Dokumentace úlohy MKA: Minimalizace konečného automatu v Pythonu 3 do IPP 2016/2017

Jméno a příjmení: Tomáš Aubrecht

Login: xaubre02

Úvod

Zpracovává se vstupní soubor obsahující textový zápis konečného automatu. Skript postupuje následujícím způsobem. Nejprve zpracuje vstupní argumenty, načte obsah vstupního souboru, transformuje získaná data do objektové reprezentace automatu a provede syntaktickou a sémantickou kontrolu. Nakonec zkontroluje, zdali se jedná o dobře specifikovaný konečný automat. Podle zadaných vstupních parametrů se určí, zdali se má provést minimalizace a výsledný formát automatu se zapíše do daného výstupního souboru.

Zpracování argumentů

Pro tuto činnost je zde samostatná třída `Arguments`, která prochází jednotlivými argumenty programu a kontroluje, zdali jsou podporované, ve správném formátu a zdali se neopakují. Dané informace si zaznamenává do svého privátního atributu třídy `Options`, která obsahuje atributy reprezentující jednotlivé přepínače implicitně nastavené na hodnotu `False`. Ten je přístupný ostatním částem skriptu pomocí získávací metody `getOptions()`.

V případě chybného přepínače se skript ukončí s daným návratovým kódem a s podrobnější informací o chybě. Pokud byl správně použit přepínač `--help`, skript vypíše nápovědu a úspěšně skončí.

Práce se soubory

Práci se soubory umožňují dvě funkce `getContent()` a `saveContent()`, které slouží pro získání obsahu souboru a zápis dat do souboru. Zároveň se tyto metody dokáží vypořádat s chybami, které mohou v rámci jejich činnosti vzniknout, jako je například neexistující vstupní soubor nebo nedostatečná práva pro čtení či zápis.

Analýza textového zápisu KA

Za tímto účelem zde byla implementována samostatná třída `Parser`, která při inicializaci vytvoří privátní atribut třídy `FSM` reprezentující konečný automat. Tento atribut je přístupný pomocí získávací metody `getAutomaton()`. Celou analýzu řídí metoda `analyze()`. Před samotnou analýzou nejprve odstraní všechny komentáře a pomocí funkce `replace()` odstraní i všechny nepotřebné bílé znaky, tj. znaky, které nepředstavují symbol vstupní abecedy. Pokud byl zadán přepínač, který určuje, že nezáleží na velikosti znaků, převede velká písmena v řetězci na malá. Nyní je vše připraveno pro analýzu.

Ta probíhá způsobem hledání konkrétních znaků pomocí funkce `find()`. Nejprve se zkontroluje přítomnost kulatých závorek, které zapouzdřují dané komponenty. Pro jednotlivé komponenty se kontroluje přítomnost složených závorek a oddělující čárky vyznačující její meze. Pouze u komponenty představující počáteční stav se kontroluje přítomnost jen oddělující čárky. Obsahy jednotlivých komponent jsou uloženy do samostatných řetězců, které jsou následně rozděleny podle oddělujících čárek na jednotlivé položky, zkontrolovány a uloženy do atributu představující automat. Usnadnění práce s pravidly zajišťuje třída `Rules`, která provede i částečnou syntaktickou kontrolu zápisu daného pravidla.

Po vytvoření automatu se provede další syntaktická kontrola zaměřující se na korektní pojmenování všech stavů a správný zápis vstupních symbolů ve vstupní abecedě a v množině pravidel. Následná sémantická kontrola zkontroluje, zdali není vstupní abeceda prázdná, zdali jsou symboly v pravidlech zastoupeny ve vstupní abecedě a zdali je počáteční stav a koncové stavy z množiny všech stavů automatu.

Pro zpracování chyb je zde implementovaná metoda, která ukončí skript s návratovým kódem a hlášením na základě toho, zdali se jedná o chybu syntaktickou či sémantickou.

