

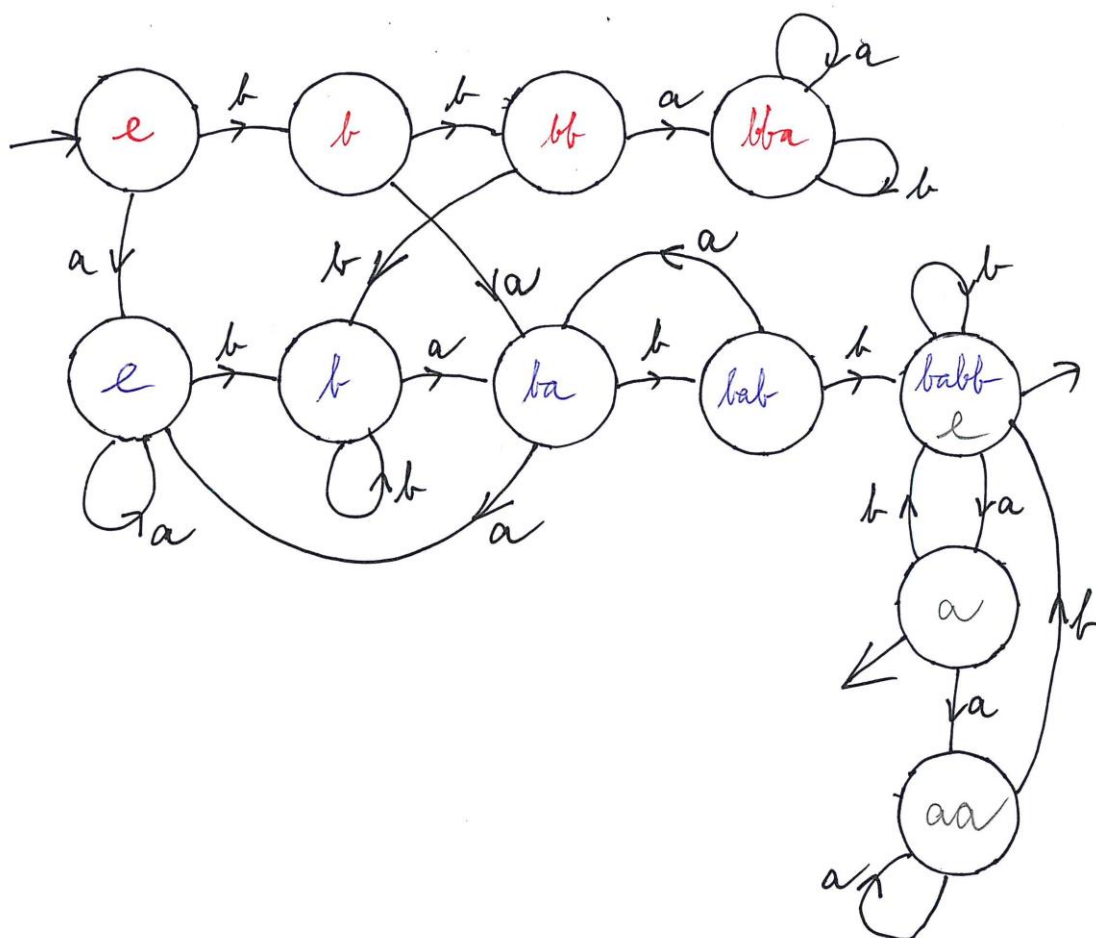
Navrhování rozpoznávacích automatů podruhé

Pro připomenutí – poslední příklad z minulého cvičení

Příklad 8.

L_8 je množina všech řetězců, které současně splňují všechny následující podmínky:

- řetězec nezačíná *bba*
- řetězec obsahuje *babb*
- řetězec nekončí *aa*



Příklad 9.

L_9 je množina všech řetězců nad abecedou $\{0,1\}$, které současně splňují následující podmínky:

- řetězec obsahuje podřetězec 1101 (podmínka 1)
- řetězec neobsahuje podřetězec 001 (podmínka 2)

Je zřejmé, že na rozdíl od předchozího příkladu nelze postupovat tak, že nejprve budeme vyhledávat podřetězec z první podmínky a až po něm podřetězec z druhé podmínky (na pořadí, v němž se hledané podřetězce objeví, v tomto případě nezáleží).

Úvaha: kdybychom měli pro každou z výše uvedených podmínek zkonstruovaný jeden KA, splňoval by vstupní řetězec obě podmínky právě tehdy, když by oba dva KA převedl do koncových stavů.

