Zadání úlohy do projektu z předmětu IPP 2011/2012

Zbyněk Křivka a Dušan Kolář
E-mail: {krivka, kolar}@fit.vutbr.cz, {54 114 1313, 54 114 1238}

MKA: Minimalizace konečného automatu

Zodpovědný cvičící: Zbyněk Křivka (krivka@fit.vutbr.cz)

1 Detailní zadání úlohy

Vytvořte skript pro zpracování a případnou minimalizaci konečného automatu. Skript bude zpracovávat textový zápis zadaného dobře specifikovaného konečného automatu a generovat ekvivalentní minimální konečný automat přesně podle algoritmu ze čtvrté přednášky (snímek 39/44) předmětu Formální jazyky a překladače (IFJ).

1.1 Formát vstupu

Komentáře do konce řádku začínají znakem #. Bílé znaky jako konec řádku (`\n` i `\r`), mezera či tabulátor jsou ignorovány (až na později definované případy).

Stav je reprezentován identifikátorem jazyka C, který nezačíná ani nekončí podtržítkem. Vstupní symbol je reprezentovaný libovolným znakem, který může, ale nemusí, být v apostrofech. U stavů i vstupních symbolů záleží na velikosti písmen[1]. Znaky, které musí být v apostrofech, jsou všechny metaznaky popisu automatu a všechny bílé znaky, tj. `'('`, `')'`, `'{'`, `'}'`, `''''`, `'-'`, `'>'`, `','`, `'.'`, `'|'`, `'#'`, `' '`, znak konce řádku v apostrofech a znak tabulátoru v apostrofech. Apostrof musí být navíc uvnitř apostrofů zdvojený. Prázdná dvojice apostrofů `''` reprezentuje prázdný řetězec[2].

Celý konečný automat je zapisován podobnou notací jako ve formálních jazycích (IFJ). Celý konečný automat je pětice uzavřená do kulatých závorek. Každá komponenta kromě komponenty určující startující stav je dále uzavřena ve složených závorkách a od ostatních komponent oddělena čárkou. Jednotlivé prvky množin reprezentujících komponenty (opět kromě startujícího stavu) jsou odděleny také čárkou.

Nejprve je definována konečná množina stavů, následuje neprázdná vstupní abeceda, poté definice množiny pravidel, dále určení startujícího stavu a nakonec množina koncových stavů. Množina pravidel je popsána seznamem pravidel. Každé pravidlo je zapsáno ve tvaru: *p a*`->`*q*, kde *p* je *výchozí stav*, *a* je *čtený vstupní symbol* (může být i vynechán nebo nahrazen reprezentací prázdného řetězce), následuje dvojznak pomlčka s větším reprezentující šipku (tento dvojznak nesmí být rozdělen jiným znakem) a poslední část pravidla *q* určuje *cílový stav*. Pokud není *a* prázdné nebo uvedeno v apostrofech, tak musí být *p* a *a* odděleno alespoň jedním bílým znakem.

Příklad vstupního zápisu konečného automatu:

```
# Příklad konečného automatu ve vstupním formátu úlohy MKA
({s, f,q4,q2 , q1, # nějaký komentář
  q3}, # nějaký komentář
  {á, ')' }, {
    s á->f, f á ->s, s')'->q3, s ')' -> q4,
    q1 á->q1, q2 á -> q2, q3 á->q4, q4 á->
```

[1] tj. definice automatu je case-sensitive

[2] V IFJ byl prázdný řetězec značen řeckým písmenem ε. V projektu z IPP tomu tak nebude.

```
  q3, q1 ')' -> s, q2')'
  ->f , q3 ')'->q1, q4')'->q2  # další komentář
}, # následuje komponenta definující startující stav
s
, {f, s } ) # koncové stavy a ukončení definice automatu
# zde může následovat libovolný počet bílých znaků nebo komentářů
```

1.1.1 Kontrola správnosti vstupu

Vstupní automat nesplňující popsaná lexikální a syntaktická pravidla ukončí skript s chybou a návratovou hodnotou `60`. Pokud je porušena neprázdnost vstupní abecedy nebo nejsou první a druhá komponenta disjunktní nebo startující stav není v množině stavů nebo množina koncových stavů není podmnožinou množiny stavů, tak skript ukončete se sémantickou chybou a návratovou hodnotou `61`. Při opakování stejných pravidel/stavů/symbolů v rámci jedné komponenty jsou tato ve výsledné množině pouze jednou (tj. množina na výstupu není multimnožinou). Při načítání vstupu je také automaticky prováděna kontrola, že se skutečně jedná o dobře specifikovaný konečný automat (viz přednášky IFJ). Pokud vstup nereprezentuje dobře specifikovaný konečný automat, skončí skript s chybou a vrátí návratovou hodnotu `62`. Kombinace více chyb nebude testována.

