COL703 - Assignment1
Scanning And Parsing

Suyash Agrawal
2015CS10262

September 13, 2017

1 Lexer

Lexer parsed the input into the following tokens:

- NOT
- OR
- AND
- IF
- THEN
- IFF
- ELSE
- TRUE
- FALSE
- LPAR (denotes '(')
- RPAR (denotes ')')
- ATOM (denotes string)
- EOL (denotes end of line)

2 Parser

In order to correctly parse the grammar without using high level directives like "%left" , I introduced additional non-terminals in my grammar. The Terminals were:

```
P_NOT, P_AND, P_OR, P_IF, P_THEN, P_IFF, P_ELSE, P_TRUE
P_FALSE, P_RPAR, P_LPAR, P_ATOM, P_ILLCH, P_EOF, P_EOL
```

The Non-Terminals were:

```
propListR,propR,iffR,iteR,orR,andR,negR,basicR,wordR
```

My grammar is as follows:

```
propListR: propR P_EOL propListR        (propR::propListR)
    | propR                             ([propR])

propR: P_IF propR P_THEN propR          (IMP(propR1,propR2))
    | iffR                              (iffR)

iffR: iteR P_IFF propR                  (IFF(iteR,propR))
    | iteR                              (iteR)

iteR: P_IF propR P_THEN iffR P_ELSE propR (ITE(propR1,iffR,propR2))
    | orR                               (orR)

orR: orR P_OR andR                      (OR(orR,andR))
    | andR                              (andR)

andR: andR P_AND negR                   (AND(andR,negR))
    | negR                              (negR)

negR: P_NOT negR                        (NOT(negR))
    | basicR                            (basicR)

basicR: P_TRUE                          (TOP)
    | P_FALSE                           (BOTTOM)
    | P_LPAR propR P_RPAR               (propR)
    | wordR                             (ATOM(wordR))

wordR: P_ATOM wordR                     ((P_ATOM)^" "^(wordR))
    | P_ATOM                            (P_ATOM)
```

The basic intuition was to first club the tokens around the high priority operators like AND OR etc. and then use these non-terminal in the grammar of other tokens. Also, ordering of these non-terminal were used to generate right and left associativity of the operators.

These were the essence of my grammar:

- Ground terms and parenthesized expressions are given highest preference.
- Priority of NOT , AND , OR are implemented using hierarchical structure of grammar
- No unparenthesised "IF THEN" expression is allowed to occur between "THEN" and "ELSE" of "IF THEN ELSE" operator.
- IFF is made right associative by constraining the leftmost IFF to contain expression of higher precedence of IFF.

3 Running

```
sml wrapper.sml <input_file> <output_file>
```

NOTE:

- The input file should not contain '.' at end of statements.
- The output file should exist. i.e. Program will not automatically create the file.