# Building Composable Domain-specific Language Extensions for Java

Eric Van Wyk

Department of Computer Science
and Engineering
University of Minnesota

www.melt.cs.umn.edu
www.cs.umn.edu/~evw
evw@cs.umn.edu

# Extensible Languages: motivation and goals

- Domain specific languages (DSLs) provide
  - high-level notations for abstractions
  - analyses (e.g. error checking at domain level),
  - optimizations

  specific to a problem domain.
  - But, problems often cover more that one domain.
- Libraries are "composable" --- programmers import the ones they want to use, but these have none of the advantages that DSLs provide.
- Extensible languages in which
  - language extensions add domain-specific language features
  - with composable extensions, programmers can import the set of features needed to address all aspects of a particular task.

# Example: SQL embedded in Java

- In JDBC, SQL commands are sent as strings to the database server via a connection.
- One writes code like the following:

```
Int value = 100 ;
ResultSet rs ;
Statement stmt = conn.createStatement();
rs = stmt.execute("SELECT cust_name FROM " +
 "customers WHERE quantity > " + value );
```

- SQL error checking and optimization is done by the server at run-time, not by the compiler at compile-time.
- A better solution: extend Java with SQL language constructs.

# Example: SQL embedded in Java

- Using an extensible language . . .

```
Int value = 100 ;
ResultSet rc ;
rc = using conn query
        ( SELECT cust_name FROM
          customers WHERE quantity > value ) ;
```

- Here `on … query` is a new expression which takes a data base connection and an SQL query expression.
- An extended language processor can statically type check the SQL queries.
- Requires semantic analysis and translation to Java.
- This is not LINQ or SQLJ.

# Example: Computational Geometry

- Geometric algorithms make extensive use of primitive expressions that return a qualitative result over geometric entities
  - *e.g.* is point p inside circle (c,r)
  - *e.g.* is point p to the left or right of vertical line l
  - implemented by taking the sign of an expression "sign (z - x * y)"
- Writing efficient robust CG programs is difficult.
  - Round off errors due to limited precision numbers.
    - Import fast unbounded-precision integers from LN
  - Degeneracies in the input data – point p is on line l
    - Perform transformations to handle degeneracies
- Can statically calculate size needed for intermediate values.
- Again, require semantic analysis and translation.

# Challenges to building extensible languages

- Composable specifications of feature semantics
  - Attribute grammars + forwarding + . . .
  - Get both explicit and implicit (via translation) specification of semantics.
  - Tool support: Silver – extensible AG system.
- Composable specifications of feature syntax
  - Context-aware scanning with LR-parsing
  - Deterministic, yet handles large class of languages
  - Monolithic and modular determinism analyses
  - Tool support: Copper – parser and scanner generator.

# AbleJ

- An extensible implementation of Java
- Built from declarative specifications used by Silver and Copper.
- Several composable language extensions
  - SQL, Computational Geometry, Condition Tables, Pizza constructs, . . .
- Supports automatic composition of language extensions to create domain-adapted versions of Java.
- Specifies all of Java 1.4 syntax and several semantic analyses such as type checking.

# More information

- … on Tuesday and Thursday

- … Thanks for your attention.