Name: Gavin Lloyd
Date: 06 Feb 2012

Milestone 2 Report

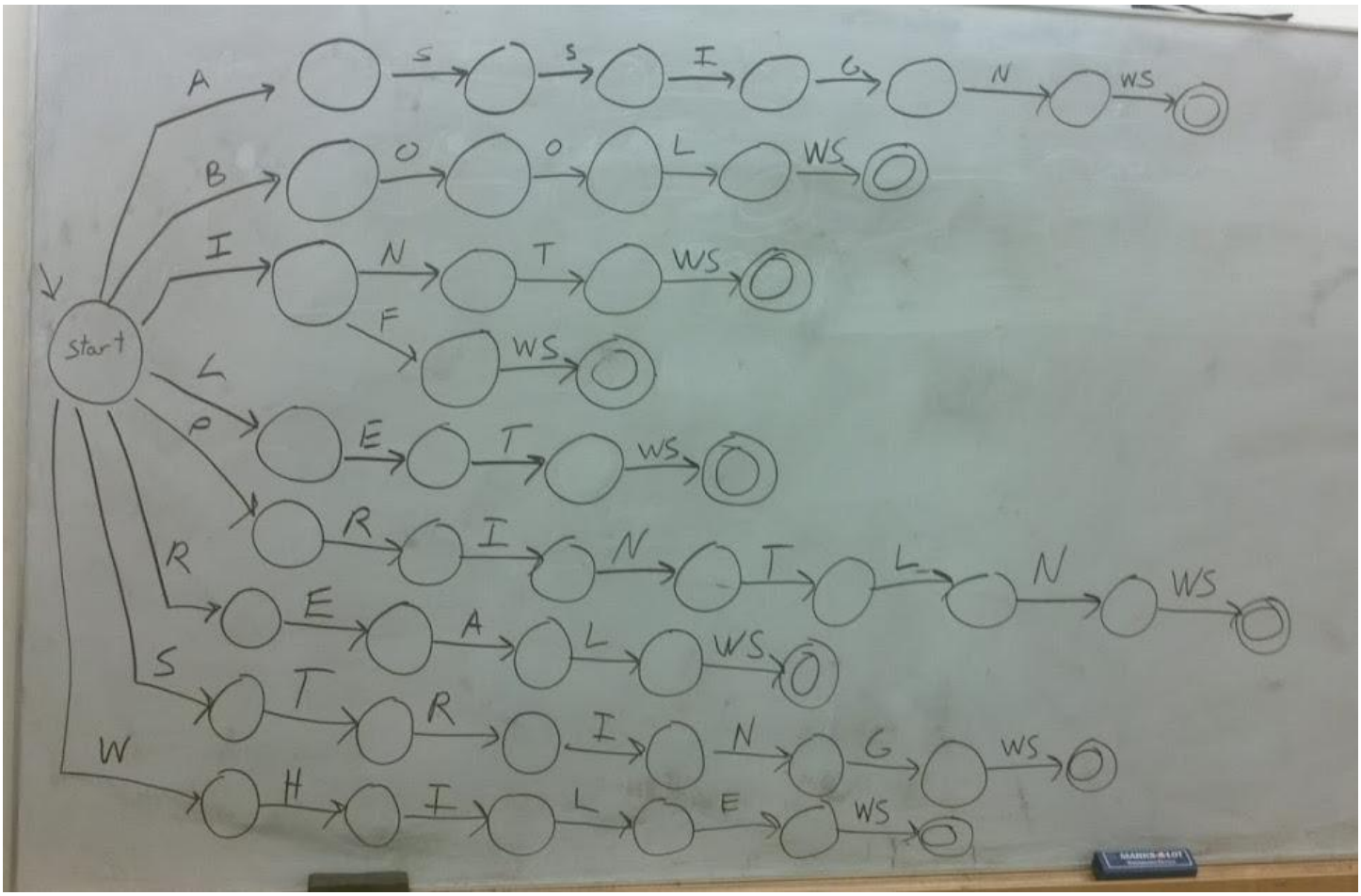
- **Milestone Questions:**

1. We're utilizing an inheritance hierarchy:

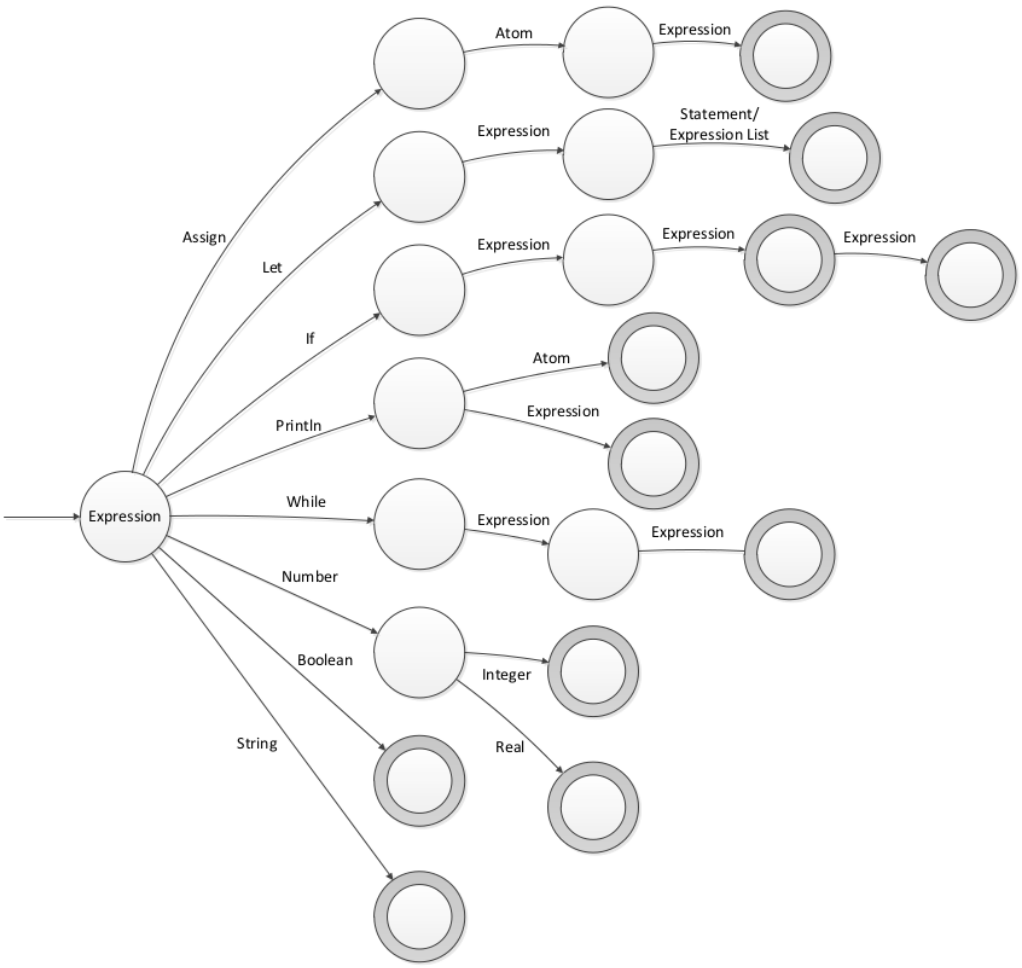
```
expression (has attribute tokenId)
      constant (abstract)
            boolean (not and or iff)
            number (abstract) (+ - * / ^ % = <)
                  int
                  real
            string (+ = <)

      statement (abstract)
            println
            if
            while
            let
            assign
```

2.



3.



- **Specification**

The purpose behind this milestone was to help us understand the concept of lexical analysis. We're using a small language (IBTL), with a small set of tokens to examine. Our lexical analyzer can break the input stream of characters stored in our input file into individual lexemes. The output is written to stdout, which can be piped to an output file (as our **`Makefile`** will do).

- **Processing**

We break the input stream up line by line, and then examine characters until we encounter whitespace. From there, our lexical analyzer checks against our regular expressions and hash table to locate known token types. For strings with quotes, we match until we find the "closing" quote to indicate the end of the string.

- **Testing Requirements**

Our input file served as our test system. **`stutest.in`** contains a sample of IBTL code based on the language specifications provided. We first examined our input file and predicted the output of our lexical analyzer, then compared our expected output with the actual output stored in **`stutest.out`**.

We tried to test cases such as strings with or without quotes, as well as real numbers.

- **Retrospective**

We learned how lexical analyzers work and how to create a rudimentary implementation of one for a simple language. This may not be something that would be fully necessary in the "real world," as preexisting and extensively tested solutions are available.