Nápad = návod: vytvoříme pro každou podmínku jeden KA a poté na jejich základě zkonstruujeme KA, který si bude „současně pamatovat stavy obou výše zmíněných KA“, tj. vytvoříme finální KA jako *kartézský součin dvou automatů*.

Vytvoření kartézského součinu automatů:

KA, které „separátně“ řeší „každou podmínku zvlášť“, budeme nazývat *parciální automaty*. Výsledný automat si musí „pamatovat“ současně stavy obou parciálních automatů. Stavy výsledného automatu tedy budou uspořádané dvojice stavů obou parciálních automatů. Protože je v každém parciálním automatu pro každou dvojici (stav, vstupní symbol) jednoznačně definován následující stav, je tomu tak i u výsledného automatu. Obecně – jestliže platí $\delta_1(q_{1,i}, a) = q_{1,j}$ a $\delta_2(q_{2,k}, a) = q_{2,l}$, pak pro přechodovou funkci výsledného automatu bude platit $\delta((q_{1,i}, q_{2,k}), a) = (q_{1,j}, q_{2,l})$. Jednoznačně je definován i počáteční stav výsledného automatu jako $(q_{1,0}, q_{2,0})$, kde $q_{1,0}$ a $q_{2,0}$ jsou počáteční stavy parciálních automatů.

Výše uvedeným způsobem se vytvoří přechodová funkce výsledného automatu, množina stavů, a počáteční stav.

Vytvoření přechodové funkce prakticky:

1. Nakreslíme přechodové grafy obou dílčích automatů tak, že se snažíme všechny stavy umístit do jedné přímky (stavy prvního automatu do přímky vodorovné, druhého automatu do přímky svislé) tak, aby vzniklo místo na zakreslení stavů výsledného automatu „v průsečících souřadnic“ (viz řešení příkladu).
2. Do pozice odpovídající „průsečíku souřadnic“ počátečních stavů obou dílčích automatů zakreslíme počáteční stav.
3. Pro každé písmeno vstupní abecedy zakreslíme do „průsečíku souřadnic“ následujících stavů dílčích automatů následující stav automatu výsledného.
4. Bod 3 opakujeme tak dlouho, dokud nám v přechodovém grafu vznikají nové stavy. Je zřejmé, že tento postup někdy skončí (počet stavů výsledného automatu je konečný).

Výše uvedený postup lze jednoduše modifikovat i pro práci s tabulkovou reprezentací automatů, pak jej lze použít i na vytvoření kartézského součinu obecného počtu n parciálních automatů.

Určení množiny koncových stavů:

Je nutné si uvědomit, že množina řetězců může být ve vztahu ke dvěma podmínkám definována různým způsobem:

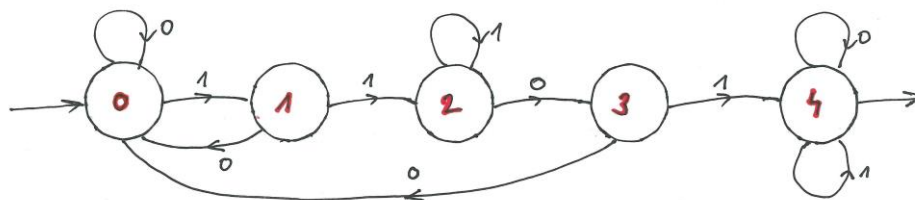
- řetězce musí splňovat alespoň jednu z podmínek
- řetězce musí splňovat právě jednu z podmínek
- řetězce musí splňovat obě podmínky

Do množiny koncových stavů výsledného automatu podle toho vybereme takové stavy, kde

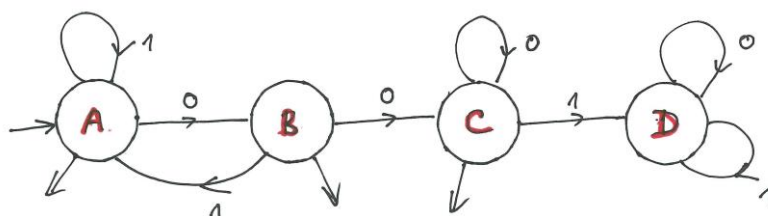
- alespoň jeden z dvojice stavů je koncovým stavem parciálního automatu
- právě jeden z dvojice stavů je koncovým stavem parciálního automatu
- oba stavy z dvojice jsou koncovými stavy parciálních automatů