Třída FSM

Tato třída obsahuje 5 atributů reprezentující jednotlivé komponenty konečného automatu. V tuto chvíli je konečný automat syntakticky a sémanticky korektní, proto se přistupuje ke kontrole, zdali se jedná o dobře specifikovaný konečný automat.

Nejprve se prochází všechna pravidla a zjišťuje se, zdali nějaké neobsahuje prázdný řetězec. Další kontrola prochází všechny stavy automatu a symboly vstupní abecedy a zjišťuje, zdali pro nějaký stav není definováno více možných přechodů pro jeden daný symbol. Následuje kontrola počtu nedostupných stavů, kde DSKA nesmí obsahovat ani jeden a počet neukončujících stavů, kde je možná přítomnost maximálně jednoho takového. Ten se případně uloží. Poslední kontrola určí, zdali se jedná o úplný DKA, kde pro každý stav musí být definovaný právě jeden přechod pro každý symbol vstupní abecedy.

V případě zadání přepínače pro výpis neukončujícího stavu se zkontroluje, zdali byl nějaký uložen. Pokud ano, tak se vypíše daný stav. V opačném případě se vypíše 0.

Správný formátovaný výstup konečného automatu zajišťuje metoda `__str__()`.

Minimalizace KA

Probíhá způsobem rozdělení množiny všech stavů DSKA na množinu koncových stavů a množinu stavů ostatních. Ty jsou pak přidány do jednoho seznamu. Ten je předáván metodě, která dané množiny štěpí do doby, dokud je co štěpit. Pokud je po štěpení ve výsledném seznamu nějaká množina obsahující více jak jeden stav, tak se tyto stavy spojí v jeden a následně se podle toho upraví i zbytek automatu.

Samotné štěpení funguje na principu procházení jednotlivých stavů množin a všech pravidel, kde se kontroluje, zdali jsou všechny cílové stavy pro danou množinu stavů a jednotlivé symboly ze stejné množiny. Pokud ano, nic se neštěpí, ale pokud ne, tak se vytvoří nové množiny, které jsou přidány do seznamu obsahující všechny množiny. Ten se pak prochází a odstraňují se ty stavy v množinách, které jsou zastoupeny v nějaké podmnožině dané množiny.

Po těchto úpravách je výsledný DSKA zapsán do výstupního souboru.

Rozšíření

Skript dále podporuje 3 volitelná rozšíření.

RLO `-r, --rules-only`

Toto rozšíření umožňuje zpracování automatu, který je zadán zkráceným zápisem obsahující pouze pravidla. Vstupní data se připraví stejně jako v případě úplného zápisu. Dále se hledají čárky oddělující jednotlivá pravidla, která jsou ukládána do výsledného automatu. Třída `Rules` zajistí částečnou syntaktickou kontrolu. Jednotlivá pravidla se následně procházejí a ukládají se symboly, výchozí a cílové stavy. Zbylá část kontrol je stejná jako v případě úplného zápisu.

MST `--analyze-string=`„řetězec“

Toto rozšíření zjistí, zdali je zadaný řetězec řetězcem jazyka přijímaného tímto konečným automatem. Nejprve se zkontroluje, zdali jsou všechny symboly v řetězci přítomny ve vstupní abecedě. Pokud ne, automat skončí s chybou. V opačném případě začne z počátečního stavu procházet další stavy podle daného řetězce. Jakmile projde celý řetězec, tak na základě toho, zdali je poslední dosažený stav z množiny koncových stavů, vypíše zprávu na výstup.

MWS `--wsfa`

Toto rozšíření převede DKA na DSKA. Zkontroluje, zdali se skutečně jedná o DKA. Dále spočítá nedostupné stavy a pokud takové nějaké jsou, tak je odstraní. Spočítá neukončující stavy a pokud je jich více jak jeden, tak je také odstraní. Poslední kontrola se týká úplnosti DKA. Pokud se jedná o úplný DKA, tak jsou splněny všechny podmínky definice DSKA. V opačném případě je nutný převod na úplný DKA, který je zajištěn pomocí přidaného stavu `qFALSE`.