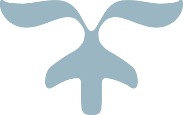


Semestrální práce z předmětu KIV/TI

K9. Demonstrační program pro simulaci nedeterministického konečného automatu



12. června 2024

Filip Valtr – A22B0107P Václav Prokop – A22B0330P

Obsah

[Zadání 2](#_Toc169102836)

[Analýza úlohy 3](#_Toc169102837)

[Automatový model 5](#_Toc169102838)

[Implementace 6](#_Toc169102839)

[Uživatelská příručka 7](#_Toc169102840)

[Závěr 8](#_Toc169102841)

## Zadání

Obsah obrázku text, dopis, papír, inkoust

Popis byl vytvořen automaticky

## Analýza úlohy

Jako první jsme se rozhodli z rozpoznávacího nedeterministického automatu naleznout ekvivalentní deterministický automat. Víme, že počáteční stav deterministického konečného automatu odpovídá množině počátečních stavů nedeterministického automatu, a navíc KA vytváříme tak, že krok po kroku vyhodnocujeme přechodovou funkci δ. Začínáme od počátečního stavu, tedy S.

Stavy nedeterministického automatu nakonec sjednotíme do jednoho stavu deterministického automatu, s tím, že navíc platí, množinou koncových stavů deterministického automatu budou všechny stavy, které v sobě obsahují některý z koncových stavů ekvivalentního nedeterministického konečného automatu.

Tabulka pro nález ekvivalentního deterministického konečného automatu bude tedy vypadat takto:

Obsah obrázku text, rukopis, dokument, papír

Popis byl vytvořen automaticky

Obsah obrázku text, rukopis, skica, kresba

Popis byl vytvořen automatickyNyní za pomocí tabulky vytvoříme deterministický konečný automat, ten bude vypadat následovně:

Dále využijeme sady obrázku, kde bude každý stav deterministického automatu vyobrazen obrázkem, který odpovídá ekvivalentnímu nedeterministickému stavu automatu. Navíc bude v obrázku vyobrazeno, zdali je daný řetězec akceptován, tedy nachází-li se v nějakém z koncových stavů, či nikoliv.

Platí, že pokud v některém stavu přijde vstupní symbol, pro který v přechodovém grafu neexistuje hrana, pak to znamená, že symbol není automatem zpracovatelný, tedy je automatem zamítnut, což se v naší aplikaci projeví vypnutím aplikace a vypsáním chybové hlášky.

## Automatový model

-popis vstupních a výstupních signálů formou tabulek

|  |  |  |
| --- | --- | --- |
| Stav | Vstup | Výstup |
| S | 0 | Je akceptován |
| S | 1-9 | Je akceptován |
| A | X | Není akceptován |
| A | 0-7 | Je akceptován |
| B | 0-9 | Je akceptován |
| C | 0-7 | Je akceptován |
| D | 0-9,A-F | Je akceptován |
| E | 0-9,A-F | Je akceptován |

|  |  |  |
| --- | --- | --- |
| Stav | Vstup | Následující stav |
| S | 0 | A |
| S | 1-9 | B |
| A | X | D |
| A | 0-7 | C |
| B | 0-9 | B |
| C | 0-7 | C |
| D | 0-9,A-F | E |
| E | 0-9,A-F | E |

**přechodový graf**

viz obrázek z 4. stránky

**popis pomocných stavových proměnných (pokud je automat používá)**

V naši aplikaci použáváme 2 stavové proměné. První proměná newState slouží pro uložení stavu do kterého se program dostane po aplikaci některého z validních symbolů. Toho využijeme hlavně při výběrů svg obrázků pro zobrazení všech možných cest (hran) kam se v nedeterministickém konečném automatu můžeme dostat. Pokud ale zadávaný symbol uživatelem je nevalidní tak tato proměná nabije hodnoty -1 pro indikaci konce programu. Druhá proměná oldState obsahuje indikátor stavu ve kterém se právě nacházíme. Obě tyto proměné spolu úzce souvisí, při převodu oldState na newState. Zajišťuje metoda searchInput.