Návrat k Příkladu 9

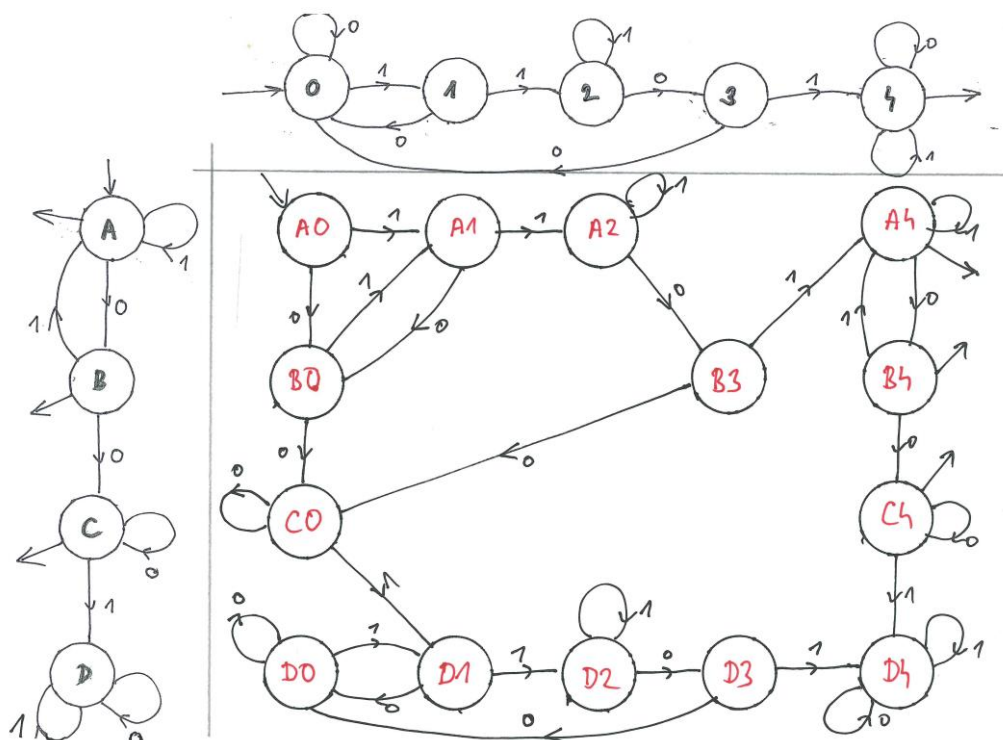
KA rozpoznávající řetězce splňující podmínku 1 (obsahuje podřetězec *1101*):



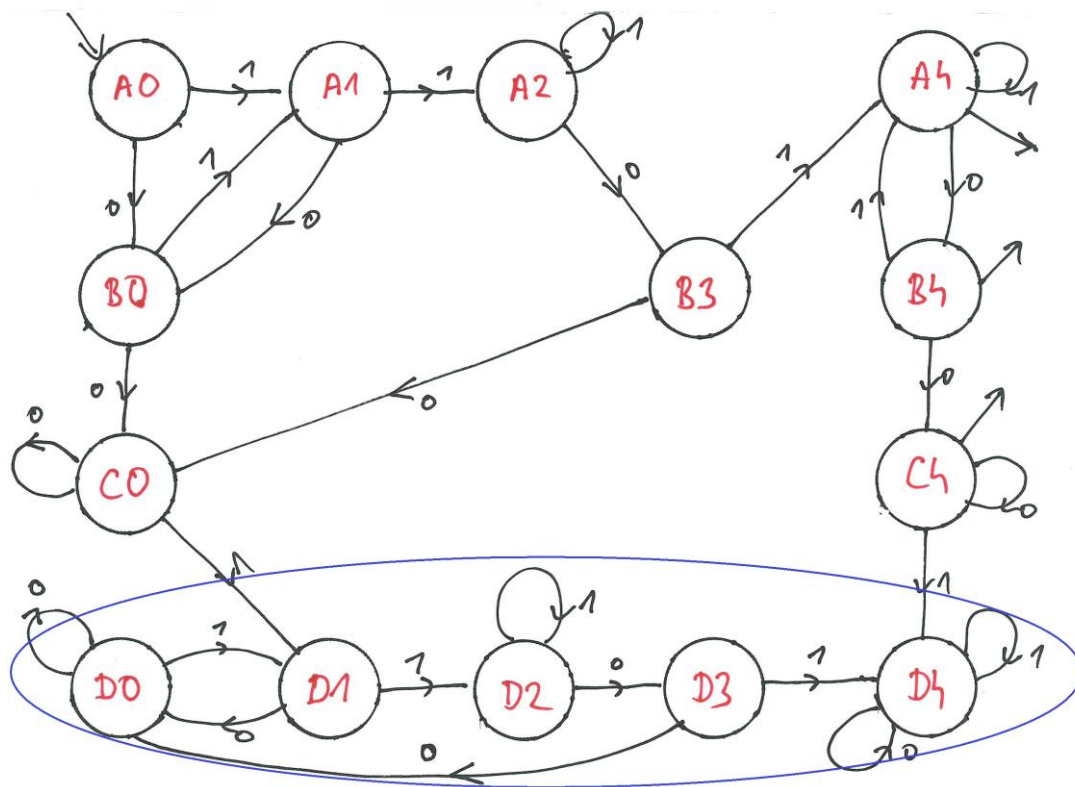
KA rozpoznávající řetězce splňující podmínku 2 (neobsahuje podřetězec *001*):



