INFO-F-403 Introduction to Language Theory and Compilation
Project Part 1: S-COBOL Lexical Analyzer

Delhaye Quentin

October 30, 2013

1 Presentation of the implementation

The program is divided into three classes: `Main`, `LexicalAnalyzer` and `SymbolsTable`.

1.1 `Main`

This is the public executable class interacting with the user. It will ask him to enter a line of S-COBOL code, will store it into a String `str`, and then will feed it to the lexical analyzer until the String is empty. The lexical analyzer will return an array of two Strings, the first being the token to subtract to `str`. If the token contains a special character from the point of view of the regular expression (such as `*` or `(`, for example), those characters need to be escaped before the subtraction, and then un-escaped after it to allow a clean output. When `str` is finally empty, the class prints on the standard output all the token/lexical unit couples sent back by the lexical analyzer.

This ends when the user stops entering code by simply hitting the `return` key without entering anything. When he does that, the class asks the lexical analyzer to print the table of symbols.

1.2 `LexicalAnalyzer`

This class receives a String `input` from the `Main` class. The first things it does with it is checking if it is a comment: `input` is in a new line and it matches the following regex: `(\*|/).*\n$`.

If it is not a comment, the method `nextToken` splits `input` at the spaces and submits the first chunck at every regex or list until a match is found. If no match is found, the next piece of `input` is concatenated using a space and submited again until a match is found or all the pieces have been put back together.

The various test are the following:

- `^\.\n$`, `ouput` is the end of the line;
- contained in the list `keywords` which is formed from the exhaustive list. An other list, `units`, holds the corresponding lexical unit at the same index.
- `^'[\w\ \-\+\*/:!\?]*'$`, String.
- `0|[\+\-]?[1-9][0-9]*`, integer.
- `(0|[\+\-]?[1-9][0-9]*)(\.[0-9]+)?`, real number. This regex is evaluated after the integer because a real number without a decimal part can be considered as a simple integer.
- `s?9(\([1-9]\))?(v9(\([1-9]\))?)?`, image.
- `^[a-zA-Z][\w\-]{0,14}`, identifier.

Each time an identifier is encountered, the token, as well as the line number on which it occurred, are sent to the table of symbols.

Each time an image is encountered, if an identifier preceded it, they are both sent the the table of symbols.

If no valid token can be found in the input String, the lexical analyzer sends `"ERROR"` as a lexical unit.

1.3 `SymbolsTable`

This class receives either variables or labels. In the first case, it must check if the identifier is not already stored before saving it in the corresponding list. In the later case, it checks if the identifier already is in the list, and if it is, if the line number is different. If the line number is different, or if the identifier is brand new, the couple is apended to the list.

When the class is asked to format the list through the `toString` method, it first sorts both of the list using the buble sort algorithm. The rest is just producing a String allowing a nice output.

2 Hypothesis

- Each line is supposed to end with a `dot`.
- Every comments occupy full line and begin with `*` or `/`.
- In order to avoid confusion, the user will leave space between the operands and the operators.
- Hyphenation is not allowed inside a String token.

3 Automaton Reduction and Regular Expressions

3.1 Image

The following will prove that the DFA shown at figure 1 can be reduced to a regular expression. The transformation from a DFA into an RE will be achieved through state elimination.

Note: `[1-9]` represents the range `[1;9]`

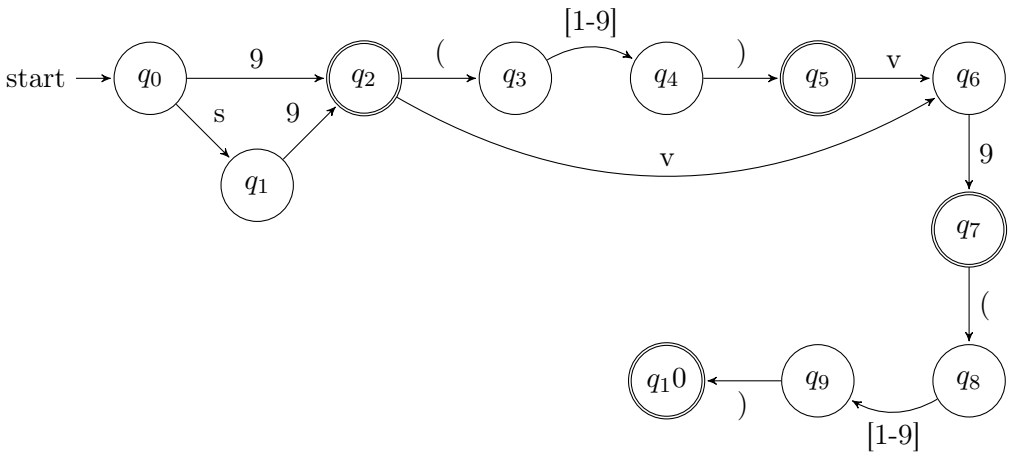
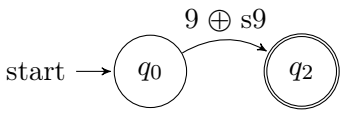
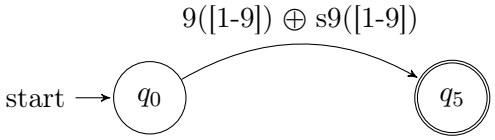


