Program transformation

A **program transformation** is any operation that takes a computer program and generates another program. In many cases the transformed program is required to be semantically equivalent to the original, relative to a particular formal semantics and in fewer cases the transformations result in programs that semantically differ from the original in predictable ways.[1]

While the transformations can be performed manually, it is often more practical to use a program transformation system that applies specifications of the required transformations. Program transformations may be specified as automated procedures that modify compiler data structures (e.g. abstract syntax trees) representing the program text, or may be specified more conveniently using patterns or templates representing parameterized source code fragments.[2]

A practical requirement for source code transformation systems is that they be able to effectively process programs written in a programming language. This usually requires integration of a full front-end for the programming language of interest, including source code parsing, building internal program representations of code structures, the meaning of program symbols, useful static analyses, and regeneration of valid source code from transformed program representations. The problem of building and integrating adequate front ends for conventional languages (Java, C++, PHP etc.) may be of equal difficulty as building the program transformation system itself because of the complexity of such languages. To be widely useful, a transformation system must be able to handle many target programming languages, and must provide some means of specifying such front ends.

A generalisation of semantic equivalence is the notion of program refinement: one program is a refinement of another if it terminates on all the initial states for which the original program terminates, and for each such state it is guaranteed to terminate in a possible final state for the original program. In other words, a refinement of a program is *more defined* and *more deterministic* than the original program. If two programs are refinements of each other, then the programs are equivalent.

See also

- List of program transformation systems
- Metaprogramming
- Program synthesis
- Source-to-source compiler
- Source code generation
- Transformation language
- Transformational grammar
- Dynamic recompilation

References

1. Ward, Martin (1989). *Proving Program Refinements and Transformations* (http://www.cse.dmu.ac.uk/~mward/martin/thesis/index.html) (DPhil Thesis). Oxford University.
2. Pawlak, Renaud; Monperrus, Martin; Petitprez, Nicolas; Noguera, Carlos; Seinturier, Lionel (2016). "SPOON: A library for implementing analyses and transformations of Java source code"

(https://hal.archives-ouvertes.fr/hal-01078532/document). *Software: Practice and Experience*. **46** (9): 1155–1179. doi:10.1002/spe.2346 (https://doi.org/10.1002%2Fspe.2346).

External links

- The Program transformation Wiki (http://www.program-transformation.org)
- Papers on program transformation theory and practice (http://www.cse.dmu.ac.uk/~mward/martin/papers/)
- Transformation Technology Bibliography (http://www.semanticdesigns.com/Company/Publications/TransformBib.pdf)
- DMS Software Reengineering Toolkit: A Program Transformation System for DSLs and modern (C++, Java, ...) and legacy (COBOL, RPG) computer languages
- Spoon (https://github.com/INRIA/spoon): A library to analyze, transform, rewrite, and transpile Java source code. It parses source files to build a well-designed AST with powerful analysis and transformation API.
- JavaParser (https://github.com/javaparser/javaparser): The JavaParser library provides you with an Abstract Syntax Tree of your Java code. The AST structure then allows you to work with your Java code in an easy programmatic way..

Retrieved from "https://en.wikipedia.org/w/index.php?title=Program_transformation&oldid=1015468705"

This page was last edited on 1 April 2021, at 17:19 (UTC).