Dnešní téma

Automaty a gramatiky TIN071

Marta Vomlelová

MS 303

Organizační záležitosti

- Přednáška:
  - moodle https://dl1.cuni.cz/course/view.php?id=5119
    - ⋆ 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: Středa 14:00-15:00 v S303.
  - Jindy bez záruky. Na maily se snažím odpovídat, na dotazy po přednášce také.

Organizační záležitosti

- Přednáška:
  - moodle https://dl1.cuni.cz/course/view.php?id=5119
    - ⋆ 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: Středa 14:00-15:00 v S303.
  - Jindy bez záruky. Na maily se snažím odpovídat, na dotazy po přednášce také.

Organizační záležitosti

- Přednáška:
  - moodle https://dl1.cuni.cz/course/view.php?id=5119
    - ⋆ 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: Středa 14:00-15:00 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 písemné přípravy a ústní části.
- Po vylosování otázek můžete být vyzván/a k rámcové odpovědi bez přípravy. Nevadí, že odpověď je v takovém případě nepřesná; jde o monitorování procesu promýšlení, tj. prevence odpovědi nadiktované pomocí nedovolených postupů.
- **Písemná příprava** 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 písemné přípravy 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 písemné přípravy, zpravidla budete dotázáni na vysvětlení-zdůvodnění-příklady k tvrzením v písemné části. Ústní část může být doplněna otázkou v rozsahu sylabu přednášky s písemnou přípravou 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*

$\Rightarrow$ moodle https://dl1.cuni.cz/course/view.php?id=5119

- textová verze slajdů
- korekce, jsou-li potřeba
- moodle testy (které ale netestují zdůvodnění a důkazy).

$\Rightarrow$ 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.

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.

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.

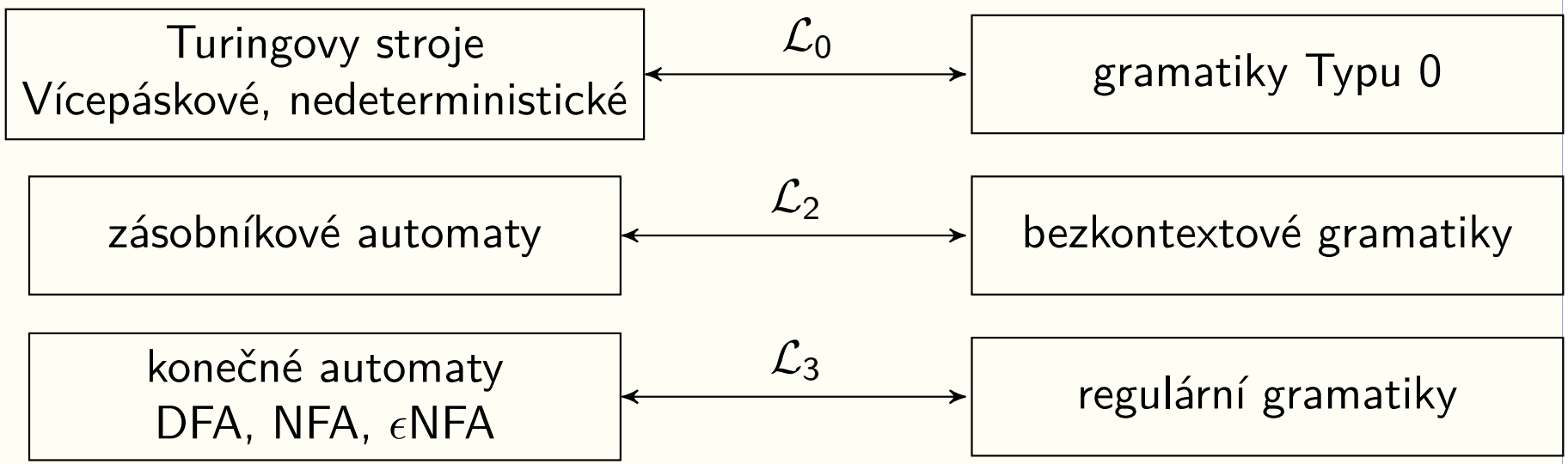
Pohled do historie

- 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 $P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$.

Automaty a gramatiky – dva způsoby popisu



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
  - transformace programů pomocí Abstract syntax trees.

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
  - transformace programů pomocí Abstract syntax trees.

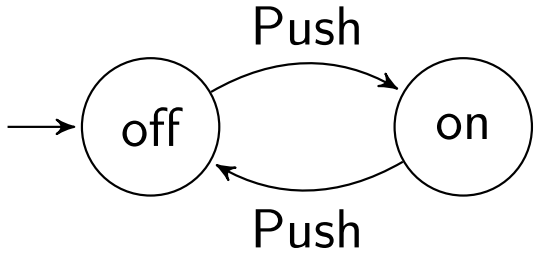
Praktické využití

- 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
  - transformace programů pomocí Abstract syntax trees.

Jednoduché příklady konečných automatů

- Návrh a verifikace integrovaných obvodů.

Konečný automat modelující spínač on/off .



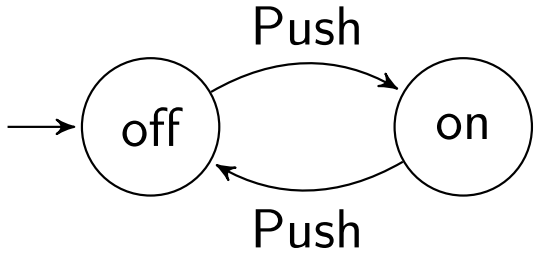
- Lexikální analýza

Konečný automat rozpoznávající slovo `then`.

Jednoduché příklady konečných automatů

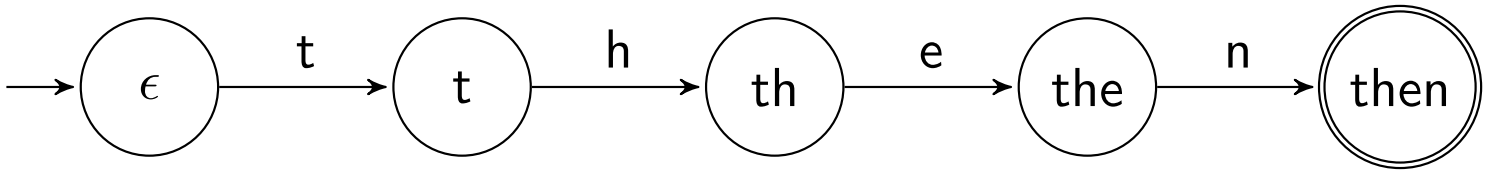
- Návrh a verifikace integrovaných obvodů.

Konečný automat modelující spínač on/off .



- Lexikální analýza

Konečný automat rozpoznávající slovo `then`.



Definice 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 (na hraně seznam $\subset \Sigma$) 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 *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. 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 $\forall s \in \Sigma$: $\delta(final, s) = final$.

Definice 1.2: 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 (na hraně seznam $\subset \Sigma$) 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'.

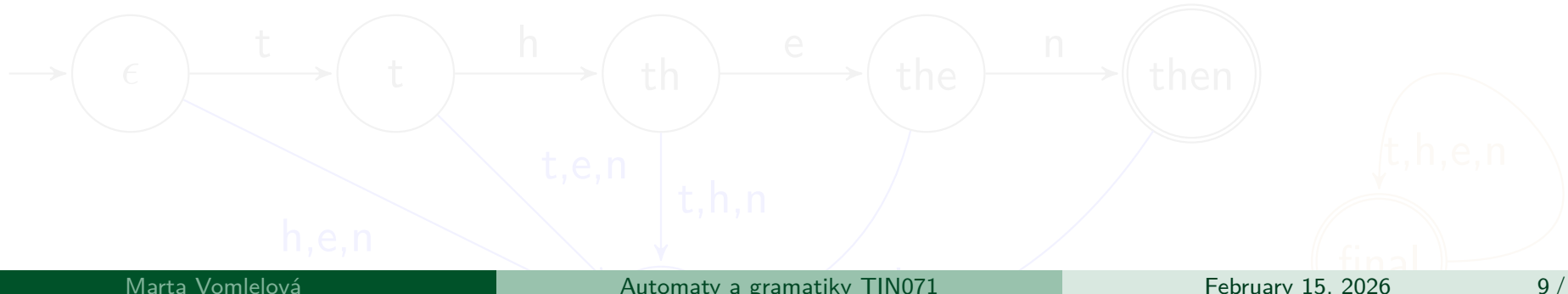
Úmluva: Pokud pro některou dvojici stavu a písmene není definovaný přechod, přidáme nový stav *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. Pokud je množina F prázdná, a je vyžadovaná neprázdná, přidáme do ní i Q nový stav *final* do kterého vedou jen přechody z něj samého $\forall s \in \Sigma$: $\delta(final, s) = final$.

Definice 1.3: 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 (na hraně seznam $\subset \Sigma$) 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 *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. 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 $\forall s \in \Sigma$: $\delta(final, s) = final$.



Definice 1.4: 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 (na hraně seznam $\subset \Sigma$) 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 *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. 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 $\forall s \in \Sigma$: $\delta(final, s) = final$.

Definice 1.5: 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 (na hraně seznam $\subset \Sigma$) 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 *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. 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 $\forall s \in \Sigma$: $\delta(final, s) = final$.

Definice 1.6: 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 (na hraně seznam $\subset \Sigma$) 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'.

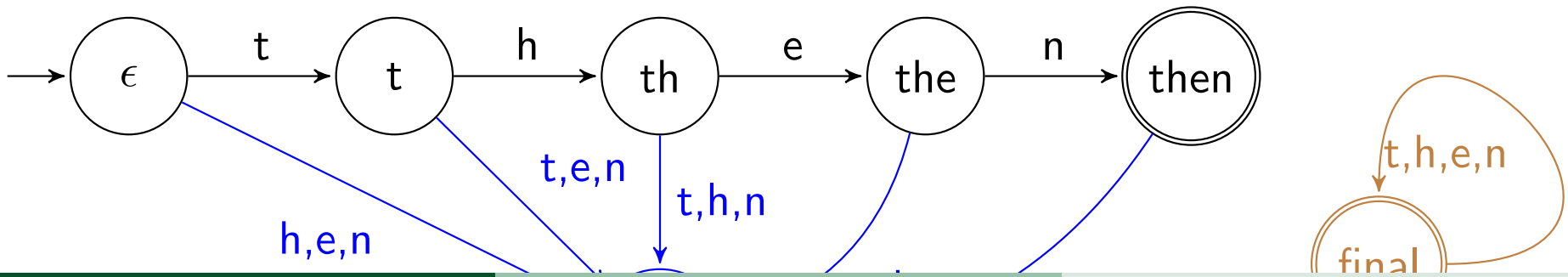
Úmluva: Pokud pro některou dvojici stavu a písmene není definovaný přechod, přidáme nový stav *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. Pokud je množina F prázdná, a je vyžadovaná neprázdná, přidáme do ní i Q nový stav *final* do kterého vedou jen přechody z něj samého $\forall s \in \Sigma$: $\delta(final, s) = final$.

Definice 1.7: 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 (na hraně seznam $\subset \Sigma$) 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 *dead* a přechodovou funkci doplníme na totální přidáním šipek do *dead*. 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 $\forall s \in \Sigma$: $\delta(final, s) = final$.

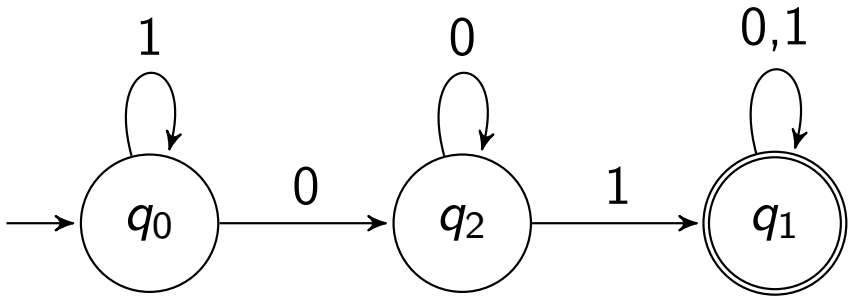


Popis konečného automatu

Příklad 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

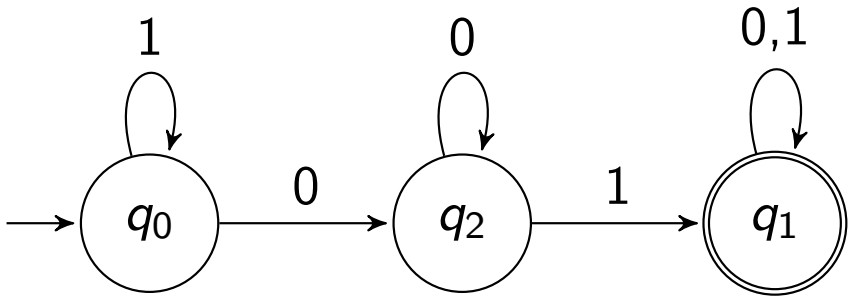
δ	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Popis konečného automatu

Příklad 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

Definice 1.8: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.9: operace zřetězení, mocnina, délka slova

Nad slovy Σ^* definujeme operace:

Abeceda, slova, jazyky

Definice 1.10: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.11: operace zřetězení, mocnina, délka slova

Nad slovy Σ^* definujeme operace:

Abeceda, slova, jazyky

Definice 1.12: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.13: operace zřetězení, mocnina, délka slova

Nad slovy Σ^* definujeme operace:

Abeceda, slova, jazyky

Definice 1.14: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.15: operace zřetězení, mocnina, délka slova

Nad slovy Σ^* definujeme operace:

- zřetězení slov $u.v$ nebo uv
- mocnina (počet opakování) u^k (z $u^0 = \epsilon$, $u^k = u.u^{k-1}$ pro $k > 0$)
- délka slova $|u|$ ($|\epsilon| = 0$, $|ua| = |u| + 1$)
- počet výskytů písmen slova $|u|_a$ ($|\epsilon|_a = 0$, $|ua|_a = |u|_a + 1$, $|ub|_a = |u|_a$ pro $a \neq b$)

Abeceda, slova, jazyky

Definice 1.16: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.17: 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 $|u|$ ($|\epsilon| = 0$, $|au| = 1 + |u|$)
- počet výskytů písmen slova $|u|_a$ ($|\epsilon|_a = 0$, $|ua|_a = 1 + |u|_a$, $|ub|_a = |u|_a$)

Abeceda, slova, jazyky

Definice 1.18: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.19: 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 $|u|_s$ ($|zmrzlina|_z = 2$).

Abeceda, slova, jazyky

Definice 1.20: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.21: 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 $|u|_s$ ($|zmrzlina|_z = 2$).

Abeceda, slova, jazyky

Definice 1.22: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.23: 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 $|u|_s$ ($|zmrzlina|_z = 2$).

Abeceda, slova, jazyky

Definice 1.24: Slovo, ϵ,λ,Σ^*,Σ^+,jazyk

Mějme neprázdnou množinu symbolů Σ.

- **Slovo, řetězec** 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.

Definice 1.25: 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

Definice 1.26: rozšířená přechodová funkce

Mějme přechodovou funkci $\delta : Q \times \Sigma \to Q$.
Rozšířenou přechodovou funkci δ^*: $Q \times \Sigma^* \to Q$ (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^*.

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

Rozšířená přechodová funkce

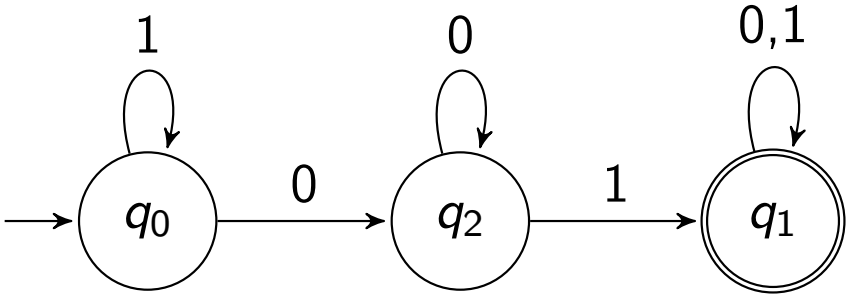
Definice 1.27: rozšířená přechodová funkce

Mějme přechodovou funkci $\delta : Q \times \Sigma \to Q$.
Rozšířenou přechodovou funkci δ^*: $Q \times \Sigma^* \to Q$ (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

Definice 1.28: jazyky rozpoznatelné DFA, 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**.

