

KIV/TI - TEORETICKÁ INFORMATIKA
**PŘEVOD NEDETERMINISTICKÉHO AUTOMATU NA
DETERMINISTICKÝ**

Jaroslav Klaus (A13B0347P), Vladimír Láznička (A13B0371P)

18. ledna 2015, Plzeň

Obsah

1	Zadání	2
1.1	Formát vstupních a výstupních dat	2
2	Analýza	3
2.1	Zpracování vstupu	3
2.2	Převod nedeterministického automatu na deterministický	3
2.3	Uložení parametrů deterministického automatu a vypsání výstupu	4
3	Implementace programu	5
3.1	Objekt Automaton	5
3.2	Načtení souboru a uložení dat	5
3.3	Algoritmus převodu	5
3.4	Uložení výsledného automatu a výpis na výstup	6
4	Uživatelská dokumentace	7
5	Ověření správnosti převodu	8
5.1	Automat 1	8
5.2	Automat 2	9
6	Závěr	10

1 Zadání

Implementujte algoritmus pro převod nedeterministického konečného rozpoznávacího automatu (umožněte i existenci e-hran) na ekvivalentní deterministický automat. Navrhněte vhodný formát vstupních a výstupních dat.

Program odladíte alespoň na 6 příkladech včetně příkladů prezentovaných na přednáškách a cvičeníh.

Všechny testovací příklady uveďte v dokumentaci včetně ručního řešení.

1.1 Formát vstupních a výstupních dat

Jako formát vstupních a výstupních dat budeme volit textové soubory `*.TI` odpovídající definici nedeterministického konečného rozpoznávacího automatu (vstup) a deterministického konečného rozpoznávacího automatu (výstup) na stránce <http://home.zcu.cz/~vais/formaty.htm>. Cílem výstupu je pak možnost jej využít pro další zpracování (např. minimalizace) daného automatu dalším programem.

2 Analýza

Řešení úlohy lze rozdělit do následujících celků - načtení dat ze vstupu a jejich zpracování do příslušné struktury, samotný převod automatu podle dané reprezentace na deterministický typ, uložení dat vzniklých z převodu do příslušné struktury a nakonec výpis výsledku na výstup.

2.1 Zpracování vstupu

Pro zpracování vstupu bude zapotřebí připravit si funkci nebo metodu, která provede **parsování vstupu na jednotlivé řetězce**, ze kterých se poté získají hodnoty důležité pro reprezentaci automatu a jeho následný převod. Formát vstupního souboru bude naprosto zásadní dodržet, neboť v opačném případě může dojít k chybnému převodu nebo také k němu nemusí dojít vůbec. Vstupní soubor nám udává **počet stavů automatu, velikost množiny vstupních symbolů, přechodovou tabulku automatu a nakonec výpis vstupních a výstupních stavů**. Tyto informace bude třeba uložit do struktury nebo objektu reprezentující daný automat.

Jako reprezentace chování automatu se pak využije zmíněná **přechodová tabulka**, která v sobě drží defacto všechny potřebné informace k jeho převodu na deterministický typ. Ostatní informace poslouží buď k vytvoření dalších celků (jako velikost polí na základě počtu stavů) nebo k závěrečné části převodu - určení vstupního stavu a výstupních stavů deterministického automatu. Samotný seznam stavů nebude ke způsobu zápisu vstupního souboru potřeba, neboť stavy vždy odpovídají **velkým písmenům a jsou řazené podle abecedy** (obdobně to platí pro množinu vstupních symbolů, nicméně tu nebudeme pro převodu automatu přímo potřebovat).

2.2 Převod nedeterministického automatu na deterministický

Převod bude probíhat pomocí **přechodové tabulky**, která se bude postupně upravovat, aby se z ní odstranily všechny nedeterminismy (více vstupních stavů, nejednoznačné přechody a přítomnost e-hran). To nám zajišťuje jistou intuitivnost a umožní snazší porovnání s ručním řešením, které bude rovněž prováděno na základě přechodové tabulky.

