Source-to-source transformations
Supporting tools and infrastructure

Nikolaos Kavvadias
nkavv@uop.gr

March 31st, 2009
By the term “source-to-source transformation” we refer to any mechanism that when applied to a SOURCE program, a functionally equivalent TARGET program is produced.

Basic assumptions
- SOURCE and TARGET programs submit to the same programming language semantics
- A “database” of the source program is generated by translating to a form of high-level intermediate representation (HIR)
- The target program is produced by pretty-printing the HIR view of the source program

Secondary assumptions
- Structural information (e.g. program layout, line numbers) may not be preserved when translating to the HIR form
Potential uses of source-to-source (also termed as “source-level”) transformations:
- Algebraic and other simplifications (e.g. matrix flattening)
- Data access improvements for enhancing data locality
- Enforcing the use of a data memory hierarchy (data reuse transformations)
- Conversion to a standard (canonicalized) form
- Enabling the application of lower level transformations (closer to the underlying machine model)

High-level view of the source-to-source translation process

Diagram: Source program → HIR database → transformation → Target program
Useful software facilities for implementing a source-to-source transformation framework

- AST builder and walker
- AST/HIR query engine
- Semantics checker and/or HIR validator
- AST2HIR and HIR2AST modules

In general, comprehensive frontend facilities would be extremely useful to build upon
An overview of tools and infrastructure

- Existing software systems
  - The C-to-C source code translator
    (ftp://theory.lcs.mit.edu/pub/c2c/), now defunct
  - Memphis tree builder and walker tool
    (http://memphis.compilertools.net/index.html)
  - EDG C/C++ frontend (http://www.edg.com)
  - TXL (http://www.txl.ca)
  - The Cetus project (http://cetus.ecn.purdue.edu/)
  - ROSE compiler infrastructure
    (http://www.rosecompiler.org)
- ...or “Roll Your Own” system/infrastructure
  - Based on extensible text transformation technology: XML + XSLT (http://www.w3.org/2001/XMLSchema)
  - Adapt to exactly fit your needs

Nikolaos Kavvadias nkavv@uop.gr
Source-to-source transformations
The C-to-C MIT source code translation tool

Features
- AST building and type checking from ANSI C
- Data flow analysis on the AST
- MIT license

Cons
- Relatively few built-in transformations
- Further development has ceased
- Distribution site now defunct (as of late 2005)

Suggestions
- Lack of features and support prevent C2C from being a reliable infrastructure
Memphis

Characteristics and features

- Intended audience are compiler writers
- Provides basic mechanisms for rule-based tree transformations
- Works well with Lex and Yacc
- Memphis personal license + GPL

Cons

- Lack of a ready-to-use C grammar
- No real world examples
- Largely unknown to the community
- Development site ceased (http://www.combo.org)

Suggestions

- Infrastructure is only minimal and not really useful for any practical use
EDG C/C++ frontend

- **Features**
  - Mature and complete C/C++ frontend
  - Covers the entire C99 and latest C++ standards
  - AST construction and a rich set of related data structures
  - Extensive documentation (~ 600 pages)
  - Actively supported by a company (Edison Design Group)
  - Proprietary open-source license; free for academic research

- **Cons**
  - Developing user tools requires a significant time investment
  - No API (which would simplify the development of extensions and plugins)

- **Suggestions**
  - Viable choice in case infrastructure development time is more or less irrelevant
 TXL

Features

- TXL is a functional programming language mainly used for domain-specific language development
- Language primitives for specifying tree rewriting rules
- Comes with many frontends (C, C++, Java, Modula-2/3, etc)
- Has been used in production environments (source code transformations for eliminating patterns of code arising Y2K problems)

Cons

- Development seems to be steered by a single person; no real community being able to contribute
- TXL is a narrow-scope language
- No previous experience with TXL

Suggestions

- Viable choice only if the source transformations involved could be easily specified by bare bones term rewriting
Cetus

- **Features**
  - Source-to-source C compiler written in Java
  - Extensive set of compiler passes working on a high-level IR
  - Supports parallelization techniques
  - Analyses and transformations
    - Data dependence analysis
    - Loop parallelizer
    - Source program canonicalization
    - Loop outlining (procedural abstraction of loops)

- **Modified Artistic License**

- **Cons**
  - Depending on external tools (Java libraries, ANTLR)
  - Very small (nonexistent?) community outside Purdue Univ.

- **Suggestions**
  - The scope of this infrastructure seems appropriate
  - Focuses on parallelism extraction for OpenMP and not loop restructuring transformations suitable for other purposes
Features

- A C++ tool for building source-to-source translators
- Builds upon the EDG frontend (included)
- Under active development
- Analyses and transformations
  - AST construction, traversal and querying, CFG construction, data flow analyses
  - Predefined loop optimizations: loop interchange, loop fusion, loop fission, loop splitting, loop unrolling
- Revised BSD license

Cons

- External dependencies (Java, compiled version of Boost)
- It is unclear whether there is an active community yet

Suggestions

- Scope and purpose of this infrastructure seem appropriate
- Heavy work for an EDG-based ecosystem already done
Custom text transformation engine based on XML

- **Features**
  - XML is a well-established and mature technology
  - Provides the means for specifying your own text manipulation and transformation primitives
  - Vast community of developers and users

- **Cons**
  - Development of the infrastructure: constructors, traversals, querying mechanisms, pretty-printers, dumping to debugging formats, fundamental analyses and transformations
  - Development effort cannot be easily estimated

- **Suggestions**
  - XML is a tool for serious work given that the development project is closely managed in respect to timeframes and human resources
Conclusion

- There exist viable solutions for creating a source-to-source translation tool fitting our needs
  - EDG: roll your own analyses and transformations
  - XML: roll your own and customize to your specific needs
  - ROSE: open and extensible infrastructure based on the EDG frontend

- The choice of one of these solutions requires taking into account subjective factors such as:
  - Communication of ideas, concepts and results to and from other parties involved
  - Extensibility of the infrastructure
  - Preexisting knowledge
  - Personal preference

Memphis tree builder and walker.
http://memphis.compilertools.net/index.html

The C-to-C source code translator.
ftp://theory.lcs.mit.edu/pub/c2c/


The Cetus project. http://cetus.ecn.purdue.edu

ROSE: A tool for building source-to-source translators.
http://www.rosecompiler.org
v0.1  (30/03/2009): Initial version