Příklad 1.2 (regulární jazyk)

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

Jazyky rozpoznatelné konečnými automaty

Definice 1.29: jazyky rozpoznatelné DFA, 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**.

Příklad 1.2 (regulární jazyk)

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

Jazyky rozpoznatelné konečnými automaty

Definice 1.30: jazyky rozpoznatelné DFA, 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**.

Příklad 1.2 (regulární jazyk)

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

Jazyky rozpoznatelné konečnými automaty

Definice 1.31: jazyky rozpoznatelné DFA, 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**.

Příklad 1.2 (regulární jazyk)

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

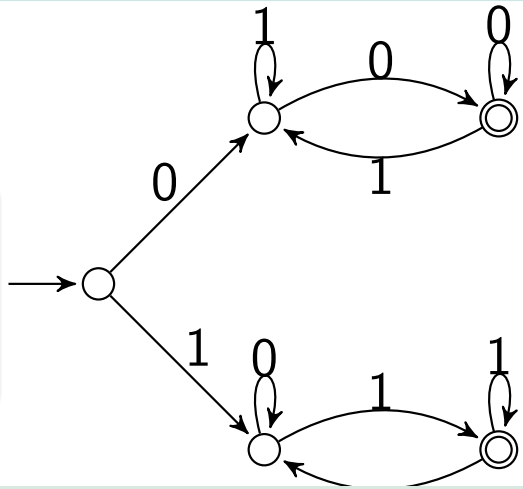
Jazyky rozpoznatelné konečnými automaty

Definice 1.32: jazyky rozpoznatelné DFA, 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**.

Příklad 1.2 (regulární jazyk)

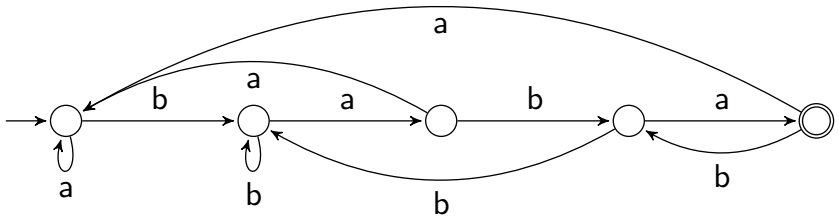
- $L = \{w \mid w = xux, w \in \{0,1\}^*, x \in \{0,1\}, u \in \{0,1\}^*\}$.



Příklady regulárních jazyků

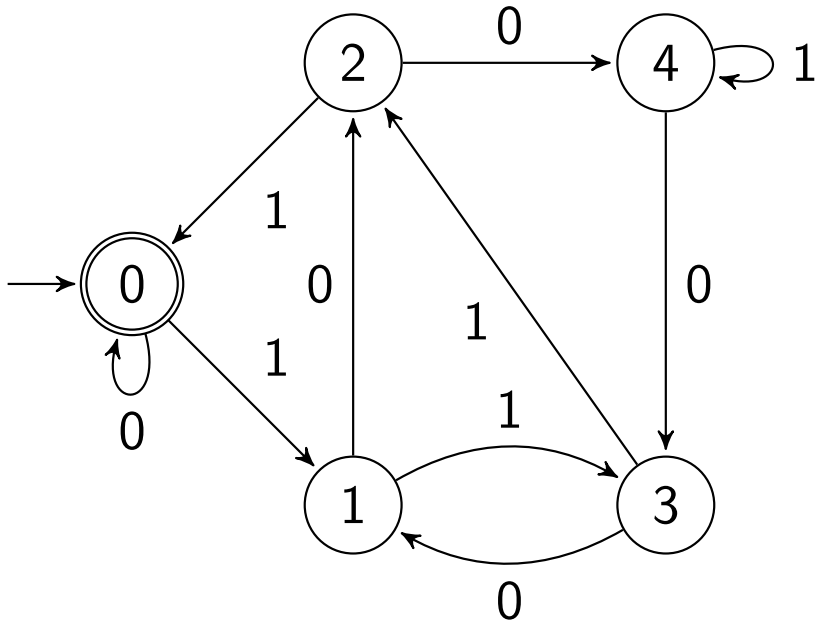
Příklad 1.3 (regulární jazyk)

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



Příklad 1.4 (regulární jazyk)

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



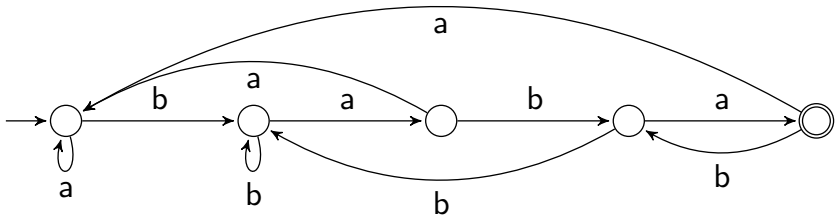
Příklad 1.5 (!NEregulární jazyk)

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

Příklady regulárních jazyků

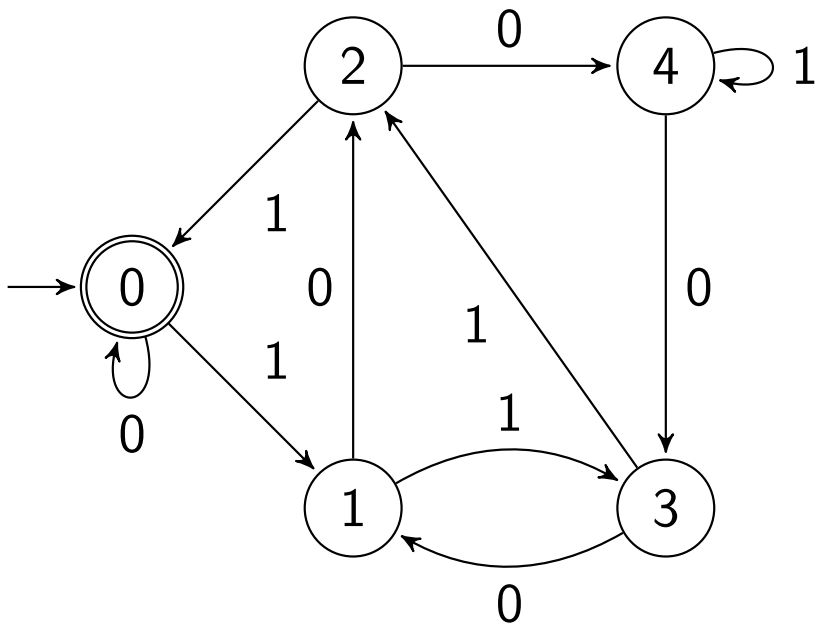
Příklad 1.3 (regulární jazyk)

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



Příklad 1.4 (regulární jazyk)

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



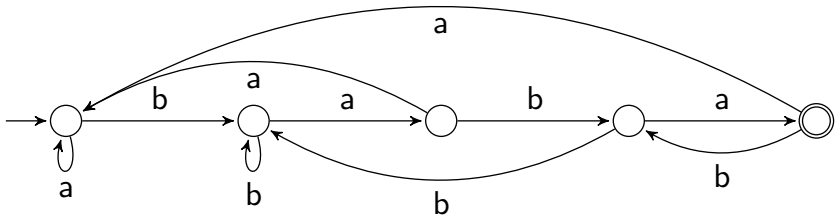
Příklad 1.5 (!NEregulární jazyk)

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

Příklady regulárních jazyků

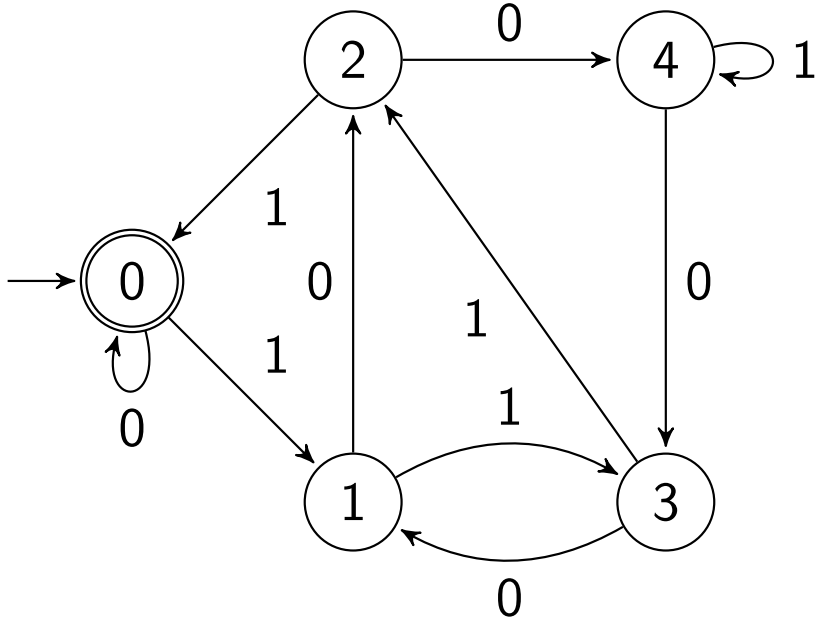
Příklad 1.3 (regulární jazyk)

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



Příklad 1.4 (regulární jazyk)

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



Příklad 1.5 (!NEregulární jazyk)

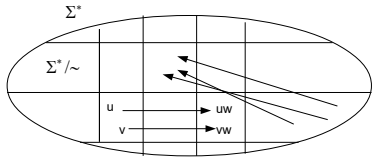
- $L = \{0^n 1^n | w \in \{0,1\}^*, n \in \mathbb{N}\}$ NENÍ regulání jazyk.

Kongruence, Myhill–Nerodova věta

Definice 1.33: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

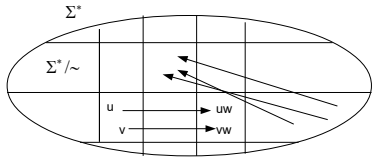
- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence, pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{|\cdot|}$ 'mají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

Kongruence, Myhill–Nerodova věta

Definice 1.34: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

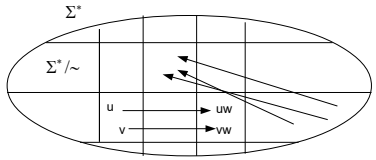
- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence, pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{|\cdot|}$ 'mají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

Kongruence, Myhill–Nerodova věta

Definice 1.35: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

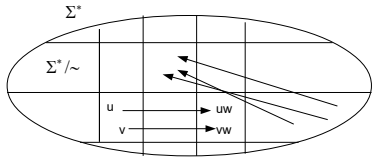
- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence,
  - pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{|\cdot|}$ 'mají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

Kongruence, Myhill–Nerodova věta

Definice 1.36: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

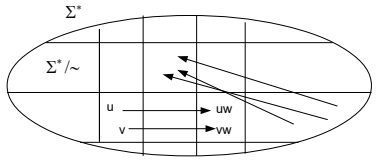
- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence,
  - pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{|\cdot|}$ 'mají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

Kongruence, Myhill–Nerodova věta

Definice 1.37: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

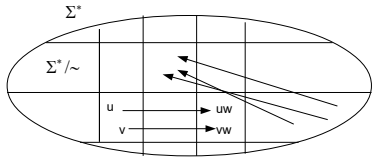
- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence,
  - pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{||}$ 'mají stejný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

Kongruence, Myhill–Nerodova věta

Definice 1.38: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

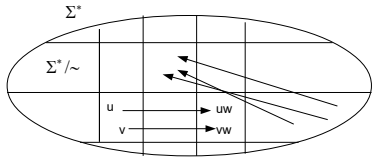
- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence,
  - pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' je ekvivalence, $aa \sim_{fl} bb$, ale $aaa \not\sim_{fl} bba$, tedy není pravá kongruence.
- Relace $\sim_{||}$ 'mají stejný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

Kongruence, Myhill–Nerodova věta

Definice 1.39: 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. $[u]$.



Příklad 1.6 (Pravá kongruence)

- Relace $\sim_{end}$ 'končí stejným písmenem' je pravá kongruence,
  - pokud $ux \sim_{end} vx$, pak i $uxw \sim_{end} vxw$.
- Relace $\sim_{fl}$ 'první a poslední písmeno stejné' 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.

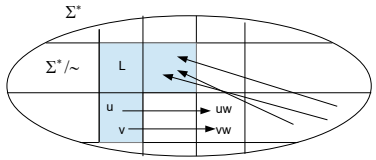
Myhill–Nerodova věta

Věta 1.1: 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 $\Sigma^*/\sim$.



Důkaz: 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, tranzitivní)

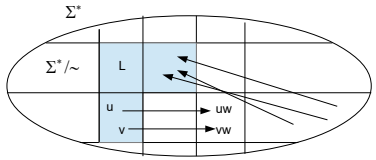
Myhill–Nerodova věta

Věta 1.2: 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 $\Sigma^*/\sim$.



Důkaz: 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)

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

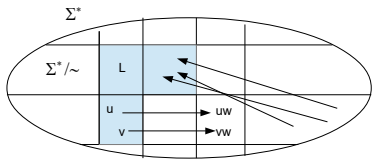
Myhill–Nerodova věta

Věta 1.3: 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 $\Sigma^*/\sim$.



Důkaz: 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)

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

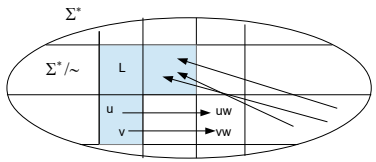
Myhill–Nerodova věta

Věta 1.4: 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 $\Sigma^*/\sim$.



Důkaz: 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)

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

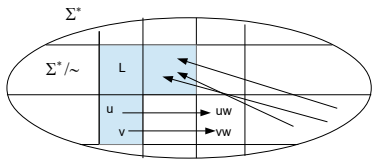
Myhill–Nerodova věta

Věta 1.5: 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 $\Sigma^*/\sim$.



