Language Design Assistants as Knowledge-Based Systems

Jan Heering
CWI
Amsterdam
E-mail: Jan.Heering@cwi.nl
WWW: http://www.cwi.nl/~jan/

Abstract

Current *language development systems* (ASF+SDF Meta-Environment, Centaur, Gem-Mex, PSG, Software Refinery, Synthesizer Generator, ...) have widely different capabilities and are in widely different stages of development. All of them can generate lexical scanners, parsers, and prettyprinters, many of them can produce syntax-directed editors, type-checkers, and interpreters, and a few can produce various kinds of software renovation tools. To this end, they support one or more meta-languages for syntax, static and dynamic semantics, and other language aspects.

Actually, these systems incorporate little language design knowledge. Their main assets are the meta-languages they support, and in some cases a meta-environment to aid in constructing and debugging language descriptions. To turn them into true *language design assistants* (LDAs), *language concepts* and *design rules* have to be incorporated.

Basically, the language designer using an LDA picks suitable language building blocks from the language knowledge base (language library), customizes them, and composes them into larger and larger language fragments. It may be necessary to add entirely new building blocks and concepts in the process, especially domain-specific ones, since it cannot be expected that everything required is already present in customizable form. The LDA provides feedback during customization and composition. Finally, the finished design is implemented by a language development system that serves as back-end to the LDA.

More specifically, the main elements and notions that would play a key role in an LDA would be:

- **Language concept** These are rather diverse. For instance, some language concepts, such as "statement", "expression", or "loop", correspond more or less directly to language constructs. These are *language building blocks*. Other ones, such as "scope" or "side-effect"

http://www.ii.uib.no/~nwpt00

have the character of an attribute to a language building block, while "state" or "call stack" refer to the dynamic behavior of programs or to the language's implementation.

- **Language building block** *Language concept* corresponding more or less directly to a language construct, such as "statement", "expression", or "loop". It may have many attributes, which themselves correspond to *language concepts* of a different kind, such as (abstract) syntax, typechecking, interpretation, data and control flow, side-effects, exceptions, among others.
- **Relation** Relation between *language concepts*. May itself be a (higher-order) *language concept*. Attributes like "scope" and "side-effect" are unary relations. "Implementation" would be an example of a binary one.
- **Language knowledge base** Knowledge base of *language concepts* and their *relations*. It may include theories of the concepts it contains. These may range from rudimentary to elaborate.
- **Customization** The process of adapting *language building blocks* during the language design process by means of instantiation, transformation, or generation.
- **Composition** (Partly) *customized language building blocks* can be composed into larger language fragments. In this way the language is constructed step by step.
- **Constraint checking** is triggered by *customization* and *composition* to check, at least to some extent, the validity of the resulting *language building block*. The constraint checker is a parameter of the LDA design.
- **Design language** Meta-language allowing the user to formulate language design questions and interrogate/browse the *language knowledge base*.
- **Language knowledge representation language** Visual or semi-visual meta-language to express *language concepts* and their *relations*. It has well-defined sublanguages to express *language building blocks* and their attributes. These sublanguages are compiled to the corresponding meta-language(s) of the *language development system* that is used as a back-end.
- **Language development system** Back-end of the LDA and parameter of the LDA design. The LDA facilitates the language description process and then uses a language development system to generate the tooling from the finished description. In our case, the ASF+SDF Meta-Environment will be the back-end. Many of the other systems mentioned above would be suitable as well.

A question immediately coming to mind is whether existing knowledge-based system technology and in particular an existing expert system shell would be suitable to build an LDA.