Kartézský součin dvou automatů

KIV/TI – Semestrální práce

Student: Kateřina Kalabzová, Ladislav Čákora
Osobní číslo: A23B0152B, A23B0149P
Email: kalabzok@students.zcu.cz, cakoral@students.zcu.cz

Obsah

1 Zadání

Navrhněte dva rozpoznávací KA A_1 a A_2, které budou rozpoznávat tyto jazyky:

$$L_1 = \{w | w \text{ obsahuje podřetězec – } bbab \text{ –}\}$$

$$L_2 = \{w | w \text{ neobsahuje podřetězec – } bbb \text{ –}\}$$

Následně vytvořte jejich kartézský součin A, který bude rozpoznávat průnik jazyků $L_1 \cap L_2$.

V jazyku Python vytvořte program, který "standardním způsobem"zobrazí přechodové grafy obou parciálních automatů i jejich kartézského součinu s vybarvenými počátečními stavy. Program umožní postupně zadávat vstupní řetězec z klávesnice. Po zadání jednoho znaku všechny automaty provedou přechod, tj. zobrazí se přechodové grafy se žlutě vybarvenými sktuálními stavy.

Kromě písmen vstupní abecedy program umožní také zpracovat 2 speciální znaky, které umožní ukončení zpracovávání řetězce a zahájení zpracování nového řetězce (RESET), respektive ukončenáí programu (STOP).

Pozn: Oproti zadání byla vyměněna žlutá barva za modrou z důvodu lepší viditelnosti.

2 Analýza úlohy

2.1 Návrh automatů

Před implementací programu je potřeba navrhnout oba konečné automaty A_1 a A_2 a následně vytvořit kartézský součin A těchto automatů.

2.2 Návrh programové implementace

Nejprve je nunté stanovit přechodovou funkci a zapsat ji ve vhodné podobě pro využití v programu, což je například tabulka dané přechodové funkce.

V dalším kroku se musí vytvořit grafická reprezentace automatů A_1, A_2 a A. To je možné buď programově, nebo zobrazováním obrázků podle stavů a jednotlivých přechodů.

Po těchto krocích už bude možné implementovat samotný program.

3 Automatový model

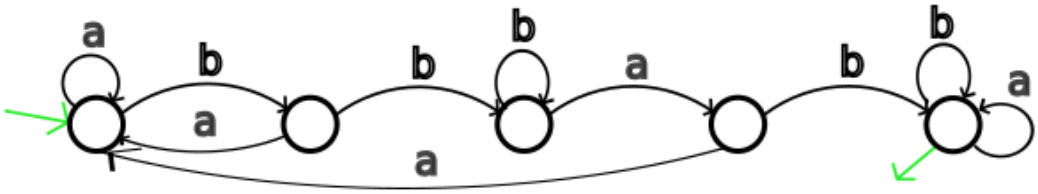
3.1 Automat A_1

Tento automat má akceptovat řetězce obsahující podřetězec $-bbab-$. Tento automat bude mít pět stavů, jeden pro každý možný prefix hledaného podřetězce. Stavy jsou číslovány zleva.

3.1.1 Popis vstupních a výstupních signálů

Znak	**0**	**1**	**2**	**3**	**4**
a	0	0	3	0	4
b	1	2	2	4	4

3.1.2 Přechodový graf



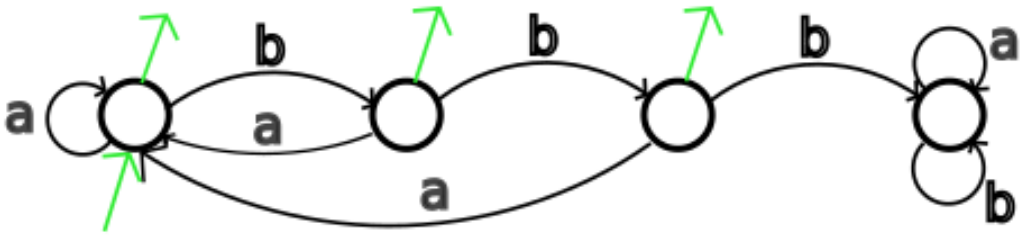
3.2 Automat A_2

Tento automat má zamítnout řetězce obsahující podřetězec $-bbb-$. Tento automat bude mít čtyři stavy, jeden pro každý možný prefix hledaného podřetězce. Stavy jsou číslovány zleva.

