Breve Illustrazione del Progetto "RegularExpressionStringRecognizer" (GIUSEPPE FAGNANI)

Grammatica

```
grammar RegularExpression;
start : regEx stringList;
regEx: choice;
choice : choice '+' concatenation #union
   | concatenation #goToConcat
   ;
concatenation : concatenation iteration #concat
   | iteration #goToIteration
   ;
iteration : iteration'*' #iterat
   | end #goToEnd
   ;

end : '('regEx')' #parens
   | ID #id
   | '_' #epsilon
   ;
stringList : ',' WORD* stringList
   |
   ;
ID : [a-zA-Z0-9];
WORD : ('a'..'z'|'A'..'Z'|'0'..'9'|'_');
WS : [ \t\r\n]+ -> skip;
```

Mi sono semplicemente limitato a "tradurre" nel linguaggio corretto la grammatica vista a lezione (ho solo aggiunto una rule (end)) per quanto riguarda la parte della regular expression; per la lista di stringhe ho dovuto aggiungere il termine WORD per indicare che la parola può anche essere vuota ('_'). Inoltre ho dato delle labels a ciascuna regola relativa alla regular expression.

Per creare le classi del Lexer, Parser, Listener e Visitor ho successivamente lanciato il comando: "antlr4 –visitor RegularExpression.g4".

Implementazione

Per prima cosa ho dovuto creare una classe per il funzionamento dell'algoritmo di Thompson, utile per la creazione di un NFA. Ho quindi creato, dopo aver modellato le classi Node e Edge, i quattro metodi utili per l'algoritmo: singleCharInput (crea un NFA semplice prendendo come input un carattere, possibilmente anche '_'), union (crea un NFA prendendo come input altri due NFAs ottenendo come risultato l'automa dell'unione tra i due), concatenation (crea un NFA prendendo come input altri due NFAs ottenendo come risultato l'automa della concatenazione tra i due) e kleeneStar (crea un NFA prendendo come input un NFA ottendendo come risultato l'automa dell'iterazione di esso).

Successivamente ho implementato il Visitor (NFABuilder) sfruttando i metodi messi a disposizione dalla classe generata da ANTLR4 (RegularExpressionBaseVisitor), per creare un modo dinamico l'NFA con un approccio bottom-up.

Infine ho usato un Listener (StringListListener) per parsare la lista di stringhe e usarle come input per il metodo recognizer della classe Thompson. In questo modo sono riuscito a stampare a video OK se la stringa è accettata dall'automa, KO altrimenti.

TEST

Alcuni screenshot dei test effettuati (solo i casi più significativi):

```
C:\Users\pc\Progetti\Compilers\RegularExpressionStringRecognizer>java main.Main
a+bc,a,b,bc
^Z
OK KO OK
C:\Users\pc\Progetti\Compilers\RegularExpressionStringRecognizer>java main.Main
a*b,aab,ba,b,abb
^Z
OK KO OK KO
C:\Users\pc\Progetti\Compilers\RegularExpressionStringRecognizer>
```

```
C:\Users\pc\Progetti\Compilers\RegularExpressionStringRecognizer>java main.Main
(a+bc)*,aaabca,_,bcbc,aaaab
^Z
OK OK OK KO
C:\Users\pc\Progetti\Compilers\RegularExpressionStringRecognizer>java main.Main
(a+bc)*d,abcd,_,d
^Z
OK KO OK
C:\Users\pc\Progetti\Compilers\RegularExpressionStringRecognizer>
```