Prednáška 6 - Algebra regulárnych výrazov, vlastnosti regulárnych jazykov, minimalizácia DKA

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

3.11.2020

Algebra regulárnych výrazov

- V praxi nás môže niekedy zaujímať otázka, či vieme regulárny výraz upraviť na jednoduchší regulárny výraz, popisujúci ten istý jazyk.
- 2 regulárne výrazy ktoré popisujú ten istý jazyk nazývame **ekvivalentné**.
- Napríklad:
  - $ab(a+b)^* + ba(a+b)^*$
  - $(ab+ba)(a+b)^*$

  sú 2 ekvivalentné regulárne výrazy, pričom ten druhý je jednoduchší, než ten prvý.
- Pri zjednodušovaní regulárnych výrazov si vieme pomôcť viacerými "algebraickými" vlastnosťami regulárnych výrazov.

Asociativita a komutativita

Nech L, M, N sú regulárne výrazy. Platia nasledovné vlastnosti:

1. (komutativita zjednotenia) $L + M = M + L$
2. (asociativita zjednotenia) $L + (M + N) = (L + M) + N$
3. (asociativita zreťazenia) $L(MN) = (LM)N$

POZOR! Komutativita vo všeobecnosti neplatí pre zreťazenie! $LM \neq ML$.

Jednotkové a nulové prvky

Nech L je regulárny výraz. Platí:

1. (jednotkový prvok zjednotenia) $\emptyset$, t.j.: $L + \emptyset = \emptyset + L = L$
2. (jednotkový prvok zreťazenia) ε, t.j.: $\varepsilon L = L\varepsilon = L$
3. (nulový prvok zreťazenia) $\emptyset$, t.j.: $\emptyset L = L\emptyset = \emptyset$

Distributívne zákony

Pre regulárne výrazy platia 2 distributívne zákony. Nech L, M, N sú regulárne výrazy. Potom platí:

1. (ľavý distributívny zákon zreťazenia nad zjednotením) $L(M + N) = LM + LN$
2. (pravý distributívny zákon zreťazenia nad zjednotením) $(M + N)L = ML + NL$

Idempotenty

Operátor sa nazýva idempotent, ak pre všetky operandy platí, že ak sa operátor aplikuje na 2 rovnaké hodnoty, tak výsledok je zase rovnaká hodnota. Pre regulárne výrazy sa ako idempotent správa:

- (idempotentný zákon zjednotenie) $L + L = L$

Uzáverové pravidlá

Platia špeciálne vlastnosti iterácie:

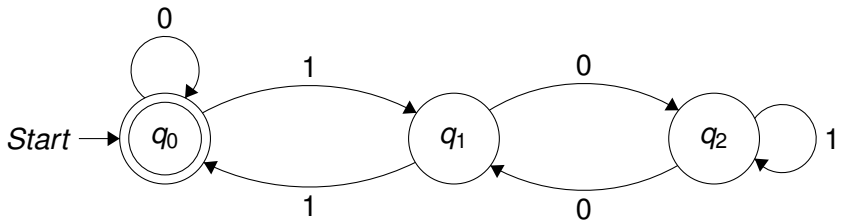
1. $(L^*)^* = L^*$
2. $\emptyset^* = \varepsilon$
3. $\varepsilon^* = \varepsilon$
4. $L^+ = LL^* = L^*L$
5. $L^* = \varepsilon + L^+$

Čo všetko je regulárny jazyk?

- **Regulárny jazyk** je každý jazyk, pre ktorý existuje konečný automat (DKA, NKA, ε-NKA) alebo regulárny výraz, ktorý ho akceptuje alebo popisuje.
- Počas posledných týždňov sme sa stretávali s viacerými príkladmi regulárnych jazykov, napríklad:
  - Reťazce z a, b začínajúce prefixom bb, t.j. $(bb)(a + b)^*$
  - Reťazce z a, b v tvare a^*b^*.
  - Binárne reťazce obsahujúce párny počet výskytov symbolu 0, t.j. $(1 + 01^*0)^*$
  - Binárne reťazce predstavujúce čísla deliteľné tromi.
- Vždy platí, že k týmto regulárnym jazykom sa dá nájsť aj konečný automat, aj regulárny výraz.

Čo všetko je regulárny jazyk?

- Na začiatku semestra sme si spomínali súvis konečných automatov s teoretickou informatikou - konečný automat je matematický model výpočtového zariadenia.
- Návrh konečného automatu pre nejaký jazyk je de facto "programovanie" tohto výpočtového zariadenia.
- Napríklad binárne reťazce predstavujúce čísla deliteľné tromi sú akceptovateľné nasledovným DKA:



- To znamená, že rozhodovací problém, či je nejaké binárne číslo deliteľné tromi, je pomocou konečných automatov **riešiteľný**, pretože uvedený DKA sa dá aplikovať na každé binárne číslo a vždy dá správnu odpoveď áno / nie.
- Jedna z klasických otázok teoretickej informatiky je, či má nejaký rozhodovací problém algoritmus, ktorý dokáže pre inštanciu problému v konečnom čase vrátiť odpoveď áno / nie.
- Ak sa bavíme o výpočtových zariadeniach na úrovni konečných automatov, tak tu algoritmus = prechodová funkcia automatu.
- Existujú však rozhodovacie problémy, ktoré sa **nedajú riešiť na konečnom automate**, t.j. nevieme zostrojiť konečný automat, ktorý by akceptoval jazyk popisujúci inštancie, pre ktoré je odpoveď na rozhodovací problém "áno".
- Teda existujú jazyky, ktoré **nie sú regulárne**.

Príklad neregulárneho jazyka

Uvažujme abecedu $A = \{0, 1\}$ a jazyk $L = \{0^n 1^n \mid n \geq 1\}$.

- Tento jazyk tvoria reťazce, ktoré sú tvorené postupnosťou núl, za ktorými nasleduje postupnosť jednotiek, pričom obe časti sú **rovnakej dĺžky**,
- $L = \{01, 0011, 000111, 00001111, 0000011111, ...\}$
- Dá sa ukázať, že sa **nedá zostrojiť** konečný automat (regulárny výraz), ktorý by akceptoval (popisoval) práve tieto reťazce.
- Teda sa **nedá zostrojiť** DKA, ktorý by dokázal zistiť, či má na vstupe reťazec, ktorý obsahuje rovnaký počet núl a jednotiek, ktoré sú usporiadané v poradí najprv nuly, potom jednotky.
- Čiže rozhodovací problém *Má tento reťazec za sebou idúce nuly a jednotky, ktorých je rovnako veľa?* sa **nedá riešiť** na konečných automatoch.

Zdôvodnenie:

- **Predpokladajme, že** jazyk L je regulárny, teda musí mať konečný automat $A = (Q, \Sigma, \delta, q_0, F)$, $L(A) = L = \{0^n 1^n \mid n \geq 1\}$
- Keďže konečný automat je **konečný**, tak má konečný počet stavov, napr $|Q| = k$.
- Keby sme dali na vstup 0^k (toľko núl, koľko má automat stavov), tak **musí nastať**, že pri čítaní núl sa nejaký stav **zopakuje**, t.j. po prečítaní 0^i a 0^j sa automat nachádza v nejakom stave q, $i \neq j$.
- Ak by sme dali na vstup $0^i 1^i$, tak zo stavu q sa prečíta 1^i a výpočet **musí** skončiť v akceptačnom stave (lebo automat akceptuje $0^n 1^n$)
- Avšak, ak dáme na vstup $0^j 1^i$, automat by **rovnako skončil v akceptačnom stave**, to znamená, že by akceptoval reťazec, ktorý akceptovať **nesmie!**
- Nastáva SPOR s predpokladom, a teda L **nie je** regulárny.

