Dokumentace úlohy DKA: Determinizace konečného automatu v PHP do IPP 2014/2015
Jméno a příjmení: Lukáš Pelánek
Login: xpelan03

Dokumentace k projektu do předmětu IPP, zadání DKA

Zadání

Mým úkolem bylo vytvořit skript v jazyce PHP, který zpracuje zadaný konečný automat a provede jeho případnou determinizaci.

Zpracování parametrů

První věc, kterou skript provede je kontrola zadaných parametrů. K tomuto účelu jsem použil knihovní funkci `getopt`. V případě vícenásobného zadání stejného parametru, zadání neznámého parametru nebo nedovolené kombinace parametrů, ukončím skript s příslušným chybovým kódem. Pokud jsou zadané parametry validní, nastavím příznaky, podle kterých provedu požadovanou operaci se zadaným konečným automatem.

Zpracování vstupního souboru

Pokud nebyl zadán žádný vstupní soubor jako parametr, čtu data ze standardního vstupu. Celý vstup načtu do paměti programu a poté jej zpracuji po znaku pomocí konečného automatu. Pokud byl zadán přepínač `-i` nebo `--case-insensitive`, převádím načítané znaky na malé.

Konečný automat, který zpracovává vstup se skládá z deseti stavů. Plní funkci lexikální, syntaktické a sémantické analýzy. Jednotlivé prvky konečného automatu jsou načítány a ukládány do paměti. Vstupní konečný automat má předem stanovenou syntaxi. Pokud vstupní automat není správně zapsán, ukončím skript s příslušným chybovým kódem. Před uložením jednotlivých stavů či pravidel zjistím, zda se tam již nevyskytují. Pokud ano, tak je ignoruji. Nakonec zkontroluji, zda počáteční stav a množina koncových stavů patří do množiny stavů zadaného konečného automatu.

Uložení dat v paměti

Množina stavů a množina koncových stavů konečného automatu jsou uloženy v obyčejných polích, kde pod každým indexem daného pole se nachází jeden stav. Počáteční stav je uložen v obyčejné proměnné. Pro uložení množiny pravidel jsem si vytvořil třídu `RulesClass`, ze které vytvářím objekty reprezentující jednotlivá pravidla. Pravidla jsou poté uložena v poli. Pokud narazím na epsilon přechod, vstupní symbol bude „eps“.

Mějme pole `rules`, kde na indexu 0 je uloženo pravidlo `b '' -> f`. Potom uložení pravidla bude vypadat následovně:

```
rules[0]->startingState = "b";
rules[0]->inputSymbol = "eps";
  rules[0]->finalState = "f";
```

Odstranění epsilon přechodů a determinizace

Implementace těchto funkcí byla určena zadáním projektu. Musel jsem využít algoritmů, které byly náplní předmětu Formální jazyky a překladače. Tedy samotná implementace spočívala pouze v přepsání daných algoritmů. Vzhledem ke způsobu uložení dat v paměti tato část projektu nebyla náročná.

Nejprve provádím odstranění epsilon přechodů. Vše provádím v jednom hlavním cyklu, který prochází všechny stavy konečného automatu. Prvně vytvořím epsilon uzávěr. Tato část se skládá z 3 vnořených cyklů. Když vytvořím epsilon uzávěr pro daný stav, následuje samotné odstranění epsilon přechodů. Tato část se skládá z 2 zanořených cyklů. Nakonec následuje přidání koncových stavů, které je také prováděno v 2 cyklech.

Dále následuje samotná determinizace, která probíhá ve 4 zanořených cyklech. Pokud se z konkrétního stavu lze dostat přes jeden symbol vstupní abecedy do více cílových stavů, ty se sloučí v jeden. Před vykonáním této funkce je potřeba zavolat funkci na odstranění epsilon přechodů.

Zpracování výstupního souboru

Pokud nebyl zadán příslušný parametr s cestou výstupního souboru, výstup je přesměrován na standardní výstup. Pokud výstupní soubor neexistuje, tak je vytvořen. Pokud již existuje, tak je bez varování přepsán. Výstupem je normalizovaný zápis konečného automatu, který odpovídá požadavkům zadání.

Rozšíření

Rozhodl jsem se implementovat 2 rozšíření. A to `STR` a `WSA`.

V případě rozšíření `STR` jsem musel zjistit, zda zadaný řetězec je řetězcem jazyka přijímaného zadaným konečným automatem. Nejprve jsem musel ověřit, zda všechny symboly zadaného řetězce patří do abecedy zadaného konečného automatu a provést determinizaci. Poté jsem začal v počátečním stavu a hledal, zda existuje takové pravidlo pro přechod do dalšího stavu. Pokud jsem přečetl celý řetězec a skončil v koncovém stavu, označil jsem analýzu jako úspěšnou.

Úkolem rozšíření `WSA` bylo transformovat zadaný konečný automat na dobře specifikovaný konečný automat. Implementace opět spočívala v pouhém přepsání algoritmu. Prvně jsem musel provést determinizaci konečného automatu a odstranit všechny neukončující stavy. To jsou takové stavy, ze kterých se nelze dostat do koncových stavů. To je provedeno ve 2 fázích, přičemž každá se skládá ze 4 vnořených cyklů. Samotná transformace na dobře specifikovaný konečný automat je provedena ve 2 vnořených cyklech. Dále jsem musel zjistit, zda počáteční stav patřil mezi neukončující stavy. Pokud ano, tak novým počátečním stavem se stává stav `qFALSE`.