme třídu

$$NTIME(t(n)) = \{L \mid L \text{ jazyk rozhodnutelný nedeterminist. TM v čase } O(t(n)).\}$$

Třída *NP*

Theorem 14.2

$$NP = \bigcup_k NTIME(n^k).$$

- Idea důkazu: převedeme verifikátor na NTM a opačně.
- NTM uhodne certifikát a simuluje verifikátor.
- Verifikátor bere přijímající větev NTM jakožto certifikát.

verif. $\Rightarrow \bigcup$.

- Předpokládáme $L \in NP$.
- Hledáme nedeterministický TM M.
- Vezmeme verifikátor V z definice NP. Nechť rozhoduje L v čase n^k.
- M na vstupu w délky n:
  - Nedeterministicky uhodne řetězec c délky nanejvýš n^k.
  - Spustí V na vstupu $\langle w, c \rangle$.
  - Pokud V přijme, M také přijme, jinak nepřijímá.

$\square$

$\bigcup \Rightarrow$ verifikátor .

- Předpokládáme $L = L(M)$ rozhodnutelný nějakým NTM M v polynomiálním čase.
- Hledáme verifikátor V.
- V na vstupu $\langle w, c \rangle$:
  - Simuluje M na vstupu w, v bodech větvení vybere větev podle c.
  - Pokud tato větev NTM přijme, V přijme, jinak nepřijímá.

$\square$

Převoditelnost v polynomiálním čase

Definition 14.8 (polynomiálně vyčíslitelná funkce)

Funkci $f : \Sigma^* \to \Sigma^*$ nazveme **polynomiálně vyčíslitelnou**, pokud existuje Turingův stroj M, který pro každý vstup w v polynomiálním čase zastaví s $f(w)$ na pásce.
Jazyk A je **převoditelný v polynomiálním čase** na jazyk B, $A \leq_P B$, pokud existuje funkce $f : \Sigma^* \to \Sigma^*$ vyčíslitelná v polynomiálním čase a pro každé $w \in \Sigma^*$

$$w \in A \Leftrightarrow f(w) \in B.$$

Funkci f pak nazýváme **polynomiální redukci A do B**.

Definition 14.9 (NP úplnost)

Jazyk B je **NP úplný**, pokud je NP a každý jazyk $A \in NP$ je na B polynomiálně převeditelný.

Theorem 14.3

Pokud B je NP-úplný a $B \in P$, pak $P = NP$.

Proof.

Přímý důsledek definice polynomiální převoditelnosti a NP. $\square$

Theorem 14.4

Pokud B je NP-úplný a $B \leq_P C$ pro nějaké $C \in NP$, pak C je NP-úplný.

Proof.

Převod problému na B dále převedeme na C, stačí polynomiální čas. $\square$

Cook-Levin-ova věta

Definition 14.10 (SAT< 3SAT)

Formuli ϕ nazveme **3-cnf formule**, pokud je formule výrokové logiky v CNF, kde v každé klauzuli jsou právě tři literály.
Formule **je splnitelná**, pokud existuje takové ohodnocení výrokových proměnných, že je hodnota formule TRUE.
Problém **3SAT** je pro každou 3-cnf formuli rozhodnout, zda je splnitelná, tj.

$$3SAT = \{\phi \mid \phi \text{ je splnitelná 3-cnf formule}\}.$$

Problém **SAT** je pro každou booleovskou formuli v rozhodnout, zda je splnitelná, tj.

$$SAT = \{\phi \mid \phi \text{ je splnitelná formule}\}.$$

Theorem 14.5 (Cook-Levin-ova věta)

SAT je NP-úplný.

- idea důkazu úplnosti: převedeme výpočet Turingova stroje na SAT.