### Implementace kódu třídy NKA

Program je postaven na promítání SVG obrázků na základě vstupu uživatele a mění tyto obrázky reprezentující možné stavy a možné kroky, kterými těchto stavů můžeme dosáhnout symbolem, který uživatel zadal na vstupu. Třída obsahuje několik statických atributů a metod, které spolupracují na zajištění této funkcionality. Níže je podrobný popis jednotlivých částí implementace.

#### Atributy třídy

1. **Seznamy znaků**:
   * list0\_9, list1\_9, list0\_7, list0\_F, list0, listX - různé seznamy znaků, které představují povolené vstupy pro různé stavy.
2. **SVG canvas**:
   * svgCanvas - objekt třídy JSVGCanvas z knihovny Apache Batik, který umožňuje manipulaci a zobrazování SVG souborů.
3. **Konstanty**:
   * END - znak K označující konec programu.
   * RESET - znak R pro restartování programu.
   * Můžou být napsány i malými písmeny.
4. **Další atributy**:
   * picture - jméno aktuálně zobrazovaného obrázku.
   * newState - proměnná uchovávající následující stav automatu.
   * table – HashMap, klíčme je stav a hodnotou jsou stavy do kterých se můžeme z klíče dostat.
   * array - dvourozměrné pole seznamů symbolů pro jednotlivé stavy.
   * outputTextField - text pro zobrazení postupně zadáváného řetězce. Vyobrazuje se v textovém poli v dolní části obrazovky na pozici South.
   * textField - textové pole pro zobrazení výstupu.

#### Metoda main

1. **Inicializace seznamů znaků**:
   * V této motdě naplníme platnými symboly listy určené pro uchovávání symbolů
2. **Naplňování pole** array **a tabulky** table:
   * Voláme metody fillArray a fillTable, které naplňují pole array seznamy znaků pro různé stavy a tabulku table definující přechody mezi stavy.

Vytvoření panelu a rozestavení jednotlivých komponent do něj

Využíváme Jframe panel na který vykreslujeme svg obrázky. Dále jsme do něj přidali text field pro zobrazení postupně zadávaného řetězce

1. **KeyListener pro vstupy uživatele**:
   * Dále metoda main obsahuje reakce na systémové události, kdy pomocí KeyListeneru tyto události obsluhujeme pro zpracování klávesových vstupů uživatele. Konkrétně v metodě keyTyped pomocí metody changeState měníme jednotlivé svg obrázky.

#### Metody

1. fillArray:
   * Naplňuje pole array seznamy znaků pro jednotlivé stavy deterministického automatu.
2. pictureSetter:
   * Nastavuje název obrázku a aktualizuje zobrazený SVG soubor voláním metody updateSVG.
3. updateSVG:
   * Aktualizuje SVG zobrazený v canvasu na základě aktuálního názvu obrázku.
4. disableTextFieldIfTextFits:
   * Zakáže focusování textového pole, pokud zadaný text se vejde do jeho šířky. A zároveň umožňuje manipulovat s textem pokud je delší než šířka textFieldu za účelem zobrazení celého řetězce
5. changeState:
   * Nejdříve otestuje jestli předaná stisknutá klávesa uživatelem slouží k ukončení nebo restartování automatu. Následně se aktualizuje proměná newState (pomocí metody searchInput). Podle obsahu této stavové proměné se určí jaký konkrétní svg obrázek se má na panelu vykreslit (pro přesnější určení se využívá i stavová proměnná oldState). Pokud je však newState roven -1, tak ukončí činnost aplikace a výpíše informující hlášku
6. searchInput:
   * Postupně projdeme stavy navazující na stav oldState, které jsme získali z table (hashMapy) a prohledáme listy příslušející k těmto stavů, jestli neobsahují klávesu zadanou uživatelem. Pokud ano, tak vrátíme identifikátor příslušného stavu nebo -1 v případě že tento symbol neobsahuje žádný stav, což naznačuje nepovolený znak.
7. fillTable:.
   * Tato metoda naplní hashMapu, kdy klíč reprezentuje stav z kterého se dostaneme do stavů uvedených ve values. Tato hashMapa je důležitou konstrukcí, kterou využíváme pro kontrolu jestli je vstupní symbol validní, neboli nachází se v listech spojenými se stavy ve values. Díky použítí hash mapy dochází k významné redukci a zpřehlehlednění kódu, také se tím významě sníží cyklomatická složitost.

## Uživatelská příručka

## Závěr