Dokumentace úlohy DKA: Determinizace konečného automatu v PHP 5 do IPP 2013/2014
Jméno a příjmení: Jan Lamacz
Login: xlamac00

# Dokumentace k projektu IPP, zadání DKA
Determinizace konečného automatu

## 1 Analýza zadání

Úkolem projektu bylo napsat skript v jazyce PHP, který načte obecný konečný automat a převede jej na ekvivalentní deterministický konečný automat bez nedostupných stavů.

## 2 Řešení

Celý projekt je rozdělen do tří souborů podle povahy jejich funkcí. Soubor `dka.php` slouží pouze jako funkce main, která volá funkce z ostatních souborů, kontroluje jejich návratové hodnoty a případně ukončuje celý skript. V druhém souboru jsou uloženy pomocné funkce pro zpracování parametrů příkazové řádky, čtení a zápis do souboru a výpis chybových hlášení včetně nápovědy. Poslední a největší soubor `automat.php` obsahuje všechny funkce potřebné k determinizaci konečného automatu.

### 2.1 Kontrola parametrů

Funkce zpracovávající parametry příkazové řádky prochází pole se vstupními parametry a porovnává jeho jednotlivé hodnoty pomocí funkce `strcmp`, přičemž načtené hodnoty ukládám jako prvky asociativního pole, které celé vracím hlavní funkci. Funkce pro zpracování parametrů zároveň hlídá všechny kolidující přepínače a případně oznámí chybu řídící funkci.

### 2.2 Zpracování automatu

Z automatu načteného do řetězce jsou nejprve odstraněny všechny komentáře začínající znakem `#`, u kterého se ovšem musí hlídat, zda to není symbol z abecedy automatu.

Poté je automat rozparserován na jednotlivé komponenty, které jsou uloženy do asociativního pole (pro každé pravidlo je v tomto poli vytvořeno druhé pole o třech prvcích pro počáteční a koncový stav a pro vstupní symbol). Po rozdělení automatu jsou z něj odstaněny duplicitní položky a automat je dále zkontrolován funkcí, která zjistí, zda je správně zadán a neobsahuje například prázdnou vstupní abecedu.

### 2.3 Odstranení epsilon přechodů

Pro správnou funkčnost všech ostatních algoritmů je nejdříve potřeba z automatu odstranit všechny epsilon přechody. Skript tedy postupně iteruje přes všechny stavy automatu a pro každý z nich vytvoří epsilon uzávěr (tedy do jakých stavů se dá dostat z tohoto stavů pomocí epsilon přechodů). Vytvoří si také nové koncové stavy a pravidla, která už ovšem neobsahují epsilon přechody.

### 2.4 Determinizace

Po odstranění epsilon přechodů již může proběhnout samotná determinizace automatu bez generování nedostupných stavů. K tomu je potřeba inicializovat několik pomocných polí a poté provádět determinizaci v dokonce čtyřnásobném cyklu.

Algoritmus si ukládá všechny stavy, do nichž se dá z právě zkoumaného stavu dojít jedním ze vstupních symbolů. V případě, že se tento stav skládá z více původních stavů, vytvoří z nich jeden společný stav skládající se z jejich jmen oddělených znakem '_'. Pokud skript zpracovává některý z těchto složených stavů, probíhá průchod nad každým z jeho původních stavů zvlášť. Tento stav se uloží do pomocného pole a funkce iteruje tak dlouho, než se celé toto pole vyprázdní.

Zároveň se spolu s novými stavy a pravidly vytváří i nové koncové stavy, kterými se stává každý nově vytvořený stav, který obsahuje alespoň jeden z původních koncových stavů.

# 3 Rozšíření

## 3.1 WSA

Rozšíření WSA jde ve zdokonalování automatu ještě dál a vytváří dobře specifikovaný automat bez nedostupných stavů a s pouze jedním nekoncovým stavem, takzvanou pastí, kam směřují všechny nekorektní kombinace stavů a vstupních symbolů.

Pro jeho vytvoření je již zapotřebí plně deterministický automat. V něm je nejdříve potřeba odstranit všechny neukončující stavy, tedy stavy, ze kterých nelze libovolným počtem přechodů dojít do libovolného koncového stavu. Toho lze nejlépe dosáhnout tak, že se automatem prochází "naopak", tedy od koncových stavů, a každý stav, do kterého jsme schopni se dostat, je zaručeně ukončující.

Po eliminaci neukončujících stavů se už pouze vytvoří jeden nový stav zvaný past (v projektu pojmenovaný qFALSE), do kterého jsou přesměrovány všechny možné kombinace stavů a přechodů, které nejsou v původním automatu obsaženy.

## 3.2 STR

Při rošíření STR probíhá kontrola, zda je zadaný řetězec přijímán daným konečným automatem.

Pro snazší analýzu řetězce nejprve proběhne determinizace konečného automatu a poté se řízení běhu skriptu předá funkci pro rozbor řetězce. Ta nejdříve zkontroluje, zda jsou všechny jeho znaky v množině vstupních symbolů automatu.

Poté vezme první symbol řetězce, najde přechod s tímto symbolem z počátečního stavu a ze stavu, do kterého se takto dostane, hledá přechod s druhým symbolem řetězce. Takto pokračuje, dokud nedojde na konec zadaného řetězce. Pokud pro zadaný symbol neexistuje přechod, nebo není poslední stav zároveň stavem koncovým, není řetězec daným automatem přijímán.

# 4 Testování

Pro potřeby tohoto projektu jsem zároveň vytvořil skript v jazyce BASH s přibližně padesáti různými testy, které by měly pokrýt všechny parametry, rozšíření, i nejrůznější chybové stavy.