FLC Lab's overview

Alberto Ercolani

University of Trento

alberto.ercolani@unitn.it

November 02, 2017

Logistic Informations

- Who?
  - Alberto Ercolani, FLC's Teaching Assistant
- Where? When?
  - Tue: 09:00 - 11:00 A105
  - Wed: 16:00 - 18:00 A105
  - Thu: 11:00 - 13:00 A206
  - Fri: 09:00 - 11:00 A105
- No office hours: write me an email to arrange a meeting.
  - alberto.ercolani@unitn.it
- Typos or errors reports, suggestions and improvements are always welcome!
  - You won't received an hexadecimal dollar though.
  - https://en.wikipedia.org/wiki/Knuth_reward_check

Course's Text Book

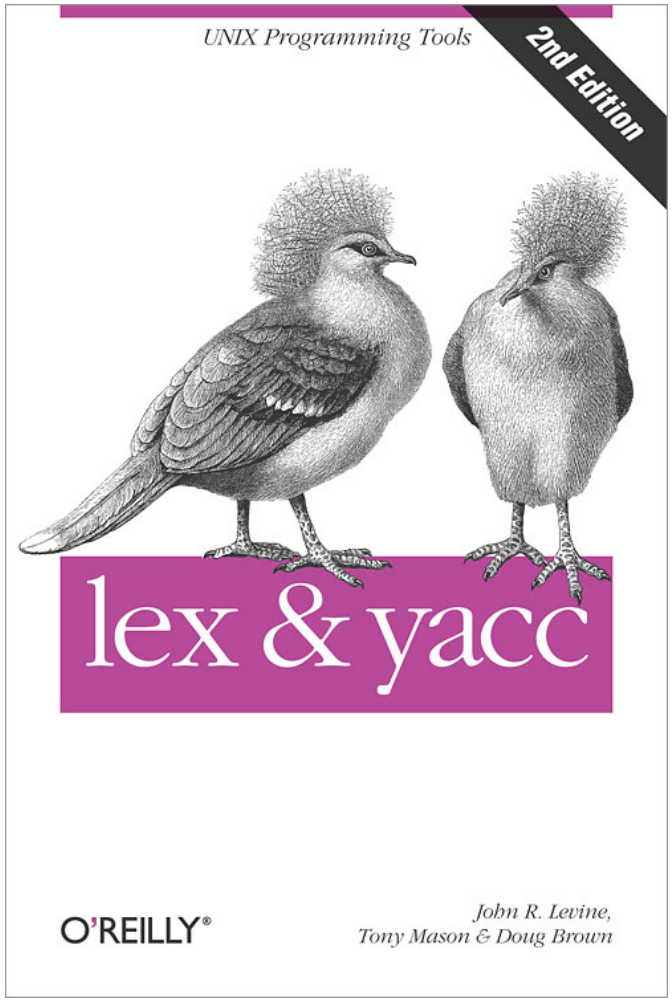

Figure: Lex and Yacc, 2ed. - Levine J.R., Manson T., Brown D. - O'Reilly

Motivational: Why should we study FLC?

- In general parsing machines are **everywhere**.
- They offer services:
  - Syntax highlighting: achieved through regular or push down automata.
  - Web Services: HTTP's grammar states messages' structure.
  - Compilation, Interpretation: push down automata.
- They ease everyday's work:
  - Data transformation: think about CPP preprocessor.
  - Data manipulation: think about SQL.
  - Data validation: think about regular expressions.
  - Document generation: think about LaTeX.

Motivational: No FLC?

- No high quality services:
  - ~~Syntax highlighting~~
  - ~~Web Services~~
  - ~~Compilation, Interpretation~~
- Tedious everyday's work:
  - Data transformation is done by hand.
  - Data manipulation, same.
  - Data validation, same.
  - Document generation, same.
- We would be stuck in 50s'.

Why a Lab?

- To familiarize with parsing generation technologies.
  - Lex & Yacc
- More sophisticated technologies are available but Lex & Yacc provide a good starting point.
- Compiler design can be challenging, using the right tools makes life easier.
- When you will write your own compiler/interpreter/preprocessor these tools will come in handy.

Within the end of the course you will...

- Design practical imperative languages
- Implement and debug:
  - Preprocessor(s)
  - Interpreter(s)
- Not compiler(s) since they are time consuming to be implemented (take months!).

- Lab is divided in two parts:
  - Lexical/Syntactic analysis
  - Semantic analysis

Lexical analysis is important

- It's at the base of any text processing tool.
- Can it be done the wrong way?
- I'll answer the question with a question:

Lexical analysis is important

```
BOOL mistery(char * s) {
        int i, len = strlen(s);
        if (len == 0) return FALSE;
        if (!(isalpha(s[0])||s[0] == '_'))
                return FALSE;
        for (i = 1; i < len; i++) {
                if (!(isalnum(s[i])||s[i] == '_'))
                        return FALSE;
        }
        return TRUE;
}
```

- Question: **What does it do?**

Lexical analysis is important

- In prose: "It checks whether or not, a string is an identifier."
- That was the wrong way. Let's check out the right one!

What is a lexer?

- A lexer λ is a DFA.
- Each final state ϕ_{i} is associated with one regular expression α_{i}.
- When a string σ belonging to $L(\lambda)$ is matched by the lexer, the pattern matching terminates in a final state ϕ_{j} and σ is therefore associated with the regular expression α_{j}.
- The couple $(\sigma, \alpha_{\mathsf{j}})$ is called a token.

What is a lexer?

