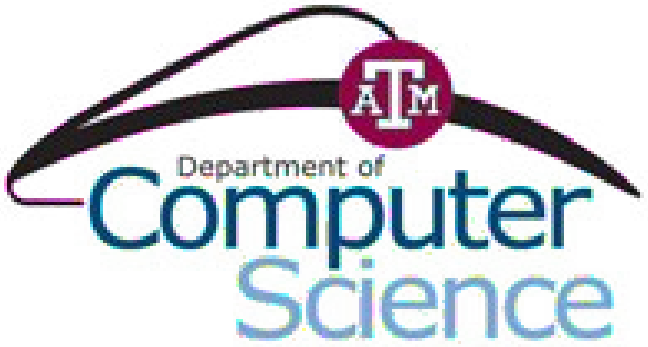
The Pivot

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



Peter Pirkelbauer,
Texas A&M University

08/23/06

The Problem

$$Z = a * X + Y$$

Elegant?

Efficient?

DSL - Approach

Domain Specific Languages

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

More Approaches

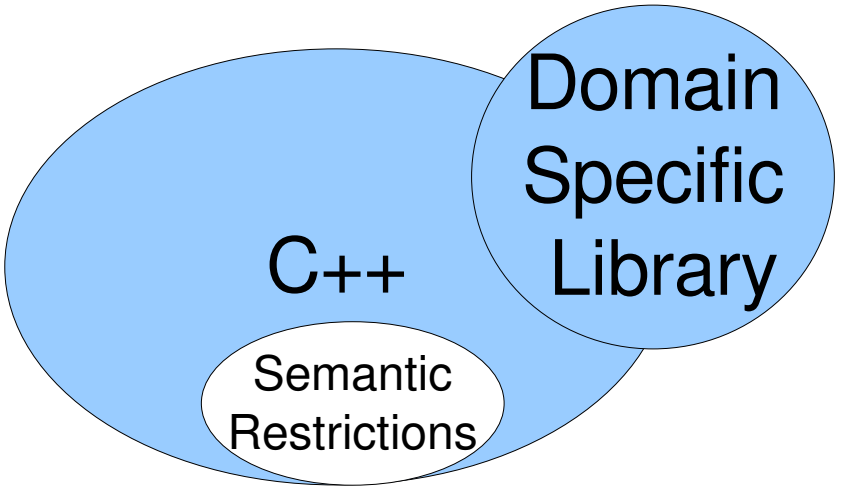
- Pragmas & Compiler Options
- Preprocessed languages
- Dialects
- Libraries

SELL Approach

Parasol
Smarter computing.

