Formal Languages and Compilers

08 July 2013

Using the JFLEX lexer generator and the CUP parser generator, realize a JAVA program capable of recognizing and executing the programming language described later.

Input language

The input file is divided into three sections: *header*, *states* and *transitions*, separated by means of the token "%%". Semantic actions are required only for the last two sections. The input file can contain C stile **comments**.

The *header* section can contain 3 types of tokens terminated with the character ";":

- <hour>: a hour with the format "HH:MM" (between 08:35 and 17:49) or in the 12 hours format with the words "am" or "pm" to indicate morning or afternoon (between the same range, i.e., between 08:35am and 05:49pm).
- <code>: it begins with the character "?", followed by an **even** number (**at least 4**) of lowercase alphabetic letters **or** an **odd** number (**at least 3**) of uppercase letters, the character "?", **optionally** followed by **3 or more** repetitions of the words ("xx", "xy", "yx" or "yy") in **any** combination.
- <ip>: an IP address, followed by the character ":", followed by an **even** number between −6 and 128.

Header section: grammar

In the *header* section the 3 tokens can appear in two ways:

- in **any order and number** with the exception of the token <code> that must appear **exactly one time**
- or, **3 or more** tokens <code>

States section: grammar and semantic

In the *states* section, the state variables are defined and initialized. The *states* section is composed of a list of <states> in **odd** number (**at least 3**).

A <state> is the word "STATE", followed by an <identifier> (i.e., a word that begins with a letter or the character "_" and followed by any number of letters, numbers and characters "_"), a "{", a list of <variables_declaration> (separated with ";") and a "}". A <variables_declaration> is an <identifier>, the symbol "=" and a signed integer number.

At the end of this section a data structure that contains, for each state, the values of the variables declared in the state, must be filled (remember that variables with the same name must appear in more than one state). **This data structure is the only global variable allowed in the project.**

Transitions section: grammar and semantic

The *transitions* section begins with the word "START" followed by an <identifier> and ended with the character ";". It sets the *current state* to <identifier>. It is followed by a list (eventually **empty**) of <commands>. Each <command> is ended with the character ";".

Two types of commands are defined:

- *PRINT*: print the *current state.*
- *IF*: manages states transitions. The *IF* command has the following two grammars:

  `IF STATE` id_1 `# <boolean_expression> THEN STATE` id_2 `;`

  `IF STATE` id_1 `# <boolean_expression> THEN STATE` id_2 `; ELSE STATE` id_3 `;`

  The behavior of the `IF` command is: if the *current state* is not equal to id_1, do nothing (i.e., the state remain the current). In the case the *current state* is equal to id_1, if <boolean_expression> is *TRUE*, the new state is id_2, else (in the case the `ELSE` part of the `IF` command is present) the new state is id_3, otherwise (i.e., without the `ELSE` part and with a *FALSE* <boolean_expression>) the current state is not changed.

  <boolean_expression> has the same syntax of an expression of a typical if instruction of the C programming language. <boolean_expression> can contain signed integer numbers or variables (variables are referred

with the syntax *a.b* where *a* represents a state, while *b* the name of a state variable; to access the value of a variable, a lookup on the data structure filled in the *states* section must be performed). The only comparison operators required are “==” and the required boolean operators are AND “&&” and OR “||” (look the example).

Goals

The realized translator must execute the program of the last two sections. The `IF` command, for its execution, must know the *current state* (**inherited attributes** must be used to this extent). **Solutions that do not use inherited attributes to access the current state value will not be accepted.**

Example

Input:

```
/* Header section: the 3 tokens can appear in any order and number, but only one <code> */
/* or 3 or more <code> */
?ABCDE?xxxyxxyy; /* <code> */
09:10;           /* <hour> */
192.168.0.1:-2;  /* <ip>   */
09:10am;         /* <hour> */

%%

/* States section */
STATE A {x=10; y=11; z=13;}
STATE B {x=3; z=5;}
STATE C {x=1;}

%%

/* Transitions section */
START A;
PRINT;     /* print A */
IF STATE A # A.x==3 || A.x==10 && B.x==3 THEN STATE B;        /* Current state is A, boolean exp.
                                                                  is TRUE -> go to STATE B */
PRINT;     /* print B */
IF STATE A # A.x==3 || A.x==10 && B.x==3 THEN STATE A;        /* Current state is B ->
                                                                  no state change */
PRINT;     /* print B */
IF STATE B # A.x==2 THEN                                       /* Current state is B, boolean exp.
                                                                  if FALSE */
   STATE A;
ELSE
   STATE C;                                                    /* -> go to STATE C */
PRINT;     /* print C */
```

Output:

```
A
B
B
C
```