Dokumentace úlohy DKA: Determinizace konečného automatu v Pythonu3 do IPP 2012/2013
Jméno a příjmení: Vojtěch Dlápal
Login: xdlapa01

Úvod

Úlohou bylo vytvořit program typu textový filtr, který zpracuje konečný automat na vstupu splňující zadaná syntaktická pravidla na základě vstupních parametrů. Hlavní funkcí skriptu je provést determinizaci pomocí známého algoritmu. Implementačním jazykem je jazyk Python 3, který umožňuje kombinaci různých programovacích paradigmat. Já jsem zvolil objektový přístup.

Při dekompozici jsem se snažil rozpoznat jednotlivé entity, které by mohly být reprezentovány třídou. Jednotlivé třídy byly rozděleny do pěti modulů podle práce, kterou zastávají (nepočítám-li moduly s testy jednotek).

Zpracování vstupních parametrů

Hlavní modul zajišťuje zpracování parametrů a na základě jejich navolení uživatelem postupně řídí běh programu pomocí vytváření příhodných objektů a volání jejich metod. Pro zpracování parametrů jsem zvolil vestavěnou knihovnu `argparse` a to hlavně pro její flexibilitu a moderní přístup. Problémem, který jsem musel řešit, byl požadavek na návratový kód 1 při zadání neplatných parametrů. Použitá knihovna však nedává možnost toto specifikovat a ukončuje běh programu s vlastním návratovým kódem a vypsáním nápovědy. Ukončení provádí pomocí vyvolání výjimky `SystemExit`, kterou odchytávám a následně skript ukončuji s vlastním návratovým kódem.

Práce se soubory

Třídy pro práci se vstupními a výstupními daty jsou uloženy v modulu `inp`. Název je zkratkou především proto, že Input je vyhrazené slovo a při jeho používání by mohlo docházet k problémům. Z důvodu požadavku na možnost zpracovat vstupní konečný automat bez ohledu na velikost písmen je pro instanciaci objektu pro práci se vstupním souborem je použita továrna, která vybírá objekt, který velikost znaků zachovává nebo normalizuje již při jejich načítání. Konkrétní implementace je taková, že třída pro ignorování velikosti znaků dědí od třídy s nezměněným přístupem ke znakům a přepisuje metody pro načtení znaku. Toto nahrazuje rozhraní, které jazyk Python nenabízí.

Po spuštění skriptu je celý soubor načten do řetězce, soubor je následně uzavřen a k řetězci je poté přistupováno jako k souboru. To je výhodné především z výkonnostního pohledu a také z důvodu snížení rizik plynoucích z práce se soubory. Třída pro výstup nabízí metody pro zápis do výstupního proudu, který je otevřen v konstruktoru.

Lexikální analýza

V modulu `lexical` se nachází třída `Lex`, která obstarává lexikální analýzu. Proces lexikální analýzy je řízen konečným automatem, který je implementován jako množina metod, kdy každá metoda je stavem tohoto automatu. Proto je možné vzájemné vnořování metod, které charakterizuje provádění přechodů a postup automatem je tak možné sledovat pomocí stavu zásobníku. Poslední získaná lexikální jednotka je reprezentována stavem objektu lexikální analýzy `state` a jeho kombinací s obsahem atributů `char` a `idStr`. Z důvodu dodržení principu zapouzdření poskytuje třída sadu metod pro dotazování se na konkrétní lexikální jednotku.

Pro rozšíření WCH je připravena třída `LexWch`, která dědí od třídy `Lex` a přepisuje metody pro vynechání bílých znaků a zjišťování přítomnosti oddělovače.

Syntaktická analýza

Syntaktickou analýzu provádí instance třídy `Syntactic` z modulu `syntactic`. Opět se jedná o konečný automat, který v každém stavu očekává některou syntaktickou konstrukci a pokud je tato konstrukce jiná, dostává se do stavu chybového. Instance třídy `Syntactic` využívá instance třídy `Lex` pro získání další lexikální jednotky a během procházení jednotlivých stavů mění hodnoty atributů instance třídy `Automata`, čímž postupně tvoří specifikaci konečného automatu, jenž je zpracováván skriptem. Jelikož úloha pro naplnění množiny všech stavů, symbolů i koncových stavů je velmi podobná, je pro všechny tyto případy použita stejná metoda, která volí metodu pro naplnění množiny zpracovávaného konečného automatu podle stavu automatu řídícího syntaktickou analýzu, který je rozpoznán podle parametru `state`.

Reprezentace konečného automatu

Jádrem celého programu je třída `Automata` sídlící v modulu `automata`. Atributy této třídy reprezentují jednotlivé složky konečného automatu. Pro pravidla byl zvolen kontejner typu slovník, kde klíčem je vstupní stav a hodnotou je seznam dvojic symbol-výstupní stav. Díky tomuto lze velice jednoduše a efektivně zjistit jaké stavy a jakým symbolem jsou z daného stavu dostupné. Pro zbylé atributy automatu byl zvolen kontejner `set()`, který je vhodný tím, že jeho chování odpovídá matematickému pojmu množiny a tím pádem bylo možné při implementaci algoritmů pro odstranění epsilon pravidel a determinizaci konečného automatu myslet abstraktně, a kód je tak pochopitelný a snadno porovnatelný s matematickým zápisem algoritmů.

Každý úkon nad zpracovávaným automatem, je implementován ve zvláštní metodě v třídě konečného automatu, která provede potřebné operace nad sebou samým.