3.2.1 Popis vstupních a výstupních signálů

Znak	**0**	**1**	**2**	**3**
a	0	0	0	3
b	1	2	3	3

3.2.2 Přechodový graf



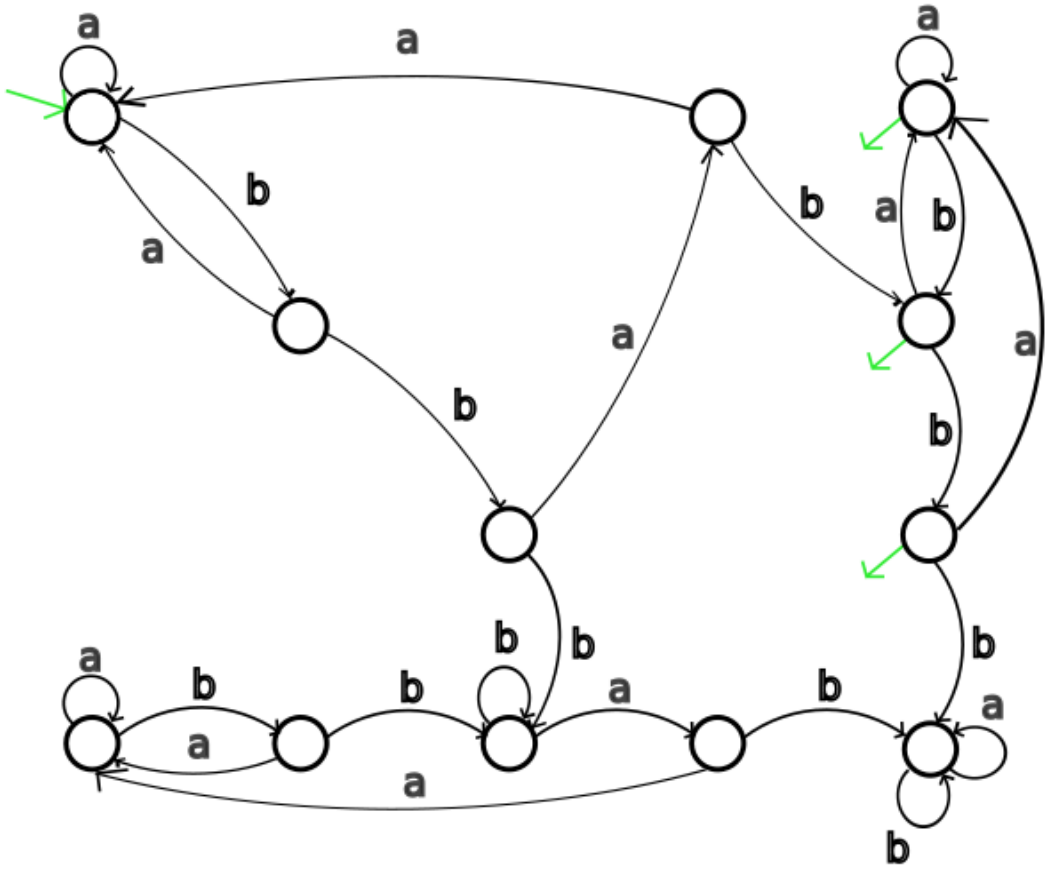
3.3 Automat A

Tento automat má akceptovat řetězce obsahující podřetězec $-bbab-$ a zároveň zamítnout všechny řetězce obsahující $-bbb-$. Tento automat byl sestaven kartézským součinem automatů A_1 a A_2. Stavy jsou číslovány po řádcích zleva (počáteční stav je 0, stav po řetězci *bba* je 1). Dále je možné zredukovat pět stavů ve spodní řádce do jednoho. Žádný z nich není koncovým stavem automatu a zároveň není možné z žádného z nich přejít do stavu, který by nepatřil do této množiny.

