ECE251 Midterm practice questions, Fall 2010

Patrick Lam

October 20, 2010

Bootstrapping

In particular, say you have a compiler from C to Pascal which runs on x86, and you want to write a self-hosting Java compiler, and you have an x86 processor. What's the best way to proceed?

Regular Expressions

Creating regular expressions and DFAs.

To practice, you can create both regular expressions and DFAs for these. I recommend that you start with DFAs.

- a DFA which recognizes strings of a's, b's, x's and y's, where every x is immediately followed by a y.
- a regular expression which accepts strings over a, x, b where at least one a follows every b.

Regexps.

Recall that regexps are not the same as regular expressions. In particular, they can match groups (subexpressions of the regular expression).

- Write a regexp for perl interpolation. It should accept print statements with this syntax:

  ```
  print "$foo" . '$bar' . 'baz';
  ```

  So we have print statements with, as arguments, a sequence of strings, joined by the . operator. You should match variables contained in double-quotes only.

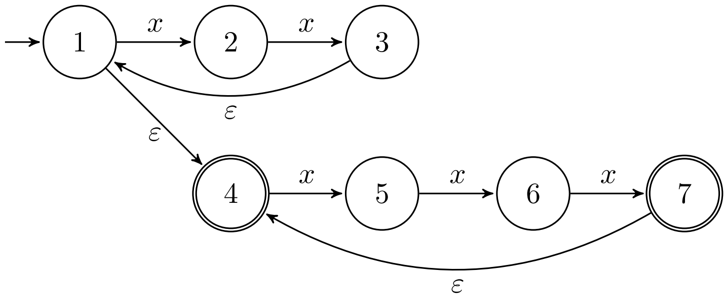
Regular expressions to NFAs.

Create any valid NFA for the following regexps:

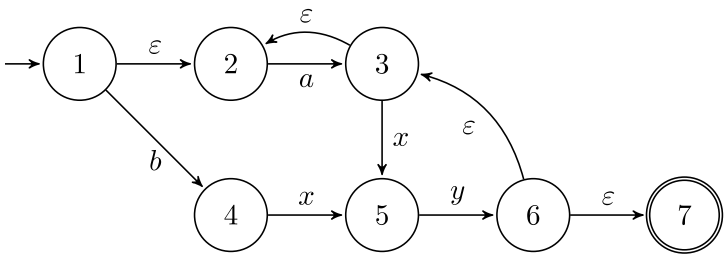
- $(ab^?c(dd)^*) \mid x$
- $(xy \mid ab)^*zw^?$

NFA execution.

Show an accepting trace or explain why the following NFA does not accept the word $xxxx$.



What about the word $bxyxy$ on this automaton?



(You might find harder questions of this flavour on the midterm.)

Lexing.

Say you have these four token definitions:

- PRINT: ``print''
- INT_LITERAL: $[+ \mid -]^?[0-9]^+$
- FLOAT_LITERAL: $[+ \mid -]^?[0-9]^+$('.' $[0-9]^*$)?
- PLUS: ``+''

What is the proper lexing for these strings:

- PRINT 5.4
- PRINT +2874.
- PRINT 27.928
- PRINT 28+99

Why?

Declarative versus operational regular expression parsing.

Recall that *declarative* regular expression parsing matches the longest possible substring match for a group, while *operational* parsing specifies that $*$ should match as many repetitions as possible while allowing the remainder to match and that $|$ should match the leftmost alternative while allowing the remainder to match.

I can think of two possible questions:

1. Give me an example of which groups the declarative and operational semantics would match on a given string (or, given matches, tell me which is which);

2. Construct an example where the declarative and operational semantics give different answers.

Interpreters

You might find short answer question on interpreters, like:

- Describe the general structure of an interpreter.

- What is the difference between interpreters and compilers?

General parsing questions

Parse trees and ASTs

Construct and compare a reasonable parse tree and an AST for the following code:

```
if (x > 4)
  x = x * y + (z * 2);
```

Briefly say why these are different.

Language design and CFGs

Propose a language and a context-free grammar for a language to describe graphs. Graphs consist of a list of vertices V and a set of edges E; each edge is a pair of vertices.

Derivations

Consider the following grammar:

$$\begin{array}{rcl}
S & := & ABC \\
A & := & \texttt{'[x'}\ D\ \texttt{'q'}\ B\ \texttt{']'}\ A \\
B & := & \texttt{'y'}\ B \\
C & := & \texttt{']'} \\
D & := & \texttt{'qq'}
\end{array}$$

1) Provide two words that can be derived from this grammar. 2) Can the following words be derived from this grammar: `[xqqqy]qq`; `[xq][qqqq]`

Regexps versus CFGs.

Short-answer questions about specific languages and if you would parse them using regular expressions or context-free grammars. Examples:

- You've saved the HTML for a bunch of Facebook profiles and want to automatically extract peoples' relationship statuses from them. How would you do this? What would change your answer?
- You have to parse some log files. Each log entry is a bunch of comma-delimited fields. What do you do?
- You need to validate user input on forms according to some simple syntax-based rules. What is the best way to do that?

Parsing: Grammar manipulations

A couple questions on parsing and grammars.

EBNF. Convert EBNF to BNF; here is a EBNF grammar:

$$\begin{array}{rcl} S & := & (AB)?C \\ A & := & t|w \\ B & := & x*y \\ C & := & z \end{array}$$

What is a corresponding BNF grammar?

FIRST sets. Compute FIRST sets and decide when a production is nullable for the grammars you've seen in this set of questions.

Recursive-descent parsing. Generate pseudocode for a recursive-descent parser on an expression grammar. Or: can you produce a recursive-descent parser for a given grammar, say the one immediately above?

Right-recursion and left-recursion. Here is a right-recursive statement list.

$$\mathit{stmt\_list} := \mathit{stmt}\text{' ; '}\,\mathit{stmt\_list}$$

What is the left-recursive equivalent?

Dangling else. Briefly describe the dangling-else problem. Draw the two parse trees for dangling else and name 2 ways for dealing with the problem.

Stratified grammars. Consider a programming language with the $\bowtie$ and $\bullet$ binary operators:

$$E := E \bowtie E \mid E \bullet E$$

Illustrate the ambiguity with two parse trees for the same expression. No w, assume that $\bowtie$ has higher precedence than $\bullet$. Give a stratified grammar which gets rid of the ambiguity.

Shift-reduce conflicts. Give me an example of a grammar which contains a shift-reduce conflict. Write down a sentence in the language of the grammar where the parser should shift, and a sentence where it should reduce. Fix the grammar by delaying decisions.