Dokumentace úlohy DKA: Determinizace konečného automatu v jazyce Python3 do IPP 2014/2015

Jméno a příjmení: Štěpán Granát
Login: xgrana02

Zadání

Cílem bylo vytvořit program v jazyce `Python3`, který načítá a optimalizuje konečný automat. Na základě vstupních parametrů provádí odstranění e-pravidel nebo determinizaci. Je implementováno i rozšíření `STR`, které umožňuje analyzovat řetězec.

Návrh a realizace

Úkol byl rozvržen do několika tříd:

Zpracování argumentů (třída `Arguments`)

Argumenty jsou zkontrolovány a objektově uspořádány. Koltrolují se chyby vyplývající ze zadání (například duplicita, nekorektní kombinace atd.). Pokud jsou argumenty v pořádku, zkotroluje také zadané soubory (zda existují a je je možné otevřít).

Zpracování vstupního souboru (třída `Parser`)

Kontroluje syntaktickou stránku vstupního souboru. Kontrola syntaktických pravidel není realizována LL tabulkou. Vzhledem k jednoduchým pravidlům jsem pouze implementoval primitivní scanner, který kontroluje lexikální chyby. Třída `Parser` přímo vytváří objekt reprezentující automat.

Generování konečného automatu (třída `Automat` a `State`)

Třída `Automat` reprezentuje celý automat včetně jeho stavů, které jsou reprezentovány třídou `State`. Třída `Automat` kontroluje většinu sémantických chyb. Například při přidávání pravidla se kontroluje, jestli existují zdrojový a koncový stav a jestli se znak nachází v abecedě.
Třída `Automat` také obsahuje algoritmy provádějící odstranění epsilon pravidel a determinizaci.

Algoritmus determinizace

Tento algoritmus je implementován přesně podle zadání. Místo množin byly využity asociatvní pole (`dict`), která se chovají velmi podobně a umožňují snadnou detekci duplicitních záznamů.
Zvlášť je implementována metoda, která odstraňuje epsilon přechody a funkce vracející epsilon uzávěr.

Rozšíření

Nad rámec základního zadání bylo implementováno rozšíření `STR`, které umožňuje programu zadat řetězec, pro který je zjištěno, jestli vyhovuje specifikovanému konečnému automatu. Toto rozšíření se přímo nabízelo k implementaci, protože všechny struktury a objekty byly již vhodně implementovány a přidaného kódu bylo naprosté minimum.

Testování

Pro účely testování byla navržena vlastní testovací sada, která byla dána veřejně k dispozici. Testy byly zaměřenyl hlavně na speciální případy, které jsou přímo zmíněny v zadání. Jsou zahrnuty i testy argumentů (jejich různé kombinace, kontrola duplicity), návratových kódů (lexikální, syntaktické a sémantické chyby) a determinizace složitějších automatů (především převzaných z IFJ přednášek).
Zveřejnění testů bylo oboustraně prospěšné, neboť moji spolužáci odhalili několik zásadních chyb v testech, kterých by jinak pravděpodobně zůstaly bez povšimnutí.

Zjednodušený diagram programu