Důkaz: 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 $\Sigma^*/\sim$.*

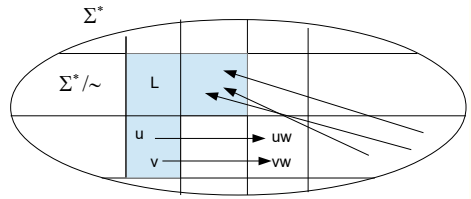
Důkaz: Důkaz Myhill–Nerodovy věty $\Leftarrow$

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).
- $L(A) = L$

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

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



□

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 $\Sigma^*/\sim$.*

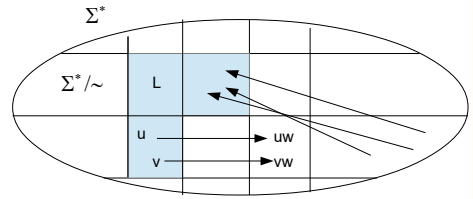
Důkaz: Důkaz Myhill–Nerodovy věty $\Leftarrow$

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

- abeceda automatu vezmeme Σ
- za stavy Q volíme třídy rozkladu $\Sigma^*/\sim$
- počáteční stav $q_0 = [\epsilon]$
- 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).
- $L(A) = L$

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

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



□

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 $\Sigma^*/\sim$.*

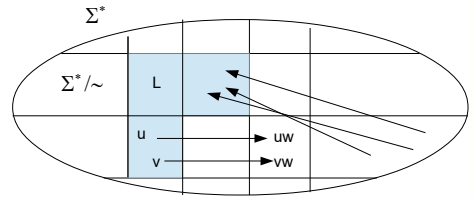
Důkaz: Důkaz Myhill–Nerodovy věty $\Leftarrow$

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).
- $L(A) = L$

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

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



□

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 $\Sigma^*/\sim$.*

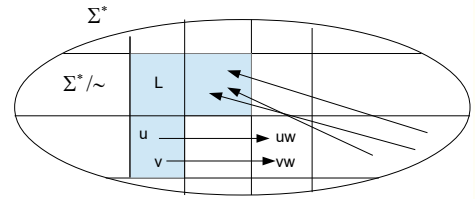
Důkaz: Důkaz Myhill–Nerodovy věty $\Leftarrow$

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).
- $L(A) = L$

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

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



□

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 $\Sigma^*/\sim$.*

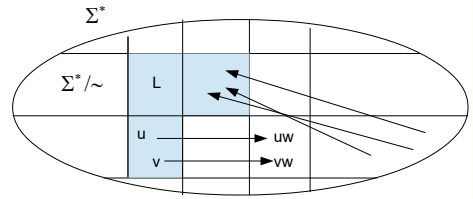
Důkaz: Důkaz Myhill–Nerodovy věty $\Leftarrow$

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).
- $L(A) = L$

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

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



□

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 $\Sigma^*/\sim$.*

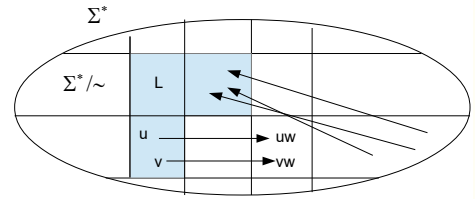
Důkaz: Důkaz Myhill–Nerodovy věty $\Leftarrow$

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).
- $L(A) = L$

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

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



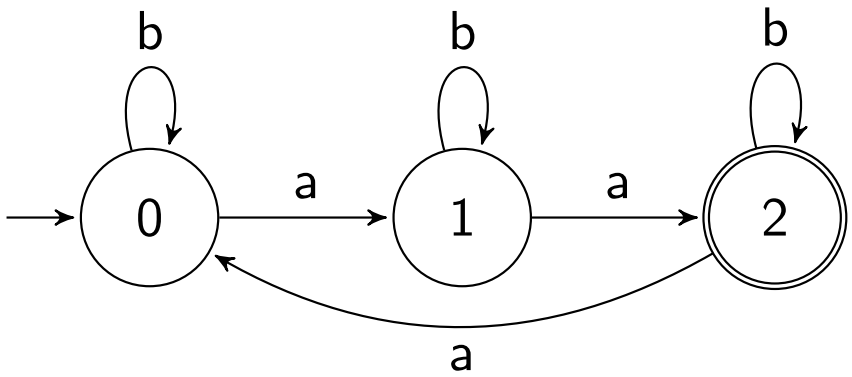
□

Použití Myhill–Nerodovy věty: Konstrukce automatů

Příklad 1.7

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.

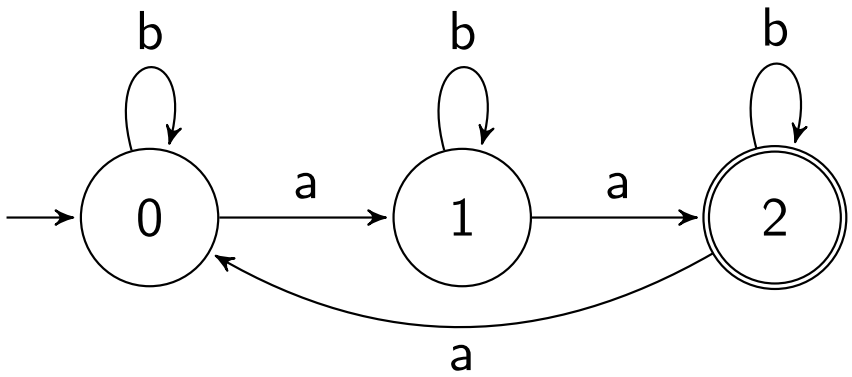


Použití Myhill–Nerodovy věty: Konstrukce automatů

Příklad 1.7

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.

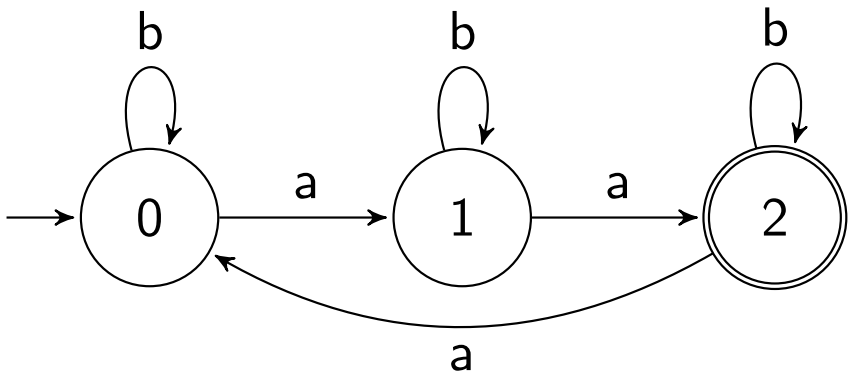


Použití Myhill–Nerodovy věty: Konstrukce automatů

Příklad 1.7

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.

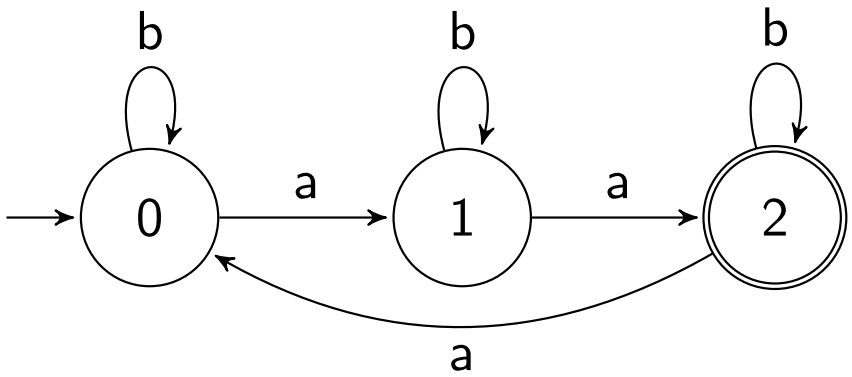


Použití Myhill–Nerodovy věty: Konstrukce automatů

Příklad 1.7

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.

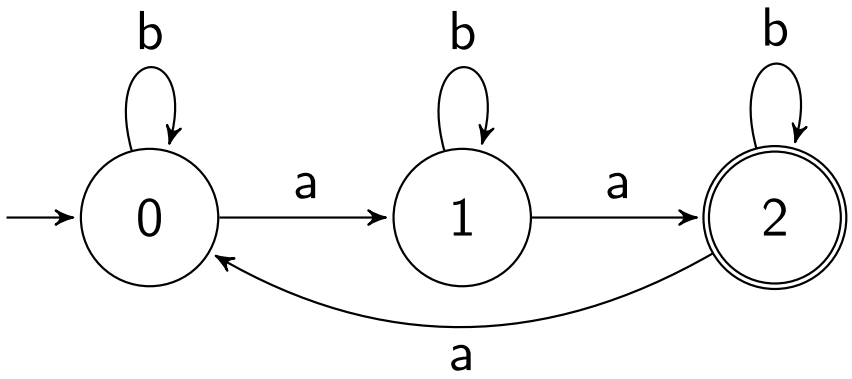


Použití Myhill–Nerodovy věty: Konstrukce automatů

Příklad 1.7

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.

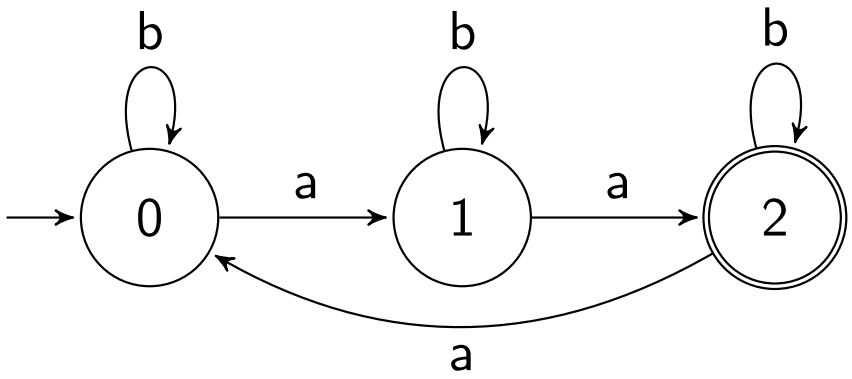


Použití Myhill–Nerodovy věty: Konstrukce automatů

Příklad 1.7

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.



Ne-regulární jazyk

Příklad 1.8 (Ne–regulární jazyk)

Jazyk $L_{a^+b^ic^i} = \{u | u = a^+b^ic^i \vee u = b^ic^j, + \in \mathbb{N}_{>0}, i, j \in \mathbb{N}_0\}$ není regulární (Myhill–Nerodova věta).

Důkaz: Jazyk $L_{a^+b^ic^i}$ není regulární

- Důkaz sporem: Předpokládejme, že L je regulární
- => pak existuje pravá kongruence $\sim_L$ konečného indexu m, L je sjednocení některých tříd $\Sigma^* / \sim_L$

Ne-regulární jazyk

Příklad 1.8 (Ne–regulární jazyk)

Jazyk $L_{a^+b^ic^i} = \{u | u = a^+b^ic^i \vee u = b^ic^j, + \in \mathbb{N}_{>0}, i, j \in \mathbb{N}_0\}$ není regulární (Myhill–Nerodova věta).

Důkaz: Jazyk $L_{a^+b^ic^i}$ není regulární

- Důkaz sporem: 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	$ab^ic^i \sim ab^jc^i$	$\sim$ je kongruence
spor	$ab^ic^i \in L$ & $ab^jc^i \notin L$	s 'L je sjednocení některých tříd Σ^* /

□

Ne-regulární jazyk

Příklad 1.8 (Ne–regulární jazyk)

Jazyk $L_{a^+b^ic^i} = \{u | u = a^+b^ic^i \vee u = b^ic^j, + \in \mathbb{N}_{>0}, i, j \in \mathbb{N}_0\}$ není regulární (Myhill–Nerodova věta).

Důkaz: Jazyk $L_{a^+b^ic^i}$ není regulární

- Důkaz sporem: 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	$ab^ic^i \sim ab^jc^i$	$\sim$ je kongruence
spor	$ab^ic^i \in L$ & $ab^jc^i \notin L$	s 'L je sjednocení některých tříd $\Sigma^*/$

□

Ne-regulární jazyk

Příklad 1.8 (Ne–regulární jazyk)

Jazyk $L_{a^+b^ic^i} = \{u | u = a^+b^ic^i \vee u = b^ic^j, + \in \mathbb{N}_{>0}, i, j \in \mathbb{N}_0\}$ není regulární (Myhill–Nerodova věta).

Důkaz: Jazyk $L_{a^+b^ic^i}$ není regulární

- Důkaz sporem: 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	$ab^ic^i \sim ab^jc^i$	$\sim$ je kongruence
spor	$ab^ic^i \in L$ & $ab^jc^i \notin L$	s 'L je sjednocení některých tříd $\Sigma^*/$

□

Ne-regulární jazyk

Příklad 1.8 (Ne–regulární jazyk)

Jazyk $L_{a^+b^ic^i} = \{u | u = a^+b^ic^i \vee u = b^ic^j, + \in \mathbb{N}_{>0}, i, j \in \mathbb{N}_0\}$ není regulární (Myhill–Nerodova věta).

Důkaz: Jazyk $L_{a^+b^ic^i}$ není regulární

- Důkaz sporem: 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

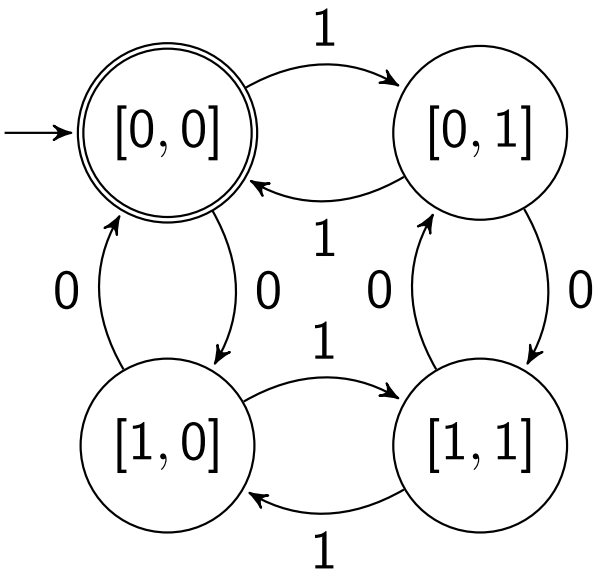
$i \neq j$	$ab^i \sim ab^j$	
přidáme c^i	$ab^ic^i \sim ab^jc^i$	$\sim$ je kongruence
spor	$ab^ic^i \in L$ & $ab^jc^i \notin L$	s 'L je sjednocení některých tříd $\Sigma^*/$

Příklad - 'součin' automatů

Příklad 1.9

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

- sudý počet 0
- a zároveň sudý počet 1.



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

Příklad (špatného) protokolu pro elektronický převod peněz

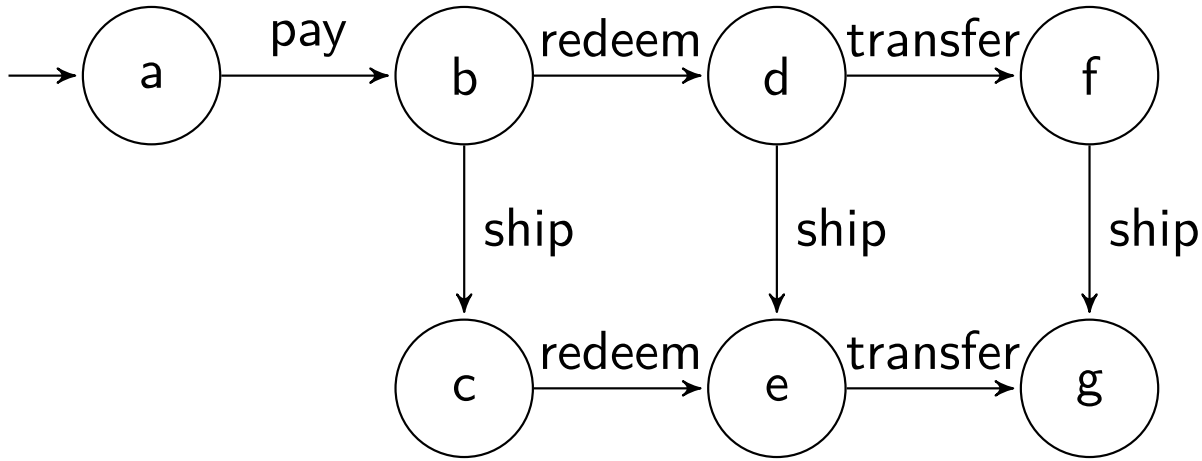
- Tři zúčastnění: zákazník, obchod, banka.
- Pro jednoduchost jen jedna platba (soubor 'money').

Příklad 1.10

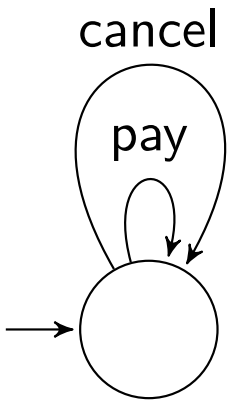
Zákazník poskytne obchodu číslo kreditní karty, obchod si vyžádá peníze od banky a pošle zboží zákazníkovi. Zákazník má možnost zablokovat kartu a žádat zrušení transakce.
Pět událostí:

- 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.

