Software Tools Exercises

Part II – Lecture/Tutorial 10

Today's Outline

- The Exam
- Grammar Exercise
- Type Derivation Exercise

The Exam

The Exam

- Two hours time in total; **50%** about second half:
  1. **Version Control (10%)**
     create diffs between versions, merge two new versions into a base version, identify conflicts
  2. **Compilers (18%)**:
     define regular expressions and a context-free grammar with actions according to a specification
  3. **Type Systems (10%)**:
     derive a given Java statement using an environment and type rules (rules are **not** given)
  4. **Text Questions (12%)**:
     short answer questions (no more than 3 sentences each) about processes, version control, compilers, type systems, static checking

Version Control Exercise

Merging

Given base version 1 and successive versions 2 and 3: What would be the result of the merge? Identify the conflicts.

Version 1

```
class X {
  int m() {
    return 0;
  }
}
```

Version 2

```
class X {
  String m() {
    return "hello";
  }
}
```

Version 3

```
class Y {
  int m() {
    int x = m();
    return x;
  }
}
```

Result

```
class Y {
  String m() {
    int x = m();
    return ???;
  }
}
```

Semantic conflict

Textual conflict

Grammar Exercise

Regular Expressions

Define regular expressions for the following tokens:

1. The **package** keyword
2. Boolean literals that are either **true** or **false**
3. Hexadecimal numbers with digits 0 to 9 and A to F
4. Identifies that start with an alphabetic lower-case character followed by an arbitrary sequence of lower-case alphanumeric characters

```
PACKAGE: 'package';
BOOLEAN: 'true' | 'false';
HEXNUM: ('0'..'9'|'A'..'F')+;
IDENTIFIER: ('a'..'z') ('a'..'z'|'0'..'9')*;
```

Context-Free Grammars

Define grammar rules for the following syntax elements:

1. Expressions that are variable accesses or use the binary operators + and *
2. The while statement
3. An interface definition with an optional **extends** clause

You can use the following tokens and subrules:

PLUS, **STAR**, **IDENTIFIER**, **WHILE**, **LPAR** ("**(**"), **RPAR** ("**)**"), **INTERFACE**, **EXTENDS**, **COMMA**, **LCURLY** ("**{**"), **RCURLY** ("**}**"), **statement**, **interfaceBody**

```
expr: expr (PLUS|STAR) expr | IDENTIFIER ;
while: WHILE LPAR expr RPAR statement ;
interface: INTERFACE IDENTIFIER
   (EXTENDS IDENTIFIER (COMMA IDENTIFIER)* )?
   LCURLY interfaceBody RCURLY ;
```


Actions

Given the following grammar rule:

```
expr: expr PLUS expr
    | expr STAR expr
    | INT
    | LPAREN expr RPAREN ;
```

Rewrite the rule using ANTLR syntax so that it returns the **int** value that is the arithmetic result of the parsed expression.

```
expr returns [int value]:
      a=expr PLUS b=expr
      { $value = $a.value + $b.value; }
    | a=expr STAR b=expr
      { $value = $a.value * $b.value; }
    | i=INT { $value = Integer.parse($i.text); }
    | LPAREN a=expr RPAREN { $value = $a.value; };
```


Type Derivation

Type Derivation

Given the environment

$\Gamma = \{$**int a; String b; int m(String s, int t);**$\}$

derive the following code: `if (a==1) a = m(b, 1);`

$$[\texttt{int}\ lit]\ \frac{\Gamma \vdash \diamond \quad x \in \texttt{int}}{\Gamma \vdash x : \texttt{int}} \qquad [\texttt{int}\ ==]\ \frac{\Gamma \vdash expr_1 : \texttt{int} \quad \Gamma \vdash expr_2 : \texttt{int}}{\Gamma \vdash expr_1 == expr_2 : \texttt{boolean}}$$

$$[var]\ \frac{\Gamma \vdash \diamond \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id : type} \qquad [if]\ \frac{\Gamma \vdash expr : \texttt{boolean} \quad \Gamma \vdash stat}{\Gamma \vdash \texttt{if}(expr)\ stat}$$

$$[call]\ \frac{\Gamma \vdash expr_1 : type_1 \quad \ldots \quad \Gamma \vdash expr_n : type_n \quad \{type_{ret}\ id(type_1\ id_1, \ldots, type_n\ id_n);\} \subseteq \Gamma}{\Gamma \vdash id(expr_1, \ldots, expr_n) : type_{ret}}$$

$$[assign]\ \frac{\Gamma \vdash expr : type \quad \{type\ id;\} \subseteq \Gamma}{\Gamma \vdash id = expr;}$$

Type Derivation Solution

Given the environment

Γ = {**`int a; String b; int m(String s, int t);`**}

derive the following code: `if (a==1) a = m(b, 1);`

$$[var]\ \frac{\Gamma \vdash \diamond \quad \{\texttt{int a;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{a:int}} \qquad [\texttt{int}\ lit]\ \frac{\Gamma \vdash \diamond \quad \texttt{1} \in \texttt{int}}{\Gamma \vdash \texttt{1:int}}$$

$$[var]\ \frac{\Gamma \vdash \diamond \quad \{\texttt{String b;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{b:String}} \qquad [\texttt{int ==}]\ \frac{\Gamma \vdash \texttt{a:int} \quad \Gamma \vdash \texttt{1:int}}{\Gamma \vdash \texttt{a==1:boolean}}$$

$$[call]\ \frac{\Gamma \vdash \texttt{b:String} \quad \Gamma \vdash \texttt{1:int} \quad \{\texttt{int m(String s, int t);}\} \subseteq \Gamma}{\Gamma \vdash \texttt{m(b, 1):int}}$$

$$[assign]\ \frac{\Gamma \vdash \texttt{m(b, 1):int} \quad \{\texttt{int a;}\} \subseteq \Gamma}{\Gamma \vdash \texttt{a=m(b, 1);}}$$

$$[if]\ \frac{\Gamma \vdash \texttt{a==1:boolean} \quad \Gamma \vdash \texttt{a=m(b, 1);}}{\Gamma \vdash \texttt{if(a==1) a=m(b, 1);}}$$