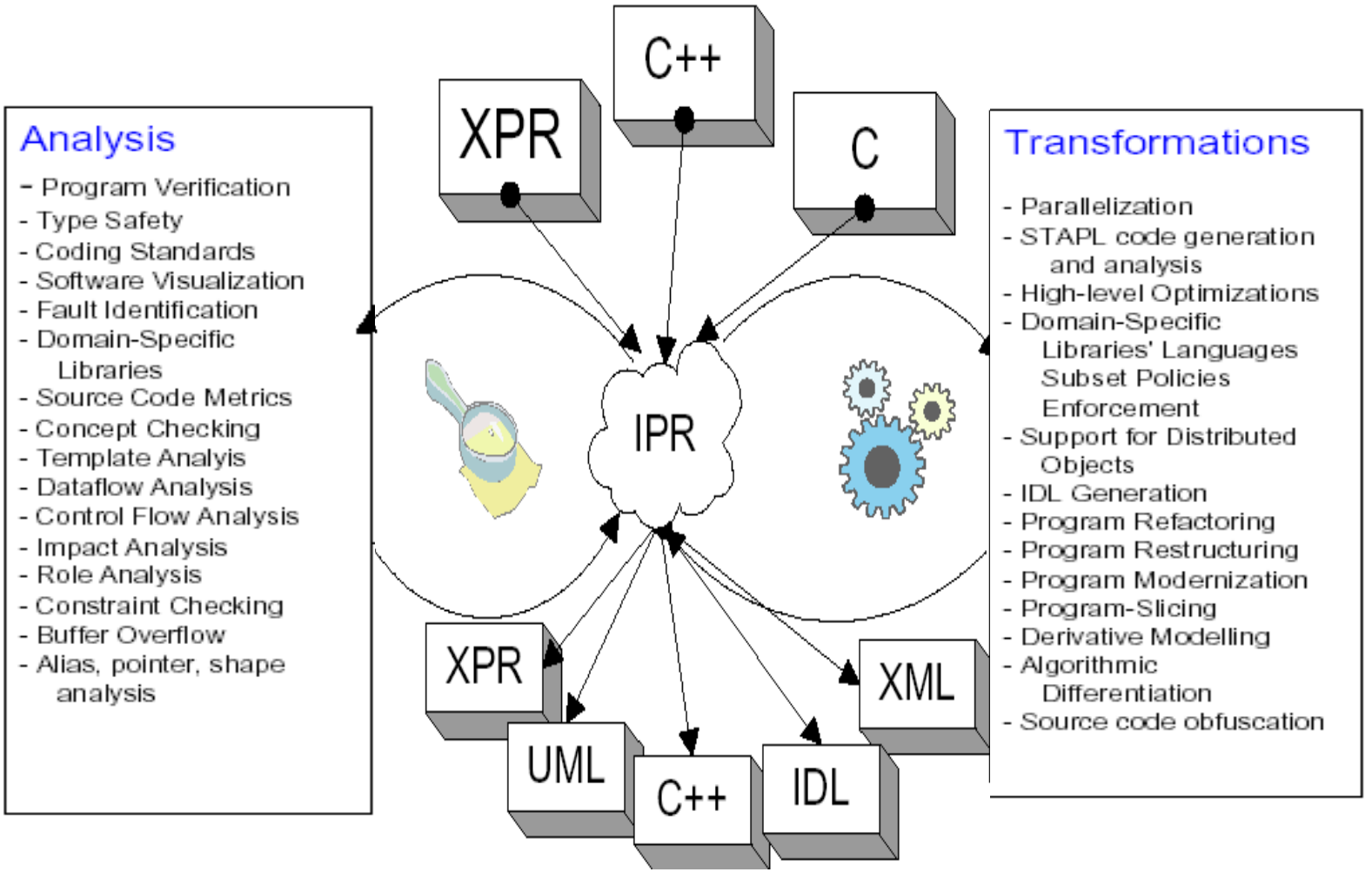
Semantically Enhanced Library Language

- Domain Specific Libraries
- Semantic Restriction
- Optimization through Transformation



The Pivot

Parasol
Smarter computing.



XPR - Principles

Parasol
Smarter computing.

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

XPR - Example

Parasol
Smarter computing.

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

Parasol
Smarter computing.

- 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

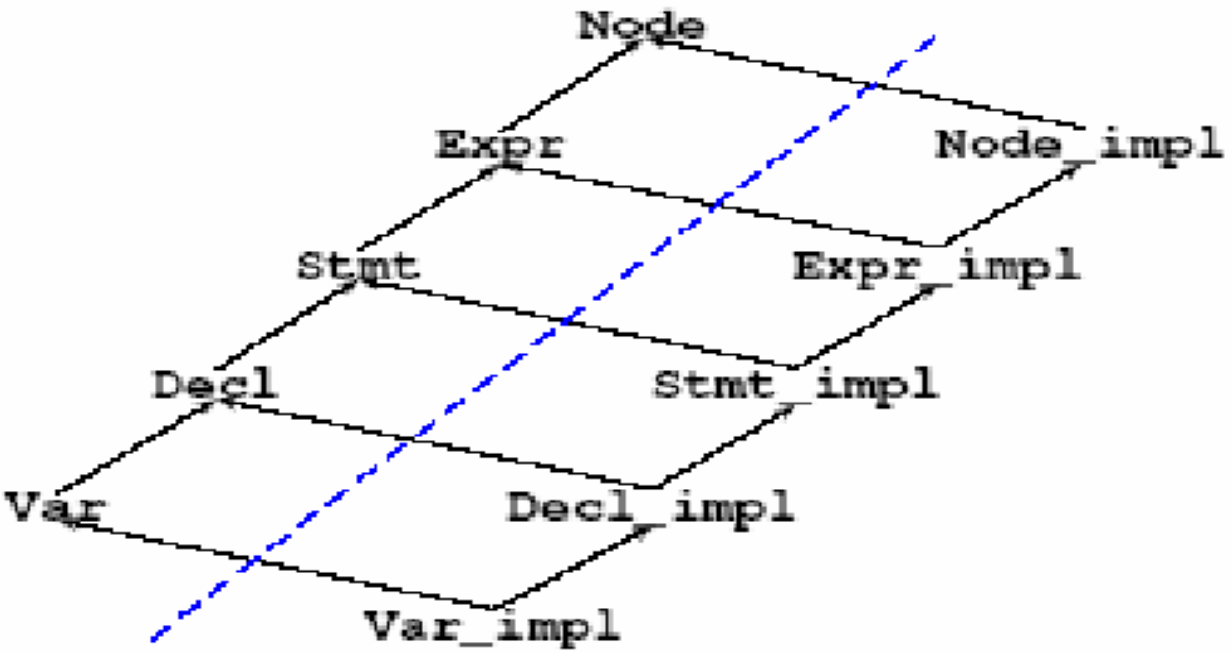
Parasol
Smarter computing.