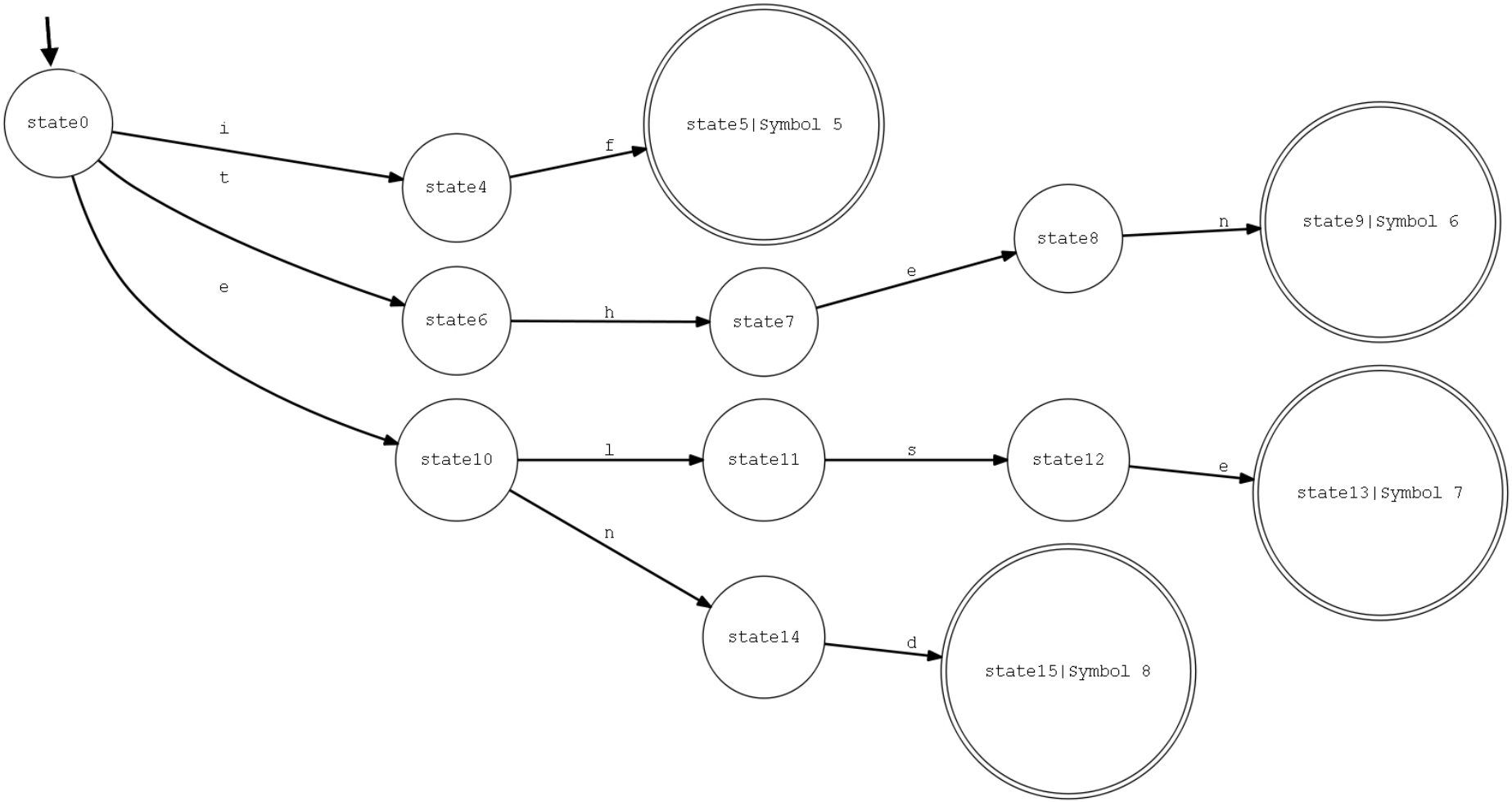

Figure: A simple lexer accepting if|then|else|end.

What is a lexer?

- The tokens produced by the lexer in the previous slide:
  - (if, 5)
  - (then, 6)
  - (else, 7)
  - (end, 8)
- If the lexer had more sophisticated regular expressions tokens would be more.

What is Lex?

- Lex is a lexer generator.
- Given the description of the language you want to match it generates C code to do so.
- The language we refer to is the set of strings, keywords, identifiers that can appear.

- You can build your own lexer generator using the algorithms seen during the course!
  - Thompson construction, Subset construction and Partition refinement.

What can Lex & Yacc achieve together?

- Despite their limited power they are expressive and efficient enough to give birth to sophisticated languages:
  - Bash
  - GCC (Previous versions)
  - Go programming language
  - MySQL
  - PHP
  - PostgreSQL
  - R
  - Ruby
- These are known examples of rather complex and successful projects.

Retrieving a copy of Lex & Yacc

The packages you are going to download provide Lex and Yacc utilities.

- Yacc version 3.0.4
- Lex version 2.6.0

Windows Users

Download Yacc/Lex from here:
https://sourceforge.net/projects/winflexbison/
In file selection choose "win_flex_bison3-latest.zip".

Unix Users

Run *sudo apt-get install bison flex gcc* from terminal.

Mac Users

Download and install XCode.

Bibliography

A. V. Aho and M. J. Corasick

Efficient String Matching: An Aid to Bibliographic Search

Comm. ACM 18, 6 (June 1975) 333 - 340.

A. V. Aho, R. Sethi and J. D. Ullman

Compilers: Principles, Techniques, and Tools

Addison Wesley, 1986.

M. E. Lesk

LEX - A Lexical Analyzer Generator

Computing Science Technical Report 39, Bell Telephone Laboratories, Murray Hill, NJ, 1975