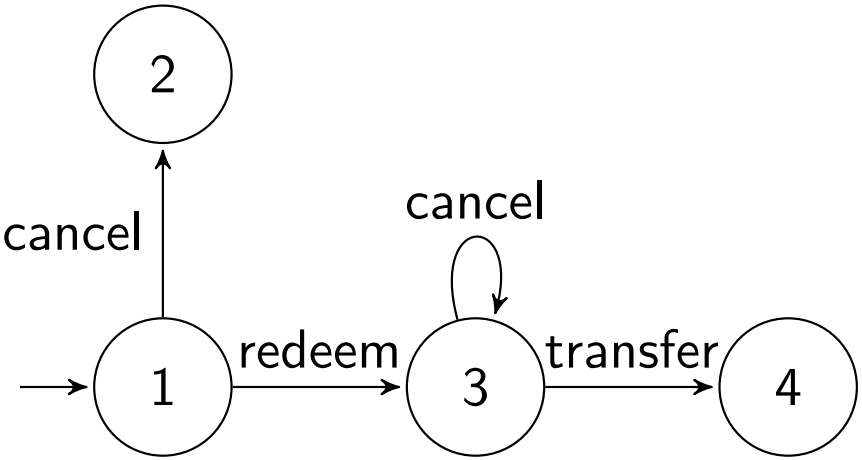
(Neúplný) konečný automat pro bankovní příklad



Obchod



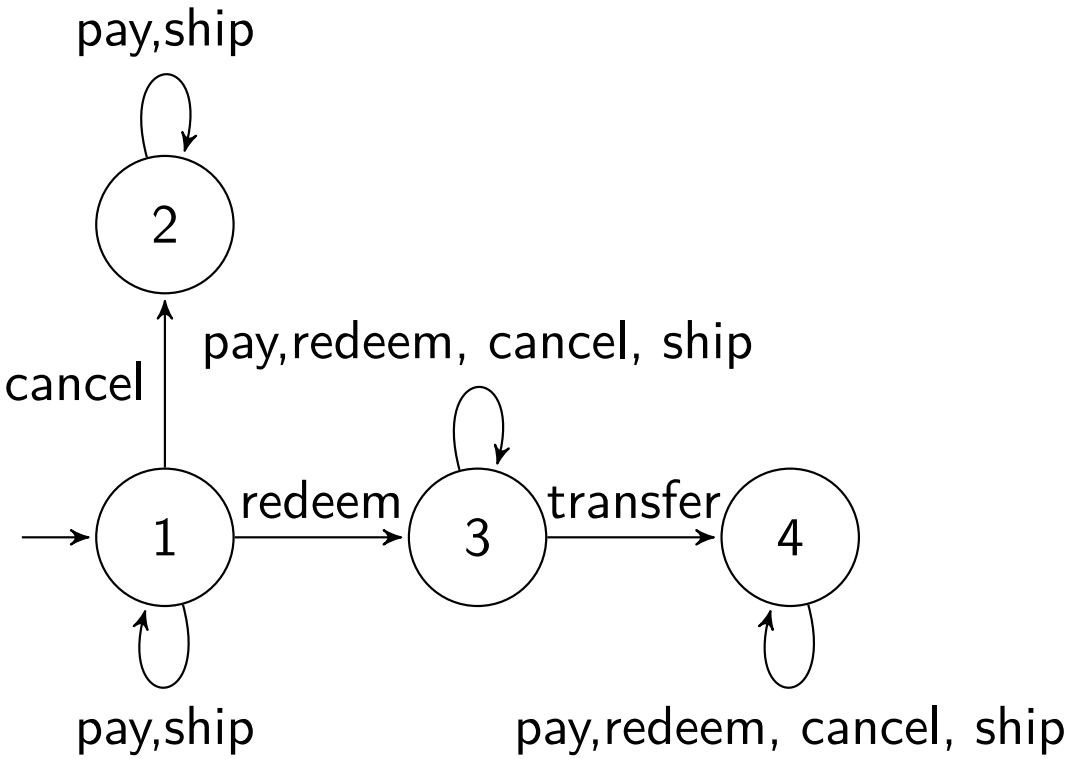
Zákazník



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.

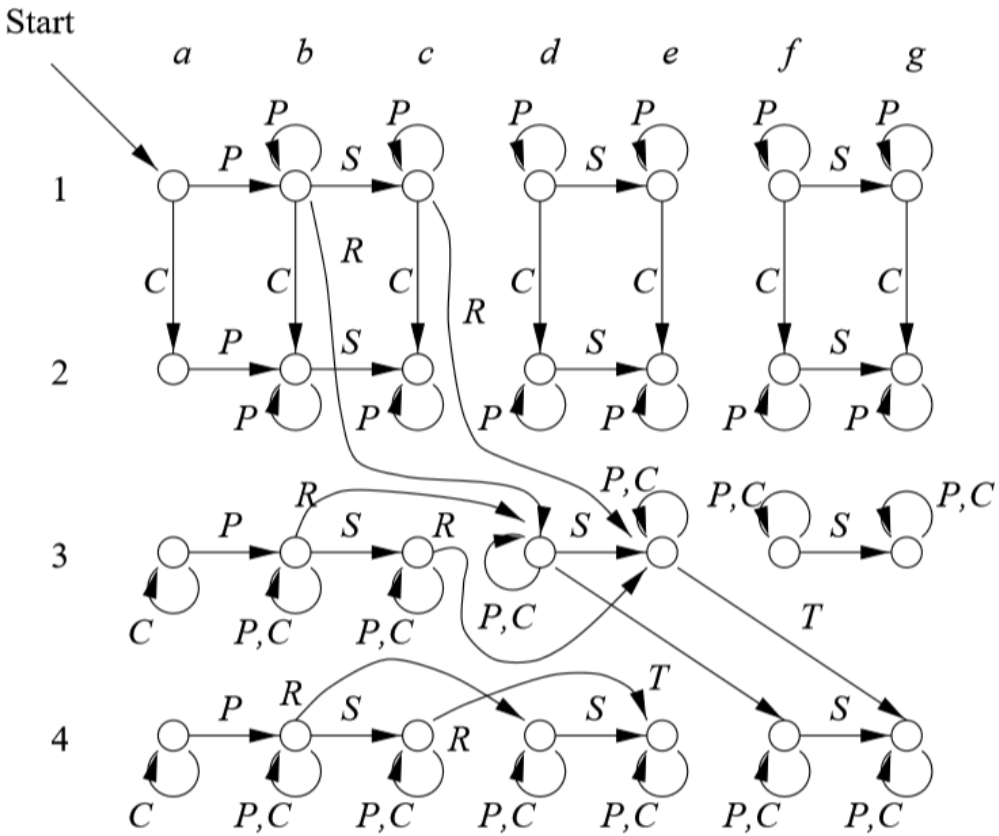


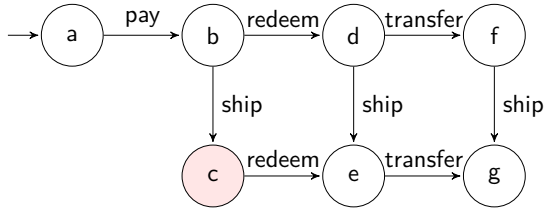
Úplnější automat pro banku.

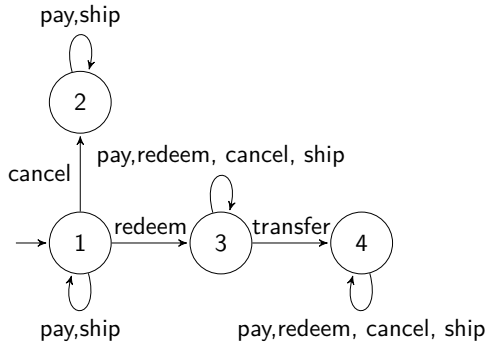
Součin automatů

- 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



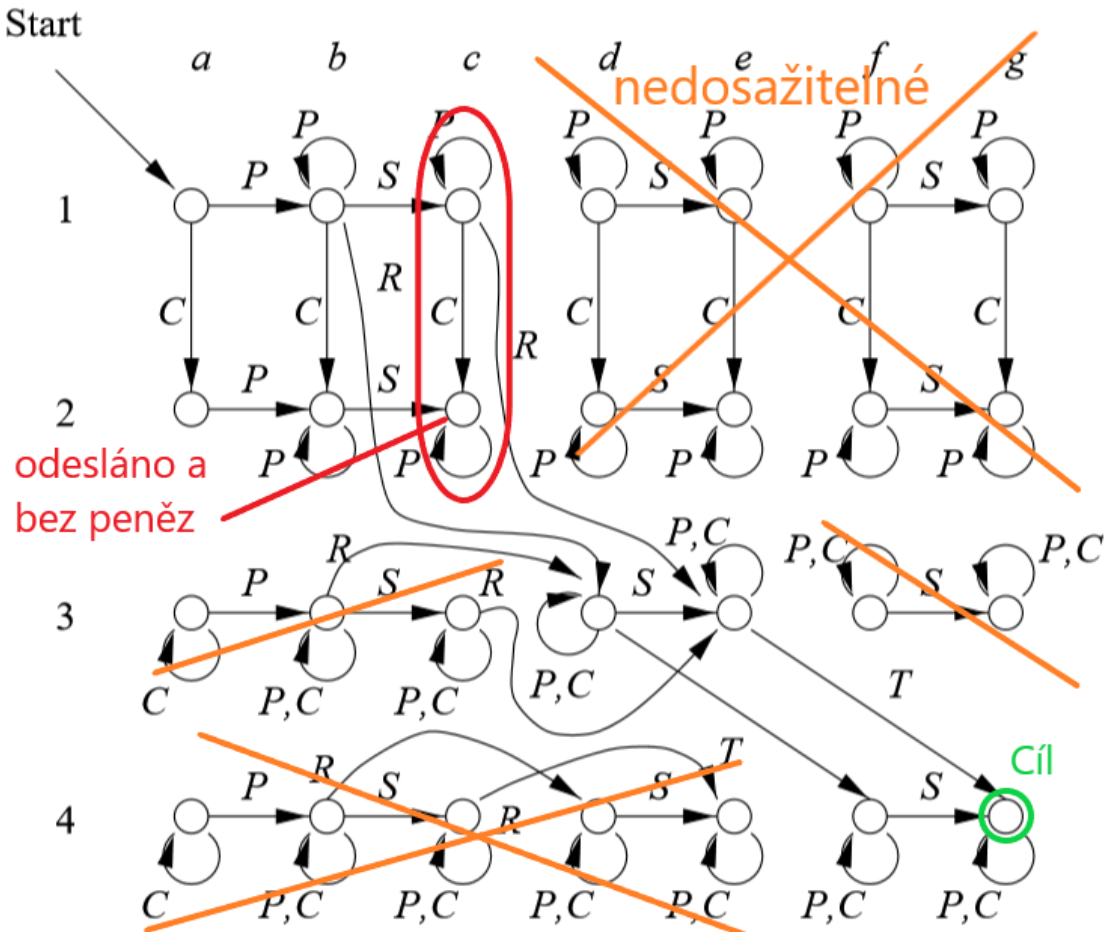


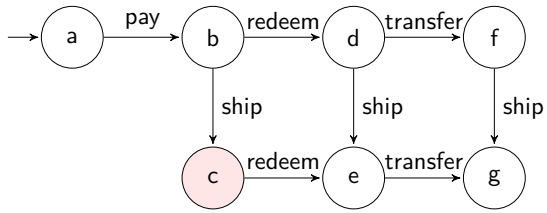


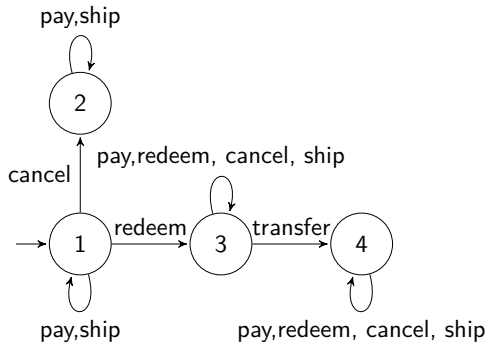
Součin automatů

- 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







Dnes jsme probrali

- Definice
  - deterministického konečného automatu $A = (Q, \Sigma, \delta, q_0, F)$
  - jazyka $L \subseteq \Sigma^*$
  - jazyka rozpoznávaného konečným automatem $L(A) = \{w \mid w \in \Sigma^* \ \& \ \delta^*(q_0, w) \in F\}$
- Myhill–Nerodova věta
- příklad důkazu ne–regulárnosti jazyka ($\{ab^i c^i | i \in \mathbb{N}\}$, $\{0^i 1^i | i \in \mathbb{N}\}$)
- příklady regulárních jazyků

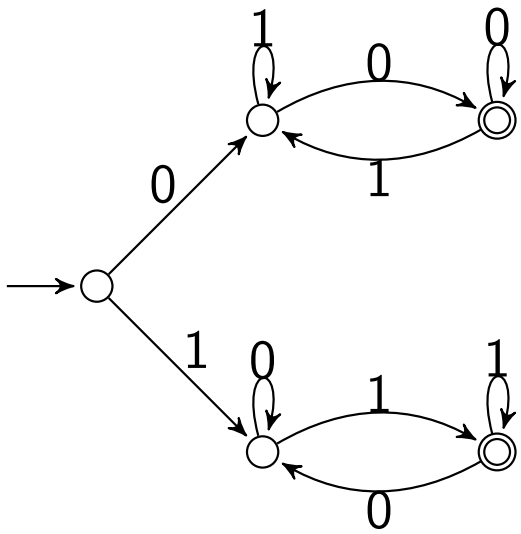
Iterační lemma, Redukovaný DFA, ϵ–NFA

Marta Vomlelová

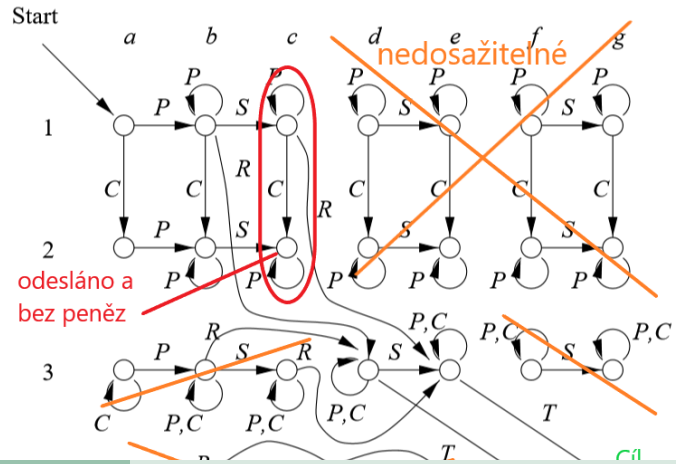
MS 303

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**.

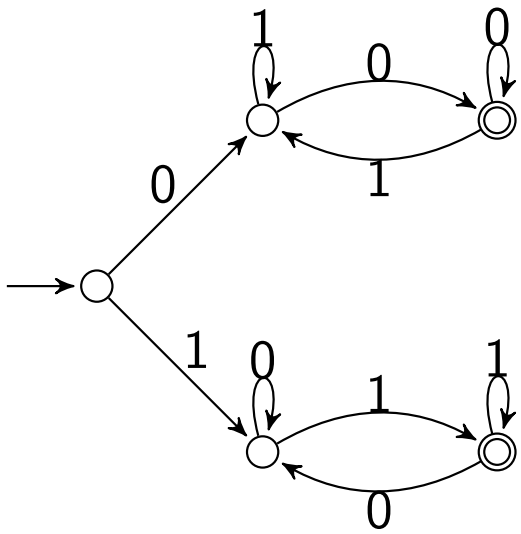


- Dnes budeme zkoumat:
  - Dá se do každého stavu dojít?
  - Je v přechodové funkci cyklus?
  - Jak automat redukovat?
  - Může být nedeterministický?

