Phase3 Parser

Phase3.1

Grammar and the .cup file

Java Cup

- Cup is a LALR parser generator for Java (Look Ahead Left to Right)
- It translates the grammar into Java code.
- LR parsers were invented by Knuth in 1965 and are potentially the fastest parsers present.

Basic Steps

cool.cup

- Run parser.JavaCup

Sym & CoolParser

- Compile Sym.java & CoolParser.java

treeNodes.*

- Flesh out the .java files for phase2.2

Run parser.Parser

- Get the AST dump. (yippie!)

Cool.cup

- Read sections 3 to 7 in the cool manual to understand the grammar.
- The grammar here needs to be added to the .cup file

e.g. CLASSES


```
class <type> [ inherits <type> ] {
    <feature_list>
};
```

class ::=

CLASS TYPEID:n LBRACE feature_list:f RBRACE SEMI

{: RESULT = new Class_(curr_lineno(), n,

AbstractTable.idtable.addString("Object"), f, curr_filename()); :}

| CLASS TYPEID:n INHERITS TYPEID:p LBRACE feature_list:f RBRACE SEMI

{: RESULT = new Class_(curr_lineno(), n, p, f, curr_filename()); :}

| CLASS OBJECTID error SEMI

| CLASS error SEMI ;

Phase 2.2
Printing out the ast Tree

treeNodes

- Is a collection of grammatically significant terminals and non terminals.
- The Parser generates the AST.
- The root node is the node of type treeNode.Program.
- Result. dumpWithTypes(out, 0) then calls others recursively.

For Block.java


```
public void dumpWithTypes(Writer out, int n) throws
    IOException {
        dumpLine(out, n);
        out.write(Utilities.pad(n) + "_block\n");
        for(AbstractExpression e : body) {
            e.dumpWithTypes(out, n + 2);
        }

        dumpType(out, n);
    }
```

Notes

- Precedence rules make grammar unambiguous.
- Rules are used to parser string of tokens.
- Action part are used to construct AST(IR), but it won't show up

  in your screen. ASR(IR) are data structures.
- Parser.java

Notes

- Only .cup file need to be modified.
- Complete .cup file are due on Oct 10.