Při vytváření tabulky deterministického automatu se nejprve použije **první řádek tabulky z původního automatu**, který se bude procházet položku po položce, z nichž se vyberou ty stavy, které ještě nemáme zaznamenány (na počátku máme pouze jeden vstupní stav). Pokud bude položka obsahovat více stavů, tyto stavy se **spojí v jeden nový stav** a ten se zaznamená. Pro každý takto zaznamenaný stav se vytvoří další řádek, jeho položky budou obsahovat stavy, do kterých bychom se z něj dostali v daném nedeterministickém automatu pomocí příslušného vstupního znaku. Pokud byl nově vzniklý stav složen z více původních stavů, v položkách pro tento stav budou zaznamenány stavy, do kterých se lze daným znakem dostat ze všech těchto původních stavů. Takto se bude pokračovat, dokud nebudou nalezeny všechny nové stavy a pro ně vytvořeny příslušné položky.

V případě **více vstupních stavů** se tyto stavy na počátku převodu spojí v jeden a takto vzniklý stav se použije jako první zaznamenaný. Pokud bude nedeterministický automat obsahovat **e-hrany**, což je indikováno další položkou pro příslušný stav v původní přechodové

tabulce ¹, budeme vytvářet navíc tzv. tabulku **e-následníků**, která bude pro každý stav z původního automatu uvádět, do jakých dalších stavů se lze dostat prostřednictvím e-hrany (včetně jeho samého). Tyto stavy se pak sjednotí do jednoho nového, který bude mít vlastnosti všech v sobě obsažených stavů a **bude reprezentovat původní stav**, pro nějž se tito e-následníci zjišťovali.

Jakmile budeme mít vytvořenou přechodovou tabulku deterministického automatu, určíme pomocí **seznamu výstupních stavů** z nedeterministického automatu, jaké stavy budou výstupní v deterministickém automatu. Budou to ty, které v sobě obsahují nějaký výstupní stav z původního automatu. Jako stav vstupní se použije buď ten původní, pokud byl jeden, nebo stav vzniklý sjednocením několika původních stavů, pokud jich bylo více.

2.3 Uložení parametrů deterministického automatu a vypsání výstupu

Vzhledem k tomu, že z předchozí části již budeme mít všechny potřebné informace - **počet stavů, přechodovou tabulku, vstupní stav a výstupní stavy**, lze je jednoduše uložit do stejné struktury nebo objektu jako u nedeterministického automatu. Pak už jen stačí použít vhodnou funkci nebo metodu, která si tyto informace vezme a zapíše je do souboru v daném formátu (v zásadě bude fungovat přesně opačně než funkce pro načtení dat ze souboru).

¹jejich počet pak o 1 přesahuje uvedenou velikost množiny vstupních znaků

3 Implementace programu

K implementaci programu jsme se rozhodli použít jazyk **Java**, který je jednak výhodný svým objektovým přístupem a také obsahuje několik knihovnických tříd umožňující používání různých struktur jako třeba seznamy, aniž bychom je museli sami implementovat. Nevýhodou je pak samozřejmě pomalejší běh než např. při použití jazyka **C**, ale pro náš případ není vysoká rychlost zpracování až tak důležitá.

3.1 Objekt Automaton

Tento objekt slouží k **uchování informací o automatu ve svých attributech**. Těmito atributy jsou:

- `String automatonType` - řetězec obsahující zkratku typu automatu
- `int statusCnt` - počet stavů automatu
- `int inputCnt` - velikost množiny vstupních znaků
- `String[] [] automatonTable` - pole uchovávající přechodovou tabulku automatu, každý řádek přísluší jednomu stavu (A... Z) a každý sloupec jednomu vstupnímu znaku (a... z).
- `ArrayList<String> inputStatuses` - seznam se vstupními stavy automatu
- `ArrayList<String> outputStatuses` - seznam s výstupními stavy automatu

Dále má samozřejmě metody pro ukládání a vracení těchto atributů, kde je to třeba. Konstruktor pak jako parametry přijímá **zkratku automatu, počet stavů a počet vstupních znaků**.

3.2 Načtení souboru a uložení dat

Načtení souboru je řešeno pomocí metody `static createAutomatonFromFile(String filePath)`, která přebírá jako parametr řetězec s cestou ke vstupnímu souboru a je obsažena ve třídě `Input_Output`. Metoda používá ke čtení souboru knihovnickou třídu `BufferedReader`, zejména její metodu `readLine()`, pomocí které získá řetězec představující obsah aktuálně načítaného řádku. Ten je pak rozdělen buď ručně nebo pomocí metody `split()`, přičemž jako dělicí znak je použita mezera. Nejprve se načte typ automatu, počet stavů a počet vstupních znaků a tyto hodnoty se poté použijí k vytvoření objektu `Automaton`. Následně se načte přechodová tabulka a postupně uloží do **dvourozměrného pole řetězců**, které se pak objektu předá. Nakonec se načtou vstupní a výstupní stavy do příslušných seznamů a rovněž se předají objektu.