Iné príklady neregulárnych jazykov

- $L = \{ww^R \mid w \in \{a, b\}^* \wedge w^R$ je zrkladový obraz reťazca $w\}$
- $L = \{w \mid w \in \{a, b\}^* \wedge w$ obsahuje rovnaký počet znakov a a $b\}$
- $L = \{a^P \mid P$ je prvočíslo $\}$
- L predstavujú "dobre-ozátvorkované" výrazy, t.j. majú korektne umiestnený rovnaký počet $(,)$
- Jazyk predstavujúci všetky syntakticky korektné programy napísané v jazyku Java
- $L = \{a^n b^n c^n \mid n \geq 0\}$
- ...a veľa ďalších.

Neregulárne jazyky

- Platí teda, že konečné automaty majú svoje "limity" a že nie každý jazyk L, ktorý si vymyslím, sa dá akceptovať konečným automatom.
- Existujú viaceré spôsoby, ako sa dá zistiť, či sa dá k danému jazyku zostrojiť konečný automat - známa je tzv. pumpovacia lemma, pomocou ktorej sa dá niekedy ukázať, že jazyk nie je regulárny.
- My si len uvedieme motivačný príklad, prečo sa nedajú konečné automaty použiť pre niektoré jazyky.

Limity konečných automatov

Uvažujme 2 jazyky nad abecedou $A = \{a, b\}$. Zápis $\sharp_a(w)$ znamená "počet výskytu symbola a v reťazci w:

1. $L_1 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv \sharp_b(w) \bmod 3\}$
2. $L_2 = \{w \in \{a, b\}^* \mid \sharp_a(w) = \sharp_b(w)\}$

- Pre jazyk L_1 vieme zostrojiť KA ľahko - pomocou stavov konečného automatu si vždy vieme pamätať, aký bol počet prečítaných (a mod 3), aký bol počet prečítaných (b mod 3) a zároveň stav, v ktorom sú počty rovnaké mod 3, bude akceptačný. Keďže zvyšky po delení 3 sú len 0, 1, 2, tým pádom je počet možných situácií konečný.
- V jazyku L_2 potrebujeme počítať a a b, avšak počet **nie je zhora obmedzený**. Preto **nemôžeme použiť** stavy konečného automatu na počítanie symbolov!

Vybrané uzáverové vlastnosti regulárnych jazykov

Nech L_1, L_2 sú 2 regulárne jazyky nad abecedou A. Medzi jednoduchšie-dokázateľné vlastnosti patria tieto:

1. Zjednotenie $L_1 \cup L_2$ je regulárny jazyk.
2. Zreťazenie $L_1 L_2$ je regulárny jazyk.
3. Iterácia L_1^* je regulárny jazyk.
4. Doplnok jazyka (komplement) L_1^C je regulárny jazyk.
5. Prienik $L_1 \cap L_2$ je regulárny jazyk.
6. Rozdiel $L_1 \setminus L_2$ je regulárny jazyk.

Dôkazy

Dôkazy tvrdení, že ak L_1, L_2 sú regulárne jazyky, tak potom

1. Zjednotenie $L_1 \cup L_2$ je regulárny jazyk.
2. Zreťazenie $L_1 L_2$ je regulárny jazyk.
3. Iterácia L_1^* je regulárny jazyk.

sú triviálne. Stačí si uvedomiť, že ak R_1, R_2 sú regulárne výrazy pre jazyky L_1, L_2, tak potom

1. Regulárny výraz $R_1 + R_2$ je regulárny výraz popisujúci jazyk $L_1 \cup L_2$, a teda $L_1 \cup L_2$ je regulárny.
2. Regulárny výraz $R_1 R_2$ je regulárny výraz popisujúci jazyk $L_1 L_2$, a teda $L_1 L_2$ je regulárny.
3. Regulárny výraz R_1* je regulárny výraz popisujúci jazyk L_1^*, a teda L_1^* je regulárny.

Dôkaz regularity komplementu

Nech A je abeceda a L_1 je regulárny jazyk nad touto abecedou. Nech L_1^C je komplement jazyka L_1, t.j. jazyk $L_1^C = A^* \setminus L_1$.

Jazyk L_1^C je teda jazyk, ktorý tvoria **všetky reťazce nad abecedou**, ktoré **nepatria do jazyka** L_1.

Napríklad ak $A = \{a, b\}$, $L_1 = a(a + b)^*$, t.j. všetky reťazce začínajúce a, tak potom $L_1^C = \varepsilon + b(a + b)^*$, teda všetky reťazce začínajúce b alebo prázdny reťazec.

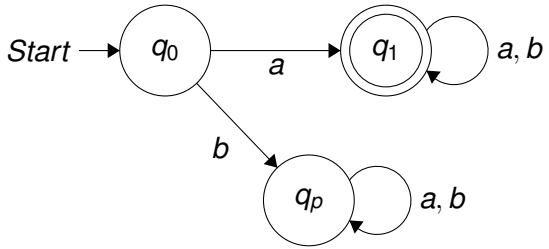
Dôkaz regularity komplementu

Neformálny dôkaz Nech A je DKA, ktorý akceptuje jazyk L_1. Ak zostrojíme automat A^C tak, že:

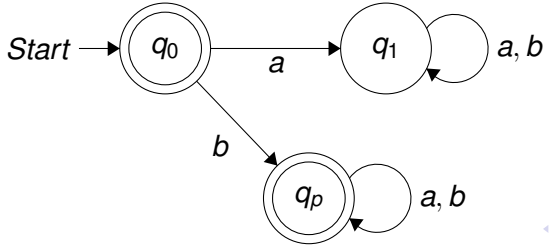
- Každý akceptačný stav označíme ako neakceptačný.
- Každý neakceptačný stav označíme ako akceptačný.

Dostávame práve automat, ktorý akceptuje jazyk L_1^C. Musí platiť, že automat A je úplný, t.j. pre každý stav sú definované prechody pre všetky symboly.

Nech A je automat pre $a(a+b)^*$:



Potom automat komplementu, A^C je:



Dôkaz regularity prieniku

Nech L_1, L_2 sú regulárne jazyky. Chceme dokázať, že aj $L_1 \cap L_2$ je regulárny jazyk. Máme 2 možnosti:

- Platí, že $L_1 \cap L_2 = (L_1^C \cup L_2^C)^C$, t.j. prienik vieme dostať pomocou zjednotenia a doplnku. A keďže už vieme, že zjednotenie aj doplnik zachovávajú regularitu, aj prienik 2 regulárnych jazykov je teda regulárny jazyk.
- Použijeme konštrukciu konečného automatu pre jazyk $L_1 \cap L_2$ z ďalšieho slajdu.