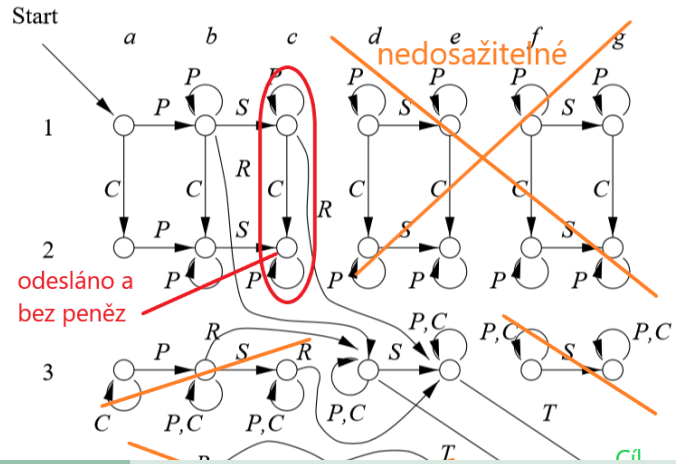


Konečné automaty, Regulární jazyky

- **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**.

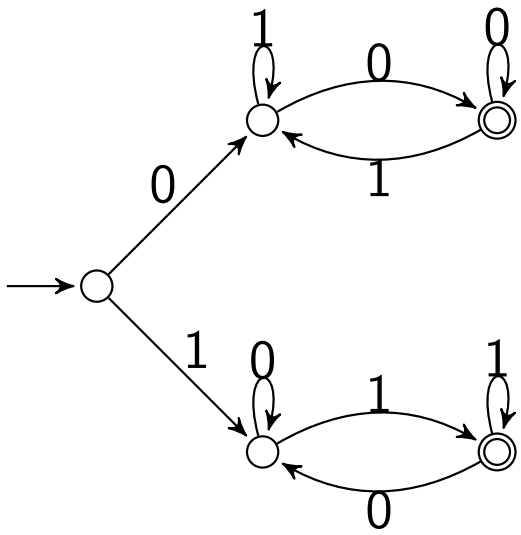


- Dnes budeme zkoumat:
  - Dá se do každého stavu dojít?
  - Je v přechodové funkci cyklus?
  - Jak automat redukovat?
  - Může být nedeterministický?

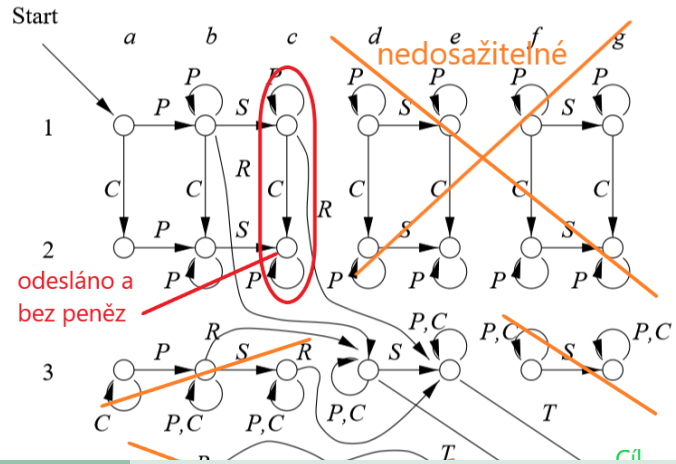


Konečné automaty, Regulární jazyky

- **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**.

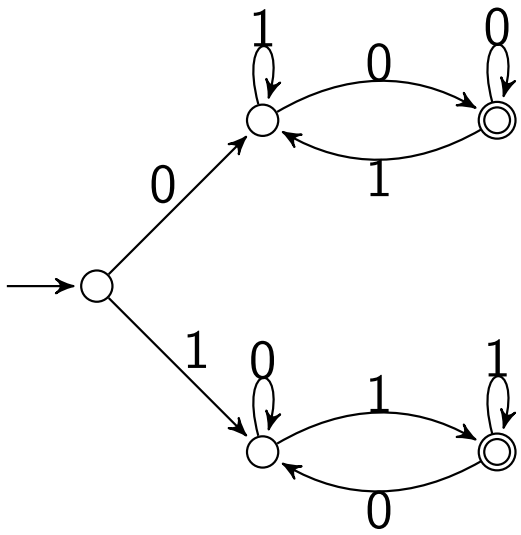


- Dnes budeme zkoumat:
  - Dá se do každého stavu dojít?
  - Je v přechodové funkci cyklus?
  - Jak automat redukovat?
  - Může být nedeterministický?

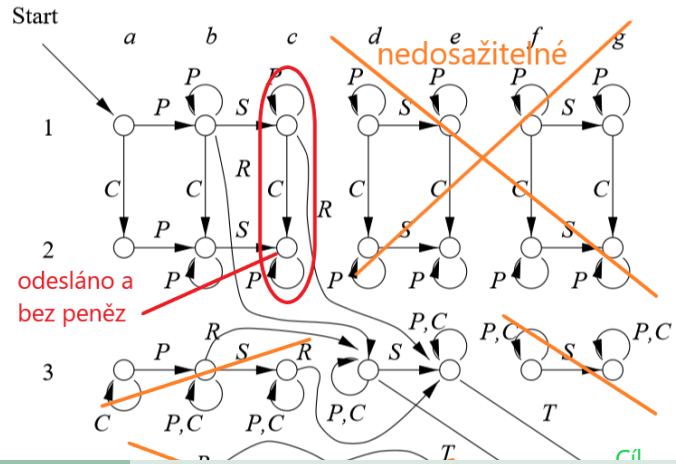


Konečné automaty, Regulární jazyky

- **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**.

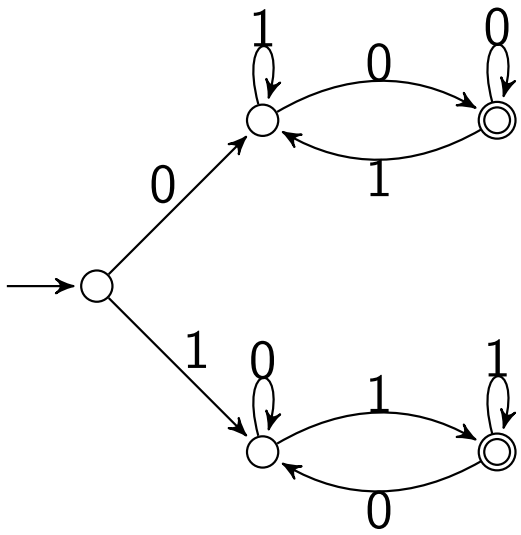


- Dnes budeme zkoumat:
  - Dá se do každého stavu dojít?
  - Je v přechodové funkci cyklus?
  - Jak automat redukovat?
  - Může být nedeterministický?

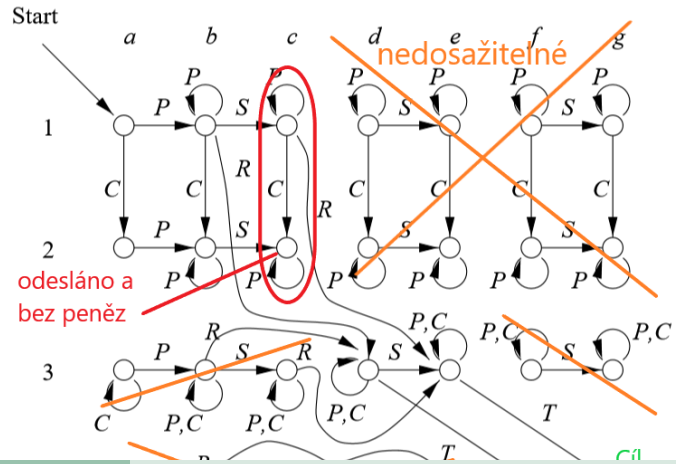


Konečné automaty, Regulární jazyky

- **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**.

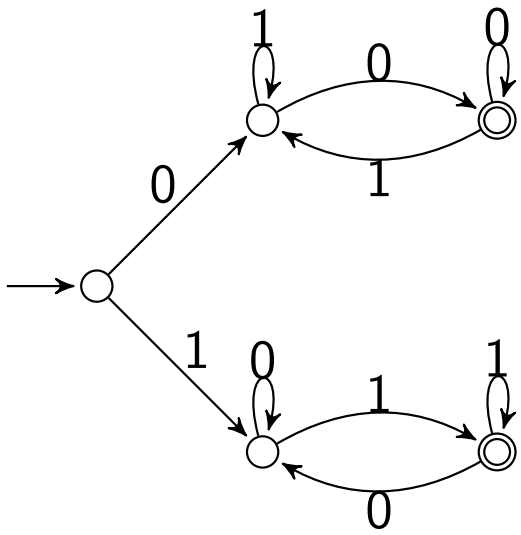


- Dnes budeme zkoumat:
  - Dá se do každého stavu dojít?
  - Je v přechodové funkci cyklus?
  - Jak automat redukovat?
  - Může být nedeterministický?

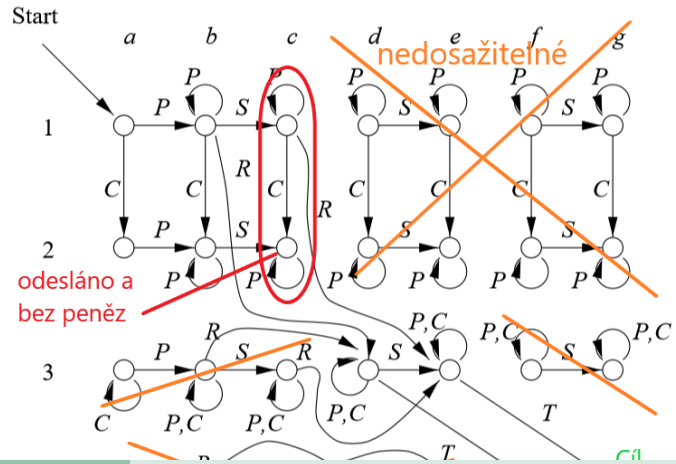


Konečné automaty, Regulární jazyky

- **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**.



- Dnes budeme zkoumat:
  - Dá se do každého stavu dojít?
  - Je v přechodové funkci cyklus?
  - Jak automat redukovat?
  - Může být nedeterministický?



Definice 2.1: 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 $\delta^*(q_0, w) = q$.

Algoritmus 2.1: 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 $M_{i+1} \neq M_i$.

Důkaz: 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 $q \in M_n$. □

Definice 2.2: 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 $\delta^*(q_0, w) = q$.

Algoritmus 2.2: 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 $M_{i+1} \neq M_i$.

Důkaz: 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 $q \in M_n$. □

Definice 2.3: 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 $\delta^*(q_0, w) = q$.

Algoritmus 2.3: Hledání dosažitelných stavů

Dosažitelné stavy hledáme iterativně.

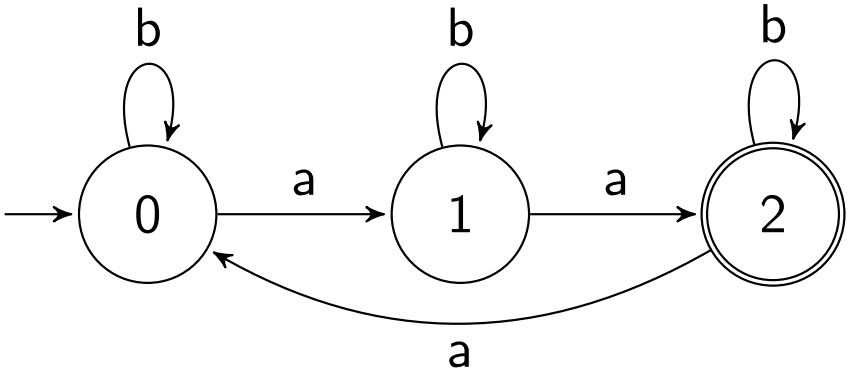
- Začátek: $M_0 = \{q_0\}$.
- Opakuj: $M_{i+1} = M_i \cup \{q | q \in Q, (\exists p \in M_i,\ \exists x \in \Sigma)\ \delta(p, x) = q\}$
- opakuj dokud $M_{i+1} \neq M_i$.

Důkaz: 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 $q \in M_n$.

Projde automat cyklus?

Automat A



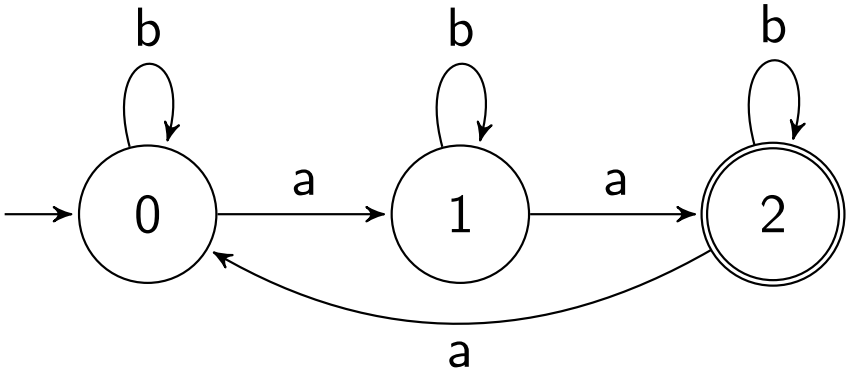
Příklad 2.1

Projde automat výše cyklus při čtení slova:

- *abbbba* ?
- $abbbba = a(b)bbba; \forall i \geq 0; a(b)^i bbba \in L(A)$.
- *aaaaba* ?
- $aaaaba = (aaa)aba; \forall i \geq 0; (aaa)^i aba \in L(A)$.
- *aa* ?
- *aa* neprojde cyklus, ale délka $|aa| <$ počet stavů.

Projde automat cyklus?

Automat A



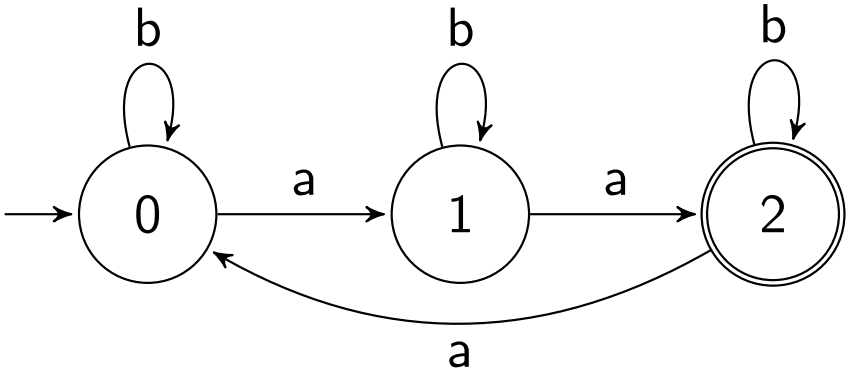
Příklad 2.1

Projde automat výše cyklus při čtení slova:

- $abbbba$?
- $abbbba = a(b)bbba$; $\forall i \geq 0; a(b)^i bbba \in L(A)$.
- $aaaaba$?
- $aaaaba = (aaa)aba$; $\forall i \geq 0; (aaa)^i aba \in L(A)$.
- aa ?
- aa neprojde cyklus, ale délka $|aa| <$ počet stavů.

Projde automat cyklus?

Automat A



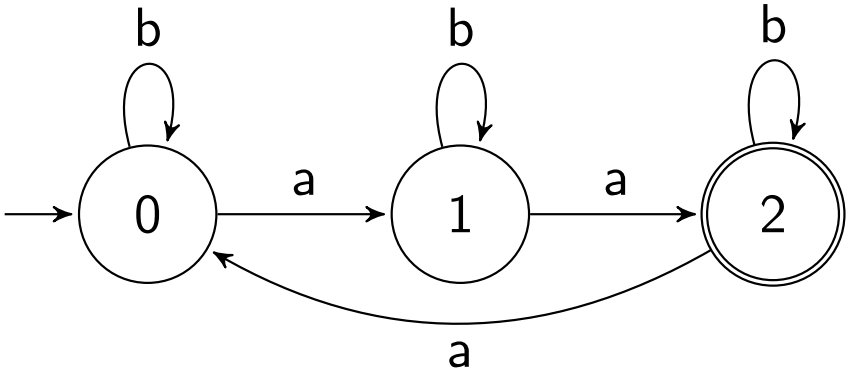
Příklad 2.1

Projde automat výše cyklus při čtení slova:

- $abbbba$?
- $abbbba = a(b)bbba$; $\forall i \geq 0; a(b)^i bbba \in L(A)$.
- $aaaaba$?
- $aaaaba = (aaa)aba$; $\forall i \geq 0; (aaa)^i aba \in L(A)$.
- aa ?
- aa neprojde cyklus, ale délka $|aa| <$ počet stavů.

