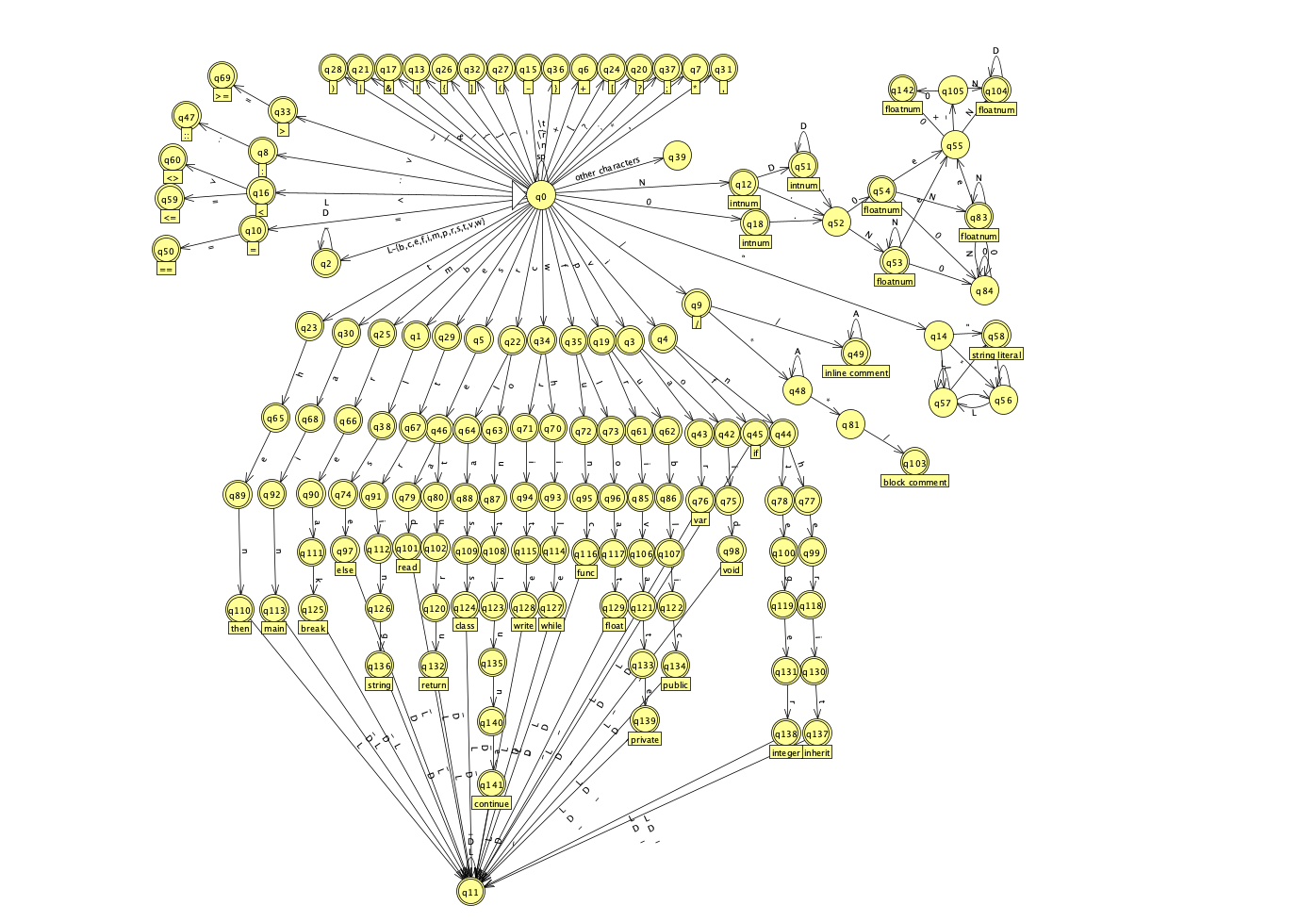
1. Lexical specification
   1. Changes on error types, added:
      1. Invalid string
      2. Invalid comment
   2. Changes to represent Finite State Automaton
2. sp: space
3. A: all ASCII printable characters
4. D: digits [0...9]
5. N: nonzero digits [1…9]
6. L: letters [a…z] & [A…Z]
7. Unlabeled final states are ID
   1. Specification
      1. Integer: ND\*|0
      2. Fraction: .D\*N|.0
      3. Float: (ND\*|0) (.D\*N|.0) [e[+|-]ND\*|0]
      4. Id: L(L|D|\_)\*
      5. String: “(L|D|\_|sp)\*”
8. Final state automaton
9. Design
   1. Token.java: Class containing tokenized value
      1. Lexeme: content of the token
      2. Type: regular type and error type
      3. Location: the token is located in which line
   2. TokenType.java: Enum class containing regular type and error type
   3. State.java: Data structure representing a state in DFA
      1. The output token type
         1. Final: regular token type
         2. Non-final: error token type
      2. Next possible states: a map with transition (character) as key, next states as values
   4. DFA.java: representation of graphic DFA using linked lists of States
      1. Using DFA instead of NFA avoids backtracks
   5. LexicalAnalyser.java: read input file, and return next token
   6. Main.java: Driver, write tokens into output files
10. Use of tools
    1. Finite state automaton: JFLAP7.1 (<http://www.jflap.org/>)
    2. State.java uses HashMap in java util (<https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>)