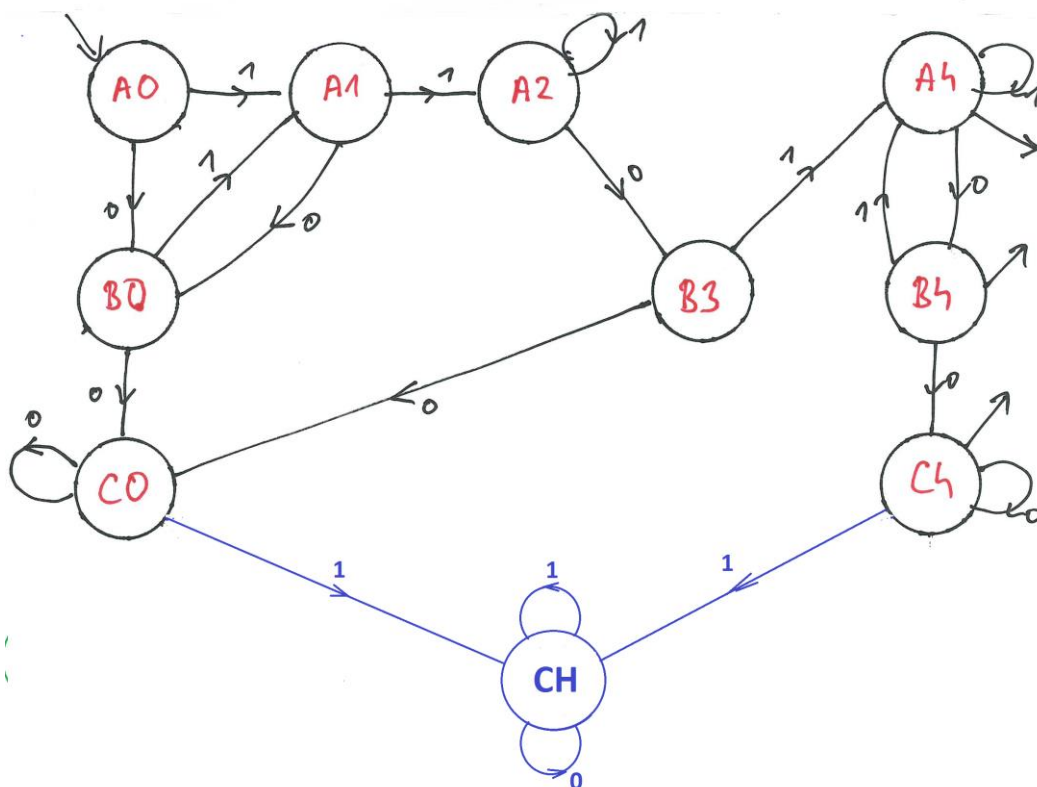
Přechodový graf výsledného KA (kartézský součin parciálních automatů výše):



Z obrázku je zřejmé, že množina stavů výsledného KA neobsahuje všechny uspořádané dvojice (q_1, q_2) , obecně je množina stavů Q kartézského součinu KA podmnožinou kartézského součinu množin stavů Q_1 a Q_2 partiálních automatů, tedy $Q \subseteq Q_1 \times Q_2$.



Z přechodového grafu je zřejmá vlastnost skupiny stavů D0 až D4 – všechny tyto stavy jsou nekoncevé a zároveň má skupina „absorpční vlastnost“ v tom smyslu, že dostane-li se automat do některého ze stavů této skupiny, již skupinu nedokáže opustit, řetězec je tedy zamítnut (souvisí to s formulací druhé podmínky – jakmile se objeví podřetězec 001, je vstupní řetězec „nenávratně pokažen“ a tedy zamítnut). Stavy D0 až D4 automat využívá pouze k vyhledávání podřetězce 1101 z první podmínky, to už ovšem není zapotřebí. Znamená to, že by skupina stavů D0 až D4 mohla být nahrazena jediným absorpčním stavem, který by nebyl koncevý:



Výsledný kartézský součin partiálních automatů obecně nemusí mít minimální počet stavů. Jinak řečeno - může existovat KA, který bude rozpoznávat stejnou množinu řetězců (tj. *ekvivalentní automat*) s menším počtem stavů. To není nikterak na závadu, existují algoritmy, které umožňují k danému automatu nalézt tzv. *redukovaný automat* (*minimální automat*), tj. automat s minimálním počtem stavů, který rozpoznává stejnou množinu řetězců, řeší tedy stejnou úlohu.

