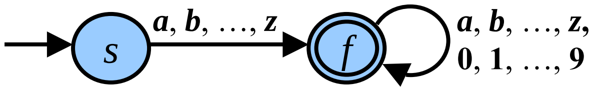
Příklady pro cvičení 2. z IFJ:
Konečné automaty, regulární výrazy

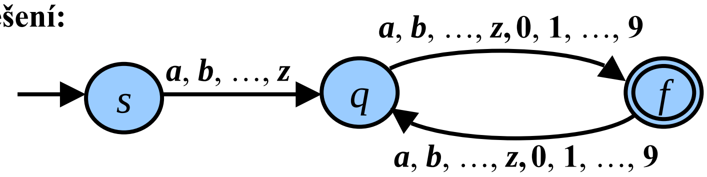
Příklad 1.
Vytvořte konečný automat, který přijímá všechny identifikátory.

Řešení:



Příklad 2.
Vytvořte konečný automat, který přijímá všechny identifikátory sudé délky.

Řešení:



Příklad 3.
Vytvořte konečný automat *M* přijímající všechny řetězce nad abecedou $\Sigma = \{a, b\}$, které neobsahují podřetězec *ab* a zároveň jejich délka je dělitelná třemi.
Formálně: $L(M) = \{x\text{: } ab \text{ is not a substring of } x \wedge |x| \bmod 3 = 0\}$

Řešení:
Konstrukce tohoto automatu je složitější, protože musíme zaručit splnění dvou podmínek. Veškeré potřebné informace o dosud přijaté části řetězce musíme vhodně zakódovat do stavů automatu. Všechny stavy označíme ve tvaru <***x***, ***y***>, kde část ***x*** bude hlídat, zda dosud přijatý řetězec neobsahoval podřetězec *ab*, a část *y* bude hlídat dělitelnost délky řetězce číslem 3.

část ***x*** tedy bude nabývat jedné z hodnot:

- — ... nebyl načten podřetězec ab, ani jeho část.
- ***a*** ... právě byl načten znak *a*, takže musíme hlídat, aby nenásledoval znak b!
- ***ab*** ... již byl načten podřetězec *ab*, takže řetězec nebude přijat!

a část ***y*** tedy bude nabývat jedné z hodnot:

- **0** ... délka dosud přečteného řetězce je dělitelná číslem 3.
- **1** ... délka dosud přečteného řetězce dává po dělení číslem 3 zbytek 1.
- **2** ... délka dosud přečteného řetězce dává po dělení číslem 3 zbytek 2.

Množina všech stavů tedy je:
Q = {<—, **0**>, <—, **1**>, <—, **2**>, <***a***, **0**>, <***a***, **1**>, <***a***, **2**>, <***ab***, **0**>, <***ab***, **1**>, <***ab***, **2**> }

Startujícím stavem bude zřejmě stav <—, **0**>, protože dosud přijatý je prázdný řetězec, který zřejmě podřetězec *ab* a ani jeho část neobsahoval a délka prázdného řetězce je 0, tedy je dělitelná číslem 3.

Množina všech koncových stavů bude obsahovat ty stavy, ve kterých řetězec neobsahoval podřetězec *ab* a dále pokud je současná délka přečteného řetězce dělitelná třemi, tedy:
F = {<—, 0>, <*a*, 0>}

Celý automat je potom vhodné popsat tabulkou:

	a	*b*
<—, 0>	{ <*a*, 1> }	{ <—, 1> }
<—, 1>	{ <*a*, 2> }	{ <—, 2> }
<—, 2>	{ <*a*, 0> }	{ <—, 0> }
<*a*, 0>	{ <*a*, 1> }	{ <*ab*, 1> }
<*a*, 1>	{ <*a*, 2> }	{ <*ab*, 2> }
<*a*, 2>	{ <*a*, 0> }	{ <*ab*, 0> }
<*ab*, 0>	{<*ab*, 1> }	{<*ab*, 1> }
<*ab*, 1>	{<*ab*, 2> }	{<*ab*, 2> }
<*ab*, 2>	{<*ab*, 0> }	{<*ab*, 0> }

Způsob efektivnějšího řešení:

- Nezavedeme na začátku komplet všechny stavy. Může se někdy stát, že touto metodou zavedeme i stavy, které nikdy nebudou dostupné. Nové stavy budeme zavádět tehdy, pokud do těchto stavů vznikne v průběhu vytváření tabulky nový přechod.
- Pokud již v průběhu čtení řetězce víme, že řetězec nebude přijat, necháme přejít automat do stavu nekoncového **<false>**, ve kterým přetrvá až do konce čtení řetězce a tím řetězec nebude přijat.