Projde automat cyklus?

Automat A



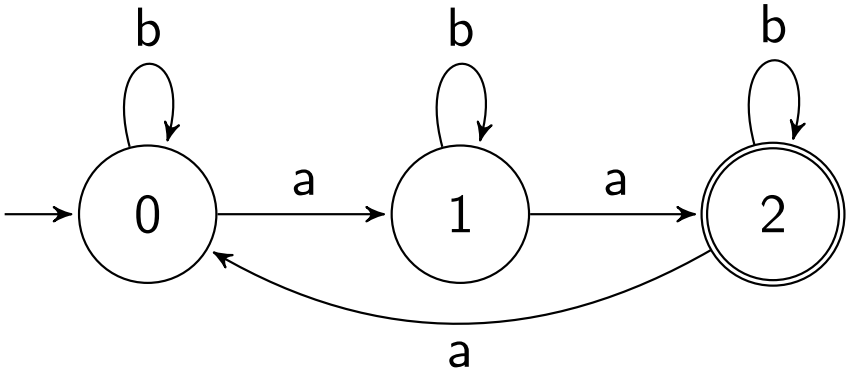
Příklad 2.1

Projde automat výše cyklus při čtení slova:

- *abbbba* ?
- $abbbba = a(b)bbba$; $\forall i \geq 0; a(b)^i bbba \in L(A)$.
- *aaaaba* ?
- $aaaaba = (aaa)aba$; $\forall i \geq 0; (aaa)^i aba \in L(A)$.
- *aa* ?
- *aa* neprojde cyklus, ale délka $|aa| <$ počet stavů.

Projde automat cyklus?

Automat A



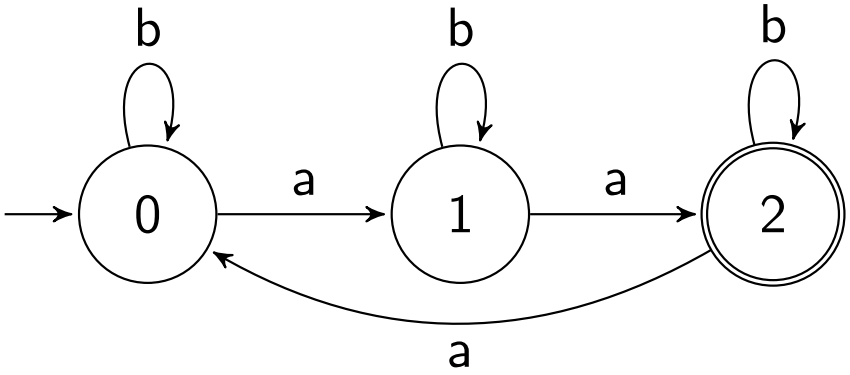
Příklad 2.1

Projde automat výše cyklus při čtení slova:

- *abbbba* ?
- $abbbba = a(b)bbba$; $\forall i \geq 0; a(b)^i bbba \in L(A)$.
- *aaaaba* ?
- $aaaaba = (aaa)aba$; $\forall i \geq 0; (aaa)^i aba \in L(A)$.
- *aa* ?
- *aa* neprojde cyklus, ale délka $|aa| <$ počet stavů.

Projde automat cyklus?

Automat A



Příklad 2.1

Projde automat výše cyklus při čtení slova:

- *abbbba* ?
- $abbbba = a(b)bbba$; $\forall i \geq 0; a(b)^i bbba \in L(A)$.
- *aaaaba* ?
- $aaaaba = (aaa)aba$; $\forall i \geq 0; (aaa)^i aba \in L(A)$.
- *aa* ?
- *aa* neprojde cyklus, ale délka $|aa| <$ počet stavů.

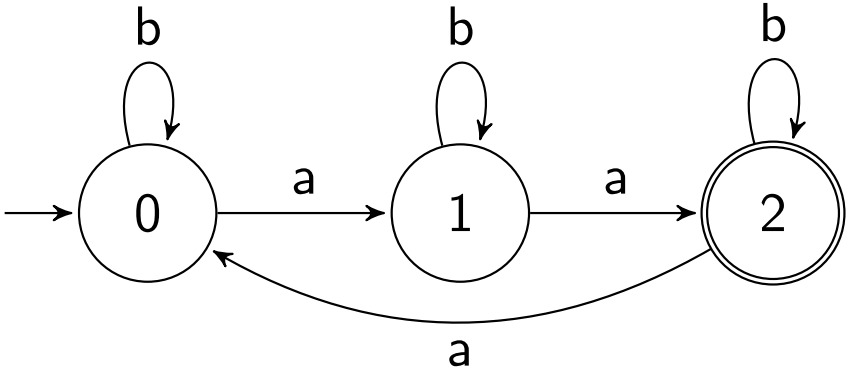
Iterační (pumping) lemma pro regulární jazyky

Věta 2.1: !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 $xy^k z$ je také v L.

Automat A



Důkaz iteračního lematu pro regulární jazyky

Důkaz: 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.

□

Důkaz iteračního lematu pro regulární jazyky

Důkaz: 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.

□

Důkaz iteračního lematu pro regulární jazyky

Důkaz: 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.

□

Důkaz iteračního lematu pro regulární jazyky

Důkaz: 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. $w = xyz$, $y \neq \epsilon$, $|xy| \leq n$.

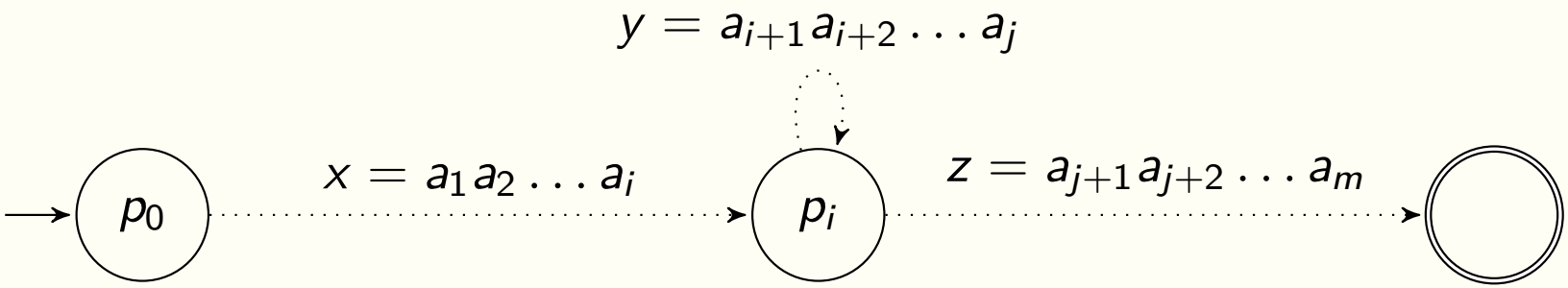
- Smyčka nad p_i se může opakovat libovolně krát a vstup je také akceptovaný.

□

Důkaz iteračního lematu pro regulární jazyky

Důkaz: 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. $w = xyz$, $y \neq \epsilon$, $|xy| \leq n$.

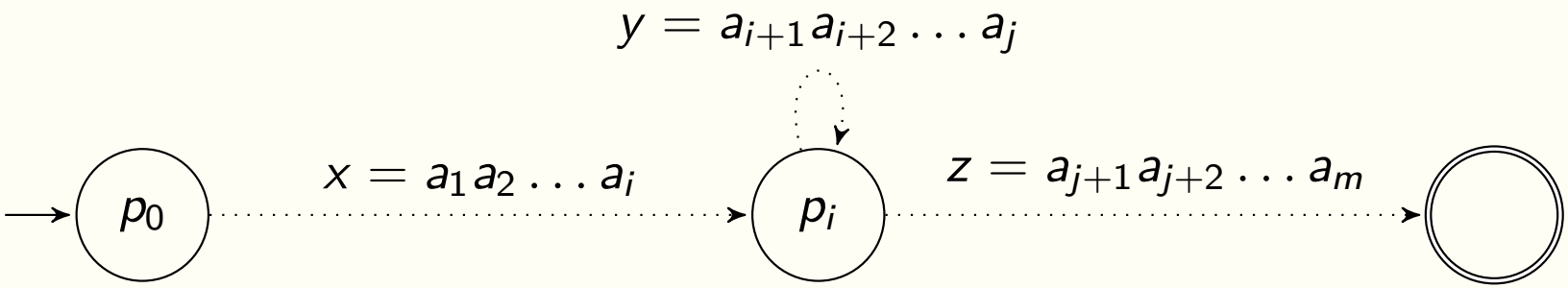


- Smyčka nad p_i se může opakovat libovolně krát a vstup je také akceptovaný.

Důkaz iteračního lematu pro regulární jazyky

Důkaz: 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

Příklad 2.2 (Pumping lemma jako hra s oponentem)

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

Důkaz: Jazyk L_{eq} není regulární.

- Předpokládejme že L_{eq} je regulární. Vezměme n z pumping lemmatu.
- Zvolme $w = 0^n1^n \in L_{eq}$.
- Rozdělme $w = xyz$ dle pumping lemmatu, $y \neq \epsilon$, $|xy| \leq n$.

□

Příklad 2.3

Jazyk $L = \{0^i1^i; i \geq 0\}$ není regulární.

Použití pumping lemmatu

Příklad 2.2 (Pumping lemma jako hra s oponentem)

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

Důkaz: 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}.

□

Příklad 2.3

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

Použití pumping lemmatu

Příklad 2.2 (Pumping lemma jako hra s oponentem)

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

Důkaz: 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}.

□

Příklad 2.3

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

Použití pumping lemmatu

Příklad 2.2 (Pumping lemma jako hra s oponentem)

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

Důkaz: 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}.

□

Příklad 2.3

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

Použití pumping lemmatu

Příklad 2.2 (Pumping lemma jako hra s oponentem)

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

Důkaz: 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}. □

Příklad 2.3

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

Použití pumping lemmatu

Příklad 2.2 (Pumping lemma jako hra s oponentem)

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

Důkaz: 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}. □

Příklad 2.3

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

Aplikace pumping lemmatu 2

Příklad 2.4

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

Důkaz: 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 $|xz| = p - m$.

Aplikace pumping lemmatu 2

Příklad 2.4

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

Důkaz: 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).

□

Aplikace pumping lemmatu 2

Příklad 2.4

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

Důkaz: 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).

Aplikace pumping lemmatu 2

Příklad 2.4

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

Důkaz: 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). □

'Pumpovatelný' ne-regulární jazyk

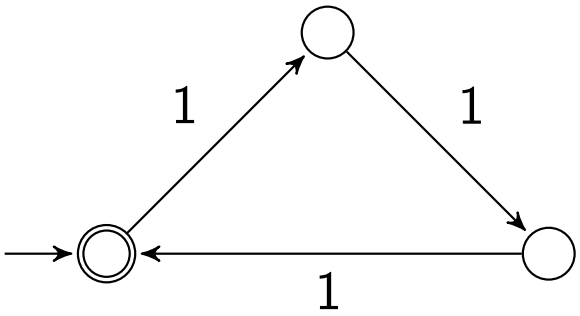
Příklad 2.5 (Ne–regulární jazyk, který lze pumpovat)

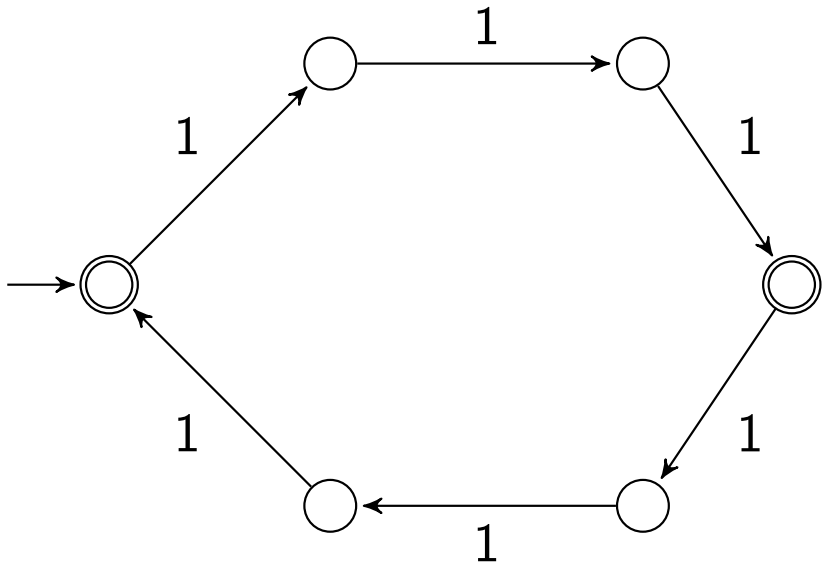
Jazyk $L = \{u | u = a^+ b^i c^i \vee u = b^i c^j, + \in \mathbb{N}_{>0}, i, j \in \mathbb{N}_0\}$ není regulární (Myhill–Nerodova věta), ale vždy lze pumpovat první písmeno.

Nejednoznačnost

Automat přijímající daný jazyk není určen jednoznačně.

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





Definice 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 $\delta^*(q, w) \in F$.

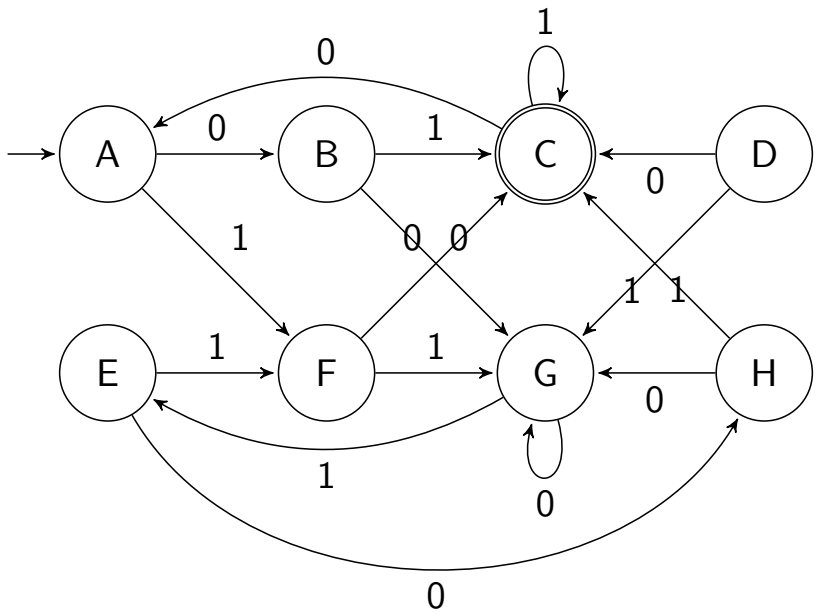
Pokud dva stavy nejsou ekvivalentní, říkáme, že jsou **rozlišitelné**.

