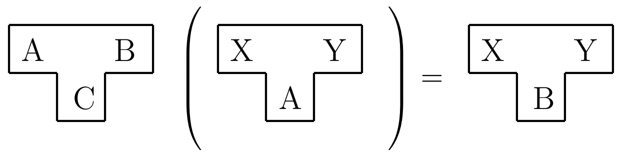
**ECE251: Programming Languages & Translators** Fall 2010

# Lecture 2 — September 15, 2010

*Patrick Lam* *version 2*

**Bootstrapping Clarification.** It seems there was some confusion.



## Scanners.

We will talk about scanners and the tools and technology we use to generate them.

**Goals.** You should be able to:

- formulate regular expressions (as on the problem sheet I'll hand out);
- convert regular expressions to NFAs; simulate NFA execution;
- give these regular expressions to a tool and create scanners;
- know when to parse and when to lex.

## Today's topic: Regular Expressions

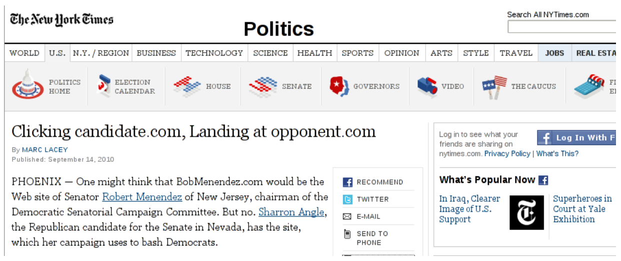
Credit: Ras Bodik, UC Berkeley.

`http://cs164fa09.pbworks.com`, `http://cs164fa10.pbworks.com`.

### Four Motivating Applications

Regular expressions are useful. Here's some evidence.

**Web scraping and rewriting.** The Web is better with print-friendly pages—no ads.



Clicking candidate.com, Landing at opponent.com

By MARC LACEY
Published: September 14, 2010

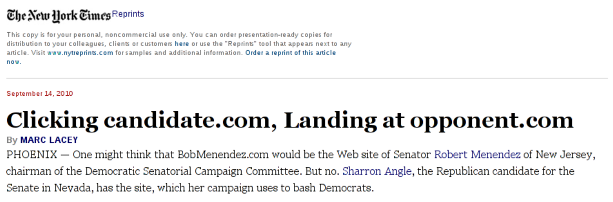
PHOENIX — One might think that BobMenendez.com would be the Web site of Senator Robert Menendez of New Jersey, chairman of the Democratic Senatorial Campaign Committee. But no. Sharron Angle, the Republican candidate for the Senate in Nevada, has the site, which her campaign uses to bash Democrats.

Log in to see what your friends are sharing on nytimes.com. Privacy Policy | What's This?

What's Popular Now

In Iraq, Clearer Image of U.S. Support

Superheroes in Court at Yale Exhibition



This copy is for your personal, noncommercial use only. You can order presentation-ready copies for distribution to your colleagues, clients or customers here or use the "Reprints" tool that appears next to any article. Visit www.nytreprints.com for samples and additional information. Order a reprint of this article now.

September 14, 2010

Clicking candidate.com, Landing at opponent.com

By MARC LACEY

PHOENIX — One might think that BobMenendez.com would be the Web site of Senator Robert Menendez of New Jersey, chairman of the Democratic Senatorial Campaign Committee. But no. Sharron Angle, the Republican candidate for the Senate in Nevada, has the site, which her campaign uses to bash Democrats.

Engineering a solution: Rewrite links to automatically go to print-friendly pages.

Implementation details: Upon loading a webpage, find links, fetch their targets, search each target for a print-friendly link, then rewrite the link.

[Berkeley students had to do this as part of their first assignment, in Greasemonkey.]

**Cucumber testing framework:** a framework for testing all sorts of things using stylized natural-language scripts, e.g. Web applications, or Excel files (`http://adomokos.blogspot.com/2010/03/testing-excel-with-cucumber.html`).

Idea: give a series of scenarios (plus additional context) and have an engine which executes scenarios.

We show one scenario here.

```
Scenario: Display column headers and captions
Given I have 2 categories
  And I have 3 child elements under the first category
When I open the Excel workbook
Then I should see "Category" in the "A1" cell
 (etc)
```

How does Cucumber actually interpret this text? Regular expressions. e.g.

```
"Then /^I should see "([^\"]*)" in the "([^\"]*)" cell$"/ do |value, cell|
@worksheet.get_cell_value(cell).strip.should == value
```

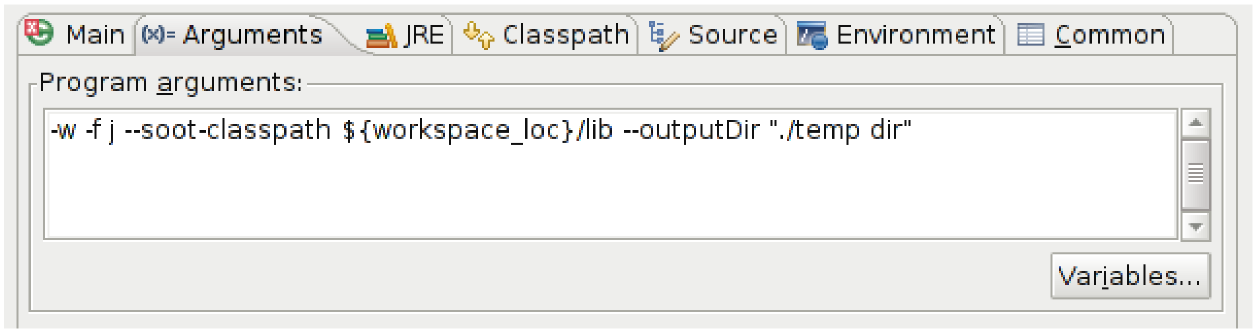
**Lexical analyzers.** (We saw this yesterday.)