Popis automatu s méně stavy:

	a	*b*
<—, 0>	{ <*a*, 1> }	{ <—, 1> }
<*a*, 1>	{ <*a*, 2> }	{ <false> }
<—, 1>	{ <*a*, 2> }	{ <—, 2> }
<*a*, 2>	{ <*a*, 0> }	{ < false> }
<—, 2>	{ <*a*, 0> }	{ <—, 0> }
<*a*, 0>	{ <*a*, 1> }	{ < false> }
< false>	{ < false> }	{ < false> }

Příklad 4.
Vytvořte konečný automat *M* přijímající všechny řetězce nad abecedou Σ = {*a*, *b*, 0}, které jsou sudé délky, neobsahují tři po sobě jdoucí písmena a obsahují lichý počet výskytů číslice 0.

Řešení:
Postupovat budeme obdobně jako v příkladu 3:
Všechny stavy označíme ve tvaru <***x***, ***y***, ***z***>, kde část ***x*** bude hlídat, zda dosud přijatý řetězec je sudé délky, část *y* bude hlídat, zda dosud přijatý řetězec neobsahoval tři po sobě jdoucí písmena a část *z* bude hlídat, zda dosud přijatý řetězec obsahoval lichý počet nul.

část ***x*** bude nabývat jedné z hodnot:

- ***ld*** ... délka dosud přijatého řetězce je lichá.
- ***sd*** ... délka dosud přijatého řetězce je sudá.

část *y* bude nabývat jedné z hodnot:

- ***0p*** … dosud přečtený řetězec nekončí písmenem.
- ***1p*** … dosud přečtený řetězec končí jedním písmenem.
- ***2p*** … dosud přečtený řetězec končí dvěma písmeny.

část *z* bude nabývat jedné z hodnot:

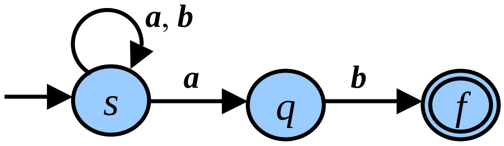
- *s0* … dosud přečtený řetězec obsahuje sudý počet nul.
- *l0* … dosud přečtený řetězec obsahuje lichý počet nul.

Celý automat potom můžeme popsat následovně:

	a, b	***0***
<sd, 0p, s0>	{ <ld, 1p, s0> }	{ <ld, 0p, l0> }
<ld, 1p, s0>	{ <sd, 2p, s0> }	{ <sd, 0p, l0> }
<ld, 0p, l0>	{ <sd, 1p, l0> }	{ <sd, 0p, s0> }
<sd, 2p, s0>	{ <false> }	{ <ld, 0p, 10> }
<sd, 0p, l0>	{ <ld, 1p, l0> }	{ <ld, 0p, s0> }
<sd, 1p, l0>	{ <ld, 2p, l0> }	{ <ld, 0p, s0> }
<ld, 1p, l0>	{ <sd, 2p, l0> }	{ <sd, 0p, s0> }
<ld, 0p, s0>	{ <sd, 1p, s0> }	{ <sd, 0p, l0> }
<ld, 2p, l0>	{ <false> }	{ <sd, 0p, s0> }
<sd, 2p, l0>	{ <false> }	{ <ld, 0p, s0> }
<sd, 1p, s0>	{ <ld, 2p, s0> }	{ <ld, 0p, l0> }
<ld, 2p, s0>	{ <false> }	{ <sd, 0p, l0> }
<false>	{ <false> }	{ <false> }

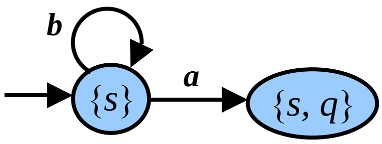
Příklad 5.
Vytvořte konečný automat *M* přijímající všechny řetězce nad abecedou Σ = {*a*, *b*}, které končí řetězcem *ab*. Formálně: *L*(*M*) = {*x*: *ab* is a suffix of *x*}. Daný automat převeďte na deterministický.