Příklad rozlišitelných stavů

Příklad 2.6

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

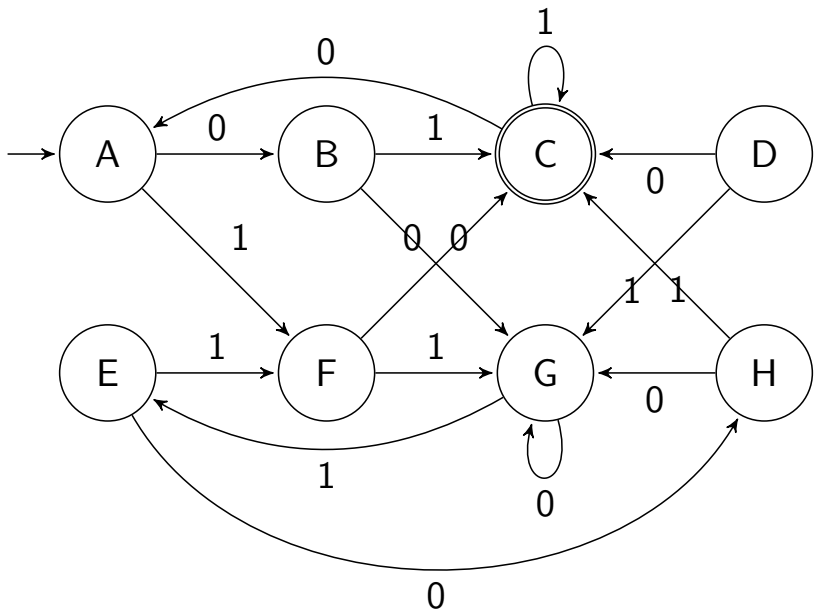
Ekvivalence na stavech je tranzitivní.

Příklad rozlišitelných stavů

Příklad 2.6

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

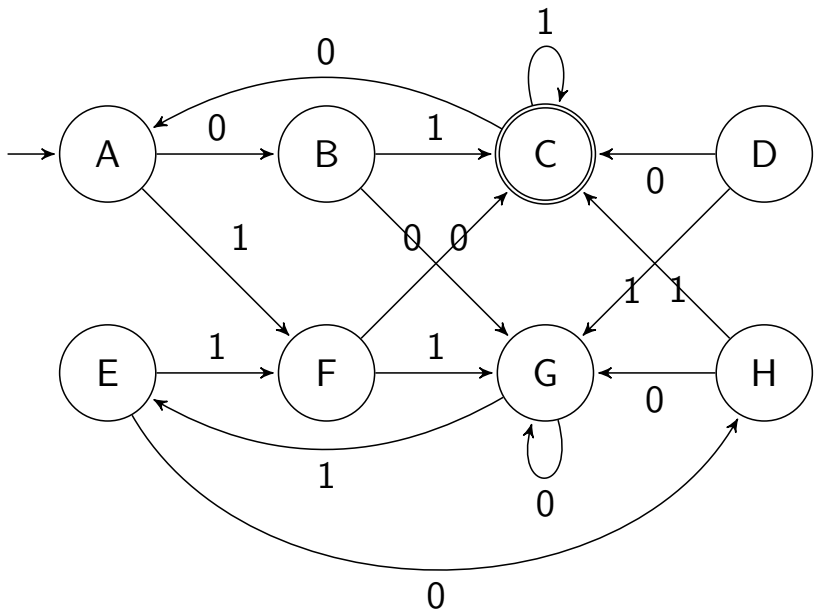
Ekvivalence na stavech je tranzitivní.

Příklad rozlišitelných stavů

Příklad 2.6

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

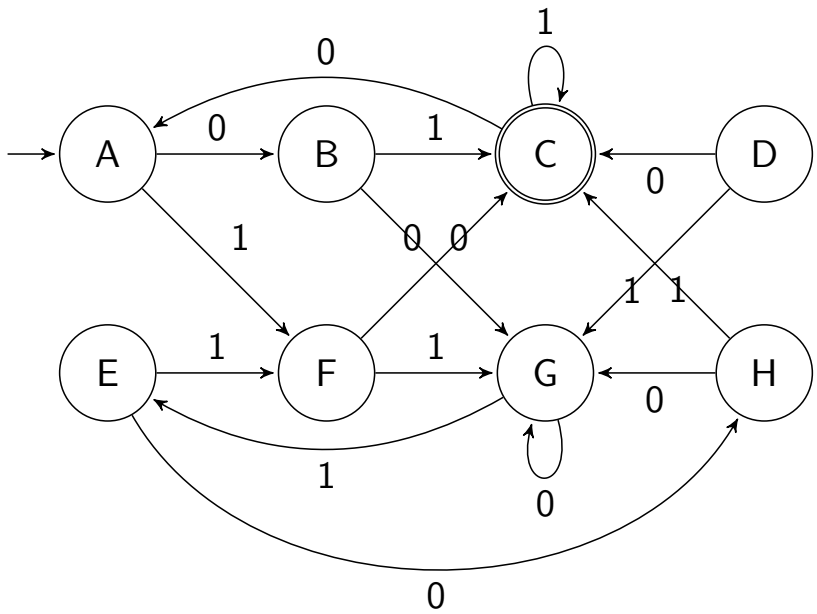
Ekvivalence na stavech je tranzitivní.

Příklad rozlišitelných stavů

Příklad 2.6

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

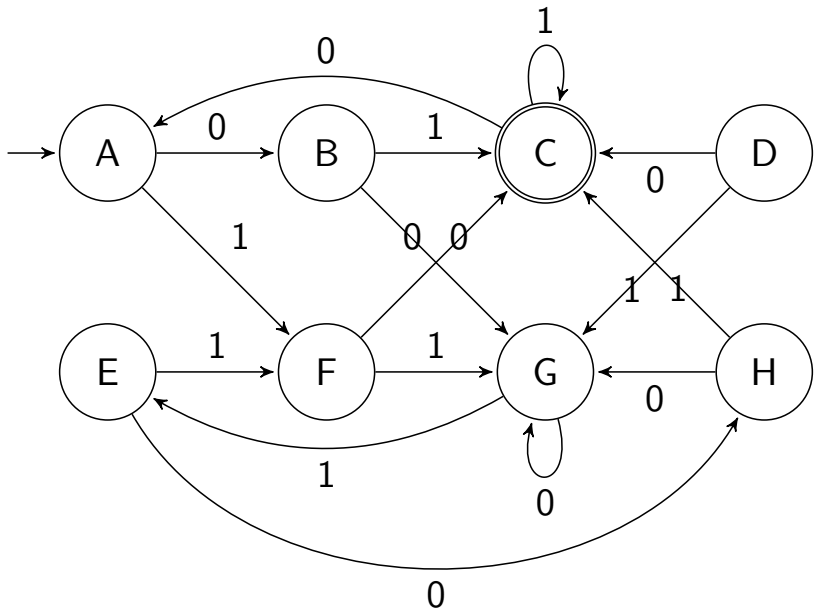
Ekvivalence na stavech je tranzitivní.

Příklad rozlišitelných stavů

Příklad 2.6

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.



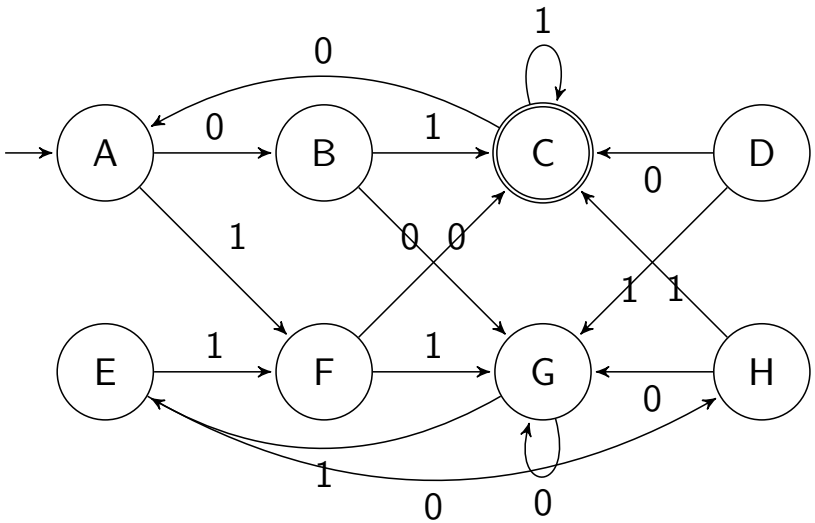
Lemma

Ekvivalence na stavech je tranzitivní.

Algoritmus 2.4: !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, pokud existuje nová dvojice $p, q \in Q$, $a \in \Sigma$.



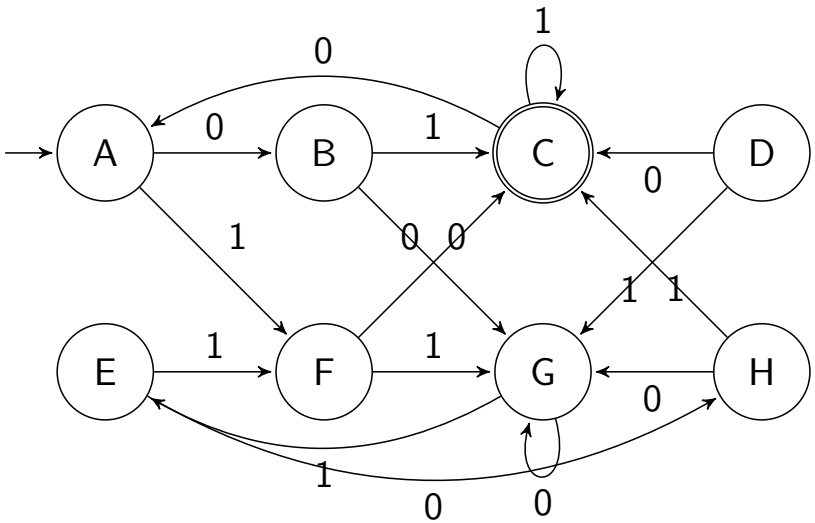
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

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

Algoritmus 2.5: !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 $p, q \in Q$, $a \in \Sigma$.



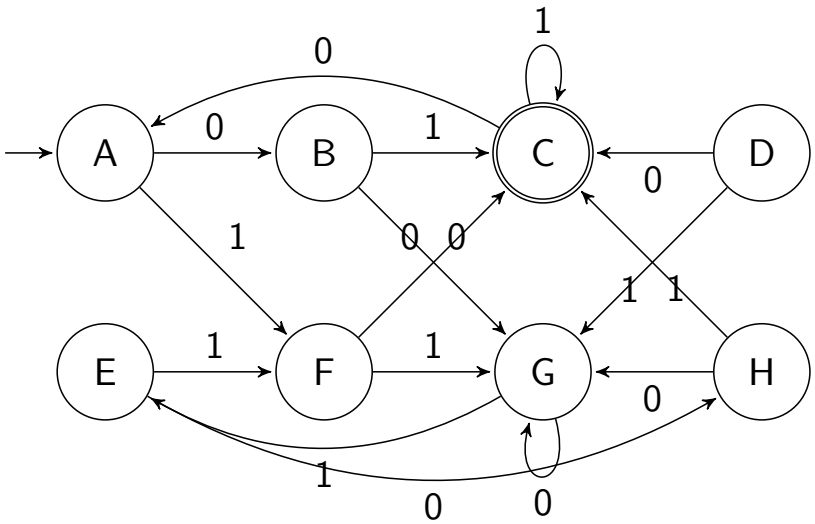
	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

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

Algoritmus 2.6: !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 $p, q \in Q$, $a \in \Sigma$.



	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

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

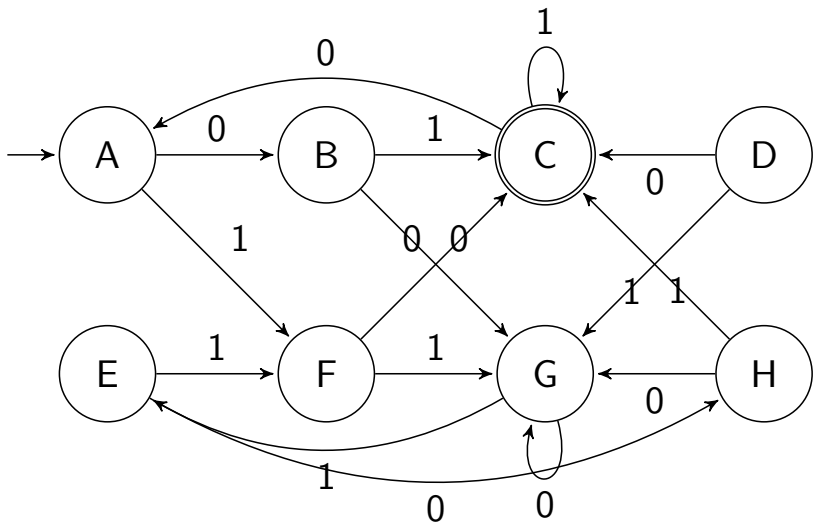
	A	B	C	D	E	F	G
B							
C	x	x					
D			x				
E			x				
F			x				
G			x				
H			x				

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.

Věta 2.2: 1

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

Důkaz: Koreknost algoritmu

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

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'.

Věta 2.3: 1

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

Důkaz: Koreknost algoritmu

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

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'.

Věta 2.4: 1

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

Důkaz: Koreknost algoritmu

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

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 rovnosti regulárních jazyků

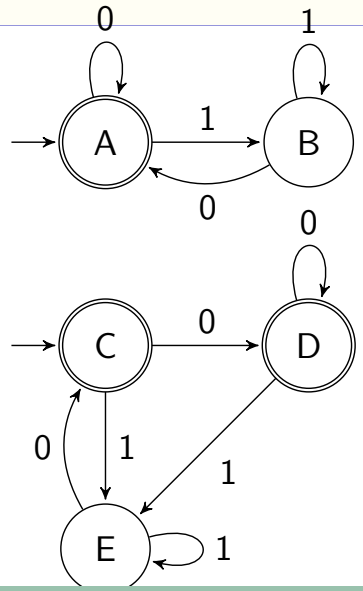
Algoritmus 2.7: !Testování rovnost regulárních jazyků

Rovnost (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 stejné právě když počáteční stavy původních DFA jsou ekvivalentní.

Příklad 2.7

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



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

Testování rovnosti regulárních jazyků

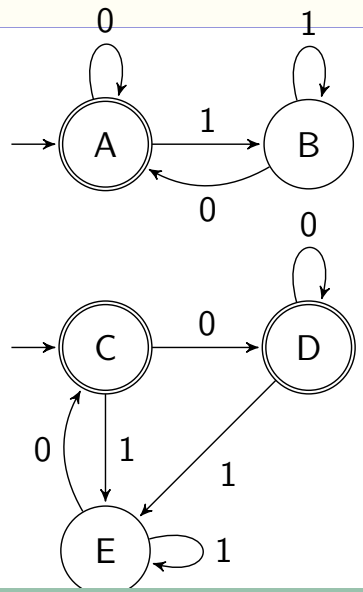
Algoritmus 2.8: !Testování rovnost regulárních jazyků

Rovnost (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 stejné právě když počáteční stavy původních DFA jsou ekvivalentní.

Příklad 2.7

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

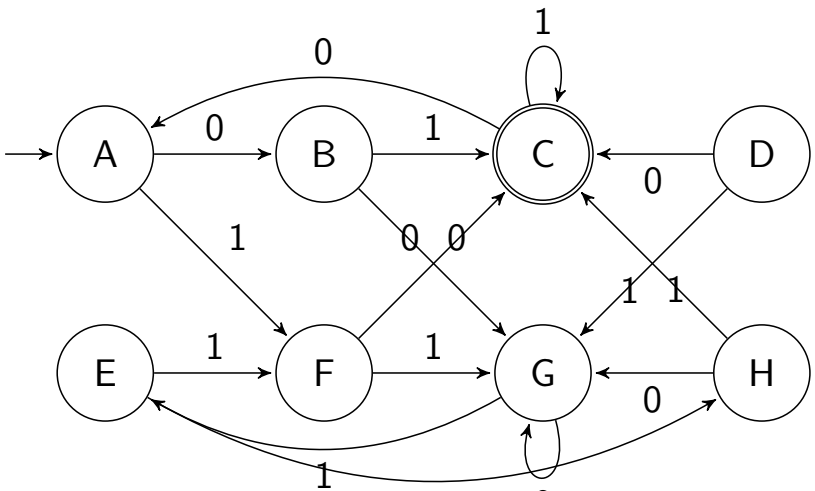
Definice 2.5: redukovaný DFA, redukt

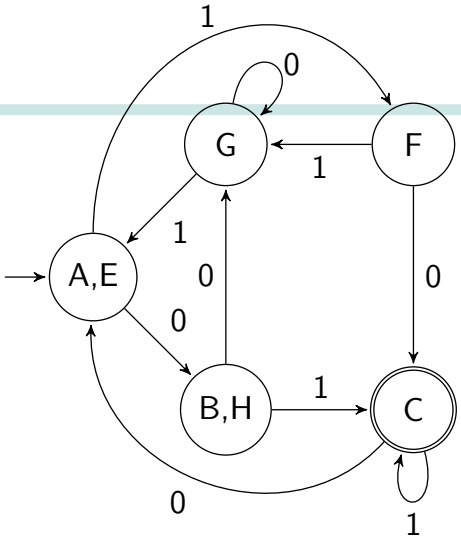
Deterministický konečný automat je **redukovaný**, pokud

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

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

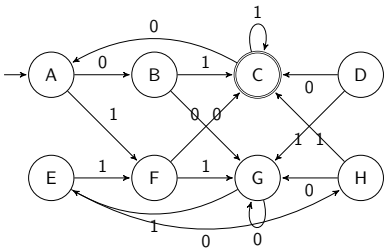
- B je redukovaný a
- A a B jsou ekvivalentní.

