RAPID LANGUAGE ENGINEERING FOR GREEN-MARL, A DSL FOR HIGHLY PARALLEL GRAPH ANALYSIS

VAAD L. VERGI
EELCO VISSE

SPOOFAX
The Spoofax toolset[2] reduces the development effort of software languages. This gives language designers the time to focus on the design decisions and language semantics.

Syntax definition: type systems, rule analysis, and core generation are specified using the Spoofax language DSLs. The type systems are expressed in the language transformation itself. The type analysis definitions extend the code generation output.

The language base languages have automatically generated code based on Eclipse. The DSLs are fully customizable and extendable. Spoofax designers are also directly variable from the base implementation.

Languages can have multiple textual or graphical core representations. This is a good starting point for domain-specific language with type systems and compiler for domain-specific languages.

The DSL allows graph analysis algorithms in the vastness of the data and allow analysts to replace the existing 38KLOC implementation in Green-Marl implementation. They will eventually become the de facto Green-Marl refer.

The Spoofax-based specification for Green-Marl required 2.5 months of development effort.

The DSL language-based system is implemented in the Stratego program transformation language.

The compiler-based transformation for Green-Marl variable and property assignments is achieved this new implementation will become the de facto Green-Marl reference.

The Spoofax-based specification for Green-Marl required 2.5 months of development effort.

Test Driven Development

Spax experiments 100 of languages. Unit tests for the core of the language are written using the Spax testing language DSLs. This includes tests for the parser, syntax analysis, code analysis and code generation. Test reports are written for the build system and test analysis.

SYNTAX DEFINITION

Green-Marl syntax is specified in GM4S (Syntax Definition For- matted, version 1.0). GM4S captures both the syntactic rules for the language and the layout preferences of the language.

Spax languages have many syntactic representations for equivalent syntactic constructs, e.g. balanced and unbalanced brackets. Spax automatically generates a same analysis, resolution and context-sensitive autocompletion for each specification. The general analysis is automatically incremental ensuring the reuse of the DSL for larger programs.

In Spax, name binding rules for languages are declaratively specified in the nbl (meta-DSL). This specifies the semantics exclusively from the usual name resolution algorithm. Spax automatically generates a same analysis, resolution and context-sensitive autocompletion for each specification. The general analysis is automatically incremental ensuring the reuse of the DSL for larger programs.

NAME AND TYPE DEFINITIONS

In Spax, name binding rules for languages are declaratively specified in the nbl (meta-DSL). This specifies the semantics exclusively from the usual name resolution algorithm. Spax automatically generates a same analysis, resolution and context-sensitive autocompletion for each specification. The general analysis is automatically incremental ensuring the reuse of the DSL for larger programs.

SPAX EXPERIMENTS

Spax experiments 100 of languages. Unit tests for the core of the language are written using the Spax testing language DSLs. This includes tests for the parser, syntax analysis, code analysis and code generation. Test reports are written for the build system and test analysis.

The DSL-based specification for Green-Marl required 2.5 months of development effort.

The compiler-based transformation for Green-Marl variable and property assignments is achieved this new implementation will become the de facto Green-Marl reference.

The Spoofax-based specification for Green-Marl required 2.5 months of development effort.

Evaluation

The Spoofax-based specification for Green-Marl required 2.5 months of development effort. The Spoofax-based specification for Green-Marl is a mix of Stratego, GM4S, and Stratego. Spax’s highly-specialised nbl DSLs for syntax and name binding definition reduce implementation effort of the parser and name analysis. Only 15% of the implementation are due to syntax graph analysis, implemented in Stratego.

The Green-Marl compiler translates stored input programs to functionally identical output programs in the Stratego program transformation language.

The Green-Marl implementation is implemented in Stratego. The transformation rules in Stratego utilize the compiled and emitted code to be emitted in a concrete syntax in the Green-Marl case, the (not) hard-coded transformation rules is compiled in a concrete syntax. The transformation rules in Stratego are simplified and improved on strategy.

Compilation

The Green-Marl compiler translates stored input programs to functionally identical output programs in the Stratego program transformation language.

The Green-Marl implementation is implemented in Stratego. The transformation rules in Stratego utilize the compiled and emitted code to be emitted in a concrete syntax in the Green-Marl case, the (not) hard-coded transformation rules is compiled in a concrete syntax. The transformation rules in Stratego are simplified and improved on strategy.

Spax experiments 100 of languages. Unit tests for the core of the language are written using the Spax testing language DSLs. This includes tests for the parser, syntax analysis, code analysis and code generation. Test reports are written for the build system and test analysis.

The DSL-based specification for Green-Marl required 2.5 months of development effort. The Spoofax-based specification for Green-Marl is a mix of Stratego, GM4S, and Stratego. Spax’s highly-specialised nbl DSLs for syntax and name binding definition reduce implementation effort of the parser and name analysis. Only 15% of the implementation are due to syntax graph analysis, implemented in Stratego.

The DSL-based specification for Green-Marl required 2.5 months of development effort. The Spoofax-based specification for Green-Marl is a mix of Stratego, GM4S, and Stratego. Spax’s highly-specialised nbl DSLs for syntax and name binding definition reduce implementation effort of the parser and name analysis. Only 15% of the implementation are due to syntax graph analysis, implemented in Stratego.