SE 550: Lecture 4

Overview

Parser generators

JavaCC

Examples, including simple HTTP

Parser generators

What is a lexical analyzer (lexer)? What is a parser? What is a static analyzer?

What kind of programs use parsers?

What is a parser generator?

Why use a parser generator rather than writing a parser by hand?

Parser generators

Metamata/WebGain/Sun's JavaCC is an LL(1) parser generator.

It takes as input a *foo*.jj file containing:

- A collection of terminals (or *tokens*)
- A collection of production rules for nonterminals
- Java code to be performed when the production rules fire (called *actions*)

It produces as output a *foo*.java file.

JavaCC is documented at javacc.dev.java.net.

Example

Consider this grammar:

```
<parse> ::= <balanced> "."
<balanced> ::= ( "(" <balanced> ")" )*
```

What are the terminals? The non-terminals?

What language does this grammar recognize?

Example

The same example in JavaCC...

```
options { STATIC=false; }
PARSER_BEGIN (TestBalancedRecognizer)
  ... the class definition ...
PARSER_END (TestBalancedRecognizer)
  ... the terminals of the grammar ...
  ... the nonterminals of the grammar ...
```

The class definition:

```
public class TestBalancedRecognizer {
    public static void main (String[] args) throws Exception {
        // JavaCC provides a default constructor which takes an InputStream argument
        TestBalancedRecognizer parser = new TestBalancedRecognizer (System.in);
        parser.parse ();
    }
}
```

Example

The terminals:

```
TOKEN : {
  <OPEN: "(">
| <CLOSE: ")">
| <PERIOD: ".">
}
```

The non-terminals:

```
void parse () : {} {
  balanced () <PERIOD>
}
void balanced () : {} {
  (
    <OPEN> balanced () <CLOSE>
  )*
}
```

The full source code is on-line.

Downloading and running JavaCC

Download JavaCC.zip from the course home page, and put it in your classpath:

```
set CLASSPATH=.;JavaCC.zip
```

Download the src.zip archive.

```
java COM.sun.labs.javacc.Main
  -OUTPUT_DIRECTORY=ajeffrey/teaching/test/
  ajeffrey/teaching/test/TestBalancedParser.jj
```

- Make sure JavaCC.zip is in your CLASSPATH.
- Make sure you're running java from inside the src directory.
- Remember to set the -OUTPUT_DIRECTORY variable.

Once it's running, try typing "()." or "()()." or "((()))." at it.

Try typing "(." or ")." or "fred." at it.

Example

Just recognizing strings isn't much use, we need to be able to do something with the strings!

```
String parse () : { String result; } {
  result = balanced ()
  <PERIOD>
  { return result + ";"; }
}

String balanced () : { StringBuffer result = new StringBuffer (); String tmp; } {
  (
    <OPEN> { result = result.append ("{"); }
    tmp = balanced () { result = result.append (tmp); }
    <CLOSE> { result = result.append ("}"); }
  )*
  { return result.toString (); }
}
```

What does this code do? The full source code is on-line.

Parsing HTTP

Here is an incomplete part of a grammar for a subset of HTTP:

```
<HTTPRequest> ::= <getRequest>
<getRequest> ::= <GET> <space> <url> <space> <protocol> <crlf>
<url> ::= (
        <HTTP> <colon> <slash> <slash> <hostName>
        ( <colon> <port> )?
    )? <fileName>
<hostName> ::= <string>
<port> ::= <number>
<fileName> ::= ( <slash> )? ( <string> ( <slash> )? )*
<protocol> ::= <HTTP> <slash> <string>
```

How can we complete this grammar?

How can we convert this grammar into JavaCC format?

Parsing HTTP

HTTPParser.jj contains the HTTP parser.

It is an *event-driven parser*: it supports the interfaces:

```
public interface HTTPRequestHandler {
    public void handleGetRequest (URL url);
}
public interface HTTPRequestMultiplexer extends HTTPRequestHandler {
    public void addHTTPRequestHandler (HTTPRequestHandler handler);
    public void removeHTTPRequestHandler (HTTPRequestHandler handler);
}
```

What happens when an incoming request is parsed?

Parsing HTTP

The JavaCCConnection class uses this parser to implement a simple HTTP server.

The run method is now:

```
public void run () {
    try {
        try {
            ...as before...
            parser = new HTTPParser (socket.getInputStream ());
            parser.addHTTPRequestHandler (this);
            parser.HTTPRequest ();
        } catch (ParseException ex) {
            out.println (error400Response);
        }
    } ... more error-handling...
}
public void handleGetRequest (final URL url) {
    ...deal with a GET request...
}
```

Parsing HTTP

Compare this code with the original Connection code.

What are the pros and cons of using a generated parser?

Summary

Parser generators can automatically generate code given a grammar and a collection of actions.

Parser generators generate efficient, maintainable code.

Parser generators have a steep learning curve, but once you've learned it, you never want to go back!

Next week: Object serialization

Week after that: Midterm