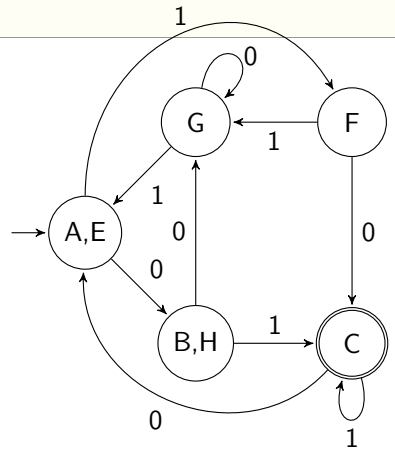




Algoritmus 2.9: !Algoritmus nalezení reduktu DFA A

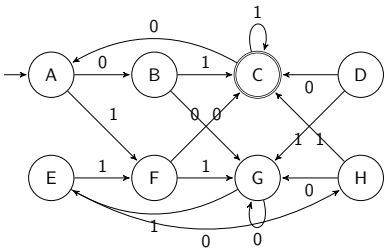
- Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
- Najdeme rozklad zbylých stavů na třídy ekvivalence.
- Konstruujeme DFA B na třídách ekvivalence jakožto stavech. Přechodovou funkci B označíme γ, mějme $S \in Q_B$. Pro libovolné $q \in S$, označíme T třídu ekvivalence $\delta(q, a)$ 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.

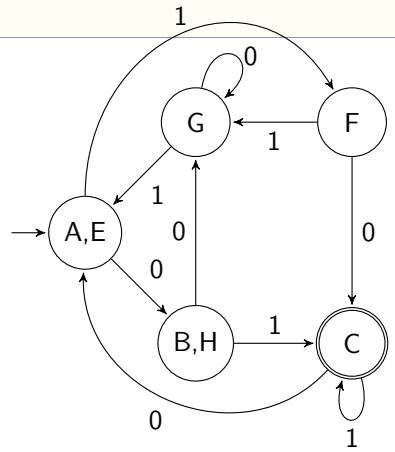




Algoritmus 2.10: !Algoritmus nalezení reduktu DFA A

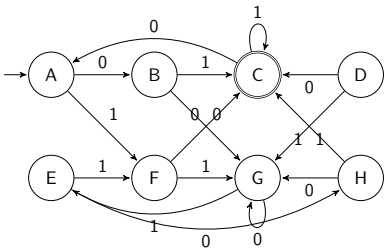
- Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
- Najdeme rozklad zbylých stavů na třídy ekvivalence.
- Konstruujeme DFA B na třídách ekvivalence jakožto stavech. Přechodovou funkci B označíme γ, mějme $S \in Q_B$. Pro libovolné $q \in S$, označíme T třídu ekvivalence $\delta(q, a)$ 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.

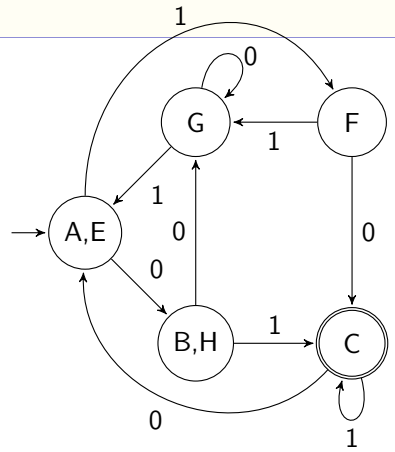




Algoritmus 2.11: !Algoritmus nalezení reduktu DFA A

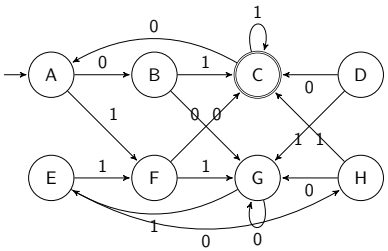
- Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
- Najdeme rozklad zbylých stavů na třídy ekvivalence.
- Konstruujeme DFA B na třídách ekvivalence jakožto stavech. Přechodovou funkci B označíme γ, mějme $S \in Q_B$. Pro libovolné $q \in S$, označíme T třídu ekvivalence $\delta(q, a)$ 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.

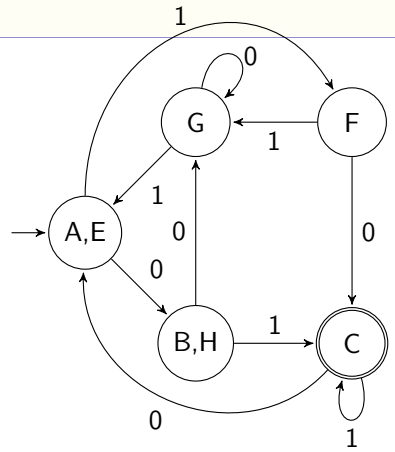




Algoritmus 2.12: !Algoritmus nalezení reduktu DFA A

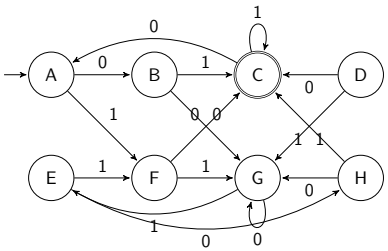
- Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
- Najdeme rozklad zbylých stavů na třídy ekvivalence.
- Konstruujeme DFA B na třídách ekvivalence jakožto stavech. Přechodovou funkci B označíme γ, mějme $S \in Q_B$. Pro libovolné $q \in S$, označíme T třídu ekvivalence $\delta(q, a)$ 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.

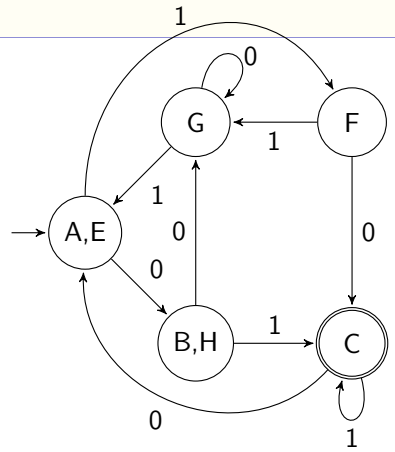




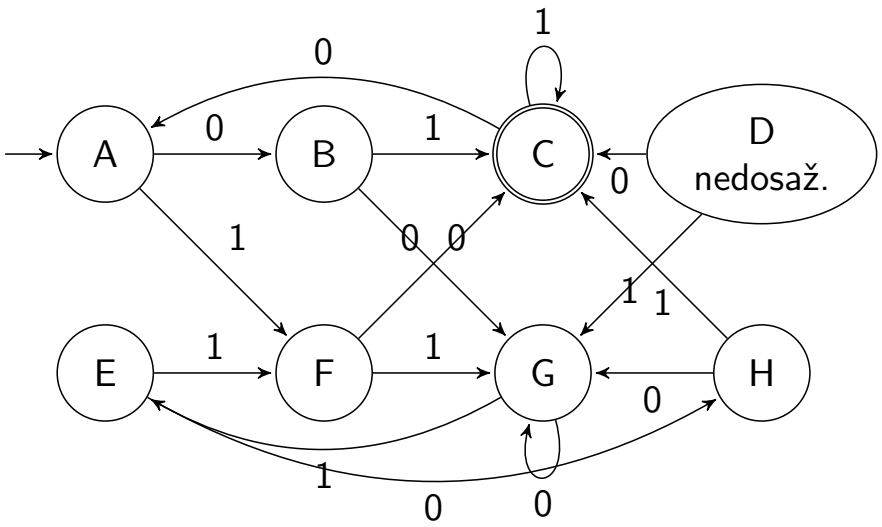
Algoritmus 2.13: !Algoritmus nalezení reduktu DFA A

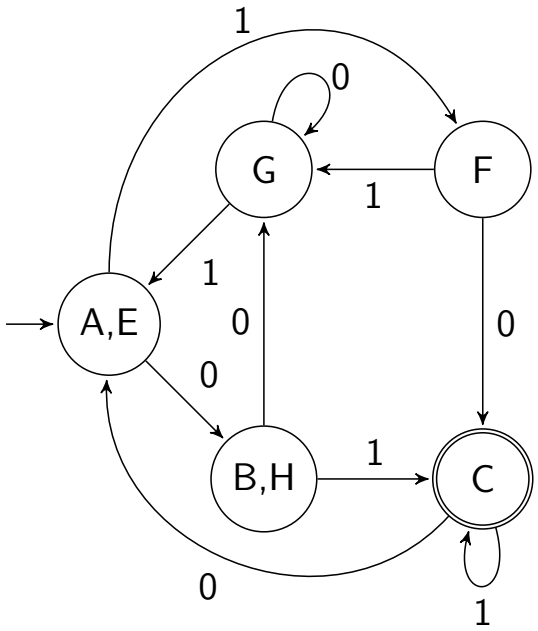
- Ze vstupního DFA A eliminujeme stavy nedosažitelné z počátečního stavu.
- Najdeme rozklad zbylých stavů na třídy ekvivalence.
- Konstruujeme DFA B na třídách ekvivalence jakožto stavech. Přechodovou funkci B označíme γ, mějme $S \in Q_B$. Pro libovolné $q \in S$, označíme T třídu ekvivalence $\delta(q, a)$ 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





	A	B	C	E	F	G
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

Třídy ekvivalence:

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

Automatový homomorfizmus

Definice 2.6: automatový homomorfizmus

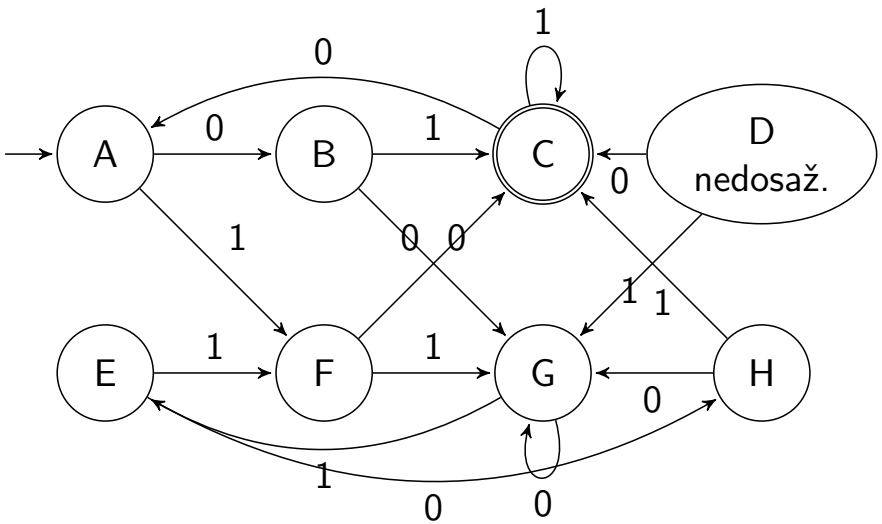
Nechť A_1, A_2 jsou DFA. Řekneme, že zobrazení $h : Q_1 \to Q_2$ Q_1 na Q_2 je **automatovým homomorfizmem**, jestliže:

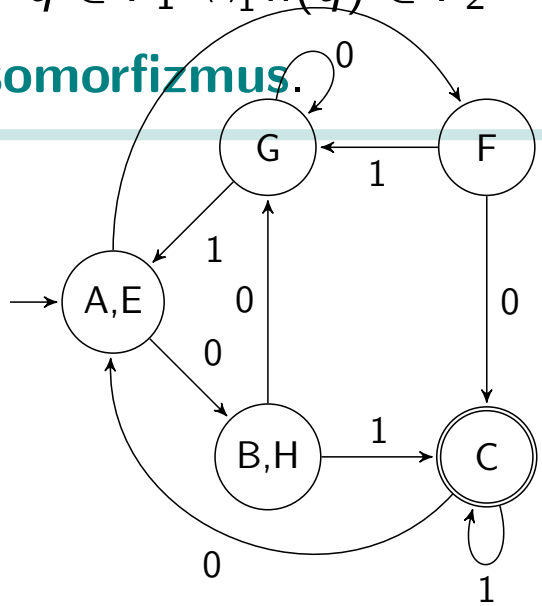
$h(q_{0_1}) = q_{0_2}$ 'stejné' počá

$h(\delta_1(q, x)) = \delta_2(h(q), x)$ 'stejné' přec

$q \in F_1 \Leftrightarrow h(q) \in F_2$ 'stejné' konc

Homomorfizmus prostý a na nazýváme **isomorfizmus**.





Ekvivalence automatů a homomorfizmus

Definice 2.7: Ekvivalence automatů!

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

Věta 2.5: Věta o ekvivalenci automatů

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

Důkaz:

- Pro libovolný řetězec $w \in \Sigma^*$ konečnou iterací
  - $h(\delta_1^*(q, w)) = \delta_2^*(h(q), w)$
- dále: $w \in L(A_1) \Leftrightarrow \delta_1^*(q_{0_1}, w) \in F_1$
  $\Leftrightarrow h(\delta_1^*(q_{0_1}, w)) \in F_2$
  $\Leftrightarrow \delta_2^*(h(q_{0_1}), w) \in F_2$

Ekvivalence automatů a homomorfizmus

Definice 2.8: Ekvivalence automatů!

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

Věta 2.6: Věta o ekvivalenci automatů

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

Důkaz:

- Pro libovolný řetězec $w \in \Sigma^*$ konečnou iterací
  - $h(\delta_1^*(q, w)) = \delta_2^*(h(q), w)$
- dále: $w \in L(A_1) \Leftrightarrow \delta_1^*(q_{0_1}, w) \in F_1$
  $\Leftrightarrow h(\delta_1^*(q_{0_1}, w)) \in F_2$
  $\Leftrightarrow \delta_2^*(h(q_{0_1}), w) \in F_2$

Ekvivalence automatů a homomorfizmus

Definice 2.9: Ekvivalence automatů!

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

Věta 2.7: Věta o ekvivalenci automatů

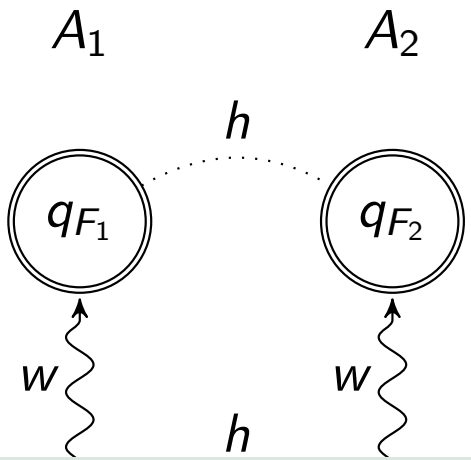
Existuje-li homomorfizmus konečných automatů A_1 do A_2, pak jsou A_1 a A_2 ekvivalentní.

Důkaz:

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

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

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



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.

□

Věta 2.8: Redukt

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

Definice 2.10: Redukt

Reduktem DFA A nazýváme redukovaný automat A ekvivalentní

Pro nedeterministické FA to tak snadné není

Příklad 2.8

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.