Formal Semantic Specification of a Core Object-Oriented Language

Project Plan

Diploma project	
Project period:	May 17, 2004 – September 16, 2004
Student name:	Thomas Bietenhader
Status:	10th Semester
Email address:	tbietenhader@student.ethz.ch
Supervisor name:	Bernd Schoeller

1 Project description

Overview

To write provable correct programs is the ultimate goal of every software engineering project. A correctness proof is preferrably done in a strict formal way. The benefits are quite obvious. Existing proofs can be veriefied automatically. Further formalism forces one to be very precise in the argumentation of proofs, thus there are no assumption that enter implicitly in the proof.

The drawback is that programming gets more demanding. It is the programmer's obligation to provide a formal specification of the job the program is supposed to do. And once the program is written, somebody has to deliver the proof of correctness according to the specification. Ideally the proof is generated (semi-)automatically, or if the program is incorrect the proof assistant gives hints to detect the flaw.

The most important role in every correctness proof is played by the semantic definition of the programming language used. In order to derive formal proofs, this semantics has to be stated completely formally.

Scope of the work

The main goal of this project is proving program correctness. However in order to achieve this goal, one needs an appropriate programming language whith a complete formal semantics. Instead of starting from an existing language a new

language together with the formal semantics should be defined. Thereby the language can be restricted to the essential.

Thus the focus is shifted from proving to formal semantics of programming languages. The language to be defined is supposed to be object-oriented in its core – it is not intended to include concepts as inheritance or genericity at this stage. An integral part of the language are contracts as a means to make the specification part of the program code. This especially raises the question of the expressive power, these contracts should have.

The issue of proving is for the most part postponed towards the end of the project, even though it cannot be ignored completely while defining the semantics. Starting from small example proofs, the power and limitations of proofs should be explored. Depending on the success of these experiments, steps towards automated proof-generation could also be a topic.

Intended results

- A small object-oriented language, preferably a subset of Eiffel. In order to keep the definition of the semantics as simple as possible, the number of language constructs is limited. The following concepts should be included:
  - objects/classes, references
  - features, attributes, feature invocation
  - creation procedure
  - control statements: conditional, loop
  - preconditions, postconditions, (class-/loop-)invariants, assertions
- A formal semantics of the language. The kind of semantics will probably be a combination of denotational and axiomatic semantics.
- Example proofs of program correctness.
- Parser/Scanner

2 Background Material

Reading list

- Bertrand Meyer: *Towards practical proofs of class correctness*, to appear in *ZB 2003: Formal Specification and Development in Z and B*, Proceedings of 3rd International Conference, Turku, Finland, June 2003, eds. Didier Bert, Jonathan P. Bowen, Steve King and Marina Waldén, *Lecture Notes in Computer Science 2651*, Springer-Verlag, 2003, pages 359-387.

- Bertrand Meyer: *Proving Pointer Program Properties, Part 1: Context and Overview*, in *Journal of Object Technology*, vol. 2, no. 2, March-April 2003, pp 87-108, at http://www.jot.fm/issues/issue_2003_03/column8

- Bertrand Meyer: *Proving Pointer Program Properties, Part 2: The overall object structure*, in *Journal of Object Technology*, vol. 2, no. 3, May-June 2003, pp 77-100, at http://www.jot.fm/issues/issue_2003_05/column8

- Bertrand Meyer: *Object Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.

- Bertrand Meyer: *Introduction to the Theory of Programming Languages*, Prentice Hall, 1990.

- Hanne Riis Nielson and Flemming Nielson: *Semantics with Applications: A Formal Introduction*, revised edition July 1999 (original edition 1992, John Wiley & Sons).

- Bernd Schoeller: *Strengthening Eiffel Contracts using Models.*

- Bernd Schoeller: *Yet Another Formal Approach to Object-Oriented Programs*, Draft version.

- R. Stärk, J. Schmid and E. Börger: *Java and the Java Virtual Machine – Definition, Verification, Validation*, Springer, 2001.