3.3.1 Popis vstupních a výstupních signálů

Znak	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
a	0	0	2	0	2	1	2	7	7	10	7	11
b	3	4	4	5	6	9	11	8	9	9	11	11

3.3.2 Přechodový graf



4 Implementace

4.1 main.py

Soubor poskytující funkce pro načtení obrázků zobrazujících jednotlivé stavy, které vykreslí pomocí knihovny pro tvorbu grafických rozhraní tkinter. Zároveň je zde zajištěna tvorba okna, přepínání obrázků a obsluha stisknutí jednotlivých kláves.

Díky povaze úlohy stačí zajistit obsluhu modelu automatu vzniklého kartézským součinem, jelikož ke každému stavu z výsledného automatu A lze přiřadit právě jeden stav z automatu A_1 a právě jeden stav z automatu A_2.

4.1.1 load_images(directory)

Funkce pro načtení obrázků z adresáře `directory`, počítá s existencí obrázků **0.png** až **11.png**. Pokud se některý z obrázků nepodaří načíst, vrátí se pole úspěšně načtených. Pokud se podaří načíst vše, vrátí se všech 12.

4.1.2 redraw(label, active_state, images)

Funkce vyvolá překreslení obrázku. Z pole obrázků `images` vybere obrázek odpovídající stavu `active_state`.

4.1.3 main()

Funkce obstarává hlavní běh programu. Vytvoří instanci modelu automatu A a vytvoří okno pro zobrazování obrázků s přechodovými grafy. Zároveň zajistí reakce na stsiknutí kláves odpovídajících znakům vstupní abecedy (a, b), resetu (r) a ukončení programu (Escape).

4.2 Třída Automat (soubor automat.py)

4.2.1 Matice přechodové funkce

Matice přechodové funkce automatu je v tomto případě matice M o rozměrech $V \times S$, kde počet řádků V odpovídá velikosti vstupní abecedy a počet sloupců S je počet stavů automatu. Prvek $M[v, s]$ je potom index stavu, do kterého automat přejde při vstupu znaku v, pokud je zrovna ve stavu s. Tato matice je v aplikaci naimplementována polem, které obsahuje odpovídající pole indexů pro daný vstupní symbol.

4.2.2 Atributy

- matrix - Matice odpovídající přechodové funkci automatu. Každý vstupní znak odpovídá jednomu řádku, každý sloupec jednomu stavu.
- current - Index současného stavu.
- start - Index počátečního stavu.
- inputs - Namapování vstupních symbolů na indexy řádků v matici.

4.2.3 __init__(matrix, start)

Kontruktor třídy. Parametr `matrix` odpovídá matici přechodové funkce, parametr `start` odpovídá indexu počátečního stavu automatu.

4.2.4 move(input)

Posune stav automatu na základě současného stavu `self.current`, matice přechodové funkce `self.matrix` a vstupního znaku `input`. Změní `self.current` na index nového stavu zároveň nový index vrátí.

4.2.5 reset()

Funkce provede návrat do počátečního stavu `self.start`. Nastaví `self.current` na hodnotu počátečního stavu a tuto hodnotu vrátí.

5 Uživatelská příručka

5.1 Spuštění programu

Pro spuštění je třeba mít nainstalované běhové prostředí jazyka Python. Poté v něm stačí spustit soubor **main.py**.

5.2 Ovládání

Aplikace reaguje na stisk kláves *"a"*, *"b"*, *"r"* a *"Escape"*. Klávesy *"a"* a *"b"* reprezentují symboly vstupní abecedy automatu. Klávesa *"r"* vrací automat do počátečního stavu a vyresetuje vizualizaci. *"Escape"* aplikaci ukončí.

6 Závěr

V této semestrální práci jsme si vyzkoušeli jiný způsob grafického zpracování, než na který jsme zvyklí, a to přepínání jednotlivých předem vytvořených obrázků podle vstupu od uživatele. Byli bychom rádi za možnost řešení semestrální práce i v jiných jazycích než Python nebo Java, vzhledem k tomu že momentálně pracujeme nejvíce v jazyce C#.

Zajímavé rozšíření by bylo například zautomatizování vykreslování automatů a jednotlivých přechodů, aby změna automatů zároveň vytvořila nový odpovídající graf.