Figure 1: DFA recognizing an image

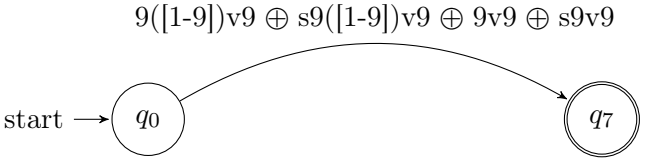
- $q_0 \rightarrow q_2$



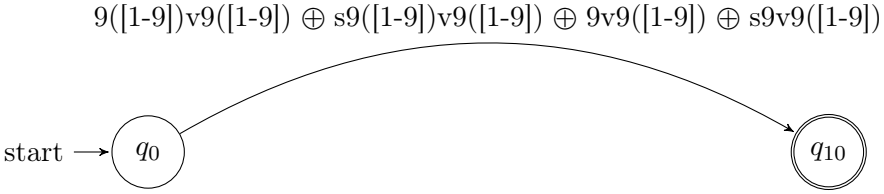
- $q_0 \rightarrow q_5$



- $q_0 \rightarrow q_7$



- $q_0 \rightarrow q_{10}$



The regular expression corresponding to the whole DFA is then:

$$RE = 9 \oplus s9 \oplus 9([1-9]) \oplus s9([1-9]) \oplus 9([1-9])v9 \oplus s9([1-9])v9 \oplus 9v9 \oplus s9v9$$
$$\oplus 9([1-9])v9([1-9]) \oplus s9([1-9])v9([1-9]) \oplus 9v9([1-9]) \oplus s9v9([1-9])$$

Knowing that $(A \oplus AB) = A(\epsilon \oplus B)$, we have:

$$RE = [\![\{\epsilon \oplus s\}9]\!] \oplus [\![\{\epsilon \oplus s\}9([1-9])]\!] \oplus [\![\{\epsilon \oplus s)9([1-9])v \oplus \{\epsilon \oplus s)9v]\!]9$$
$$\oplus [\![\{\epsilon \oplus s\}9([1-9])v \oplus \{\epsilon \oplus s\}9v]\!]9([1-9])$$
$$\Leftrightarrow RE = \{\epsilon \oplus s\}9[\![\epsilon \oplus \{\epsilon \oplus ([1-9])\} \oplus \{([1-9])v \oplus v\}9 \oplus \{([1-9])v \oplus v\}9([1-9])]\!]$$
$$\Leftrightarrow RE = \{\epsilon \oplus s\}9[\![\epsilon \oplus \{\epsilon + ([1-9])\} \oplus \{([1-9])v \oplus v\}9\{\epsilon \oplus ([1-9])\}]\!]$$
$$\Leftrightarrow RE = \{\epsilon \oplus s\}9[\![\epsilon \oplus \{\epsilon \oplus ([1-9])\} \oplus \{([1-9]) \oplus \epsilon\}v9\{\epsilon \oplus ([1-9])\}]\!]$$

Using the fact that $A \oplus \epsilon = \epsilon \oplus A$, we find:

$$RE = \{\epsilon \oplus s\}9[\![\epsilon \oplus \{\epsilon \oplus ([1-9])\}\{\epsilon \oplus v9\{\epsilon \oplus ([1-9])\}\}]\!]$$

According to the extended regular expressions, $\epsilon \oplus A$ corresponds to $A?$. Using that relation, we can deduce the following expression:

```
s?9([1-9]?(v9[1-9]?)?)?
```