1.2 Transformace a formát výstupu

Popis algoritmu je odkázán v referencích (snímek 39/44). Algoritmus je kvůli testování výsledků závazný. Po načtení automatu je třeba provést kontrola, že se jedná skutečně o dobře specifikovaný konečný automat (viz snímek 34/44), a pak pomocí algoritmu ze snímku 39/44 můžeme provést minimalizaci.

Při minimalizaci se provádí tzv. *slučování a štěpení stavů*. Pro jednotný výsledek bude potom výsledný identifikátor pro štěpený stav definován jako spojení všech původních stavů (resp. jejich identifikátorů) pomocí znaku podtržítko v lexikografickém vzestupném pořadí (pořadí jednotlivých znaků je určeno jejich ordinální hodnotou). Pokud dojde k atomickému štěpení až na původní stav, tak se samozřejmě jeho identifikátor využije jako identifikátor i ve výsledném automatu. Například, pokud dostaneme po minimalizaci stav reprezentující množinu původních stavů {`s1`, `p2`, `p`, `P2`}, tak výsledný stav bude mít identifikátor `P2_p_p2_s1`, kdy jsou jednotlivé stavy spojeny podtržítkem v lexikografickém pořadí.

1.2.1 Normální forma výstupu

Výstupní formát výsledného konečného automatu vychází ze vstupního formátu a je definován následující normální formou. Všechny komentáře a nadbytečné bílé znaky budou vypuštěny. Automat bude vypsán v úplném tvaru a s každou komponentou začínající na zvláštním řádku. Kromě množiny pravidel budou všechny komponenty právě na jednom řádku. Za každou komponentou bezprostředně následuje čárka a odřádkování[3]. Za poslední komponentou nebude čárka ale bude pouze odřádkování, takže uzavírací kulatá závorka bude na novém řádku. Stavy v množině stavů, resp. symboly ve vstupní abecedě, budou odděleny čárkou a jednou mezerou (za posledním prvkem nebude čárka ani mezera) a ve svých komponentách seřazeny lexikograficky vzestupně. Každý symbol vstupní abecedy bude navíc uveden v apostrofech.

Každé pravidlo z množiny pravidel bude začínat na novém řádku (tj. odřádkování bude i za otevírající levou složenou závorkou; uzavírající pravá složená závorka bude též na novém řádku).

[3]Odřádkování provádějte unixovým způsobem (znakem `LF`).

Oddělující čárka bude bezprostředně za identifikátorem cílového stavu a za ní rovnou odřádkování. Množina výsledných pravidel bude seřazena vzestupně do seznamu pravidel podle sdruženého klíče ze tří podklíčů. Primární klíč je identifikátor výchozího stavu (řazeno lexikograficky podle ordinálních hodnot znaků), sekundární je znak reprezentující vstupní symbol bez případných uvozujících apostrofů (řazeno podle ordinální hodnoty znaku, prázdný řetězec má hodnotu 0), posledním podklíčem je identifikátor cílového stavu (řazeno analogicky jako výchozí stavy).

Vstupní symbol obsažený v pravidlech bude na výstupu vždy uveden v apostrofech (stejně tak prázdný řetězec, který bude též v apostrofech). Části pravidla budou odděleny právě jednou mezerou, jak je vidět na příkladu formátování jednoho pravidla na výstupu (včetně oddělující čárky za pravidlem):

```
stav1_stav2 'a' -> stav2_stav3,
```

Tento skript bude pracovat s těmito parametry:

- `--help` viz společné zadání všech úloh.
- `--input=filename` zadaný vstupní textový soubor v UTF-8 s popisem dobře specifikovaného konečného automatu.
- `--output=filename` textový výstupní soubor (opět v UTF-8) s popisem výsledného ekvivalentního[4] konečného automatu v předepsaném formátu.
- `-f`, `--find-non-finishing` hledá neukončující stav zadaného dobře specifikovaného konečného automatu. Nalezne-li jej, vypíše jej na výstup; jinak skript nevypíše nic. (Před hledáním se provede validace na dobrou specifikovanost automatu.) Parametr nelze kombinovat s parametrem `-m` (resp. `--minimize`).
- `-m`, `--minimize` provede minimalizaci dobře specifikovaného konečného automatu (viz algoritmus IFJ, přednáška 4, snímek 39/44). Parametr nelze kombinovat s parametrem `-f` (resp. `--find-non-finishing`).
- `-i`, `--case-insensitive` nebude brán ohled na velikost znaků při porovnávání symbolů či stavů (tj. `a = A`, `ahoj = AhOj` nebo `A_b = a_B`); ve výstupu potom budou všechna velká písmena převedena na malá.

Pokud nebude uveden parametr `-m`, tak dojde pouze k validaci načteného dobře specifikovaného konečného automatu a k jeho normalizovanému výpisu.

2 Bonusová rozšíření

- **WHT** (0,5 bodu): Volba `-w`, `--white-char` ve vstupu lze oddělující čárku nahradit libovolným bílým znakem (i komentář je brán jako bílý znak).
- **RLO** (0,5 bodu): Volba `-r`, `--rules-only` vstupní soubor obsahuje zkrácený vstupní zápis tj. pouze množinu pravidel automatu (nikoli ostatní komponenty). Ve zkráceném zápisu jsou na vstupu pouze jednotlivá pravidla (bez složených závorek ohraničujících množinu pravidel v úplném zápisu). U pravidla může navíc za cílovým stavem následovat tečka, která označuje cílový stav zároveň jako koncový (tečku není třeba opakovat u každého výskytu stejného

[4] Ekvivalentní konečný automat je definován jako automat přijímající stejný jazyk.

stavu). Výchozí stav prvního pravidla je definován jako startující stav. Zkráceně zapsaný konečný automat obsahuje právě všechny stavy a symboly použité v seznamu pravidel. Všechny komentáře jsou zahazovány.

Příklad:

```
start a -> start, start b -> stav2, stav2 'a' -> stav2, # nějaký komentář
stav2 b -> stav1 ,stav1 a -> stav1 ,stav1 b -> start. # start je koncový stav
```

- **MST** (1 bod): Po zadání bonusového parametru `--analyze-string="retezec"` (nelze kombinovat s `-m` a `-f`) provede algoritmus analýzu, zda je zadaný `retezec` řetězcem jazyka přijímaného zadaným konečným automatem. Pokud ano, vypíše na výstup samotnou číslici `1`, jinak vypíše `0`. POZOR! Návratový kód je vzhledem k validní funkčnosti skriptu v obou případech `0`. V zadaném řetězci nemusíte uvažovat výskyt znaků uvozovek a apostrofů.
- **MWS** (1,5 bodu): Po zadání bonusového parametru `--wsfa` bude skript akceptovat i obyčejný deterministický konečný automat[5] (neskončí s chybou) a transformovat jej na dobře specifikovaný konečný automat (pomocí algoritmu ze 4. přednášky IFJ na snímku 35/44; případný maximálně jediný neukončující stav bude mít identifikátor `qFALSE`). Tento automat pak bude skript v případě kombinace s parametrem `-m` dále minimalizovat, jinak jej pouze normalizovaně vypíše.

Reference:

- A. Meduna, R. Lukáš. *Přednášky předmětu Formální jazyky a překladače (IFJ): Kapitola IV. Speciální typy konečných automatů.* FIT VUT v Brně, 2011. [cit. 2012-02-09]. Dostupné z: `https://www.fit.vutbr.cz/study/courses/IFJ/private/prednesy/Ifj04-cz.pdf`

3 Specifické požadavky na dokumentaci

Popište techniku ověřování lexikální a syntaktické správnosti vstupu. V případě minimalizace nepopisujte obecný algoritmus, ale specifika Vaší konkrétní implementace.

4 Poznámky k hodnocení

Výsledný automat bude nejprve porovnán na přesnou shodu pomocí nástroje `diff`. V případě neúspěchu bude s bodovou srážkou provedena normalizace jinak syntakticky správného výstupního konečného automatu a uskutečněno nové porovnání.

[5] Předpokládejte, že vstupní konečný automat nemá stav `qFALSE`.