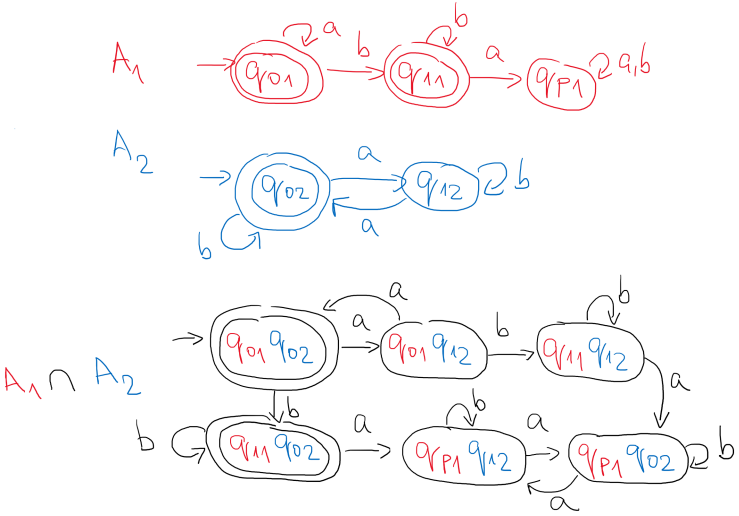
Konštrukcia DKA pre $L_1 \cap L_2$

Nech L_1, L_2 sú 2 regulárne jazyky nad abecedou Σ a A_1, A_2 sú ich DKA, t.j. $L(A_1) = L_1, L(A_2) = L_2$. $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$, $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

- Zostrojíme DKA, ktorý bude počas svojej činnosti sledovať, ako sa správajú paralelne DKA A_1 a A_2. Inými slovami, bude súčasne simulovať činnosť ako A_1, tak aj A_2.
- V situácií, že by A_1 aj A_2 súčasne akceptovali vstupný reťazec, bude aj výsledný automat signalizovať akceptáciu.
- Podobný princíp sme použili na prednáške číslo 2, kde sme vytvorili automat pre e-peňažný systém z automatov pre banku, obchod a zákazníka.
- Formálne zostrojíme automat: $(Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F_1 \times F_2)$, kde $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$

Príklad

Nech $L_1 = a^*b^*$, $L_2 = \{w \in \{a,b\}^* \mid \sharp_a(w) \equiv 0 \bmod 2\}$. Zostrojte DKA pre $L_1 \cap L_2$.



Dôkaz regularity rozdielu

Nech L_1, L_2 sú regulárne jazyky. Ak chceme dokázať, že ich (množinový) rozdiel $L_1 \setminus L_2$ je tiež regulárny jazyk, stačí nám ukázať, že rozdiel vieme popísať pomocou iných operácií, o ktorých vieme, že zachovávajú regularitu:

- Z teórie množín je zrejmé, že platí: $L_1 \setminus L_2 = L_1 \cap L_2^C$.
- Ak by sme teda napríklad chceli nájsť DKA pre $L_1 \setminus L_2$, tak najprv nájdeme DKA pre L_1, L_2.
- Následne nájdeme DKA pre L_2^C podľa návodu zo slajdov pre komplement.
- A následne zostrojíme DKA pre $L_1 \cap L_2^C$ podľa návodu zo slajdov pre prienik.

Súhrn modelov regulárnych jazykov

Povedali sme si, že regulárne jazyky vieme popísať:

- Deterministickým konečným automatom
- Nedeterministickým konečným automatom
- ε-nedeterministickým konečným automatom
- Regulárnym výrazom

Taktiež sme si povedali, akými rôznymi spôsobmi vieme medzi sebou tieto reprezentácie konvertovať:

- Subset-konštrukcia prevedie ε-NKA alebo NKA na DKA (prednášky 3, 4)
- Odstraňovanie stavov prevedie DKA na regulárny výraz (prednáška 5)
- Konverzia regulárneho výrazu na ε-NKA postupnou konštrukciou ε-NKA na základe operátorov zjednotenie, zreťazenie, iterácia v regulárnom výraze (prednáška 5)

Rozhodovacie problémy regulárnych jazykov

S teóriou formálnych jazykov úzko súvisí viacero rozhodovacích problémov. Medzi dôležité problémy s praktickými implikáciami patria aj tieto:

- Je jazyk L prázdny, t.j. $L = \emptyset$?
- Platí pre nejaký reťazec w a jazyk L, že w patrí do jazyka L ?
- Sú dva jazyky L_1, L_2 ekvivalentné? T.j. platí $L_1 = L_2$?

Tieto otázky majú zmysel najmä v prípade, že sa bavíme o nekonečných jazykoch. Existujú triedy jazykov, pre ktoré sa dá ukázať, že **neexistuje algoritmus**, ktorý by pre ne dané problémy vedel riešiť.

Avšak platí, že ak uvažujeme regulárne jazyky, tak všetky tri uvedené problémy **majú pre regulárne jazyky algoritmy, ktoré ich vedia riešiť**.

Rozhodovací problém prázdnosti (emptiness problem)

- "Emptiness problem" je rozhodovací problém formálneho jazyka, kde sa pre jazyk L pýtame, či je prázdny, t.j. či $L = \emptyset$?
- Máme k dispozícii reprezentáciu jazyka a pýtame sa, či vieme z danej reprezentácie zistiť, či je jazyk prázdny alebo nie.
- Pre regulárne jazyky sme doteraz uvažovali ich reprezentácie vo forme konečných automatov alebo regulárnych výrazov
- Dá sa ukázať, že pre regulárne jazyky **existuje algoritmus**, ktorý vie zistiť, či je daný jazyk prázdny alebo nie. Vychádzame z toho, že ku každému regulárnemu jazyku existuje DKA, ktorý ho vie akceptovať.

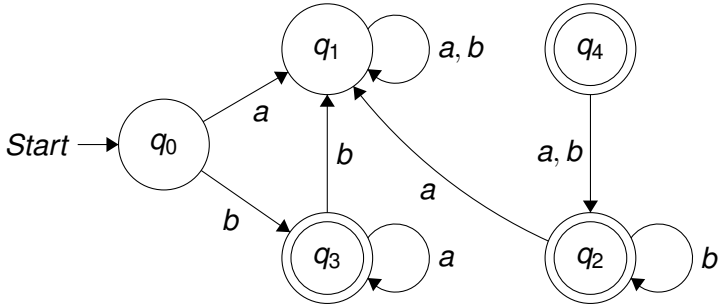
Riešenie emptiness problému z DKA

Ak máme daný nejaký regulárny jazyk konečným automatom, to, či je prázdny, zistíme nasledovným spôsobom:

- Aby jazyk **nebol prázdny**, musí **akceptovať nejaký reťazec**.
- Teda v DKA musí existovať **cesta z počiatočného stavu** do niektorého z **akceptačných stavov**.
- Jednoduchý rekurzívny algoritmus je riešenie tohto problému je nasledovný:
  1. **Základný krok:** Počiatočný stav je určite dosiahnuteľný z počiatočného stavu.
  2. **Rekurzívny krok:** Ak stav q je dosiahnuteľný z počiatočného stavu a existuje z neho prechod do stavu p (na ľubovoľný symbol), tak aj p je dosiahnuteľný z počiatočného stavu.
- Ak je v množine stavov dosiahnuteľných z počiatočného stavu aspoň jeden akceptačný stav, tak jazyk **nie je prázdny**. V opačnom prípade **je prázdny.**

Príklad

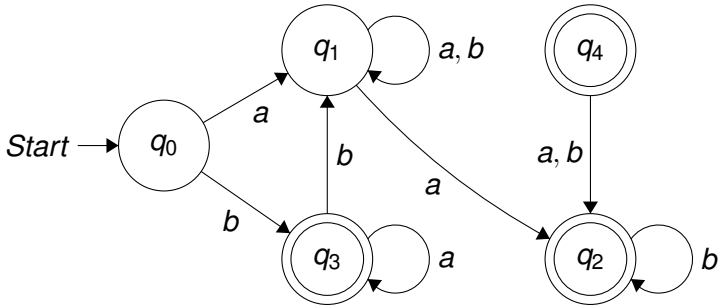
Zistite, či je jazyk akceptovaný nasledovným DKA prázdny:



- Je zrejmé, že je dosiahnuteľný počiatočný stav q_0.
- Z neho sú ďalej dosiahnuteľné stavy q_1, q_3. Z nich už ďalej nič.
- Dosiahnuteľné stavy z počiatočného stavu sú teda $\{q_0, q_1, q_3\}$. Keďže ani jeden akceptačný stav tam nie je, jazyk je **prázdny**.