- 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

IPR - Idea

Parasol
Smarter computing.

Every interface class **Xyz** should have a corresponding implementation class **Xyz_impl**.

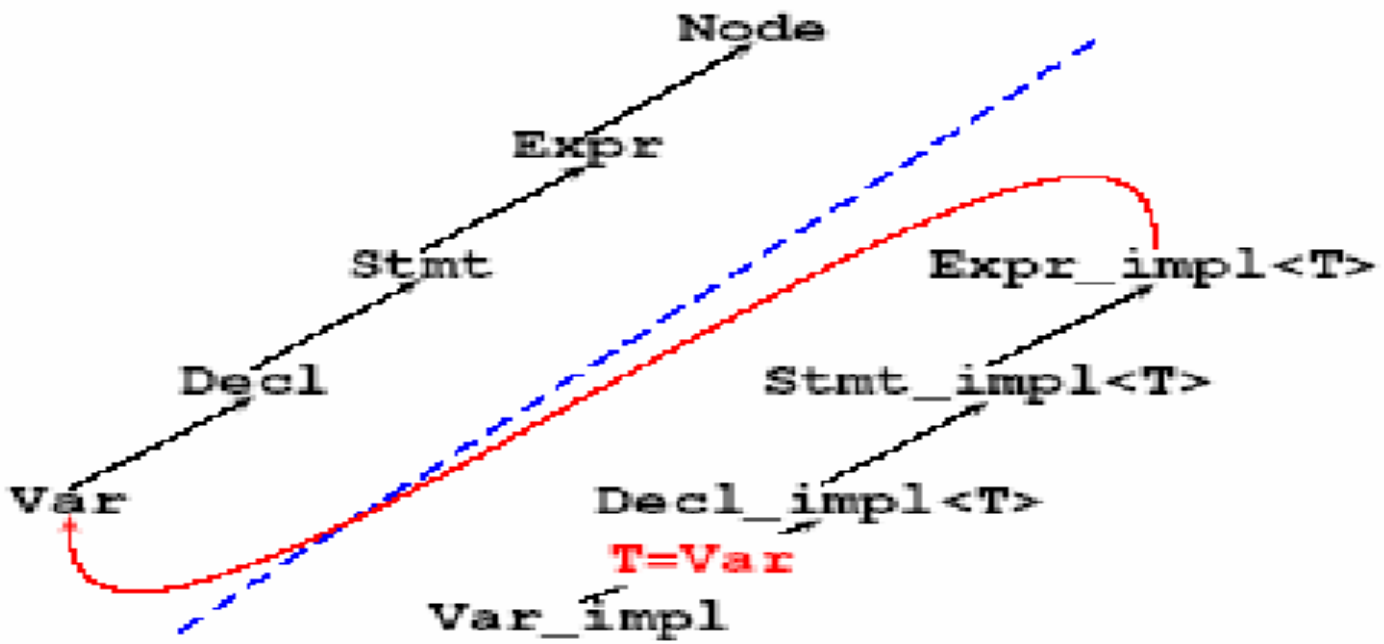


IPR - Implementation

Parasol
Smarter computing.

