

# Fortress Boot Camp: Parser and AST

**Sukyoung Ryu**

Programming Language Research Group

May 23, 2008

Copyright © 2008 Sun Microsystems, Inc. (“Sun”). All rights are reserved by Sun except as expressly stated as follows. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted, provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers, or to redistribute to lists, requires prior specific written permission of Sun.

# Fortress Parser Trials

- JavaCC: LL(k)
- Javacup: LALR(1)
- Hand-written recursive descent
- OCaml/Elkhound: GLR(1)
- Rats!: Packrat

# Fortress Parser Trials

- JavaCC: LL(k)
- Javacup: LALR(1)
- Hand-written recursive descent
- OCaml/Elkhound: GLR(1)
- Rats!: Packrat:

`ProjectFortress/src/com/sun/fortress/parser/`

# Advantages of Rats!

- Module modification reduces copy & paste:
  - > {Field, AbsField, LocalDecl}.rats modify Variable.rats.
  - > {TraitObject, MethodParam}.rats modify Param.rats.
  - > {NoNewlineExpr, NoSpaceExpr}.rats modify Expr.rats.
  - > NoNewlineType.rats modifies Type.rats.
- Rats! productions are very close to the concrete syntax grammar in the Fortress specification.

# Advantages of Rats! (Example)

- Concrete syntax

```

VarDecl ::= VarMods? VarWTypes InitVal
           | VarImmutableMods? BindIdOrBindIdTuple = Expr
           | VarMods? BindIdOrBindIdTuple : Type ... InitVal
           | VarMods? BindIdOrBindIdTuple : TupleType InitVal

```

- `Variable.rats`

```

VarDecl VarDecl =
  a1:VarMods? a2:VarWTypes w a3:InitVal { ... }
  / a1:VarImmutableMods? a2:BindIdOrBindIdTuple w equals w a3:NoNewlineExpr { ... }
  / a1:VarMods? a2:BindIdOrBindIdTuple w colon w a3:Type w ellipses w a4:InitVal { ..
  / a1:VarMods? a2:BindIdOrBindIdTuple w colon w a3:TupleType w a4:InitVal { ... }
  / var w BindIdOrBindIdTuple w equals w NoNewlineExpr // Error production
  { ... "Mutable variables should be declared with their types." ... };

```

- `Field.rats`

```

modify Variable;
...
VarDecl FldDecl = VarDecl ;
List<Modifier> VarMods := FldMods ;
List<Modifier> VarImmutableMods := FldImmutableMods ;

```

# Disadvantages of Rats!

- Manual transformation of left-recursive productions
  - > No guarantee for the transformation correctness.
  - > Big gaps between two grammars.
- Ordered productions
  - > Longer productions should come before shorter ones (when the longer ones have shorter ones as their prefix)
- Bad syntax error messages

# Disadvantages of Rats! (Example)

- Concrete syntax

```

IntExpr ::= IntVal
           | IntExpr + IntExpr
           | IntExpr - IntExpr
           | ...
  
```

- MayNewlineHeader.rats

```

IntExpr IntExpr = seed:IntExprFront list:IntExprTail* { ... };
private IntExpr IntExprFront = IntVal ... ;
private constant transient Action<IntExpr> IntExprTail =
    SumIntExpr
  / MinusIntExpr
  / ... ;
private constant inline Action<IntExpr> SumIntExpr = w plus w a1:IntExpr { ... };
private constant inline Action<IntExpr> MinusIntExpr = w minus w a1:IntExpr { ... };
  
```

- Longer productions should come before shorter ones

```

Fndef ::= AbsFndef = Expr
Fndef ::= Fndef
           | AbsFndef
  
```

# Fortress Parser in Rats!

- Completeness of the parser
  - > The parser is complete with respect to the *whole* language in the Fortress 1.0 $\beta$  specification.
  - > **Please do not remove parser productions.**
- Improving syntax error messages
  - > Bad syntax error messages:  
`./tst.fss:29:25: Syntax Error`
  - > **Please send me the example Fortress program.**

# Fortress Abstract Syntax Tree (AST)

- Java files representing Fortress AST nodes are located in:  
`ProjectFortress/src/com/sun/fortress/nodes/`
- Automatically generated by ASTGen
- From `ProjectFortress/astgen/Fortress.ast`
- Fortress AST is complete with respect to the *whole* language in the Fortress 1.0 $\beta$  specification.
- **Please do not remove AST nodes.**

# Fortress.ast (Example)

```
abstract Name();
  /**
   * unstructured sequence of ids naming an API or component
   * APIName ::= Id(.Id)*
   * e.g.) com.sun.fortress.nodes_util
   * Names in the list must be unqualified.
   */
  APIName(List<Id> ids);
  abstract IdOrOpOrAnonymousName(Option<APIName> api = Option.<APIName>none());
    abstract IdOrOpName();
      Id(String text);
      abstract OpName();
        Op(String text, Option<Fixity> fixity = Option.<Fixity>none());
        Enclosing(Op open, Op close);
  abstract AnonymousName();
    AnonymousFnName();
    ConstructorFnName(GenericWithParams def);
```

# Op.java (Example)

```
package com.sun.fortress.nodes;
...
public class Op extends OpName {
    private final String _text;
    private final Option<Fixity> _fixity;
...
    public Op(Span in_span, Option<APIName> in_api, String in_text,
              Option<Fixity> in_fixity) {
...
    public Op(Span in_span, Option<APIName> in_api, String in_text) {
...
    final public String getText() { return _text; }
    final public Option<Fixity> getFixity() { return _fixity; }
...
}
```

# Utilities for Parser and AST

- Parser utilities
  - > `ProjectFortress/src/com/sun/fortress/parser_util/`
  - > `FortressUtil.java`
- AST utilities:
  - > `ProjectFortress/src/com/sun/fortress/nodes_util`
  - > `{ OprUtil, NodeUtil, NodeFactory, ExprFactory }.java`