Semestrální práce z předmětu KIV/TI

Konvertor z formátu JFLAP do formátu DKAR

Jméno a příjmení: Vitalij Atamanjuk
Osobní číslo: A23B0136P
Datum odevzdání: 16. června 2025

Obsah

Zadání

N1. Konvertor z formátu JFLAP do formátu DKAR

Na https://www.jflap.org/ se seznamte s nástrojem JFLAP a s využitím tutoriálu a vlastních experimentů prostudujte formát, v němž editor generuje popisy rozpoznávacího automatu, Mealyho automatu a Moorova automatu. Poté navrhněte a realizujte program, jež umožní konverzi mezi formátem JFLAP (data v XML) a formátem DKAR (viz http://home.zcu.cz/-vais/). Předpokládejte, že uživatel programu JFLAP bude používat implicitní vstupní a výstupní symboly z formátů TI, může ale používat i implicitní symboly JFLAPU, provádějte tedy také případnou konverzi mezi implicitními názvy stavů (q0 bude A, q1 bude B, ...). Smysl konverzního programu: Uživatel navrhne automat grafickým editorem programu JFLAP, konverzní program jej z XML převede do struktury formátu DKAR, takže s ním budou moci pracovat další programy odladěné v rámci projektů z KIV/TI. Pro formáty DKAME a DKAMO budou potřebné úpravy zdrojového programu popsány v části Future works. Program lze realizovat v Javě nebo Pythonu.

1 Analýza úlohy

1.1 Cíl a motivace

Hlavním cílem je vytvořit nástroj, který umožní uživateli jednoduše exportovat deterministický automat z JFLAPu do textového formátu DKAR, používaného v projektech předmětu KIV/TI. Díky tomu mohou studenti navrhovat a testovat automaty v GUI a poté je zpracovat ve skriptech či dalších nástrojích.

1.2 Formát JFLAP (.jff)

JFLAP je vzdělávací nástroj pro práci s formálními automaty. Exportuje popis v XML se strukturou:

- `<structure>`, `<type>` určuje druh (fa, mealy, moore).
- `<automaton>` s poduzly:
  - `<state id="i"name="qi»` + značka `<initial/>`, `<final/>`,
  - `<transition>` obsahuje `<from>`, `<to>`, `<read>`, případně `<output>`.

1.3 Teoretický rozbor

Deterministický konečný automat (DFA) je pětice $(Q, \Sigma, \delta, q_0, F)$. V překladu do DKAR:

- Q – množina stavů, přejmenovaná na $\{A, B, \dots\}$,
- Σ – vstupní abeceda,
- δ – tabulka přechodů ve formě řádek `<stav>: <n`$_1$ `>` ...,`q`$_0$ `- počáteční stav,`
- F `- množina koncových stavů.`

1.4 Formát DKAR

Výstupním formátem je:

```
Deterministický konečný automat (rozpoznávací)
DKAR
6
7
A: A C A A B C F
B: A C D A A B E
C: A A B C D A E
D: A C D A A B D
E: C D A A B B D
F: A D A B E F F
D
3 A C E
```

2 Implementace

Projekt je rozdělen do tří modulů:

- `parser` – třída `JflapParser` pro načítání XML.
- `converter` – třída `Converter` provádějící mapping.
- `formatter` – třída `DkarFormatter` pro zápis DKAR.

2.1 Struktura a závislosti

- `pom.xml` deklaruje:
  - `jackson-dataformat-xml:2.18.1`, `woodstox-core:6.5.0`,
  - `maven-compiler-plugin:3.11.0`, `maven-assembly-plugin:3.6.0`.
- Balíčky v `src/main/java/`: `parser`, `converter`, `formatter`.

2.2 Parser: `JflapParser`

- Používá Jackson XML (`XmlMapper`) s Woodstox pro streaming.
- Deserializuje do POJO `Automaton`, `State`, `Transition`.
- Validuje existenci počátečního stavu a alespoň jednoho koncového.

2.3 Converter: `Converter`

- Dependency injection parseru a formatteru.
- Přejmenování stavů: $q_i \rightarrow$`(char)('A'+i)`.
- Volání `formatter.format(model, output)`.

2.4 Formatter: `DkarFormatter`

- Zápis pomocí NIO `BufferedWriter` s UTF-8.
- Hlavička, počet stavů, počet symbolů, tabulka přechodů, počáteční a koncové stavy.

3 Uživatelská příručka

3.1 Požadavky

- **Java 21** (JDK nebo JRE) – doporučeno OpenJDK 21.
- **Apache Maven 3.8+** – pro sestavení projektu.
- **Přístup k internetu** – pro stažení závislostí z Maven Central.