Linearization:

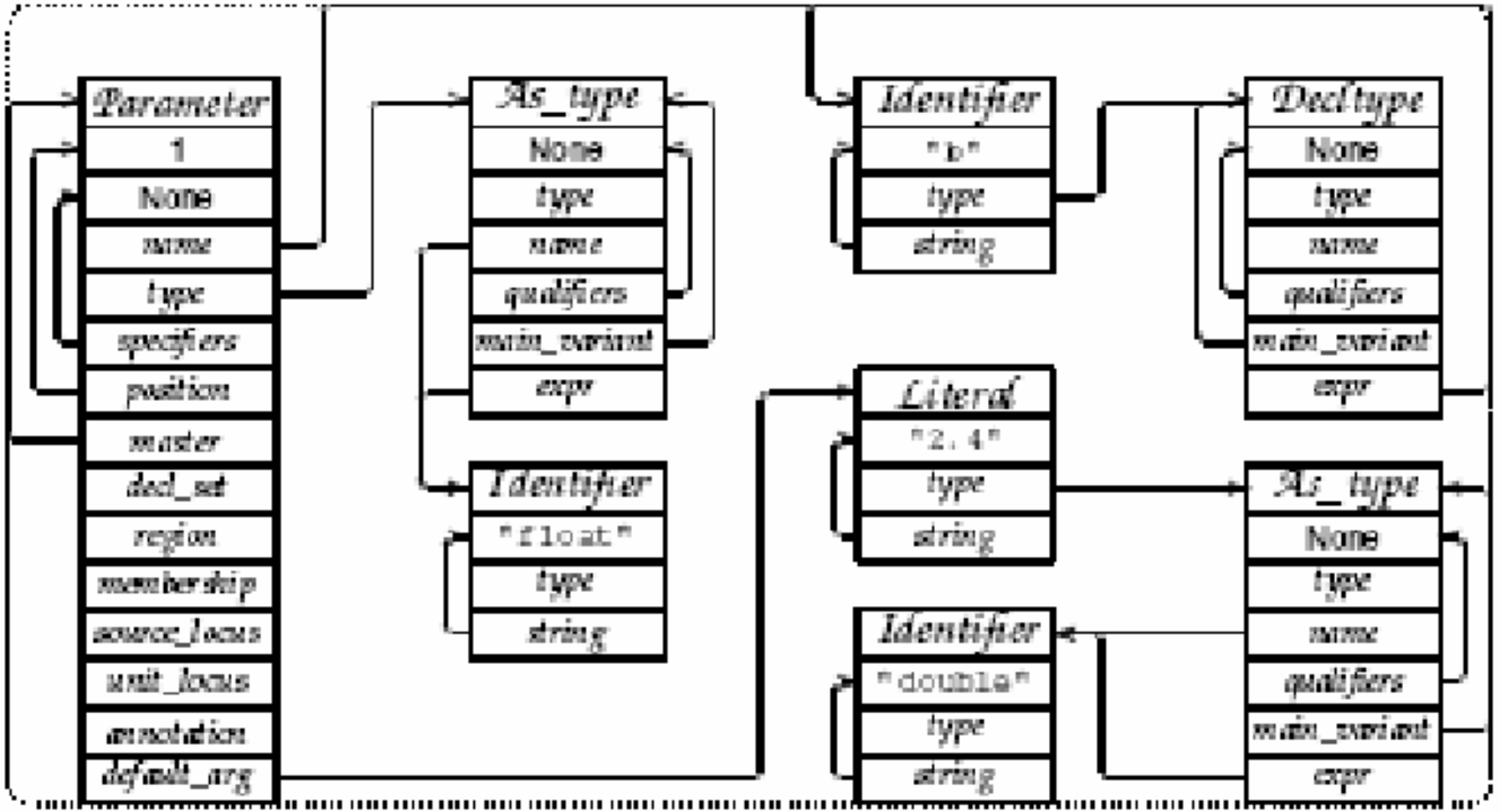
Parameterize implementations by interfaces