3.2 Integer

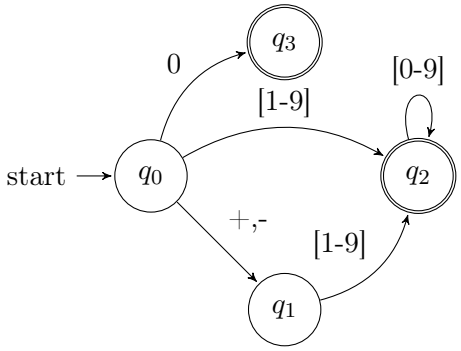
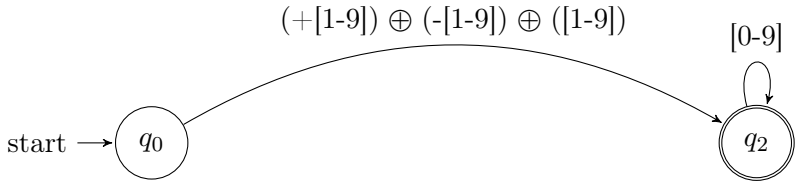


Figure 2: Automaton recognizing an integer

- $q_0 \rightarrow q_2$



$$RE = [\![(+[1-9]) \oplus (-[1-9]) \oplus ([1-9])]\!] \oplus [\![(+[1-9]) \oplus (-[1-9]) \oplus ([1-9])]\!][0-9]^*$$
$$\Leftrightarrow RE = [\![(+[1-9]) \oplus (-[1-9]) \oplus ([1-9])]\!][\![\epsilon \oplus [0-9]^*]\!]$$
$$\Leftrightarrow RE = [\![+ \oplus - \oplus \epsilon]\!][1-9][\![\epsilon \oplus [0-9]^*]\!]$$

Since `a` $\oplus$ `b` is equivalent to the extended regular expression `[ab]`, we have:

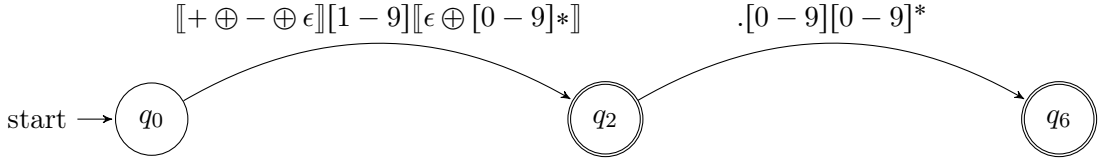
```
0 | [+-]?[1-9][0-9]*
```

3.3 Real

- $q_0 \rightarrow q_2$ Same result as for the integer:

$$[\![+ \oplus - \oplus \epsilon]\!][1-9][\![\epsilon \oplus [0-9]*]\!]$$

- $q_0 \rightarrow q_6$



$$RE = [\![+ \oplus - \oplus \epsilon]\!][1-9][\![\epsilon \oplus [0-9]^*]\!] \oplus [\![+ \oplus - \oplus \epsilon]\!][1-9][\![\epsilon \oplus [0-9]^*]\!].[0-9][0-9]^*$$
$$\Leftrightarrow RE = [\![+ \oplus - \oplus \epsilon]\!][1-9][\![\epsilon \oplus [0-9]*]\!] \oplus [\![\epsilon \oplus .[0-9][0-9]^*]\!]$$

Knowing that AA^* is equivalent to the extended regular expression `A+`, we have:

```
(0 | [+-]?[1-9][0-9]*)(.[0-9]+)?
```

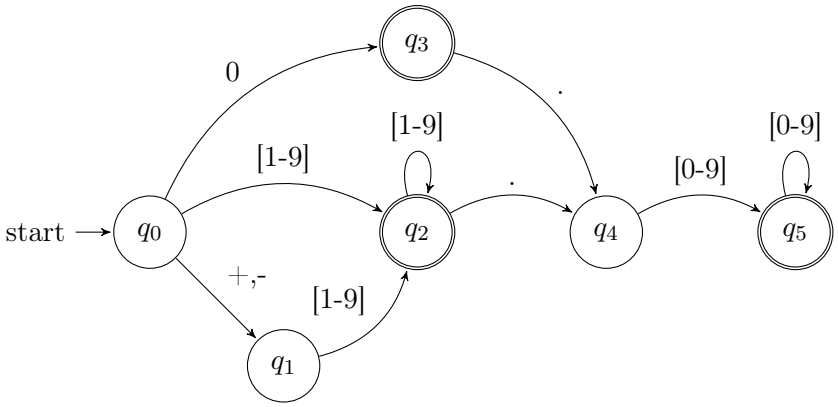


Figure 3: Automaton recognizing a real

4 Notes on the DFA

It is to be noted that the DFA Simulator can't recognize the comma – , – nor the "end of line" character.

The rejecting state is label `ERROR`.