Řešení:



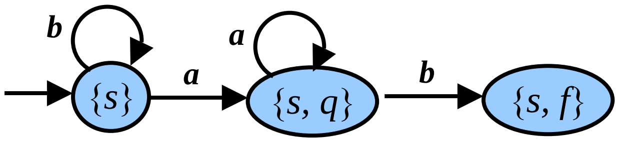
Postupná konstrukce deterministického automatu:

- Začneme konstruovat automat ze startovacího stavu *s*, který označme jako {*s*}. Ze stavu {*s*} můžeme přejít při přečtení symbolu *b* do pouze do stavu {*s*} ale při přečtení symbolu *a* do dvou stavů *s* a *q*. Vytvoříme tedy nový stav {*s*, *q*}, do kterého vstoupíme ze stavu {*s*} při přečtení symbolu *a*:

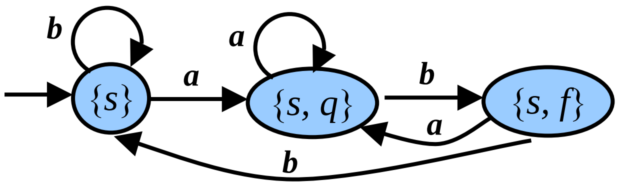


- Nyní musíme pro nový stav {*s*, *q*} vytvořit všechny přechody a to tak, že postupně pro všechny symboly vstupní abecedy = {*a*, *b*} budeme vytvářet nové stavy, které vniknou sloučením jednoho a více stavů z původního nedeterministickém automatu a to následovně:

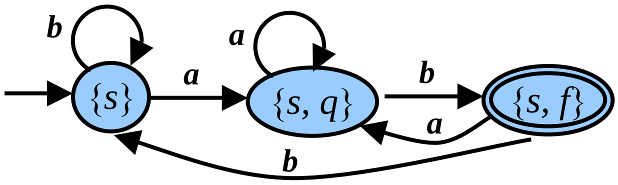
Vezmeme první symbol ze vstupní abecedy = *a*. Stav {*s*, *q*} se skládá ze stavů *s*, *q*, tedy zjistíme, do kterých stavů jsme se mohli dostat v původním nedeterministickém automatu ze stavů *s*, *q* při přečtení symbolu *a*. Zjistíme, že ze stavu *s* jsme mohli přejít do stavů *s*, *q* a ze stavu *q* nemůžeme nikam. Celkem jsme se tedy mohli dostat opět do stavů *s*, *q*, vytvoříme tedy v automatu novou hranu označenou symbolem *a* do stavu {*s*, *q*}, který již existuje. Obdobně pro *b* zjistíme, že ze stavů *s*, *q* můžeme přejít do stavů *s*, *f*, stav {*s*, *f*} ale ještě neexistuje, proto jej musíme vytvořit jako nový:



- Obdobným způsobem přidáme hrany novému stavu {*s*, *f*}:



- Až dospějeme do situace, kdy jsme ke všem stavům již všechny hrany přidali a přitom nevznikl žádný nový stav (což nastalo nyní), máme konstrukci hotovou. Koncové stavy jsou ty stavy, které „obsahují" aspoň jeden koncový stav z původního automatu. Výsledný deterministický automat tedy je:



Příklad 6.
Vytvořte regulární výraz popisující všechny identifikátory.

Řešení:
- Nechť výraz *l* reprezentuje regulární výraz $a + b + \ldots + z$.
- Nechť výraz *d* reprezentuje regulární výraz $0 + 1 + \ldots + 9$.

Potom regulární výraz $l(l + d)^*$ reprezentuje všechny identifikátory.

Příklad 7.
Vytvořte regulární výraz popisující všechny identifikátory sudé délky.