Proof

- SAT is NP
  - Nedeterministický TM uhodne správné ohodnocení a v polynomiálním čase ověří, je je pro něj formule pravdivá.
- SAT je NP-úplný
  - Vezmeme libovolný $L \in NP$
  - nechť M je nedeterministický TM který rozhoduje jazyk L v čase $n^k - 3$ pro nějaké k.
  - Vytvoříme tabulku (tableau) $n^k \times n^k$, každý řádek odpovídá konfiguraci M na vstupu w
  - můžeme předpokládat (opatřit) každou konfiguraci ohraničenou zarážkami $\#$.

$\#$	q_0	w_1	w_2	$\ldots$	w_n	$\_$	$\_$	$\ldots$	$\#$
$\#$									$\#$
$\#$	$\vdots$								$\#$
$\#$	$\vdots$								$\#$
$\#$									$\#$

  - Výpočet budeme skládat po okýnkách 2×3.

Proof:

- Vybraná dovolená okénka, $(a, b, c, d \in \Gamma)$

$\delta(q_1, b) \ni (q_2, c, L)$

a	q_1	b
q_2	a	c

$\delta(q_1, b) \ni (q_2, c, R)$

a	q_1	b
a	c	q_2

$\delta(q_1, b) \ni (q_2, c, R)$

d	a	q_1
d	a	c

přenos beze změny

#	a	b
#	a	b

$\delta(\_, \_) \ni (q_2, \_, L)$

a	b	c
a	b	q_2

$\delta(\_, \_) \ni (\_, c, L)$

a	b	d
c	b	d

- Přenos beze změny všude, kde není v okolí stav (hlava)
- na každém řádku nejvýše jeden stav
- okénko musí být částí povoleného přechodu
- rozbor technický, udělali za nás jiní.
- **Tvrzení:** Pokud je první řádek tabulky počáteční konfigurace a každé okénko je legální, pak každý řádek odpovídá legální konfiguraci dosažitelné jedním krokem z předchozího řádku.
  - V horním řádku není stav, pak se prostřední symbol musí opsat beze změny.
  - V horním řádku stav vprostřed: okénko garantuje korektnost přepisu obou stran.

$\square$

- Z tabulky vytvoříme formuli $\phi = \phi_{cell} \;\&\; \phi_{start} \;\&\; \phi_{move} \;\&\; \phi_{accept}$.
- pro každé políčko tabulky (i,j) a písmeno $a \in \Gamma \cup Q \cup \{\#\}$ vytvoříme výrokovou proměnnou $x_{i,j,a}$

$$\phi_{cell} = \bigwedge_{1\le i,j\le n^k} \left[\left(\bigvee_{a\in\Gamma\cup Q\cup\{\#\}} x_{i,j,a} \right) \& \bigwedge_{s\neq t\in\Gamma\cup Q\cup\{\#\}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}\right) \right] \quad \#\text{právě jedno}$$

$$\phi_{start} = x_{1,1,\#} \& x_{1,2,q_0} \& x_{1,3,w_1} \& x_{1,4,w_2} \& \dots \& x_{1,n+2,w_n} \& x_{1,n+3,\_} \& \dots \& x_{1,n^k,\#}$$

$$\phi_{accept} = \bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$$

- Celková formule ϕ_{move} bude konjunkce, že každé okénko s horním středem na pozici i,j je legální

$$\phi_{move} = \bigwedge_{1\le i<n^k, 1<j<n^k} \phi_{i,j} \qquad \#\text{ okénko } (i,j) \text{ je legální}$$

- legálnost okénka (i,j) zajistíme disjunkcí legálních okének

$$\phi_{i,j} = \bigvee_{(a_1,\dots,a_6)\in LEGAL} \left(x_{i,j-1,a_1} \& x_{i,j,a_2} \& x_{i,j+1,a_3} \& x_{i+1,j-1,a_4} \& x_{i+1,j,a_5} \& x_{i+1,j+1,a_6}\right)$$

- **Tvrzení:** Převod má polynomiální složitost, konkrétně $O(n^{2k} \log n)$.
  - $\phi_{cell} \in O(n^{2k})$, procházíme dvojice buněk
  - $\phi_{start} \in O(n^2)$, procházíme první řádek
  - $\phi_{move}, \phi_{accept} \in O(n^{2k})$,
    - $|\delta|$ je pro nás konstanta, protože nezávisí na délce vstupu
    - počet buněk n^{2k}, pro každou konstantní velikost formule.
- pro šťouraly $\log n$ pro zápis indexu proměnných, jehož délka závisí na n.

