Program Generation and Modification Using Multiple Domains

Ira D. Baxter
DMS Software Reengineering Toolkit

• *Automated* source code analysis and *modification*
  – *Leverage transformation machinery needed to build DMS vision*

• Enables wide variety of SE tasks to be automated
  – *Commercial applications*
    • Source Formatters, Hyperlinked Source Browsers
    • Documentation extraction
    • Metrics
    • Preprocessor conditional simplification
    • Test Coverage and Profiling tools
    • Clone Detection and removal
    • XML DTD compilation
    • DSL code generation: Factory Automation
    • Migrations (JOVIAL to C)
  – *Research applications*
    • Aspect-weaving (U. Alabama Birmingham)
    • Large-scale C++ component restructuring (SD/Boeing)  **OOPSLA DEMO**
    • Code generation/quality checking for spacecraft (NASA/JPL)
DMS Software Reengineering Toolkit = Generalized Compiler

- Underlying Hypergraph representation: trees, graphs, …
- Parsing/Prettyprinting
  - UNICODE lexer with binary conversions, lexical format/comment capture
  - GLR (context-free) parser with automatic tree builder
  - "Text Box" building language; reproduces comments!
- Analysis
  - Multipass attribute grammars
  - Generalized symbol table support: inheritance, overloading, …
- Transformation
  - Complete AST interface => procedural transforms (& analyzers)
  - Conditional Source to Source transforms w/ associative/commutative laws
- Predefined Domains
  - Specification, Technology, and Legacy languages
    Spectrum, .MDL, XML, SQL, IDL, C/C++, C#, Java, COBOL, many more…
Knowledge Capture in Manageable Chunks:

DMS Domain Parts

• Syntax
  – External Form  Lexical Specification and Grammar Rules
  – Parser  GLR parser + custom code (preprocessor…)
  – AntiParser  Formatting Rules + custom lexeme generators

• Semantics
  – Analyzers  Typically includes Name/Type resolution
  – Optimizations  (Source to Source) transforms in domain
  – Refinements  (Source to Source) transforms between domains

*Draco “domain” paradigm… James Neighbors, 1978*
DMS C++ Symbol Table for a function

```c
void fnct(int param)
{
    int local;
    11:
    if (param + local)
        { int nested1;
            return;
        }
    else
        { int nested2;
            goto 12;
            if (0)
                12:
                { int deep;
                    return;
                }
        }
}
```

Not possible to do much without this!
How to represent multiple domains?

• Union grammar? … for C and C#
  – Combine rules of two grammars

  \[
  \begin{align*}
  \text{program_element}_C &= \text{function}_C \text{Declaration} \\
  \text{expression}_C &= \text{identifier} \, \text{'['} \text{exp} \, \text{']'} \\
  \text{program_element}_C &= \text{class}_C \text{Declaration} \\
  \text{expression}_C &= \text{identifier} \, \text{'['} \text{exp} \, \text{']'}
  \end{align*}
  \]
  – Simplifies problem of writing transformation rules
    • Only one “syntax”
  – Makes separating semantics more difficult
    • What if \( a[i] \) has different index origin?

• Separate grammars!