Input: stream of characters.

Output: stream of tokens.

$\boxed{\texttt{let}}\ \boxed{\texttt{rec}}\ \boxed{\texttt{tcs}}\ \boxed{\texttt{s}}\ \boxed{\texttt{=}}\ \boxed{\texttt{match}}\ \boxed{\texttt{s}}\ \boxed{\texttt{with}}\ \boxed{\texttt{|}}\ \boxed{\texttt{EmptyStmt}}\ \boxed{\texttt{->}}\boxed{\texttt{EmptyStmt}}$

**Filename/commandline processing languages.** Consider the following:



```
-w -f j --soot-classpath ${workspace_loc}/lib --outputDir "./temp dir"
```

Eclipse should convert this to:

```
args = { "-w", "-f", "j", "--soot-classpath", "/home/plam/lib", "--outputDir",
         "\"./temp dir\"" }
```

This is surprisingly complicated, since it requires variable substitution, escaping, quotes, etc.; Eclipse apparently gets this wrong in some cases.

## The Plan[1]

Design a small string processing language to *find* and *extract* substrings. Implement this language efficiently. Explore applications of this language and algorithms supporting its implementation.

**The Language.** Arithmetic expressions over variables.

| Token | Lexeme |
|---|---|
| ID | sequence of one or more letters/digits starting with a letter. |
| EQUALS | "==" |
| PLUS | "+" |
| TIMES | "*" |

**An Imperative Scanner.** Note that this primarily states *how* to scan.

```
c = nextChar();
if (c == '=') { c = nextChar(); if (c == '=') { return EQUALS; } }
if (c == '+') return PLUS;
if (c == '*') return TIMES;
if (isalpha(c)) {
  c = nextChar();
  while (isalnum(c)) { c = nextChar(); }
  undoNextChar(c);
  return ID;
}
```

[1]See Section 2.1 of the text for more.

Plusses and minuses:

Here, enjoy some real hand-written scanner code:

`http://mxr.mozilla.org/mozilla/source/js/src/jsscan.c`

```
        } else {
            /* Enforce the http://www.w3.org/TR/REC-xml/#wf-Legalchar WFC. */
            if (c != 0x9 && c != 0xA && c != 0xD &&
                !(0x20 <= c && c <= 0xD7FF) &&
                !(0xE000 <= c && c <= 0xFFFD)) {
                goto badncr;
            }
        }
    } else {
        /* Try to match one of the five XML 1.0 predefined entities. */
        switch (length) {
          case 3:
            if (bp[2] == 't') {
                if (bp[1] == 'l')
                    c = '<';
                else if (bp[1] == 'g')
                    c = '>';
            }
            break;
```

Also, you can see page 53 of the textbook for another pseudocode example.

**Idea.** Instead of *how*, say *what*—in our example, the token definitions for '==', EQUALS, IDs...

Let's look at the plumbing in the scanner implementation.

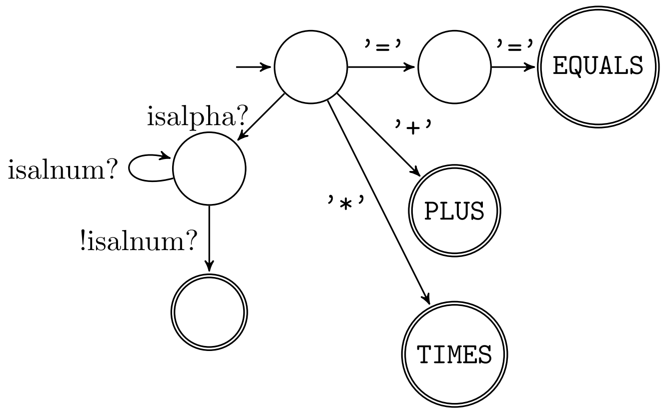
1. `c = nextChar();`: reading characters
2. `return TOK;`: tokens always returned
3. `undoNextChar(c)`: lookahead explicit (programmer-managed)
4. ifs and whiles: explicit control-flow.

We would like to hide the plumbing and specify just the structure of the language (a DSL!)

Here is the code structure:

```
READ a character
COMPARE with our targets;
  IF found, READ and COMPARE with newly-appropriate targets
REPEAT while we have inputs; RETURN tokens
```

This is actually a *finite automaton*.



**Declarative Scanner.** We want to separate the (1) declarative part (*what* are the tokens?) from (2) the imperative part (*how* to process the input?).