COMP 304B – Object-Oriented Software Design
Assignment 3 – Solution
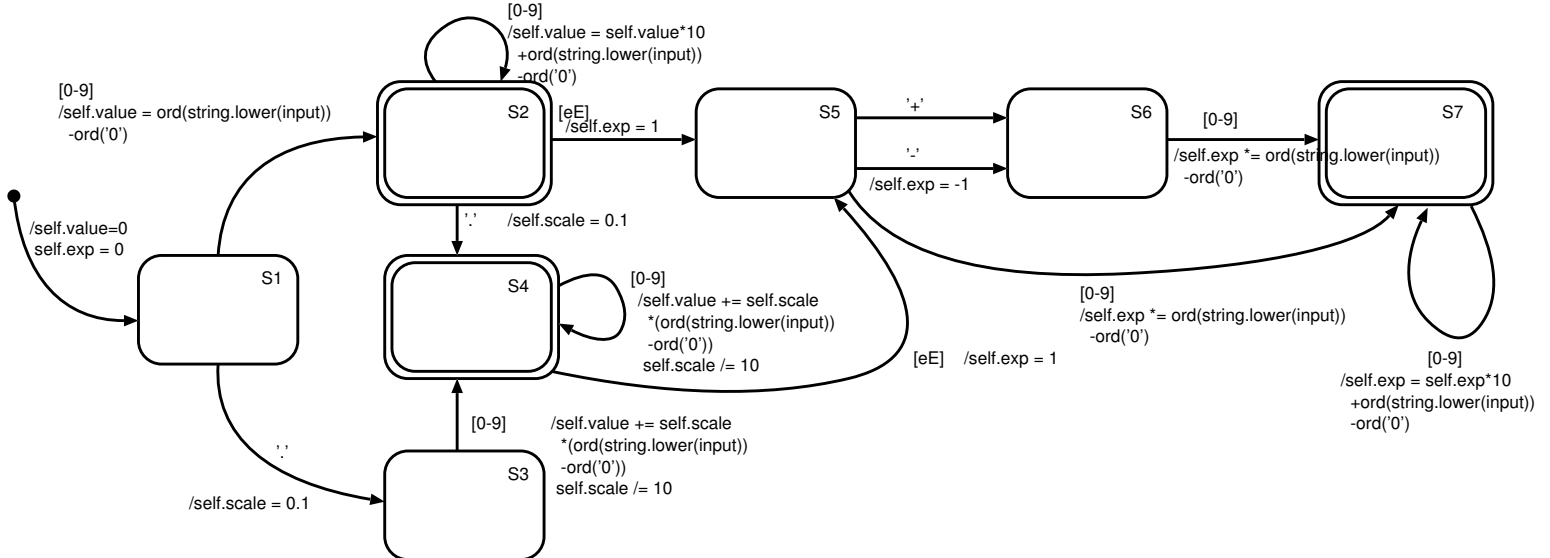
Number

A spreadsheet `Number` is specified by the following regular expression:

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

From this specification, we derive the following Finite State Machine

Note how the specification not only describes how characters in the input stream trigger automaton transitions. It also describes *actions* to be taken upon transition. In particular, these actions set/update `self.value` and `self.exp` attributes to hold the mantissa and exponent respectively of the recognized number.
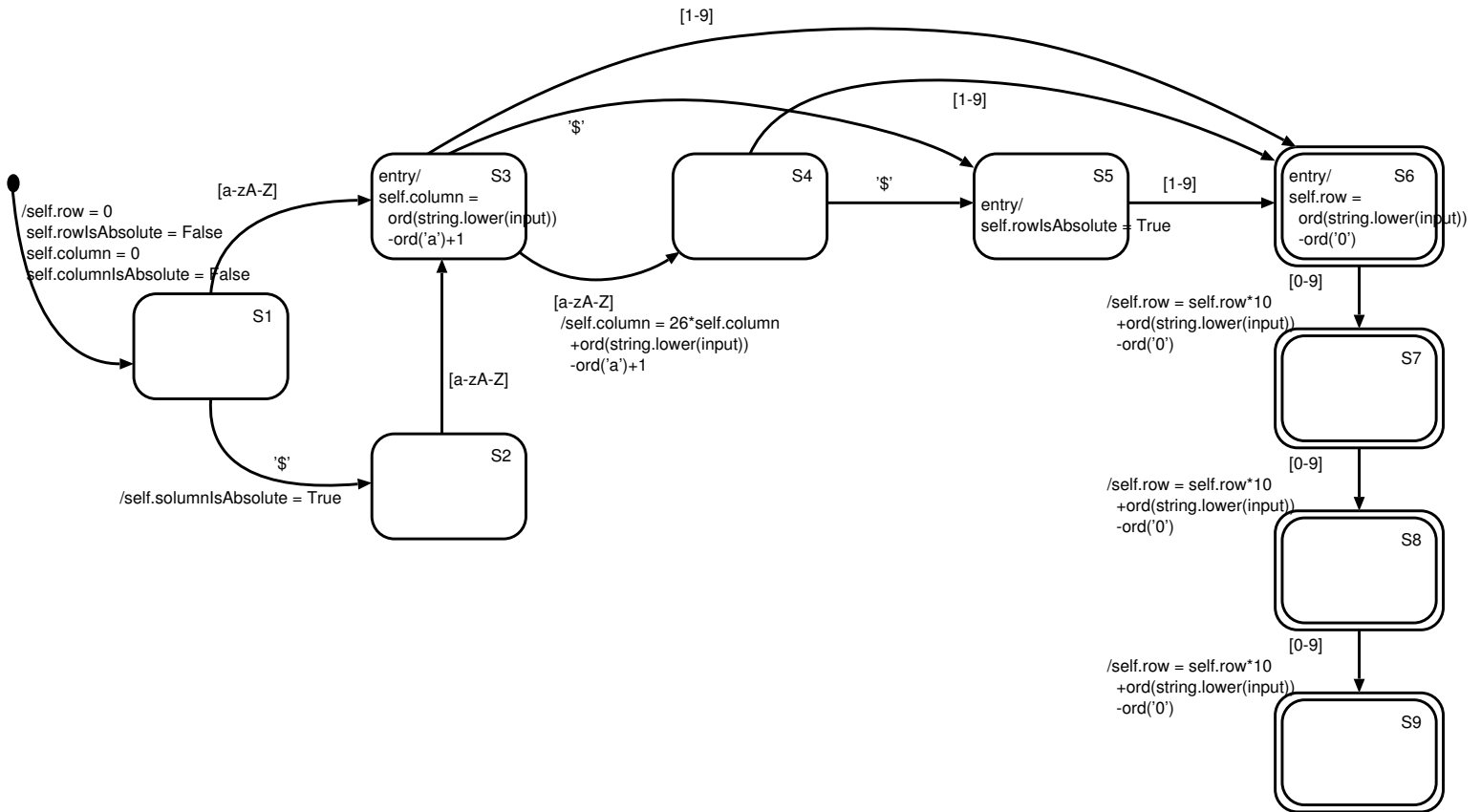
The scanner is encoded in the class `NumberScanner` in `scanner.py`. This requires an input stream class `CharacterStream` found in `charstream.py`.

CellRef

A spreadsheet `CellRef` is specified by the following regular expression:

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

From this specification, we derive the following Finite State Machine

[1-9]

[1-9]

'$'

entry/   S3
self.column =
  ord(string.lower(input))
  -ord('a')+1

S4

'$'

entry/   S5
self.rowIsAbsolute = True

[1-9]

entry/   S6
self.row =
  ord(string.lower(input))
  -ord('0')

[a-zA-Z]

/self.row = 0
 self.rowIsAbsolute = False
 self.column = 0
 self.columnIsAbsolute = False

S1

[a-zA-Z]
/self.column = 26*self.column
  +ord(string.lower(input))
  -ord('a')+1

[a-zA-Z]

[0-9]

/self.row = self.row*10
  +ord(string.lower(input))
  -ord('0')

'$'

S2

/self.solumnIsAbsolute = True

S7

[0-9]

/self.row = self.row*10
  +ord(string.lower(input))
  -ord('0')

S8

[0-9]

/self.row = self.row*10
  +ord(string.lower(input))
  -ord('0')

S9

Note how the specification not only describes how characters in the input stream trigger automaton transitions. It also describes *actions* to be taken upon transition. In particular, these actions set/update `self.row`, `self.rowIsAbsolute`, `self.column` and `self.columnIsAbsolute` attributes to hold appropriate integer values.

The scanner is encoded in the class `CellRefScanner` in `scanner.py`. This requires an input stream class `CharacterStream` found in `charstream.py`.

Test Results

The test script `tests.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`.

Note how the scanner will only commit the part of the input `stream` which was recognized. The remainder of the input stream remains ready for future scanning. This is necessary as the different scanners will be used by a *parser* which will drive the *scanner* which will try to recognize different tokens as it recognizes a grammar (in this case, the spreadsheet formula syntax).