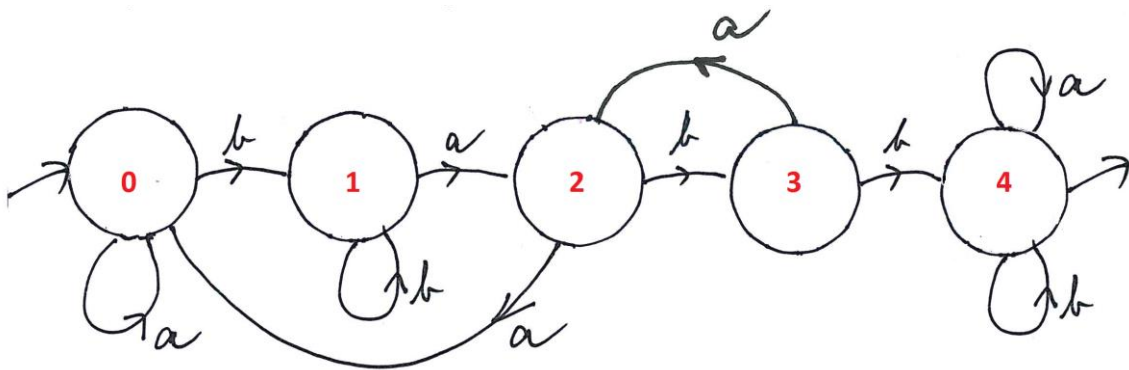
Příklad 10.

L_{10} je množina všech řetězců nad abecedou $\{a,b\}$, které současně splňují následující podmínky:

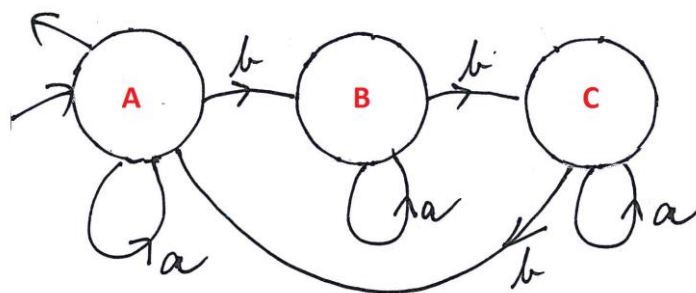
- řetězec obsahuje podřetězec $babb$ (podmínka 1)
- počet znaků b v řetězci je dělitelný 3 (podmínka 2)

Parciální automaty:

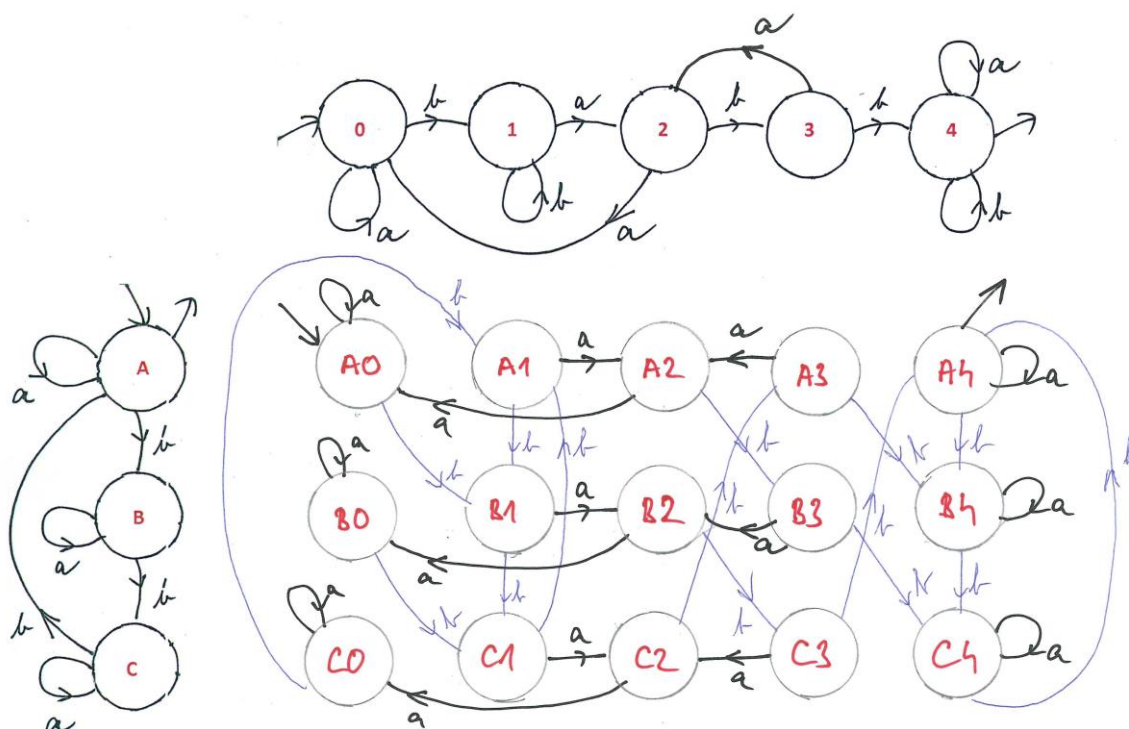
Podmínka 1:



Podmínka 2:



Výsledný automat:



Využití konečného automatu ke vstupní konverzi konstant

Příklad 11.

Navrhněte konečněautomatový model algoritmu pro softwarovou implementaci vstupní konverze textového řetězce, reprezentujícího zápis konstanty typu INTEGER, do vnitřní reprezentace počítače (formát v pevné řádové čárce).

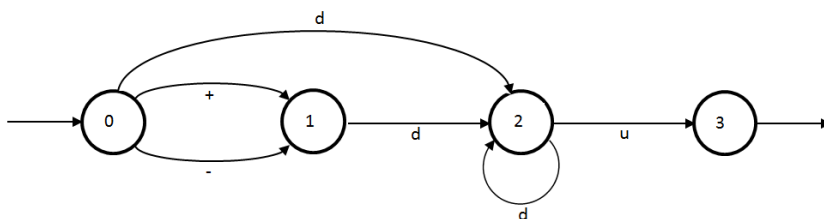
Ilustrace – reprezentace některých konstant ve znakové reprezentaci a ve formátu INTEGER s délkou slova 4 byty (každý řetězec je zakončen ukončovací mezerou 20_H, obsahy jednotlivých bytů jsou zapsány hexadecimálně):

Konstanta	Reprezentace ve znakovém poli							Reprezentace INTEGER			
	(kódování ASCII)										
314	33	31	34	20				00	00	01	3A
-314	2D	33	31	34	20			FF	FF	FE	C6
+27123	2B	32	37	31	32	33	20	00	00	69	F3

Úvahy

1. Rozpoznávací automat vydává o vstupním řetězci pouze odpověď akceptuji/zamítám, nicméně prostřednictvím svých stavů si o vstupním řetězci může „pamatovat“ veškerou informaci nezbytnou pro řešení dané úlohy.
2. V počítači můžeme ve formátu INTEGER zobrazit celá čísla pouze z konečného intervalu <MININT, MAXINT> (v případě zobrazení na 4 bytech je to z intervalu <-2.147.483.648, 2.147.483.647>).
3. Konečný počet stavů k řešení této úlohy stačí, otázkou je pouze to, jakým způsobem takový automat navrhnout tak, aby si v průběhu zpracování vstupního řetězce pamatoval i „hodnotu dosud zpracované části“ a jak popsat jeho přechodovou funkci.

Dílčí problém – návrh rozpoznávacího automatu, jenž bude akceptovat řetězce reprezentující konstanty typu INTEGER.



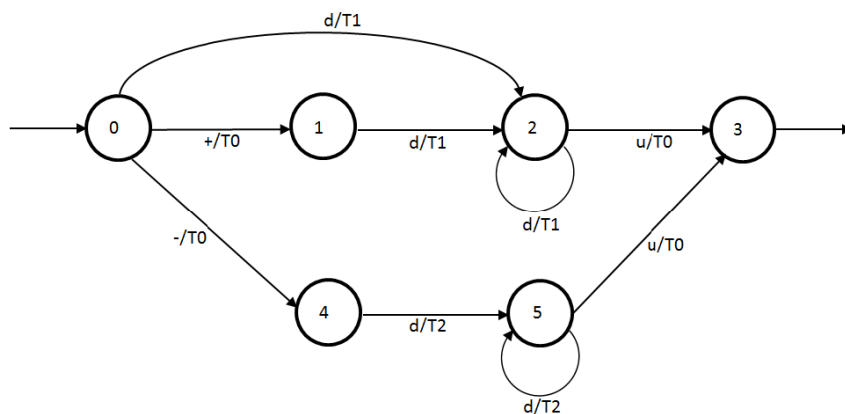
(symbol d představuje libovolnou dekadickou číslici, symbol u ukončovací znak řetězce; přijde-li v některém stavu vstupní symbol, pro který v přechodovém grafu neexistuje hrana, znamená to, že řetězec nepředstavuje korektní zápis celočíselné konstanty a je automatem zamítnut).