Príklad

Zistite, či je jazyk akceptovaný nasledovným DKA prázdny:



- Je zrejmé, že je dosiahnuteľný počiatočný stav q_0.
- Z neho sú ďalej dosiahnuteľné stavy q_1, q_3. Z q_1 je ďalej dosiahnuteľný q_2.
- Dosiahnuteľné stavy z počiatočného stavu sú teda $\{q_0, q_1, q_2, q_3\}$. Keďže sa tam nachádza aj akceptačný stav q_2, jazyk **nie je prázdny**

Rozhodovací problém príslušnosti (membership problem)

- "Membership problem" je rozhodovací problém formálneho jazyka, kde sa pre jazyk L a reťazec w pýtame, či reťazec w patrí do jazyka L, t.j. či $w \in L$.
- V prípade, že je jazyk regulárny, znamená to, že je daný konečným automatom alebo regulárnym výrazom.
- V prípade, že je jazyk daný konečným automatom, tento problém vieme riešiť jednoduchým výpočtom príslušného konečného automatu, t.j. dáme na jeho vstup reťazec w a sledujeme, ako daný KA spracuje reťazec w a či skončí / neskončí v akceptačnom stave, a teda, či w patrí / nepatrí do jazyka.
- V prípade, že je jazyk daný regulárnym výrazom, vykonáme konverziu regulárneho výrazu na konečný automat a aplikujeme postup z predchádzajúceho bodu.

Rozhodovací problém ekvivalencie (equivalence problem)

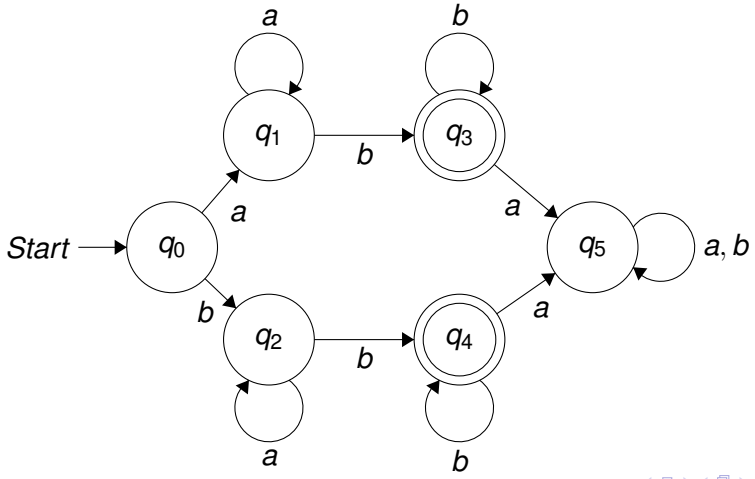
- "Equivalence problem" je rozhodovací problém, v ktorom sa pýtame, či 2 reprezentácie formálnych jazykov L_1 a L_2 popisujú ten istý jazyk, t.j. či $L_1 = L_2$.
- V prípade regulárnych jazykov platí, že tento problém **má algoritmus**, ktorý ho rieši.
- Riešenie tohto problému si ukážeme na situácii, že sú obe reprezentácie jazykov dané konečným automatom.
- V prípade, že by boli obe reprezentácie dané regulárnymi výrazmi, tak z nich najprv vyrobíme DKA a následne aplikujeme uvedený algoritmus.
- Najprv si však povedzme niečo o *ekvivalencii stavov DKA*.

Ekvivalencia stavov v DKA

- Uvažujme nasledovný problém: dá sa v nejakom DKA zistiť, či sa 2 rôzne stavy "správajú rovnako" z hľadiska akceptácie reťazcov?
- Ak áno, tak potom by sa 2 stavy nejakého deterministického konečného automatu p a q mohli nahradiť jedným stavom, ktorý sa bude správať ako *p a q súčasne*, pretože by to **nezmenilo jazyk akceptovaný automatom**.

Ekvivalencia stavov v DKA

Napríklad uvažujme nasledovný DKA:



Vidíme, že stavy q_3 a q_4 majú tú vlastnosť, že:

- Sú oba akceptačné
- Uvažujme, že sa počas výpočtu ocitneme v stave q_3 a chceme ešte dočítať do konca reťazec w.
- Uvažujme, že sa počas výpočtu ocitneme v stave q_4 a chceme ešte dočítať do konca reťazec w.
- **Oba stavy** majú tú vlastnosť, že:
  - Ak w je postupnosť b^*, tak po jej prečítaní budem stále v akceptačnom stave, t.j. celý vstup by som akceptoval.
  - Ak w je postupnosť s aspoň jedným a, tak prejdem do stavu q_5 a celý vstup by som neakceptoval.
- To znamená, že z hľadiska akceptácie reťazcov v celom DKA hrajú stavy q_3, q_4 **ekvivalentnú** úlohu. **Neexistuje reťazec** w, ktorý by mal tú vlastnosť, že ak ho spracujem z q_3, tak skončím v akceptačnom stave, ale ak by som ho spracoval z q_4 tak neskončím v akceptačnom stave - a naopak.

Keď viem, že stavy q_3 a q_4 sú ekvivalentné, môžem si položiť otázku, či náhodou v automate nie je ešte iná dvojica stavov, ktorá sa správa "rovnako". Napríklad q_1 a q_2:

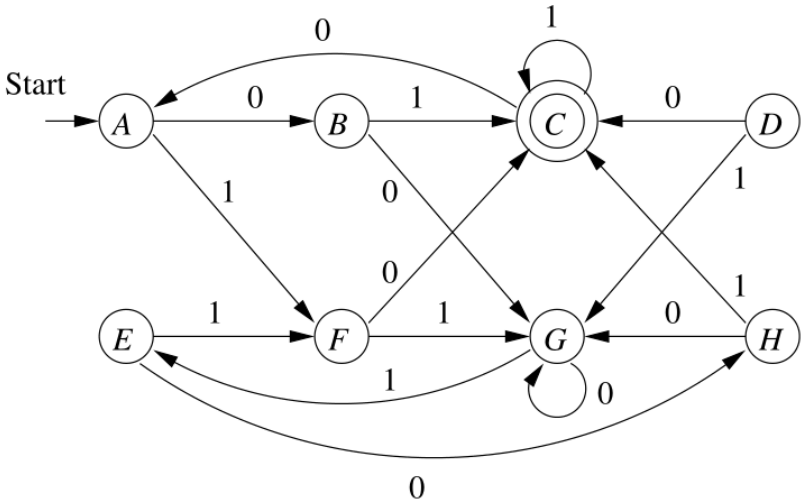
- Oba q_1, q_2 sú neakceptačné.
- Uvažujme, že sa počas výpočtu ocitneme v stave q_1 a chceme ešte dočítať do konca reťazec w.
- Uvažujme, že sa počas výpočtu ocitneme v stave q_2 a chceme ešte dočítať do konca reťazec w.
- **Oba stavy** majú tú vlastnosť, že:
  - Ak w je postupnosť a^*, tak po jej prečítaní budeme v oboch prípadoch v neakceptačnom stave, t.j. celý vstup by sme neakceptovali.
  - Ak w je postupnosť začínajúca a^*b tak z q_1 prejdeme do q_3 a z q_2 do q_4. Keďže vieme, že q_3 a q_4 sa správajú rovnako, tak pre w sa budú správať rovnako q_1 a q_2.
- To znamená, že z hľadiska akceptácie reťazcov v celom DKA hrajú stavy q_1, q_2 **ekvivalentnú** úlohu. **Neexistuje reťazec** w, ktorý by mal tú vlastnosť, že ak ho spracujem z q_1, tak skončím v akceptačnom stave, ale ak by som ho spracoval z q_2 tak neskončím v akceptačnom stave - a naopak.

Ekvivalencia stavov v DKA

- Dva stavy p a q DKA nazývame **ekvivalentné**, ak pre ne platí, že pre všetky vstupné reťazce w: $\hat{\delta}(p, w)$ je akceptačný stav vtedy a len vtedy, ak $\hat{\delta}(q, w)$ je akceptačný stav
- Inými slovami, **nedokážeme rozlíšiť** stavy p a q tým, že by alebo zo stavu p, alebo zo stavu q spustili výpočet pre reťazec w a pýtali sa, či sme skončili v akceptačnom stave, alebo nie.
- V takom prípade totižto oba stavy hrajú presne **tú istú úlohu** pri akceptácii reťazcov a teda je možné oba stavy nahradiť jedným, ktorý bude robiť to isté.
- Ak 2 stavy nie sú ekvivalentné, hovoríme, že sú **odlíšiteľné**. T.j. existuje aspoň jeden reťazec w, taký, že $\hat{\delta}(p, w)$ je akceptačný stav a $\hat{\delta}(q, w)$ nie je akceptačný stav, alebo naopak.

Príklad

Uvažujme DKA na obrázku ([1]):



Príklad

Z obrázka je zrejmé, že napríklad:

- Stavy A, C **nie sú ekvivalentné**, t.j. tú odlíšiteľné, pretože A nie je akceptačný stav a C je akceptačný stav, a teda pre prázdny reťazec $w = \varepsilon$ platí, že $\hat{\delta}(A, \varepsilon)$ nie je akceptačný stav a $\hat{\delta}(C, \varepsilon)$ je akceptačný stav.
- Podobne sú odlíšiteľné dvojice (B, C), (C, D), (C, E), (C, F), (C, G), (C, H).
- Dvojica (A, G) je neodlíšiteľná reťazcami $\varepsilon, 0, 1$, avšak je odlíšiteľná reťazcom 01, pretože $\hat{\delta}(A, 01)$ je akceptačný stav a $\hat{\delta}(G, 01)$ nie je akceptačný stav. Teda aj táto dvojica **nie je ekvivalentná**.

Príklad

Z obrázka je ďalej zrejmé, že napríklad:

- Dvojica (A, E):
  - nie je odlíšiteľná ε (lebo oba sú neakceptačné).
  - pre reťazce začínajúce jednotkou, $w = 1x$, bude určite $\hat{\delta}(A, 1x) = \hat{\delta}(E, 1x)$, teda ani tie neodlíšia stavy A, E
  - Reťazce začínajúce nulou síce urobia to, že zo stavu A sa po prvej nule automat dostane do B a zo stavu E sa dostane do H, avšak vidíme, že stavy B a H majú tú vlastnosť, že sa z nich ide do rovnakých stavov ako na 0, tak na 1, a teda v konečnom dôsledku neexistuje ani reťazec začínajúci nulou, ktorý by odlíšil stavy A a E
- Dvojica (A, E) je teda v tomto automate **ekvivalentná** a je teoreticky možné oba stavy nahradiť jedným stavom.
- Takto môžeme vyšetriť všetky dvojice stavov v automate.

Ekvivalencia stavov

Ak máme teda daný DKA $A = (Q, \Sigma, \delta, q_0, F)$, tak v ňom môžeme zistiť pre všetky dvojice stavov, či sú ekvivalentné alebo nie pomocou nasledovného algoritmu, zvaného aj *table-filling algorithm*:

1. **(Základný krok:)** Ak p je akceptačný stav a q nie je akceptačný stav, pár (p, q) **nie je ekvivalentný**
2. **(Rekurzívny krok:)** Nech p a q sú také stavy, že pre vstupný symbol a platí, že $r = \delta(p, a)$ a $s = \delta(q, a)$ sú nie sú ekvivalentné. Potom ani p a q nie sú ekvivalentné.

Rekurzívny krok musí platiť, pretože ak vieme, že r a s sú odlíšiteľné, a teda existuje reťazec w taký, že len 1 z dvojice $\hat{\delta}(r, w), \hat{\delta}(s, w)$ je akceptačný stav, potom logicky len 1 z dvojice $\hat{\delta}(p, aw), \hat{\delta}(q, aw)$ musí byť akceptačný stav.

Ekvivalencia stavov

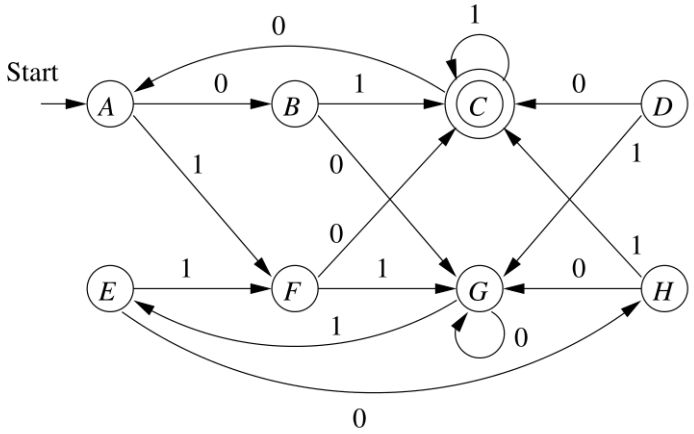
Dá sa formálne ukázať, uvedený algoritmus správne rozpozná v každom DKA ekvivalentné a odlíšiteľná stavy, t.j. že platí veta:

Veta

Ak table-filling algoritmus neodlíši 2 stavy, tieto 2 stavy sú ekvivalentné.