3.2 Sestavení projektu

```
git clone https://github.com/VitaliyAtmnk/TI
cd SP
mvn clean package
```

Po úspěšném dokončení se v adresáři `target/` vytvoří:

- `SP-1.0-jar-with-dependencies.jar` – tzv. *uber-jar*, obsahující vlastní kód i všechny potřebné knihovny.

3.3 Spuštění konvertoru

Pro převod automatu spusťte vytvořený JAR následujícím příkazem:

```
java -jar target/SP-1.0-jar-with-dependencies.jar \
    <vstupn_soubor.jff> < vstupn_soubor .dkar>
```

kde:

`<vstupní_soubor.jff>` Cesta k XML souboru exportovanému z JFLAPu. Může být relativní i absolutní; doporučená přípona je `.jff`.

`<výstupní_soubor.dkar>` Cesta, kam se uloží vygenerovaný DKAR soubor. Pokud soubor již existuje, přepíše se.

Příklad

```
# Soubory ve složce examples/
java -jar target/SP-1.0-jar-with-dependencies.jar \
    examples/sample.jff examples/sample.dkar
```

3.4 Průběh a výstup

`stdout` Program vypíše informaci o dokončení programu:

- *Převod byl úspěšně dokončen.*

`stderr` V případě chyby (špatný formát souboru, neexistující cesta, práva k zápisu) se vypíše diagnostická zpráva a stack trace.

3.5 Návratové kódy

- `0` – úspěšné dokončení konverze.
- `1` – chybný počet argumentů (program očekává právě 2).
- `2` – chyba během čtení, validace nebo zápisu (I/O nebo formát).

4 Future works

Pro přidání podpory DKAME (Mealy) a DKAMO (Moore) je třeba:

4.1 Rozšíření modelu

- Přidat `MealyTransition` (input, output, to) a `MooreState` (output).
- Rozšířit `Automaton` o kolekce těchto typů.
- Enum `AutomatonType` = {DFA, MEALY, MOORE}.

4.2 Parser

- Číst `<type>` pro určení typu.
- Metody `parseMealyTransitions()` a `parseMooreStates()`.
- Anotovat POJO: `@JacksonXmlProperty(localName="output")`.

4.3 Converter

- `convertMealy()` a `convertMoore()` s logikou mappingu vstup/výstup.
- Přejmenovat symboly a výstupy podle implicitních pravidel TI.

4.4 Formátovače

- `DkarMealyFormatter`: hlavička DKAME, počet stavů/vstupů/výstupů, řádky `<stav>: i/o ...`.
- `DkarMooreFormatter`: hlavička DKAMO, stav/výstup, přechodová tabulka.
- Factory pro volbu formátovače v `Main` (–format).

4.5 CLI a testy

- Commons CLI pro přepínače `-input`, `-output`, `-format`.
- Jednotkové testy (JUnit5) pro všechny typy automatů.
- Ukázkové `.jff` a očekávané výstupy.

5 Zhodnocení

V této části zhodnocuji dosažené cíle, vlastní posun ve znalostech a kvalitu výsledného řešení.

- **Úspěšné vyřešení zadaného problému:** Navržený a implementovaný konvertor spolehlivě překládá popis deterministického automatu z JFLAP XML formátu do přesného textového zápisu DKAR. Tím plní svůj primární účel a umožňuje další zpracování automatu v nástrojích KIV/TI.
- **Osobní rozvoj a dovednosti:** Při vývoji jsem procvičil práci s XML – zejména deserializaci pomocí Jackson Dataformat XML. Díky tomu jsem lépe porozuměl struktuře konečných automatů a principům jejich popisu.
- **Modularita a čitelnost kódu:** Rozdělení na tři samostatné moduly (parser, converter, formatter) zaručuje přehlednost, usnadňuje údržbu a podporuje případné rozšíření o další formáty (Mealy, Moore).

Celkově hodnotím tuto semestrální práci jako velmi přínosnou – vedle naplnění zadání mi umožnila prohloubit praktické zkušenosti s prací s XML, architekturou modulárních aplikací a detailnějším pohledem na teorii konečných automatů.

6 Závěr

Navržený konvertor splňuje zadání a poskytuje efektivní nástroj pro převod JFLAP XML do formátu DKAR. Díky modulární architektuře je snadno rozšiřitelný a udržovatelný. Je připravený na rozšíření o další formáty jako jsou DKAME, nebo DKAMO