Dokumentace úlohy MKA: Minimalizace konečného automatu v Pythonu 3 do IPP 2016/2017
Jméno a příjmení: Patrik Michalák
Login: xmicha65

Zadání

Úkolem projektu je implementování skriptu v Pythonu 3, který načte na vstupu textový zápis konečného automatu, zpracuje a případně minimalizuje podle vstupních požadavků.

Řešení

Parser

Na začátku skript zpracuje pomocí parseru zadaný vstupní automat. Parser čte ze vstupu znak po znaku a z načtených dát vytváří elementární objekty. Parser kontroluje úplnost zadaného vstupního automatu. Kvůli rozšíření `RLO`, existuje ještě `rloParser()`, který přijímá na vstupu jenom množinu pravidel automatu.

Načtená vstupní data rozdělená do elementárních objektů projdou funkci `cleanFsm()`, kde se zahodí nepotřebný obsah. Funkce `checkFsm()` provádí kontrolu lexikálních a syntaktických pravidel (při chybě návratový kód `60`). Vytvářejí se zde množiny stavů a vstupů při rozšíření `RLO` a provádí se znaky na lower case když je zadán argument case insensitive. Na konci funkce se zavolá instance třídy `Fsm()` která vytváří samotný konečný automat.

Třída Fsm

Při inicializaci třídy `Fsm()` se vytváří jednotlivé množiny stavů, vstupů a pravidel. Kontrolují se sémantické chyby s návratovým kódem `61` - jestli se počáteční a koncové stavy a stavy v pravidlech nachází v množině stavů a přechody pravidel v množině vstupů. Následně se volají metody na validaci, že je automat dobře specifikovaný:

- `reachStates()` zjišťuje, jestli jsou všechny stavy dosažitelné,
- `inputRule()` kontroluje, jestli jednomu vstupnímu symbolu připadá právě jedno pravidlo,
- `fnf()` hledá, neukončující stavy konečného automatu. Pokud je zadán argument find non finishing, vypíše neukončující stav na výstup jinak končí s chybovým kódem `62`. Když je zadán argument rozšíření `MWS`, automat se transformuje na dobře specifikovaný konečný automat (podle přednášek IFJ).

Když je zadán argument rozšíření `MST`, zavolá se metoda `analyzeStr()`, která na základě zadaného řetězce prochází pravidly a zjišťuje, jestli řetězec je řetězcem jazyka přijímaného zadaným konečným automatem.

Když je zadán argument pro minimalizaci konečného automatu, proběhne algoritmus z přednášek IFJ, který vytváří minimální stavy konečného automatu. Přepíše se množina stavů automatu a vytváří se nové, minimalizované pravidla automatu. Poté se zavolá instance třídy `Fsm()` nového automatu.

Jestli není argumenty zadáno jinak, na konci skriptu se vykoná metoda `printFsm()`, která vypíše dobře specifikovaný, případně požadavky upravený automat.

Implementována rozšíření projektu: `RLO`, `MST`, `MWS`