Další úvahy

1. Jednomu stavu rozpoznávacího automatu může odpovídat větší počet stavů navrhovaného konverzního automatu (jinak by si automat nedokázal „zapamatovat“ hodnotu dosud zpracované části řetězce). Jedno kolečko v přechodovém grafu tedy může představovat více stavů „srovnaných za sebou ve třetím rozměru dvourozměrné nákresey“, tedy *makrostav*.
2. Stavy tvořící jeden makrostav mezi sebou budeme rozlišovat pomocí hodnot *pomocných stavových proměnných*. Pomocné stavové proměnné jsou „součástí stavu“, proto je třeba definovat jejich počáteční hodnoty.
3. Je třeba definovat, jak se hodnoty pomocných stavových proměnných budou transformovat při přechodech.

Konkrétně

4. Použijeme pomocnou stavovou proměnnou *hodnota*; bude typu INTEGER, počáteční hodnota 0. Tato pomocná stavová proměnná bude reprezentovat číselnou hodnotu dosud zpracované části řetězce.
5. K výpočtu *hodnota* použijeme Hornerovo schéma pro výpočet hodnoty mnohočlenu. Koeficienty mnohočlenu jsou číslice ve znakovém poli, mnohočlen vyčíslíme pro $x = 10$, tj. „dosud vypočtenou *hodnota* vynásobíme deseti a přičteme (resp. v případě záporného čísla odečteme) vstupující číslici.
6. Protože vstupující číslici zpracováváme jinak, představuje-li zpracovávaný řetězec konstantu kladnou, jinak, zpracováváme-li konstantu zápornou, upravíme přechodový graf tak, že zpracování kladného a záporného znaménka oddělíme.



„Stavy“ 2, 3 a 5 jsou makrostavy, tedy množiny stavů. Symboly T0, T1, T2 u přechodů za lomítkem budeme obecně chápat jako nějaké transformace pomocných stavových proměnných, které budou realizovány v souvislosti s přechodem mezi makrostavy. Tyto transformace budou vypadat takto:

T0 : prázdná;

T1 : *hodnota* := *hodnota**10 + (VSTUP – ord("0"));

T2 : *hodnota* := *hodnota**10 - (VSTUP – ord("0"));

Konstrukce $VSTUP - \text{ord}("0")$ od ASCII kódu vstupující číslice odečítá ASCII kód znaku 0, čímž získáme hodnoty z intervalu 0 až 9 vhodné pro přičtení/odečtení.

Příklad 12.

Navrhněte konečněautomatový model algoritmu pro softwarovou implementaci vstupní konverze textového řetězce, reprezentujícího zápis konstanty typu REAL, do vnitřní reprezentace v pohyblivé řádové čárce.

Příklady správných zápisů konstant:

1315. -221. +16. -0.4 .26 -.314 2.5E10 10E6 10.E6

Příklady nesprávných zápisů konstant:

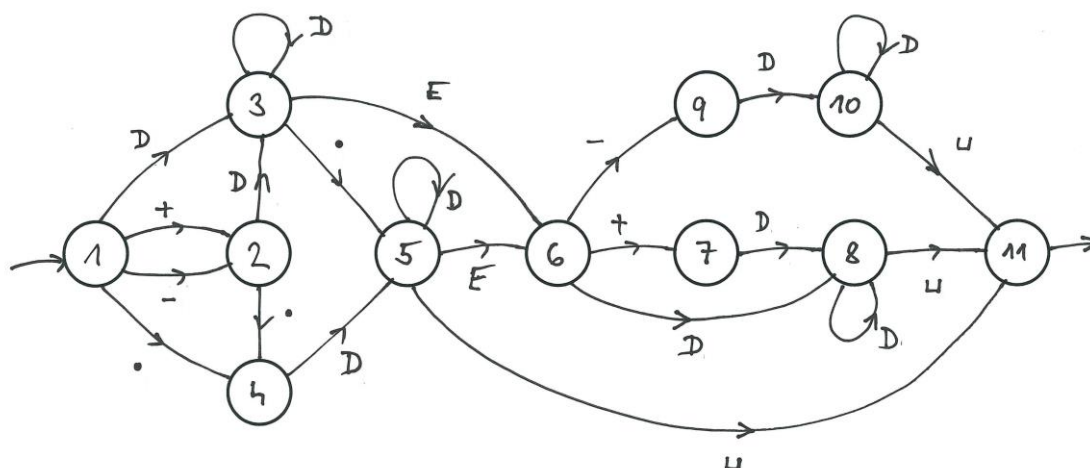
314 -5 0 E20 .

