Perfection is achieved on the verge of collapse... so, don't aim for it.

–Nicolae Vintila

14 Conclusions

The main contributions of this dissertation are two novel and complementary techniques for improving language independence of program transformations: program object model adapters for decoupling the transformation engine from the program model and aspects for adapting generic transformation algorithm skeletons to specific subject languages.

Survey of Transformation Systems – A detailed survey of the state-of-the-art in software transformation systems was presented. It employed feature models and concrete examples taken from about a dozen research systems for describing central parts of the design space for transformation system. The survey indicated that good abstraction facilities for the program model are necessary for language independence.

Program Object Model Adapters – Large-scale reuse of transformations and transformation systems across subject language infrastructures is supported by the program object model adapters. They weld together transformation system runtimes with the abstract syntax tree of an existing language infrastructure such as the front-end of a compiler. This is done by on-the-fly translation of rewriting operations in the transformation system to sequences of equivalent method calls on the AST API. This obviates the need for data serialisation and thus enables efficient integration between transformation engines and front-ends. The technique can be applied to most tree-like APIs and is applicable for many term-based rewriting systems.

Aspects – The AspectStratego language offers a declarative mechanism for adding support for subject language families to transformation libraries. This improves the genericity and language-independence of transformations. The aspects also capture many cross-cutting concerns found in transformation programs. Well-defined points in the execution of a transformation program can be identified, parametrised and modified using aspects. By extracting and parametrising execution points, a transformations can be formulated, even retroactively in cases where grey box reuse is accepted.

Strategic Graph Rewriting – The System S rewriting calculus has been extended to handle graphs. Its implementation, the GraphStratego language, provides new language abstractions for capturing graph-like program models while still maintaining the benefits from strategic rewriting: the separation of rewrite rules from traversal strategies, thus providing a language for strategic graph rewriting.

An Extensible Transformation Language Framework – The MetaStratego framework used to prototype the proposed language extensions is available as a general language prototyping framework. It is provided so that other developers may experiment with new transformation language extensions.

A Flexible Transformation Runtime – The Stratego/J interpreter provides a transformation engine capable of abstracting over term representations. It may easily be plugged into existing language infrastructures and is supported by jsglr, an implementation of a scannerless GLR parser.

The main contributions are supported by several case studies which serve to demonstrate their usefulness. The presented case studies include:

- An implementation of a domain-specific aspect language for alert handling.
- An interactive development environment for program transformations that has served as a test-bed for the techniques and language abstractions presented in this dissertation.
- A collection of examples of framework-specific transformation and analysis, showing that advanced framework developers may benefit from the above contributions since writing custom language processing tools is now fairly simple.
- A generator of executable unit tests from algebraic specifications that demonstrates how the proposed techniques may be used to extend development environments with new, interactive program transformations.

Spoofax – The Spoofax interactive development environment for strategic programming is useful in its own right. It provides a modern and effective editing environment for transformation developers. Scripts, written in Stratego (with any of the extensions discussed above), may be used to extend the development environment.