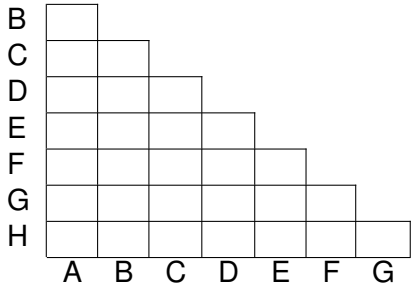
Príklad

Znovu uvažujme DKA:

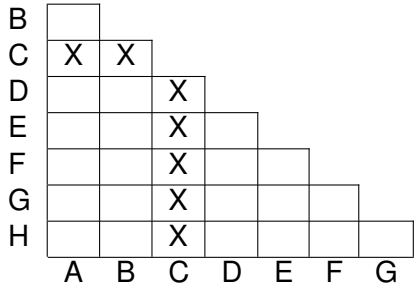


Nájdeme dvojice ekvivalentných stavov.

Budeme vypĺňať nasledovnú tabuľku, do ktorej budeme značiť, ktoré stavy sú odlíšiteľné (preto *table-filling algorithm*):



Najprv vyplníme tie stavy, o ktorých vieme, že sú určite odlíšiteľné - akceptačný stav C od neakceptačných stavov:



Teraz vezmeme nejaký pár, o ktorom sme zistili, že je odlíšiteľný - napríklad (B, C) a v DKA sledujeme:

- Z akých stavov ideme do B na symbol 0: A
- Z akých stavov ideme do C na symbol 0: D, F.
- Keďže (B, C) sú odlíšiteľné, budú odlíšiteľné aj (A, D) a (A, F).

- Z akých stavov ideme do B na symbol 1: žiadne
- Z akých stavov ideme do C na symbol 1: B, C, H.
- Keďže do B na symbol 1 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Zistenia zapíšeme do tabuľky.

Zároveň v tabuľke označím pár, ktorý sme už "vyšetrili", t.j. (B, C).

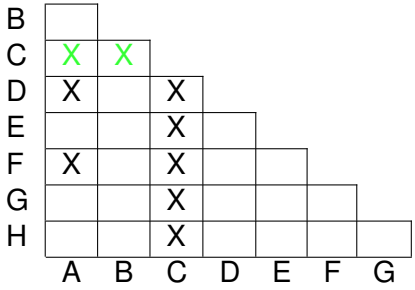
	A	B	C	D	E	F	G
B							
C	X	X					
D	X		X				
E			X				
F	X		X				
G			X				
H			X				

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (A, C) a v DKA sledujeme:

- Z akých stavov ideme do A na symbol 0: C
- Z akých stavov ideme do C na symbol 0: D, F.
- Keďže (A, C) sú odlíšiteľné, budú odlíšiteľné aj (C, D) a (C, F). To však už vieme...

- Z akých stavov ideme do A na symbol 1: žiadne
- Z akých stavov ideme do C na symbol 1: B, C, H.
- Keďže do A na symbol 1 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Nič sme nezistili, akurát si označíme, že sme vyšetrili dvojicu (A, C).

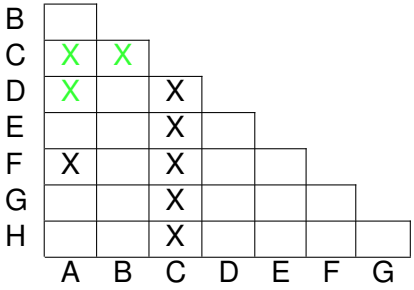

B
C X X
D X X
E X
F X X
G X
H X
A B C D E F G

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (A, D) a v DKA sledujeme:

- Z akých stavov ideme do A na symbol 0: C
- Z akých stavov ideme do D na symbol 0: žiadne
- Keďže do D na symbol 0 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Z akých stavov ideme do A na symbol 1: žiadne
- Z akých stavov ideme do D na symbol 1: žiadne
- Keďže do A na symbol 1 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Nič sme nezistili, akurát si označíme, že sme vyšetrili dvojicu (A, D).

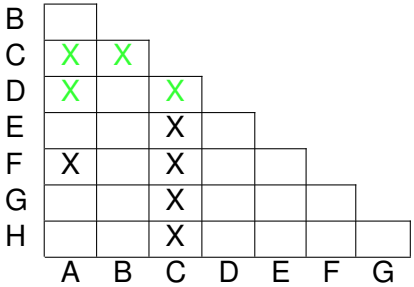

B
C
D
E
F
G
H
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
X
A
B
C
D
E
F
G

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (C, D) a v DKA sledujeme:

- Z akých stavov ideme do C na symbol 0: D, F
- Z akých stavov ideme do D na symbol 0: žiadne
- Keďže do D na symbol 0 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Z akých stavov ideme do C na symbol 1: B, C, H
- Z akých stavov ideme do D na symbol 1: žiadne
- Keďže do D na symbol 1 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Nič sme nezistili, akurát si označíme, že sme vyšetrili dvojicu (C, D).


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

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (C, E) a v DKA sledujeme:

- Z akých stavov ideme do C na symbol 0: D, F
- Z akých stavov ideme do E na symbol 0: žiadne
- Keďže do E na symbol 0 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Z akých stavov ideme do C na symbol 1: B, C, H
- Z akých stavov ideme do E na symbol 1: G
- Našli sme teda odlíšiteľné dvojice stavov (B, G), (C, G) a (G, H). Tie, ktoré ešte nemáme, si zaznačíme do tabuľky.

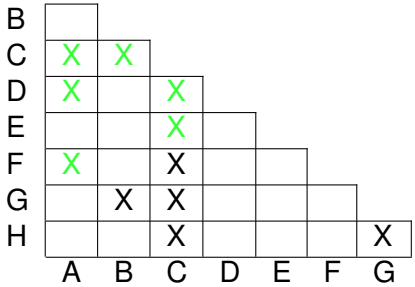
	A	B	C	D	E	F	G
B							
C	X	X					
D	X		X				
E			X				
F	X		X				
G		X	X				
H			X				X

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (A, F) a v DKA sledujeme:

- Z akých stavov ideme do A na symbol 0: C
- Z akých stavov ideme do F na symbol 0: žiadne
- Keďže do F na symbol 0 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

- Z akých stavov ideme do A na symbol 1: žiadne
- Z akých stavov ideme do F na symbol 1: A, E
- Keďže do A na symbol 1 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

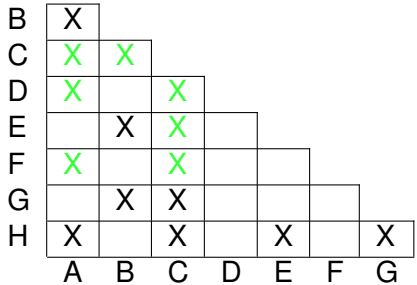
- Tak si aspoň v tabuľke zaznačíme, že sme vyšetrili (A, F).


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

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (C, F) a v DKA sledujeme:

- Z akých stavov ideme do C na symbol 0: D, F
- Z akých stavov ideme do F na symbol 0: žiadne
- Keďže do F na symbol 0 nejdeme zo žiadneho stavu, v tomto kroku sme nezistili nič.

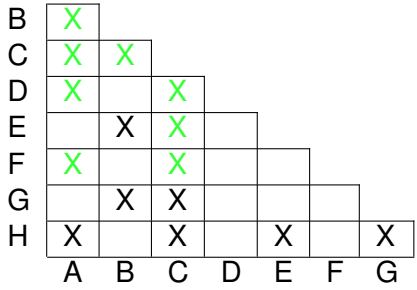
- Z akých stavov ideme do C na symbol 1: B, C, H
- Z akých stavov ideme do F na symbol 1: A, E
- Dostávame odlíšiteľné páry $(A, B), (B, E), (A, C), (C, E), (A, H), (E, H)$. Ktoré ešte nemáme v tabuľke, tak zaznačíme.