$\square$

co-NP, Tautologičnost

Definition 14.11

Jazyk $L \subseteq \Sigma^*$ patří do třídy **co-NP** právě když jeho doplněk $\Sigma^* - L$ patří do NP.

- P je částí NP i co-NP.
- Domníváme se, že NP-úplné problémy nejsou v co-NP.
  - pokud $P = NP$, tak jsou.

Definition 14.12 (tautologičnost)

Problém, zda je daná formule výrokové logiky, nazýváme **tautologičnost TAUT**.

Theorem 14.6

Problém tautologičnosti TAUT je co-NP.

- Důkaz z pozorování, že doplněk TAUT je SAT a SAT je v NP.
- Doplněk SAT je otázka, jestli negace dané formule je tautologie.
- Doplněk SAT je co-NP.
- SAT rozhoduje všechny formule, tedy i jejich negace.

Prostorová složitost

- Podobně jako časovou složitost měříme prostor potřebný k výpočtu.
- Turingův stroj je jednoduchý a dost podobný reálným počítačům, proto je často používán k definici tříd složitosti.

Definition 14.13 (prostorová složitost)

Pro deterministický Turingův stroj M, který zastaví na každém vstupu, je **prostorová složitost** M funkce $f : \mathbb{N} \to \mathbb{N}$, kde $f(n)$ je maximální počet buněk pásky, které M přečte při jakémkoli vstupu délky n.
Pro nedeterministický Turingův stroj M, který všechny větve výpočtu zastaví na každém vstupu, je **prostorová složitost** M je funkce $f : \mathbb{N} \to \mathbb{N}$, kde $f(n)$ je maximální počet buněk pásky, které M přečte při jakémkolivstupu délky n na libovolné větvi výpočtu.

Třídy prostorové složitosti

Definition 14.14 (třídy prostorové složitosti)

Mějme funkci $t : \mathbb{N} \to \mathbb{R}^+$. Definujeme **třídy prostorové složitosti** $SPACE(f(n))$ a $NSPACE(f(n))$:

- $SPACE(f(n)) =$
  $\{L \mid L$je jazyk rozhodnutelný v prostoru $O(f(n))$ deterministickým TM$\}$,
- $NSPACE(f(n)) =$
  $\{L \mid L$je jazyk rozhodnutelný v prostoru $O(f(n))$ nedeterministickým TM$\}$.

Theorem 14.7

$$P \subseteq NP \subseteq PSPACE \subseteq NPSPACE \subseteq EXPTIME = \bigcup_k TIME(2^{n^k}).$$

Přehled kapitol

1. Úvod, Iterační lemma pro reg. jazyky
2. Redukovaný DFA a ekvivalence automatů, stavů
3. Nedeterministické ϵ–NFA, Operace zachovávající regularitu
4. Regulární výrazy, Kleeneova věta, Substituce, Homomorfizmus
5. Dvousměrné FA, Mealy a Moore stroje
6. Gramatiky, Chomského hierarchie, víceznačnost
7. Chomského NF, Pumping Lemma pro CFL
8. CYK – náležení do CFL
9. Zásobníkové automaty, Deterministické PDA
10. Uzávěrové vlastnosti, Dykovy jazyky
11. Turingův stroj, rozšíření
12. Lineárně omezené automaty, Univerzální TM, Diagonální jazyk
13. Nerozhodnutelné problémy, Postův korespondenční p.
14. Časová složitost