Řešení:
- Nechť výraz *l* reprezentuje regulární výraz $a + b + \ldots + z$.
- Nechť výraz *d* reprezentuje regulární výraz $0 + 1 + \ldots + 9$.

Potom regulární výraz $l(l + d)((l + d)(l + d))^*$ reprezentuje všechny identifikátory sudé délky.

Příklad 8.
Vytvořte regulární výraz r nad abecedou $\Sigma = \{a, b\}$, který popisuje jazyk všech řetězců obsahující podřetězec aa. Formálně: $L(r) = \{x\text{: } aa \text{ is substring of } x\}$

Řešení:
Regulární výraz $r = (a + b)^*aa(a + b)^*$ popisuje tento jazyk.

Příklad 9.
Vytvořte regulární výraz r nad abecedou $\Sigma = \{a, b\}$, který popisuje jazyk všech řetězců neobsahující aa. Formálně: $L(r) = \{x\text{: } aa \text{ is not substring of } x\}$

Řešení:
Je zřejmé, že můžeme libovolně za sebe kopírovat symbol (respektive řetězec) b, aniž bychom vytvořili podřetězec aa. Pokud chceme vložit symbol a, musíme nejprve zaručit, že za symbolem a se nebude nacházet další symbol a, tedy bude se nacházet symbol b. To zaručíme tak, že za symbol a pevně „vnutíme" symbol b. Tedy můžeme jakýmsi způsobem za sebe libovolně skládat řetězce b a ab, aniž bychom vytvořili řetězec se dvěma symboly a za sebou. Regulárním výrazem můžeme popsat tuto skutečnost jako: $(b + ab)^*$
Tento regulární výraz však ještě není přesný – pokud chceme vložit symbol a, pevně vnutí symbol b. Tímto regulárním výrazem tedy ještě nepopisujeme žádný řetězec, který končí symbolem a. Například ba je zřejmě korektně vytvořený řetězec neobsahují podřetězec aa. To můžeme ošetřit například tak, že za již vytvořený regulární výraz přidáme $(\varepsilon + a)$, což nám říká, že na konec vytvořeného řetězce máme právo buď přidat pouze jeden! symbol a a nebo nic.
Výsledný regulární výraz tedy je: $r = (b + ab)^*(\varepsilon + a)$

Příklad 10.
Vytvořte regulární výraz r nad abecedou $\Sigma = \{a, b, c\}$, který popisuje jazyk všech řetězců neobsahující aba. Formálně: $L(r) = \{x\text{: } aba \text{ is not substring of } x\}$

Řešení:
Postupovat budeme obdobně jako v příkladu 8:

Řešení „hlavního těla" výrazu:
- Řetězece b a c můžeme libovolně za sebe kopírovat, aniž by vznikl podřetězec aba.
- Pokud je vložen jeden nebo i více po sobě symbolů a, musí za nimi následovat: řetězec c nebo řetězec bb nebo řetězec bc

Regulárním výrazem můžeme popsat tuto skutečnost jako: $(b + c + a^+c + a^+bb + a^+bc)^*$

Řešení „konce" výrazu:
Nyní procházejme jednotlivé elementy těla výrazu:
- a^+c – vnuceno c, ale můžeme zakončit pouze sekvencí symbolů a, tj. a^+
- a^+bb – vnuceno bb, ale můžeme zakončit pouze sekvencí symbolů a ukončenou jedním nebo žádným symbolem b, tj. a^+b nebo a^+
- a^+bc – vnuceno bc, ale můžeme zakončit pouze sekvencí symbolů a ukončenou jedním nebo žádným symbolem b, tj. a^+b nebo a^+

Tedy řetězec můžeme zakončit I. prázdným řetězcem, II. sekvencí symbolů a, III. Sekvencí symbolů a následovanou jedním symbolem b. Regulárním výrazem můžeme popsat tuto skutečnost jako: $(\varepsilon + a^+ + a^+b)$, což lze zjednodušit na: $(a^* + a^+b)$.

Výsledný regulární výraz tedy je: $r = (b + c + a^+c + a^+bb + a^+bc)^*(a^* + a^+b)$