The Pivot

a source-to-source framework for more elegant and efficient code

Peter Pirkelbauer,
Texas A&M University

08/23/06
The Problem

\[ Z = a \times X + Y \]

Elegant?
Efficient?
DSL - Approach

Domain Specific Languages

- Tool chain support
- Interfacing code
- Maintenance
- “Edge effect” problems
More Approaches

- Pragmas & Compiler Options
- Preprocessed languages
- Dialects
- Libraries
SELL Approach

Semantically Enhanced Library Language

- Domain Specific Libraries
- Semantic Restriction
- Optimization through Transformation
XPR - Principles

- Compact representation
  - ~ size of the C++ source
- Portable
- Human read- and writable
- Simple and fast parser
  - LL(1) grammar
  - No Symbol Tables
XPR - Example

**XPR**

```
vector : <T: class> class {
  p : *T
  sz : const static int = 0
}
```

**C++**

```
template <class T>
class vector {
  T* p;
  const static int sz = 0;
};
```
IPR Principles

- High Level
  - templates, partial specializations, concepts

- Complete and Regular
  - Able to represent erroneous and incomplete code
  - Full C++ but does not mimic irregularities

- Extensible
  - Concepts

- Application effort proportional to task
  - IPR is more than a data structure

- Compiler independent
IPR – Design Choice

- Type safe
- IPR manages memory
- Optimal runtime / space
  - Minimal number of nodes
  - Minimal number of checked indirections
- Expression based
  - Types / Statements / Declarations
- Programmer Interface
  - Mutating / Non-Mutating
  - Traversal- / Query- Framework
Every interface class `xyz` should have a corresponding implementation class `Xyz_impl`.
Linearization:

Parameterize implementations by interfaces

Diagram:

- Node
- Expr
- Stmt
- Decl
- Var
- Expr_impl<T>
- Stmt_impl<T>
- Decl_impl<T>
- T=Var
- Var_impl
void foo(float b = 2.4)
Applications

- Frontend
- Refactoring
- Concept-Checking
- Controlflow-Analysis
- Program-Verification
- SEL-Languages
- IDL-Generation
- Dataflow-Analysis
- High-Level Optimizations
IPR – Two Levels

• **Source Level**
  - after macro preprocessing

• **High Level Representation**
  - after template instantiations
Levels of Representation

C++ Source

IPR – source level
Rose
IPR – high level
EDG – high level
EDG – low level
LLVM

regularization
template instantiation
this as explicit parameter
lvalue/rvalue distinction
C++ to C
typed pseudo-assembly
Current Stage

• handles “Hello World”
  – >20000 lines of preprocessed code
  – classes, unions, enumerations
  – overloaded functions
  – heavy use of templates
SELL example

\[ Z = a \times X + Y \]

template <Parallelizable T>
void f(const T& v)
{
    double d = v[2]; // OK
    double* d = &v[2]; // NOT OK
};
[GJS+06] Gregor, Douglas; Järv, Jaako; Siek, Jeremy; Lumsdaine, Andrew; Dos Reis, Gabriel; Stroustrup, Bjarne: Concepts: First-Class Language Support for Generic Programming in C++. to appear OOPSLA'06.


[SQ03] Schordan, Markus; Quinlan, Daniel: A Source-to-Source Architecture for User-Defined Optimizations. JMLC '03.

[SDR05] Stroustrup, Bjarne; Dos Reis, Gabriel: Supporting SELL for High Performance Computing. LCPC '05.

[Str05] Stroustrup, Bjarne: A rational for semantically enhanced libraries. LCSD '05.