Dokumentace úlohy MKA: Minimalizace konečného automatu v Python 3.6 do IPP 2016/2017
Jméno a příjmení: Jiří Matějka
Login: xmatej52

1 Zadání, použité funkce a třídy

Úkolem bylo vytvořit script, který provede minimalizaci zadaného konečného automatu, včetně kontroly, zda zadaný automat odpovídá definici dobře specifikovaného konečného automatu. Pro tvorbu scriptu byly hojně využity abstraktní datové typy, které python nabízí, zejména řetězec, slovník, seznam a množina. Dále byla použita knihovna re pro kontrolu, zda jméno stavu konečného automatu je validní.

2 Postup řešení

2.1 Zpracování parametrů programu

Parametry (argumenty) programu jsou zpracovány pomocí třídy Arguments. Objekt této třídy uchovává veškeré nastavení programu. Construktor třídy zároveň kontroluje správnost argumentů.

2.2 Načítání vstupních dat

Při načítání vstupního souboru byly odstraněny veškeré bílé znaky a komentáře. Následně byl takto načtený řetězec zpracován pomocí rozsáhlého stavového automatu, během kterého probíhala kontrola syntaktických a sémantických chyb.

2.3 Zpracování načtených vstupních dat

Zpracování načtených dat byla nejnáročnější část úlohy a zpracovávala se pomocí metod třídy FiniteStateMachine, která sloužila i k reprezentaci konečného automatu. Nejdříve byly nalezeny dostupné a nedostupné stavy, následně neukončující stavy, poté nedeterminismy a následně epsilon přechody. Pokud kontrola, zda se jedná o dobře specifikovaný konečný automat proběhla v pořádku, mohlo dojít k výpisu výsledků (pokud nebyla požadována minimalizace). Samotná tvorba algoritmu pro minimilizaci zabrala nejvíce času. Metoda pro minimalizaci obsahuje 2 pomocné funkce a 5 – 6 do sebe vnořených cyklů. Algoritmus je založen na algoritmu probíraném v rámci předmětu IFJ.

2.4 Vytvoření výstupního souboru

Výstupní soubor se otvírá až poté, co proběhla kontrola syntaktických a sémantických chyb a nepředpokládá se, že by během programu mohla nastat chyba.

2.5 Řešení chybových stavů

Chyby otevření souborů jsem řešil pomocí odchycení vyjímky. Ostatní chyby jsem detekoval pomocí podmíněných skoků. Volání exit zastávala vlastní funkce, která vytiskla požadovanou zprávu na standartní chybový výstup a ukončila program se zadaným chybovým kódem.

2.6 Testování

Během tvorby scriptu jsem script testoval na svém stroji, (Ubuntu 16.04) a to interpretem Python 3.5. Poté, co mi script pracoval na mém stroji, jsem testování spustil na školním serveru Merlin, kde proběhlo, až na jeden referenční test (chybějící odřádkování, které nakonec bylo správné), v pořádku.

2.7 Odevzdání

Úlohu jsem se snažil dokončit tak, abych stihl pokusné odevzdání. Až na pár drobných úprav se mi to podařilo i s rezervou jednoho dne. Výsledné hodnocení programu bylo v rozmezí od 80% – 100%. Dokumentaci jsem tvořil v den pokusného (po pokusném odevzdání probělo pár drobných úprav) a to pomocí LaTeXu. Při dokumentaci kódu jsem se snažil používat dokumentační řetězce ke všem funkcím a třídám a snažil jsem se přehledně komentovat svůj kód, aby byl co nejlépe čitelný a to nejen pro mě. Dokumentace kódu je v anglickém jazyce.

3 Shrnutí z pohledu autora

Tento projekt nebyl mou první zkušeností s jazykem Python, a už jsem byl při jeho tvorbě zvyklý na syntax a měl jsem alespoň minimální povědomí, jak interpret Pythonu funguje. Zkušenosti s Pythonem jsem měl v rámci předmětu scriptovací jazyky a spolupráce s výzkumnou skupinou KNOT. Nicméně přesto Python není u mě nijak oblíbený jazyk, zejména kvůli syntaxy. Programy v Pythonu mi osobně přijdou značně nečitelné, zejména, když odsazení od začátku stránky je už příliš velké.