- *Java Modelling Language (JML) Homepage*: www.jmlspecs.org

3 Project management

Objectives and priorities

The language definition – including the formal semantics – and formal proofs are the major parts of the project. To think about proof automatization is desirable. The parser is considered rather as a byproduct. In the follwoing table 1 stays for highest priority.

Language definition	1
Proof examples	1
Proof automatization	2
Parser	2

Criteria for success

1. There are no severe limitations to the expressiveness of the language. In particular basic data structures as *linked lists*, *stacks* or *trees* can be represented in a reasonable way.

2. The definition of the semantics is well suited for proving program properties. On the other hand the complexity of the semantics should be reduced to a minimum.

3. Interesting proof examples – ideally full proofs of program correctness.

Method of work

The language definition is developped incrementally. For validation the first proving experiments are already done in an early stage. As much of the work is done on an experimental basis, flexibility is important.

Quality management

Documentation

Each project step is documented. This includes particularly the justification of design decisions and documentation of software produced.

Validation steps

At the beginning of each step the intended results are formulated. Validation is done according to this requirements specification.

4 Plan with milestones

Tasks

1. *Language definition – "classical" constructs*
   This task is concerned with the program constructs which are of a rather instructive nature, i.e. constructs which enable the programmer to tell the computer *how* something has to be done.

2. *Language definition – logical assertions*
   In contrast to task 1, the program constructs treated in this task allow the programmer to reason about *what* a certain piece of code should do. This includes concepts as e.g. *contracts* and *assertions*.

3. *Example proofs*
   The formal semantics is supposed to be well-suited for proving. In order to validate this requirement, example proofs have to be done.

4. *Proof automatization*

5. *Parser*
   The parser is developped to an extent as considered helpful for the proving experiments during the project.

6. *Documentation*

Project steps and milestones

Step 1: Definition of a core language

The first step should primarly help to get acquainted with the topic. In particularly the issues of how to formulate the semantics and the kind of assertions to be allowed are addressed. In the end a first prototype of the language has to be provided.
Milestone 1: June 21, 2004

Step 2: Extending the language

The language is extended according to the gained experience from step 1. Towards the end of this step proving will become more and more important. The intended result of this step is the full language definition.
Milestone 2: July 26, 2004

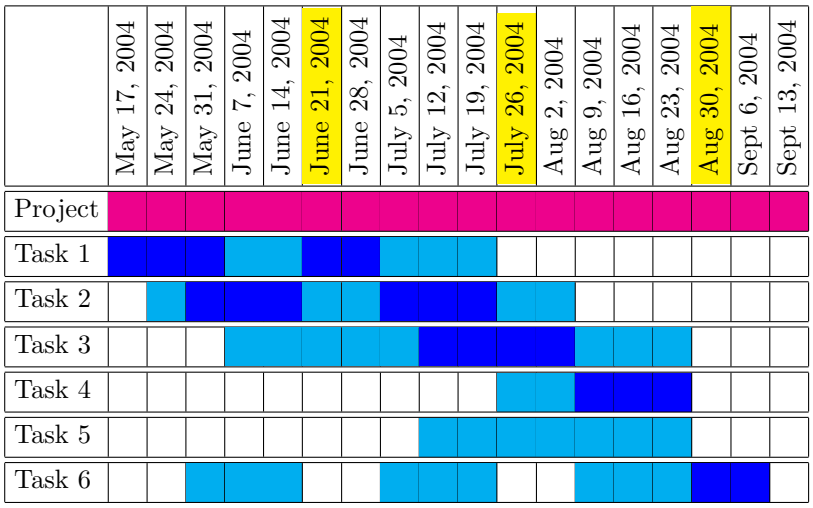
Step 3: Proofs

Once the language definition is complete, the focus is set on proofs. The priority is on constructing example proofs for interesting programs. If this goal is attained successfully, the issue of proof automatization will also be addressed.
Milestone 3: August 30, 2004

Step 4: Finishing the work

The last two weeks are reserved for collecting all results and finishing the documentation.
Project deadline: September 16, 2004

Tentative schedule



References

[1] Chair of Software Engineering: Semester-/Diplomarbeiten, http://se.inf.ethz.ch/projects/index.html, consulted in May 2004.