Outline of Research Plans in the Context of WG 2.11

Yannis Smaragdakis
htttp://www.cc.gatech.edu/~yannis

April 16, 2004

My research interests are both in the development of domain-specific program generators and in the design of infrastructure for generators. Such infrastructure can be language abstractions and type system support, transformation systems, notations for transformations, etc. In the context of WG 2.11, the infrastructure direction may prove to be quite fruitful. I am particularly interested in infrastructure for generators because it is the domain-independent part of generators research. As such, it can be claimed to be conceptually general and important to the entire generators community, regardless of domain expertise. At the same time, infrastructure tools have significant potential for valuable applications. Many interesting domain-specific languages are getting developed every year. Their authors mostly do not benefit from any research work in the generators community. Program translation is typically done via ad hoc means—there is no established piece of infrastructure for manipulating programs.

An example of my most recent work in this area is Meta-AspectJ (MAJ): a language tool for generating AspectJ programs using code templates. MAJ itself is an extension of Java, so users can interleave arbitrary Java code with AspectJ code templates. MAJ is a structured meta-programming tool: a well-typed generator implies a syntactically correct generated program. MAJ promotes a methodology that combines aspect-oriented and generative programming. Potential applications range from implementing domain-specific languages with AspectJ as a back-end to enhancing AspectJ with more powerful general-purpose constructs. In addition to its practical value, MAJ offers valuable insights to meta-programming tool designers. It is a mature meta-programming tool for AspectJ (and, by extension, Java): a lot of emphasis has been placed on context-sensitive parsing and error-reporting. As a result, MAJ minimizes the number of meta-programming (quote/unquote) operators and uses type inference to reduce the need to remember type names for syntactic entities.