# COMP 304B – Object-Oriented Software Design
# Assignment 3 – Scanner Finite State Machine

Due date: Sunday March 23, 2003 before 23:55

## Practical information

- Team size == 2 (pair design) !
- Each team submits only *one full solution*. Use the `index.html` template provided on the assignments page. Use **exactly** this format to specify names and IDs of the team members. The other team member *must* submit a single `index.html` file containing only the coordinates of both team members. This will allow us to put in grades for both team members in WebCT. Beware: after the submission deadline there is no way of adding the other team member's `index.html` file and thus no way of entering a grade !
- Your submission must be in the form of a simple HTML file (`index.html`) with explicit references to *all submitted files* as well as inline inclusion of images. See the general assignments page for an `index.html` template.
- The submission medium is WebCT.

## Goal

In this assignment you will use UML *State Machines* to specify whether to accept or reject a series of input characters from an input character stream. The specification will be derived by you from a Regular Expression specification. The automaton will not only accept/reject input, but will also determine pertinent properties (such as value) of the recognized token. For the spreadsheet application, we need to be able to recognize `Number` and `CellRef` tokens. You will encodecode both automata in the given Python framework and run the simple provided tests.

Your assignment solution should contain:

1. A state automaton *completely* specifying recognition of

   - A spreadsheet `Number`. A `Number` is specified as follows:

     ```
     D            [0-9]
     E            [eE][+-]?({D})+
     Number       [({D}+{E}?)
                   ({D}*'.'{D}+({E})?)
                   ({D}+'.'{D}*({E})?)]
     ```

     Note how this specification is taken from the ANSI C grammar, Lex specification.
     `D` and `E` are macros which are expanded literally wherever `{D}` and `{E}` occur. Wherever no ambiguity exists, characters such as `0` stand for themselves `'0'`. `[]` denotes *or*. `*` denotes 0 or more times repeated. `+` denotes 1 or more times repeated. `?` denotes exactly 0 or 1 occurrences. Brackets `()` allow for grouping.
     The automaton should set `self.value` and `self.exp` attributes to hold the mantissa and exponent respectively.
   - A spreadsheet `CellRef`. A `CellRef` is specified as follows:

```
'$'?[a-zA-Z][a-zA-Z]?'$'?[1-9][0-9]?[0-9]?[0-9]?
```

The automaton should set `self.row`, `self.rowIsAbsolute`, `self.column` and `self.columnIsAbsolute` attributes to hold appropriate integer values.

You may use any drawing tool to produce the automaton.

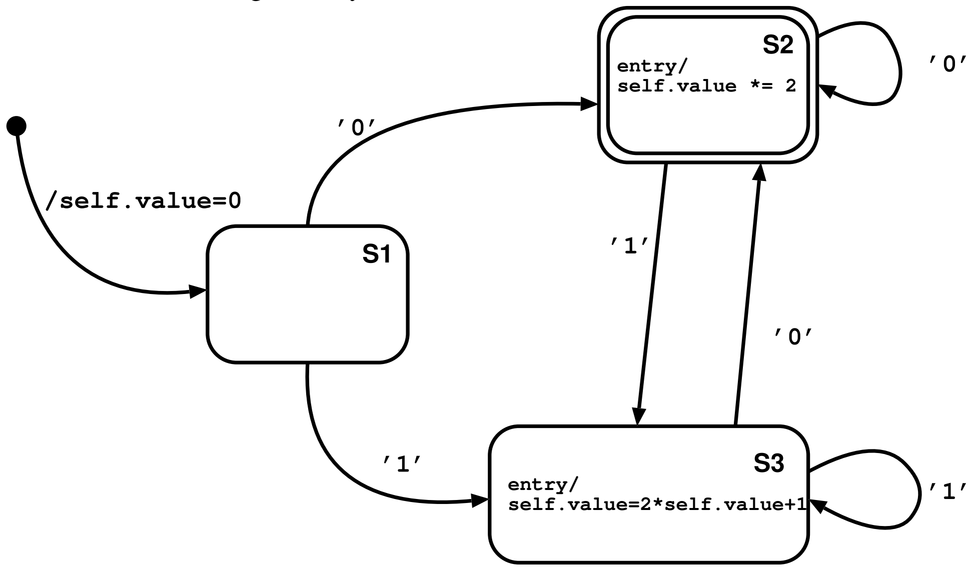
2. An encoding, in the file `scanner.py`, of both patterns in the form of the classes `NumberScanner` and `CellRefScanner`, derived from the `Scanner` class given below.

3. The results of the simple tests in `tests.py`.

Upload *all* files to WebCT and provide links to them from your `index.html` file.

## Starting Point and Example

The Regular Expression `[10]*0` is recognized by the automaton below.



The scanner is encoded in the class `EvenBinaryScanner` in `scanner_evenBinary.py`. This requires an input stream class `CharacterStream` found in `charstream.py`. The test script `test_evenBinary.py` produces the following output when the `__trace__` variable is set to `False`. It produces the following output when the `__trace__` variable is set to `True`.