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

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (A, B) a v DKA sledujeme:

- Z akých stavov ideme do A na symbol 0: C
- Z akých stavov ideme do B na symbol 0: A
- Odlíšiteľné sú teda (A, C). Ale to už vieme.

- Z akých stavov ideme do A na symbol 1: žiadne
- Z akých stavov ideme do B na symbol 1: žiadne
- Nič sme nezistili. Tak aspoň zaznačíme (A, B) ako vyšetrený.


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

Vyšetrením páru (B, E) nedostaneme nič nové, lebo do E nejdú prechody na 0 a zase do B nejdú prechody na 1:

	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

Vyšetrením páru (B, G) dostaneme pre symbol 0 odlíšiteľné páry $(A, B), (A, G), (A, H)$ Z nich (A, G) ešte nemáme. Pre symbol 1 nedostaneme nič. Všetko zaznačíme v tabuľke.

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

Vyšetrením páru (A, G) dostaneme pre symbol 0 odlíšiteľné páry $(C, G), (B, C), (C, H)$. Tu teda nič nové. Pre symbol 1 nedostaneme nič. Všetko zaznačíme v tabuľke.

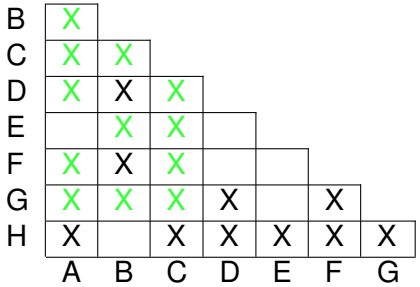
	A	B	C	D	E	F	G
B	X						
C	X	X					
D	X		X				
E		X	X				
F	X		X				
G	X	X	X				
H	X		X		X		X

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (C, G) a v DKA sledujeme:

- Z akých stavov ideme do C na symbol 0: D, F
- Z akých stavov ideme do G na symbol 0: B, G, H
- Odlíšiteľné sú teda $(B, D), (B, F), (D, G), (F, G), (D, H), (F, H)$.

- Z akých stavov ideme do C na symbol 1: B, C, H
- Z akých stavov ideme do G na symbol 1: D, F
- Odlíšiteľné sú $(B, D), (B, F), (C, D), (C, F), (D, H), (F, H)$.

- Zaznačíme si nové odlíšiteľné stavy a vyšetrený (C, G).


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

Vyšetrením (B, D) nezískame nič nové.

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

Vyšetrením (B, F) nezískame nič nové.

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

Vyšetrením (D, G), resp. (D, H) nezískame nič nové.

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

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (F, G) a v DKA sledujeme:

- Z akých stavov ideme do F na symbol 0: žiadne
- Z akých stavov ideme do G na symbol 0: B, G, H
- Tu nezistím nič.

- Z akých stavov ideme do F na symbol 1: A, E
- Z akých stavov ideme do G na symbol 1: D, F
- Odlíšiteľné sú $(A, D), (D, E), (A, F), (E, F)$.

- Zaznačíme si nové odlíšiteľné stavy a vyšetrený (F, 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	
H	X		X	X	X	X	X

Vezmime ďalší nevyšetrený odlíšiteľný pár - napríklad (E, F) a v DKA sledujeme:

- Z akých stavov ideme do E na symbol 0: žiadne
- Z akých stavov ideme do F na symbol 0: žiadne
- Tu nezistíme nič.

- Z akých stavov ideme do E na symbol 1: G
- Z akých stavov ideme do F na symbol 1: E, A
- Odlíšiteľné sú $(E, G), (A, G)$.

- Zaznačíme si nové odlíšiteľné stavy a vyšetrený (E, 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	X	X	
H	X		X	X	X	X	X

Ďalej by sme zistili, že analýzou ostatných párov nám už nepribudnú žiadne nové odlíšiteľné stavy. Tabuľka by teda mala finálnu podobu:

	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

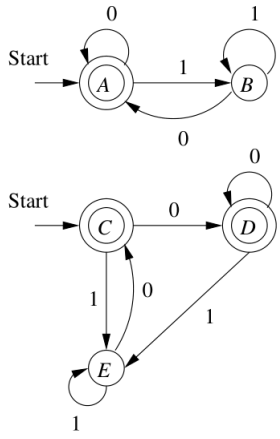
V tabuľke je teda signalizované, že páry (A, E), (B, H) a (D, F) predstavujú ekvivalentné stavy.

Rozhodovací problém ekvivalencie (equivalence problem)

- Vráťme sa teraz k problému ekvivalencie. Uvažujme, že máme 2 DKA A_1, A_2, ktoré akceptujú nejaké jazyky L_1, L_2. Chceme zistiť, či sú jazyky oboch automatov ekvivalentné, t.j. $L(A_1) = L(A_2)$.
- Na riešenie tejto úlohy vieme použiť algoritmus ekvivalencie stavov.
- Uvažujme, že zostrojíme z automatov A_1, A_2 väčší automat tak, že len zjednotíme stavy oboch automatov. Zachováme prechodové funkcie, počiatočné a akceptačné stavy.
- V tomto novom automate vyšetríme ekvivalencie stavov. Ak nastane situácia, že počiatočný stav q_{01} automatu A_1 bude ekvivalentný počiatočnému stavu q_{02} automatu A_2, znamená to podľa definície, že $\hat{\delta}(q_{01}, w)$ je akceptačný stav práve pre tie isté reťazce, ako je $\hat{\delta}(q_{02}, w)$. To však logicky znamená, že akceptujú práve tie isté reťazce.

Príklad

Chceme zistiť, či automaty na obrázku ([1]) akceptujú ten istý jazyk. Prvý automat pozostáva zo stavov A, B, druhý zo stavov C, D, E



Tvárime sa, že sú oba súčasťou jedného veľkého automatu a spustíme algoritmus na hľadanie ekvivalentných stavov. Výsledkom je tabuľka:

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

Z tabuľky vidíme, že oba počiatočné stavy oboch automatov boli označené ako ekvivalentné (v tabuľke nie je x pri stavoch A a C). Znamená to teda, že oba automaty sú **ekvivalentné**, t.j. akceptujú ten istý jazyk.

Minimalizácia konečných automatov

- Ekvivalencia stavov má okrem ekvivalencie automatov ďalšie dôležité použitie.
- Umožňuje konštrukciu **minimálneho** DKA, t.j. konečného automatu, ktorý má najmenší možný počet stavov.
- Ak zistíme, ktoré automaty v DKA sú ekvivalentné, môžeme ich nahradiť jedným stavom, čím sa nám podarí znížiť počet stavov.
- Problém nájdenia minimálneho automatu má praktický význam všade tam, kde je potrebné optimalizovať zdroje.

Minimalizácia konečných automatov

Algoritmus minimalizácie zadaného DKA má 2 hlavné kroky:

1. Najprv sa odstránia z automatu všetky nedosiahnuteľné stavy.
2. Následne sa pre automat zostrojí tabuľka ekvivalentných stavov. Tie stavy, ktoré sú označené ako ekvivalentné, sa nahradia jedným novým stavom.

