Introduction to Generic Language Technology

Mark van den Brand
Paul Klint
Jurgen Vinju

Today

- Tools for software analysis and manipulation
- Programming language independent (parametric)
- The story is from the IDE/PE perspective
- First a general overview, then introduction to specific technologies

What ...

- ... is a language?
- ... is a Programming Environment (PE)?
- ... is Generic Language Technology (GLT)?
- ... is a Program Generator?
- ... is a Programming Environment Generator?
What is a Language?

- A programming language
  - Assembler, Cobol, PL/I, C, C++, Java, C# ...
- A Domain-Specific Language (DSL)
  - SQL for queries
  - BibTex for entries in a bibliography
  - Euris for railroad emplacement safety
  - Risla for financial products

What is a Programming Environment (PE)?

A system that supports the development of programs in order to:

- Increase productivity:
  - Uniform user-interface (UI); integrated tools
  - Increased interaction; early error detection
- Increase quality:
  - Integrated version management
  - Integrated testing
  - Integrated documentation

Aspects of a Language

- Syntax
  - Textual form of declarations, statements, etc.
- Static Semantics
  - Scope and type of variables, conversions, formal/actual parameters, etc.
  - Queries: who calls who, who uses variable X, ...
- Dynamic Semantics
  - Program execution
Classical PE

- Text editor only
- Programs stored in files
- Complete recompilation after each change
- Late error detection
- Debugging requires recompilation with different options
- Example:
  - xemacs or vim
  - gcc or javac

Integrated PE (IPE)
also: Integrated Development Environment (IDE)

- Specialized, syntax-directed, editor for each language
- Common intermediate representation for all tools
- Incremental processing (reusing results of previous calculations)
- One GUI
- Early error detection
  - Syntax errors
  - Undeclared variables
  - Type errors in expressions

Functionality of an IPE

- Syntax-directed editing/highlighting, pretty printing
- Typechecking
- Restructuring
- Versioning
- Executing, debugging, profiling
- Testing
- Documenting

Simple, External, View of IPE
Simple, Internal, View of IPE

Examples of IPEs

- Eclipse: www.eclipse.org
  - Integrated Development Environment (IDE) for Java
  - Plug-in mechanism for extensions
- MS Visual Studio: msdn.microsoft.com/vstudio
  - IDE for various languages VB, C, C++, C#

What ...

- ... is a language?
- ... is a Programming Environment (PE)?
- ... is Generic Language Technology (GLT)?
- ... is a Program Generator?
- ... is a Programming Environment Generator?

What is Generic Language Technology?

- Goal: Enable the easy creation of language-specific tools and programming environments
- Separate language-specific aspects from generic aspects
- Approach:
  - Find good, reusable, solutions for generic aspects
  - Find ways to define language-specific aspects
  - Find ways to generate tools from language-specific definitions
Generic aspects

- User-interface
- Text editor
- Program storage
- Documentation

Defining Language Aspects

- Syntax
  - Lexical and context-free grammar
- Static semantics
  - Algebraic specification/rewrite rules
- Dynamic semantics
  - Algebraic specification/rewrite rules

From Definition to Tool

- Syntax
  - Scanner and parser generation
- Static semantics
  - Term rewriting
- Dynamic semantics
  - Term rewriting

Examples of Generic IPEs

  - A framework for Generic Integrated Development Environments (IDE)
  - The goal of the Eclipse IMP is to provide an extensible platform for the development of high-quality, fully-featured, language-specific IDE's in Eclipse.
What ...

- ... is a language?
- ... is a Programming Environment (PE)?
- ... is Generic Language Technology (GLT)?
- ... is a Program Generator?
- ... is a Programming Environment Generator?

What is a Program Generator?

- Definition of problem P
- Generator
- Generated program that solves P

Examples of Program Generators (1)

- Regular expression matching:
  - Problem: recognize regular expressions $R_1, \ldots, R_n$ in a text
  - Generates: finite automaton
- Web sites
  - Problem: create uniform web site for set of HTML pages
  - Generate: HTML code with standard layout and site map

Examples of Program Generators (2)

Generate bibliographic entries; input:

```latex
@article{BJKO00, author = {Brand, (M.G.J. van den) and Jong, (H.A. de) and P. Klint and P. Olivier},
   title = {(E)fficient (A)notated (T)erms),
   journal = {Software, Practice & Experience},
   year = {2000},
   pages = {259--291},
   volume = {30}}
```

generates:

Examples of Program Generators (3)

- Compiler:
  - Input: Java program
  - Generates: JVM code
- C preprocessor:
  - Input C program with `#include`, `#define` directives
  - Generates C program with directives replaced.

Program Generators (summary)

- Problem description is specific and is usually written in a Domain-Specific Language (DSL)
- Generator contains generic algorithms and information about application domain.
- A PG isolates a problem description from its implementation ⇒ easier to switch to other implementation methods.
- Improvements/optimizations in the generator are good for all generated programs.

What ...

- ... is a language?
- ... is a Programming Environment (PE)?
- ... is Generic Language Technology (GLT)?
- ... is a Program Generator?
- ... is a Programming Environment Generator?

What is a Programming Environment Generator (PEG)?

- A PEG is a program generator applied in the domain of programming environments
- Input: description of a desired language $L$
- Output: (parts of) a dedicated $L$ environment
- Advantages:
  - Uniform interface across different languages
  - Generator contains generic, re-usable, implementation knowledge
- Disadvantage: some UI optimizations are hard
Programming Environment Generator

Formal definition of language $L$

Generator

Dedicated environment for editing, manipulating and executing $L$ programs

PEG = collection of program generators

Definition of $L$ syntax
Definition of $L$ static semantics
Definition of $L$ dynamic semantics

Parser Generator
Typechecker Generator
Evaluator Generator

$L$-parser
$L$-typechecker
$L$-evaluator

Integrated $L$-programming environment

From Definitions to Components

Syntax Definition

Text
Edit Commands
Text
Error Messages
Values

Parser
Editor
Pretty Printer
Type Checker
Evaluator

Syntax Tree Manager

Static Semantics

Dynamic Semantics

PEG: further definitions

- Lexical syntax
- Concrete syntax
- Abstract syntax
- Pretty printing
- Editor behaviour
- Dataflow
- Control flow
- Program Analysis
- Program Queries
- Evaluation rules
- Compilation rules
- User Interface
- Help rules
- ...

Introduction to Generic Language Technology
ASF+SDF Meta-Environment (1)

- An interactive development environment for generating tools from formal language definitions
- Based on:
  - Full context-free grammars
  - Conditional term rewriting
- Language definitions written in ASF+SDF
  - SDF: Syntax definition Formalism
  - ASF: Algebraic Specification Formalism

Summary

- Generic Language Technology helps to build tools for language processing quickly
- Programming Environment Generators are an application of GLT
- The ASF+SDF Meta-Environment is an Interactive Development Environment for language definitions and a Programming Environment Generator

Further reading (1)

Further reading (2)
application areas


See: www.meta-environment.org