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

Jméno a příjmení: Jan Pavlica
Login: xpavli78

Zadání úlohy

Úkolem projektu bylo vytvořit skript pro zpracování a determinizaci konečného automatu. Skript načítá automat ze vstupního souboru popřípadě ze stdin a výsledný automat vypisuje v normální formě.

Řešení

Kontrola parametrů

Jako první věc je nutné zpracovat zadané parametry. Skript jako první zkontroluje, zda je požadována nápověda a s tímto požadavkem odpovídající počet parametrů. Dále kontroluji jednotlivé parametrů v jednoduchém switchi. V případě, že jsou detekovány jiné parametry, je skript ukončen chybou. Každý zaznamenaný parametr je uložen do proměnné pro případnou kontrolu opakovaných výskytů parametrů nebo jejich nepovolené kombinace.

Zpracování vstupního souboru

Vstupní soubor se načítá do proměnné pomocí funkce `file_get_contents()` a s touto proměnnou poté dále pracuji. V případě zadaní parametru `--case-insensitive` se celý vstup převede na malá písmena. Tímto jsem vyřešil všechnny operace spojené s tímto parametrem. Z proměnné jsem poté načítal jednotlivé znaky a dále s nimi pracoval. V případě, že jsem narazil na bílý znak, jsem tento znak přeskočil. Když jsem narazil na znak "#" byl přeskočen celý řádek. Předchozí dva případy se samozřejmě nevztahují na případy, kdy se jedná o znak abecedy. V případě, že se v abecedě objeví prázdný řetězec je do pravidla uložen jako řetězec "eps".

Syntaktická a sémantická kontrola je řešena konečným automatem, který je implementován formou switche. Automat načítá jednotlivé znaky, dokud nenarazí na konec souboru. Samotný automat má 23 stavů, nicméně mnohé stavy jsou si velice podobné, neboť se jedná stále o načítání stavů, či znaků abecedy.

Uložení dat

Pro množiny stavů, vstupní abecedy a koncových stavů jsem použil klasické pole. Pro uložení pravidel jsem musel vymyslet jiné řešení. Nakonec jsem zvolil vytvoření vlastí třídy `rule`. Která obsahuje proměnné `from`, `symbol` a `to` pro uložení výchozího stavu, čteného vstupního symbolu a cílové stavu. Tímto byl zajištěn bezproblémový přístup k jednotlivým položkám požadovaného pravidla.

Odstranění epsilon přechodů

Nejprve bylo zapotřebí vytvořit epsilon uzávěr. Při psaní algoritmu jsem vycházel z opory předmětu IFJ. Výsledné řešení zahrnuje čtyři vnořené cykly `foreach` a jeden cyklus `do-while`. Toto řešení může působit nevzhledně, nicméně mi připadalo vcelku jednoduché.

Samotné odstranění epsilon přechodů bylo poté opsáno z opory předmětu IFJ a nenarazil jsem při tom na žádný problém.

Determinizace

Poslední částí byla determinizace. Algoritmus byl opět převzat z opory předmětu IFJ. Při zpracování jsem narazil na jeden problém, a to uložení stavů, ze kterých se skládá nově vytvořený mezistav. Tento problém jsem vyřešil vytvořením dvourozměrného pole, kde klíčem byl nově vytvořený mezistav a jeho hodnotami byla množina stavů, ze kterých je tvořen. Tímto jsem byl schopen kontrolovat jednotlivé stavy, které zahrnoval nově vytvořený mezistav.

Závěr

Ačkoliv se jednalo o mé první setkání s jazykem PHP, tak nebyl problém se s jazykem sžít. Se samotnými algoritmy z opory předmětu IFJ jsem neměl, až na zmíněné tvoření mezistavů, žádný problém.