Ekvivalencia stavov

Uvádzame tu zaujímavý fakt pre znalcov diskrétnej matematiky: ak si zadefinujeme reláciu na množine stavov tak, že 2 stavy sú v relácii vtedy, ak sú ekvivalentné, tak táto relácia je:

- reflexívna
- symetrická
- tranzitívna

Preto triedy ekvivalencie stavov tvoria aj rozklad množiny stavov.

Ekvivalentné stavy sa na základe toho, ktorý je s ktorým ekvivalentný, dajú zoskupiť do blokov. V našom príklade:

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

vzniknú bloky: $\{A, C, D\}, \{B, E\}$.

V prvom príklade sme mali ekvivalenciu stavov:

	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

t.j. vzniknú bloky: $\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}$

Algoritmus minimalizácie DKA

Nech je daný DKA $A = (Q, \Sigma, \delta, q_0, F)$, ktorý chceme minimalizovať, t.j. nájsť k nemu ekvivalentný minimálny automat B.

1. Použi table-filling algoritmus na nájdenie tabuľky ekvivalencie stavov.
2. Rozdeľ množinu Q stavov do blokov vzájomne ekvivalentných stavov podľa ekvivalencií.
3. Stavy minimálneho automatu budú predstavovať bloky ekvivalentných stavov. Prechodová funkcia γ je definovaná nasledovne. Nech S je blok ekvivalentných stavov, a je vstupný symbol. Musí existovať blok T taký, že $\delta(p, a) \in T$ pre všetky $p \in S$. Potom $\gamma(S, a) = T$.
4. Počiatočný stav B je ten blok, ktorý obsahuje počiatočný stav A.
5. Akceptačné stavy B sú tie bloky, ktoré obsahujú akceptačné stavy A.

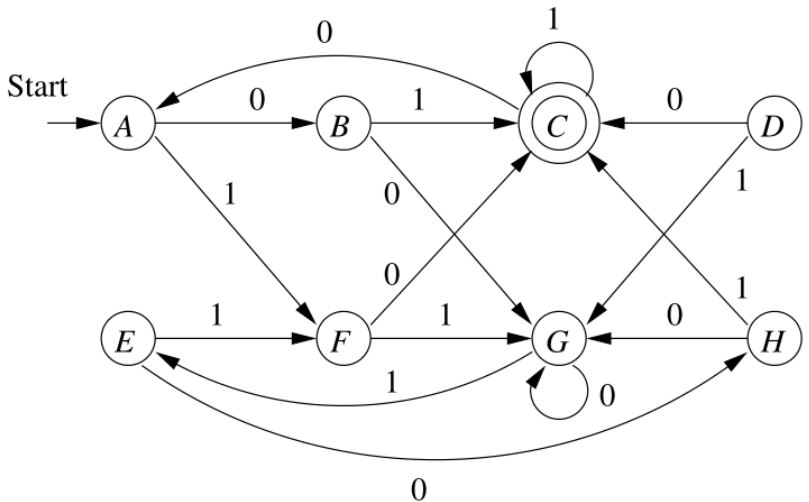
Minimalizácia

Inými slovami, stavy automatu sa nahradia blokmi ekvivalentných stavov, pričom sa príslušne upraví aj prechodová funkcia.

Všetko sa teda udeje na základe ekvivalencie stavov.

Príklad

Minimalizujte DKA:



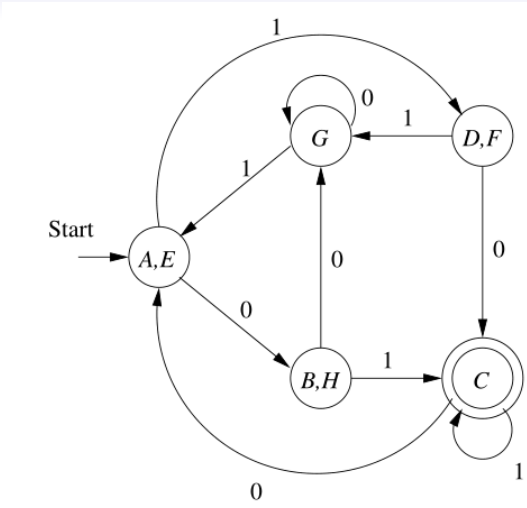
Tento automat neobsahuje nedosiahnuteľné stavy. Preto pristúpime ku konštrukcii tabuľky ekvivalentných stavov (tú sme už robili, preto ju tu rovno uvediem):

	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

Z tabuľky je zrejmé, že bloky ekvivalentných stavov sú:

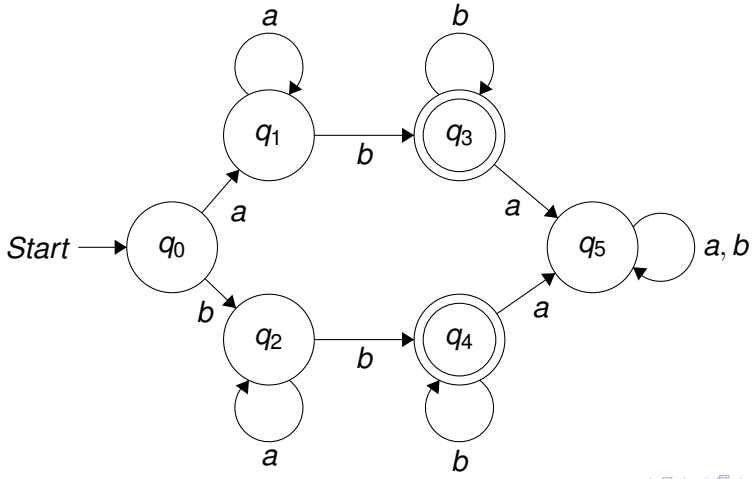
- $\{A, E\}$
- $\{B, H\}$
- $\{D, F\}$
- $\{C\}$
- $\{G\}$.

Tieto bloky budú teda predstavovať stavy minimálneho automatu. Počiatočný stav bude blok s počiatočným stavom, t.j. $\{A, E\}$, akceptačné stavy budú bloky s akceptačnými stavmi, t.j. $\{C\}$. Výsledný minimálny automat bude ([1]):


Start
A,E
G
D,F
B,H
C
0
1
0
1
1
0
0
1
0
1

Príklad

Minimalizujte automat:



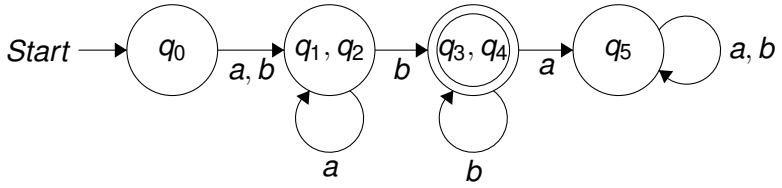
Tabuľka ekvivalencií stavov by vyzerala nasledovne:

q1	X				
q2	X				
q3	X	X	X		
q4	X	X	X		
q5	X	X	X	X	X
	q0	q1	q2	q3	q4

Teda bloky ekvivalentných stavov sú:

- $\{q_0\}$
- $\{q_1, q_2\}$
- $\{q_3, q_4\}$
- $\{q_5\}$

A výsledný minimálny DKA:



Použitá literatúra

1 Hopcroft, Motwani, Ullman - Introduction to Automata Theory, Languages and Computations, 3rd Ed.