SILVER: AN EXTENSIBLE ATTRIBUTE GRAMMAR SYSTEM

Eric Van Wyk, Derek Bodin, Jimin Gao, Lijesh Krishnan

by David Dufour

CS 846: Model-Based Software Engineering

Main Author

Eric Van Wyk

- Head of the Minnesota Extensible Language Tools (MELT) group.
- Primary research interests are in programming languages, especially extensible languages.

http://melt.cs.umn.edu/index.html

Outline

- Introduction
- Silver
- Extensions to Silver
- Discussion

Silver

- an attribute grammar specification language.

- Silver supports many extensions to attribute grammars:
  - Forwarding
  - Higher-order attributes
  - Collections
  - Pattern matching
  - Parametric Polymorphism
  - Collections

- Silver comes bundled with the Copper parser and context-aware scanner generator.

Copper

- an integrated context-aware scanner and LR parser generator.

- The unique feature in Copper is that the generated parser provides contextual information to the scanner in the form of the current LR parse state when the scanner is called to return the next token.

INTRODUCTION

Domain Specific Languages

Advantages

- Domain specific notation
- Concise programs
- Optimizations

Disadvantages

- Design, implementation and maintenance costs
- Choosing a scope

Extensible Language

- new language features are implemented as modular language extensions
- host language is implemented as an AG specification
- language extensions are implemented as AG fragments

SILVER ATTRIBUTE GRAMMAR SPECIFICATION LANGUAGE

```
grammar edu:umn:cs:melt:simplec;
start nonterminal Prog ;
nonterminal Dcl, Dcls, Type,
 Stmt, Stmts, Expr ;
terminal Id /[a-zA-Z] [a-zA-Z0-9]+/;

syn attr c :: String occurs on
 Prog, Dcl, Dcls, Type,
 Stmt, Stmts, Expr;

nonterminal TRep ;
syn attr typerep :: TRep
 occurs on Expr ;

autocopy inh attr env::[ Binding ];

syn attr errors :: [ String ]
 collect with ++ ;
attr errors occurs on Prog, Dcl,
 Dcls, Type, Stmt, Stmts, Expr ;

concrete prod program
p::Prog ::= d::Dcls
{ p.c = "#include<stdio.h>" ++ d.c;
  p.errors := d.errors;
  d.env = [ :: Binding ];}

terminal AndOp '&&' precedence = 10,
               association = none ;
```

```
concrete prod logical_and
e::Expr ::= l::Expr '&&' r::Expr
{ e.c = ...;  e.errors := ... ;
  e.typerep = booleanType(); }

concrete prod funcCall
e::Expr ::= f::Id '(' arg::Expr ')'
{ e.c =...;  e.errors := [ :: Error ];}

abstract prod funcType
 ft::TRep ::= in::TRep out::TRep {...}
abstract prod booleanType
 bt::TRep ::=    {...}
abstract prod arrayType
 at::TRep ::= component::TRep {...}
abstract prod errorType
 et::TRep ::=    {...}

aspect prod funcCall
e::Expr ::= f::Id '(' arg::Expr ')'
{ e.typerep = case ftype of
     funcType(in, out) => out
   | _                 => errorType();
  e.errors <- case ftype of
     funcType(in, out) => [ :: Error ]
   | _  => [ "Error: " ++ f.pp ++
             " must be a function."];
  local attr ftype :: TRep;
  ftype = ... lookup f in env ... ; }
```

SILVER AND IT'S LANGUAGE EXTENSIONS

The With-Clause

- A simple extension that requires only a local transformation to translate into core Silver.

- Specificies a higher-order function to use with the attribute.

Pattern Matching

- Matching the structure of the input to a select function.
- Popular in functional languages.
- Good example of adding functionality to the core Silver program.

Data-flow analysis in Silver

- Not a general purpose feature.
- Domain of analysis of imperative programs.
- Extensions using other logics and model checkers can also be easily written.

DISCUSSION

Discussion

- Worth hiring an expert in Silver to extend Silver to create your DSL?
- Silver is just one example of many AG systems out there, more of a test bed for features.
- Newest version of Silver released last Monday, still in development.

Thank You!