3.3 Algoritmus převodu

Samotný převod je implementován ve třídě `Convert` a v její hlavní metodě `public static Automaton nkaToDka(Automaton nka)`, která jako parametr přebírá nedeterministický konečný automat (dále NKA), který se načtl ze vstupního souboru. Nejprve se vytvoří tabulka

e-následníků pomocí metody `private static void createETable(String[] [] nkaTable, int inputCount, String[] eTable)`. Jako první stav deterministického konečného automatu (dále DKA) se použije sjednocení vstupních stavů NKA a jejich e-následníků. Tento první stav se vloží do fronty `statuses`. Dokud bude ve frontě `statuses` nějaká položka, vybere se první položka a do nové řádky přechodové tabulky DKA se zapisují nové stavy, do kterých se dostaneme příslušným vstupním symbolem. Zjišťování těchto stavů probíhá tak, že sjednotíme stavy, do kterých se z původních stavů NKA dostaneme pomocí vstupního symbolu, a přidáme jejich e-následníky. Pokud takto nově vzniklý stav není ve frontě `statuses`, tak jej do ní přidám a pokračuji pro další sloupec tabulky (další vstupní symbol). Toto opakuji, dokud mi vznikají nové stavy v přechodové tabulce DKA, respektive dokud není fronta `statuses` prázdná.

3.4 Uložení výsledného automatu a výpis na výstup

Výsledný deterministický automat je vytvořen na konci metody pro převod. Nejprve se vytvoří objekt tohoto automatu, přičemž jako parametry jsou použity řetězec "DKAR" (deterministický konečný automat rozpoznávací - dle požadovaného formátu souboru), velikost seznamu s vytvořenými stavy a počet vstupních znaků z původního automatu (ten se převodem nijak nemění). Poté se uloží pomocí metody `static void setOutputStatusesToDka(Automaton nka, Automaton dka, LinkedList<String> statuses)` seznam výstupních stavů automatu postupem víceméně popsáním v analýze. V metodě také dojde k přejmenování stavů tak, aby vyhovovaly formátu souboru, který bude výstupem. Přejmenování probíhá procházením vytvořených výstupních stavů, které mohou být v tu chvíli označené jako složení původních stavů, přičemž se tyto stavy porovnávají se seznamem všech nově vytvořených stavů a jakmile dojde ke shodě, složený stav se přejmenuje podle formule 'A'+ index stavu v seznamu vytvořených stavů. K podobnému přejmenování dojde i ve vytvořené přechodové tabulce v metodě `static void renameStatuses(String[] [] dkaTable, LinkedList<String> statuses)`. Nakonec se tabulka předá objektu automatu a vytvoří se list s jedním stavem "A", který je uložen jako vstupní (prezentuje množinu původní vstupních stavů).

Objekt se pak použije k výpisu informací do výstupního souboru pomocí metody `static void writeAutomatonToFile(Automaton a, String filepath)` ze třídy `Input.Output`. V zásadě funguje opačně než metoda pro získání dat ze souboru. Postupně skládá informace získané z objektu automatu do řetězců, představující jednotlivé řádky (podle požadovaného formátu) a ty pomocí metody `write(<retezec>)` z knihovny třídy `BufferedWriter` zapisuje do zadaného souboru.

4 Uživatelská dokumentace

Pro spuštění programu je zapotřebí mít na počítači nainstalované prostředí **Java Runtime Environment 1.8 nebo novější** (na této verzi bylo vykonána kompilace programu). Spuštění lze pak provést z příkazové řádky, odkud se spouští JAR archiv aplikace pojmenovaný **NKAR_to_DKAR_App.jar**.