Nápověda:

- Při zpracování čísla před exponentovou částí není třeba oddělovat zpracování kladného a záporného čísla, stačí si v pomocné stavové proměnné zapamatovat znaménko.
- Je třeba oddělit zpracování číslic před desetinnou tečkou a za desetinnou tečkou (každá má jiný význam, zpracovává se jinak).
- Exponentová část je celé číslo, jako část přechodového grafu použijte výsledek Příkladu 11.

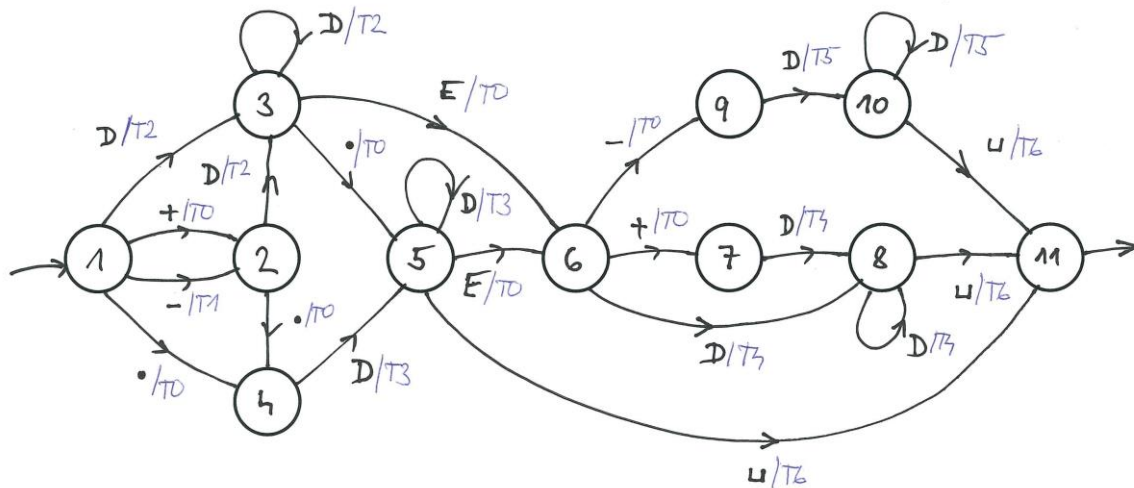
Řešení:

Rozpoznávací automat



Pomocné stavové proměnné

Jméno	typ	počáteční hodnota
<i>Znaménko</i>	REAL {-1.,1.}	1.
<i>Hodnota</i>	REAL	0.
<i>Vaha</i>	REAL	0.1
<i>Exponent</i>	INTEGER	0



Transformace pomocných stavových proměnných:

T0 : prázdná;

T1 : *znamenko* := MINUSJEDNA;

T2 : *hodnota* := *hodnota* * DESET + POLE[(VSTUP – ord("0"))];

T3 : *hodnota* := *hodnota* + *vaha* * POLE[(VSTUP – ord("0"))];

vaha := *vaha* / DESET;

T4 : *exponent* := *exponent* + (VSTUP – ord("0"));

T5 : *exponent* := *exponent* - (VSTUP – ord("0"));

T6 : *hodnota* := *znamenko* * *hodnota* * DESET^{*exponent*};

Několik vysvětlujících poznámek z programátorského pohledu

1. Při psaní programů, které jsou součástí základního programového vybavení, nemá programátor k dispozici nic jiného, než instrukční sadu stroje a formát reprezentace dat. K dispozici nejsou žádné knihovny (ty teprve píše pro uživatele on). Proto si programátor sám (na kalkulačce) musí předem spočítat obrazy všech konstant typu REAL, které ve svých výpočtech použije, tedy
 - konstanty MINUSJEDNA, DESET a pro nastavení počátečních hodnot také konstanty NULA, JEDNA a DESETINA.
 - pole konstant POLE, což je jednorozměrné pole, v němž jsou uloženy obrazy čísel 0 až 9. Pole je indexováno od 0 do 9, takže konstrukce `POLE[(VSTUP – ord("0"))]` pouze převádí dekadickou číslici ze znakové reprezentace v ASCII kódu na formát v pohyblivé řádové čárce.
2. Není třeba lekat se operace *hodnota* *DESET^{exponent}. Nezvládne-li to hardware jinak, vyřeší se násobením deseti v cyklu.