Dokumentace úlohy DKA: Determinizace konečného automatu v PHP 5 do IPP 2015/2016

Jméno a příjmení: David Kozák

Login: xkozak15

Cílem projektu bylo napsat v jazyce PHP5 skript pro determinizaci konečného automatu. Skript má být schopný automat načíst a provést determinizaci či pouze odstranění epsilon přechodů, případně automat pouze vypsat v jednotné formě. Vstupní soubor totiž může obsahovat komentáře a jeho forma je mnohem volnější než výsledná forma výstupu. Kromě základní funkcionality jsem implementoval i rozšíření RUL, STR a WSA.

Postup řešení

Skript nejdříve zpracuje a vyhodnotí argumenty, se kterými byl spuštěn. V rámci této fáze též načte data ze vstupního souboru a otevře výstupní soubor pro zápis. Také dojde ke kontrole správnosti dané kombinace argumentů, neboť ne všechny kombinace jsou dovoleny. Řídící pokyny jsou uloženy do globální proměnné $arguments.

Pro zpracovávání vstupních dat jsem se rozhodl využít lexikální a syntaktickou analýzu. Pro lexikální analýzu jsem využil konečného automatu. Syntaktická analýza je implementována na základě LL gramatiky.

V další fázi dojde k sémantickým kontrolám. Zde se ověří například neprázdnost vstupní abecedy, či zda se nachází počáteční stav a koncové stavy v množině stavů.

START->RULE RULES\_N

RULE->state symbol -> state DOT

DOT-> .

DOT-> epsilon

RULES\_N->,RULE RULES\_N

RULES\_N->epsilon

Poté dojde v závislosti na vstupních argumentech k příslušné operaci (determinizace, odstranění epsilon přechodů, převod na dobře specifikovaný konečný automat, či pouze vypsání automatu ve specifikované formě).

Nakonec dojde k vypsání výsledku na standardní výstup či do souboru.

Implementační detaily

Lexikální analýza je implementována klasicky za pomoci switch uvnitř cyklu while. Pro syntaktickou analýzu jsem zvolil metodu rekurzivního sestupu. Syntaktická analýza též zároveň shromažďuje informace o stavech, symbolech, pravidlech, počátečním stavu a koncových stavech. Samotný konečný automat je implementován jako třída, která dostane právě výše zmíněnou pětici parametrů v konstruktoru. Ověření sémantické správnosti je implementováno jako jedna z veřejných metod instance třídy konečného automatu. Algoritmy pro odstranění epsilon přechodů, determinizaci jsou další veřejné metody této instance. Dále jsem v projektu využil ještě další dvě třídy, jednu pro modelování pravidla a druhou pro složené vztahy, které vznikají při algoritmu determinizace.

S->(START\_TWO)

START\_TWO->{STATES},{ALPHABET},{RULES},state,{FINISH}

STATES -> state STATES\_N

STATES\_N -> epsilon

STATES\_N -> ,state STATES\_N

ALPHABET -> symbol ALPHABET\_N

ALPHABET\_N -> epsilon

ALPHABET\_N -> ,symbol ALPHABET\_N

FINISH -> state FINISH\_N

FINISH\_N -> epsilon

FINISH\_N -> ,state FINISH\_N

RULES -> RULE RULES\_N

RULES -> epsilon

RULE -> state symbol -> state

RULES\_N -> epsilon

RULES\_N -> ,RULE RULES\_N

Rozšíření

Rozšíření WSA zahrnovalo implementování algoritmu pro vytvoření dobře specifikovaného konečného automatu. Toto rozšíření jsem implementoval jako další metodu instance třídy konečného automatu. Rozšíření STR pro ověření, zda je daný řetězec přijímaný konečným automatem, jsem implementoval podobně. Pro rozšíření RUL jsem přidal speciální stav do konečného automatu lexikální analýzy pro načítání konečného stavu. Pro ověření syntaktické správnosti jsem implementoval vlastní syntaktickou analýzu. Třídě konečného automatu jsem poté přidal statickou tovární metodu, která jako parametr obdrží množinu stavů a konstruktor zavolá interně po vypreparování všech potřebných informací.

Závěr