Příkaz ke spuštění je následující: `java -jar NKAR_to_DKAR_App.jar <vstupni_soubor> <vystupni_soubor>`

Vstupní soubor musí obsahovat informace ve specifickém formátu zobrazeném na stránkách zmíněných v části **Zadání**. Výstupní soubor pak označuje soubor, do kterého se v obdobném formátu zapíše převedený automat. Pokud je toto splněno, proběhne převod a vytvoří se výstupní soubor s převedeným automatem (při každém dalším použití programu se soubor přepíše, pokud použijeme stejný název).

```
C:\Users\Vlada47\Desktop>java -jar NKAR_to_DKAR_App.jar automat.TI result.TI
Nedeterministicky konecny automat uspesne nacten ze souboru.
Prekonvertovani automatu na deterministicky probehlo uspesne.
Prekonvertovany automat uspesne ulozen do souboru.
```

Obrázek 1: Korektní spuštění programu

Pokud nedodržíme uvedené podmínky, program skončí chybovým hlášením. Stejně tak, pokud dojde během zpracování vstupu nebo převodu k nějaké výjimce.

```
C:\Users\Vlada47\Desktop>java -jar NKAR_to_DKAR_App.jar automat.TI
Musite zadat dva parametry. Prvni parametr je cesta k souboru s NKA a druhy para
metr cesta k souboru, kam se zapise DKA.
```

Obrázek 2: Chyba při zadání příliš málo vstupní parametrů

5 Ověření správnosti převodu

Zde se pokusíme ověřit, zda vytvořený program splňuje korektně svůj účel, totiž zda ze zadaného nedeterministického automatu dokáže vytvořit **ekvivalentní** deterministický automat. Správnost budeme testovat na celkem šesti různých nedeterministických automatech, které se vyskytli na přednáškách či cvičeních předmětu Teoretická informatika. Testování bude prováděno nějakým řetězcem složeným ze znaků vstupní abecedy, kterou dokáže daný automat zpracovat, pro přechodovou tabulku nedeterministického automatu a ten samý test se pak provede pro tabulku deterministického automatu. Vstupní a výstupní stavy jsou pak vypsány mimo tabulku, přičemž u deterministického automatu je vstupním vždy pouze stav A.

5.1 Automat 1

Následující automat přijímá řetězce složené z písmen a, b, přičemž mezi jeho nedeterminismy patří **více vstupních stavů a nejednoznačné přechody**.

Vstupní stavy: A, B

Výstupní stavy: A, D

Přechodová tabulka nedeterministického automatu:

	a	b
A	A,B	C
B	C	D
C	A,C	B
D	A	C

Konfigurace nedeterministického automatu pro řetězec **aaabba**:

(A, aaabba) — (B, aabba) — (C, abba) — (C, bba) — (B, ba) — (D, a) — (A, -)

- řetězec **je akceptovaný** automatem

Přechodová tabulka deterministického automatu:

	a	b
A	B	C
B	B	D
C	E	F
D	E	D
E	B	F
F	E	G
G	E	C

Výstupní stavy: A, B, C, D, E, G

Konfigurace deterministického automatu pro řetězec **aaabba**:

(A, aaabba) — (B, aabba) — (B, abba) — (B, bba) — (D, ba) — (D, a) — (E, -)

- řetězec **je akceptovaný** automatem

5.2 Automat 2

Následující automat přijímá řetězce složené z písmen **a**, **b**, **c**, přičemž mezi jeho nedeterminismy patří pouze **výskyt e-hran**. Automat vychází z regulárního výrazu $ab + c^*$.

Vstupní stavy: **A**

Výstupní stavy: **C**

Přechodová tabulka nedeterministického automatu:

	a	b	c	e
A	D	-	-	B
B	-	-	B	C
C	-	-	-	
D	-	C	-	

Konfigurace nedeterministického automatu pro řetězec **ab**:

(A, ab) — (D, b) — (C, -)

- řetězec **je akceptovaný** automatem

Konfigurace nedeterministického automatu pro řetězec **cccc**:

(A, cccc) — (B, cccc) — (B, ccc) — (B, cc) — (B, c) — (B, -) — (C, -)

- řetězec **je akceptovaný** automatem

Přechodová tabulka deterministického automatu:

	a	b	c
A	B	-	C
B	-	D	-
C	-	-	C
D	-	-	-

Výstupní stavy: **A**, **C**, **D**

Konfigurace nedeterministického automatu pro řetězec **ab**:

(A, ab) — (B, b) — (D, -)

- řetězec **je akceptovaný** automatem

Konfigurace nedeterministického automatu pro řetězec **cccc**:

(A, cccc) — (C, ccc) — (C, cc) — (C, c) — (C, -)

- řetězec **je akceptovaný** automatem

6 Závěr

-doplnit-