IPR – Example 1

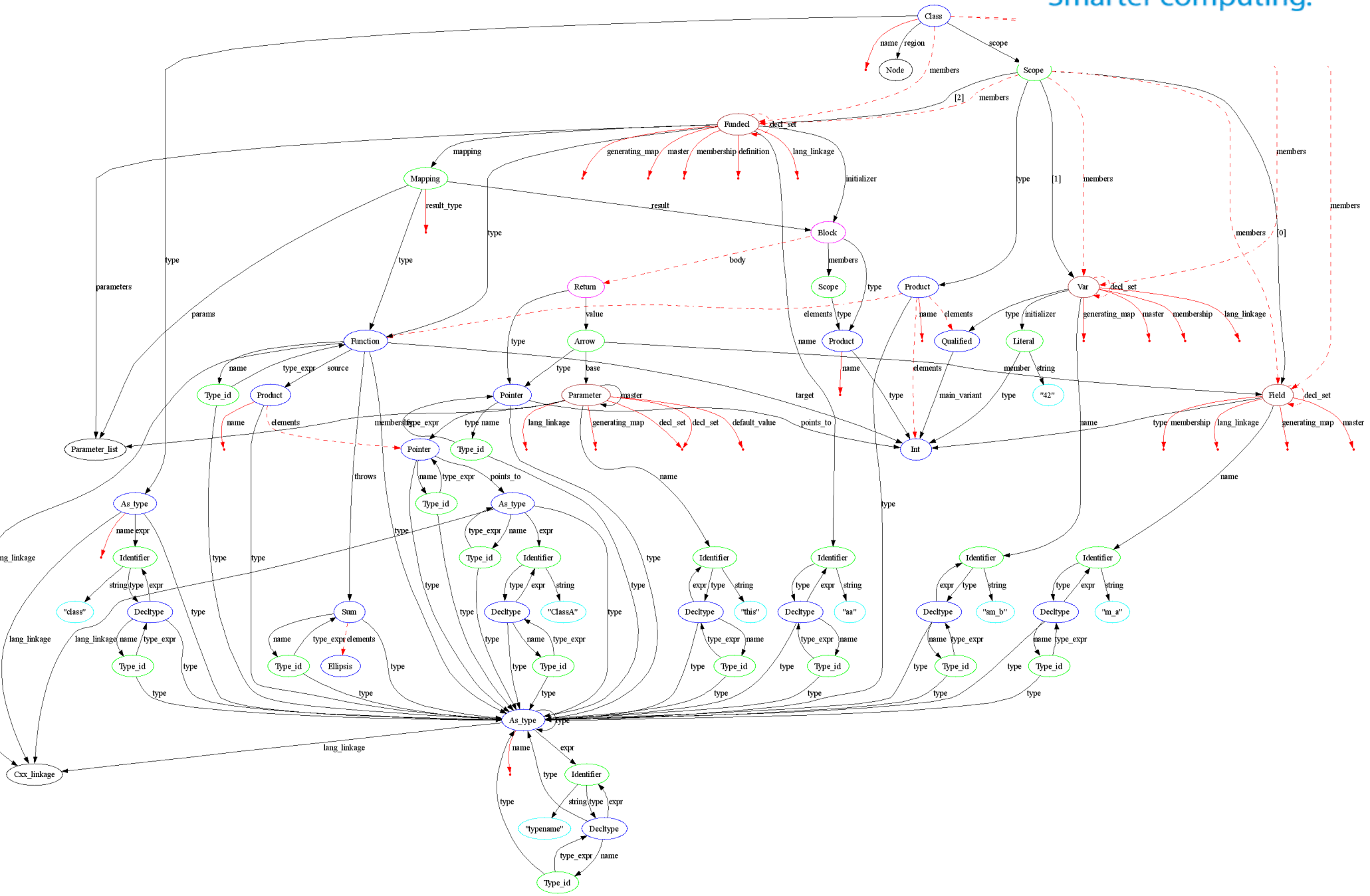
Parasol
Smarter computing.

void foo(float b = 2.4)



IPR – Example 2

Parasol
Smarter computing.



Applications

Frontend

Program-Verification

SEL-Languages

Refactoring

IDL-Generation

Concept-Checking

Dataflow-Analysis

Controlflow-Analysis

High-Level Optimizations

IPR – Two Levels

Parasol
Smarter computing.

- 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

Machine-code

Current Stage

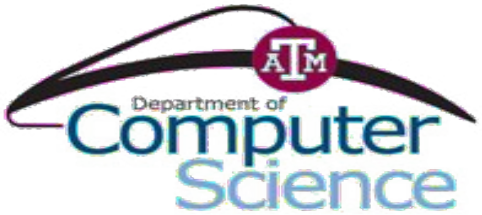
Parasol
Smarter computing.

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

SELL example

$$Z = a * X + Y$$

```
template <Parallelizable T>
void f(const T& v)
{
   double d = v[2];     // OK
   double* d = &v[2];    // NOT OK
};
```



References

[GJS+06] Gregor, Douglas; Järvi, Jaako; Siek, Jeremy; Lumsdaine, Andrew; Dos Reis, Gabriel; Stroustrup, Bjarne: Concepts: First-Class Language Support for Generic Programming in C++. to appear OOPSLA'06.

[DRS05] Stroustrup, Bjarne; Dos Reis, Gabriel: A concept design. C++ Committee, paper N1782. April 2005.

[LV04] Lattner, Chris; Adve, Vikram: LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. CGO '04.

[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.