Interpreter for the SCOOP Programming Model

October 18, 2011

PROJECT PLAN

Type: Master Thesis

Period: October 2011 until March 2012

Student: Mischael Schill (me@mschill.ch), 5th Semester

Mentor: Prof. Bertrand Meyer

Supervisor: Benjamin Morandi

1 PROJECT DESCRIPTION

Overview

Benjamin Morandi, Sebastian Nanz, and Bertrand Meyer defined a comprehensive operational semantics of the SCOOP programming model[4]. The next step is to create an interpreter which strictly follows the semantics. This interpreter is useful to develop and test the SCOOP semantics. The interpreter can be used as a reference implementation for current and future developments (e. g. optimizing compilers) and as a research basis for different strategies and new features.

The Maude System[2] is a programming language supporting equational and rewriting logic. This makes it very suitable for the task of implementing operational semantics in the form of executable operational semantics. In fact, there are several published examples[5] of structural operational semantics implemented using Maude. Maude also provides a model checker which can check properties either formulated as invariants or in Linear Time Logic.

Scope of the work

Implementation of the structural operational semantics of the SCOOP programming model[4].

Intended results

A complete interpreter for the SCOOP programming model according to the defined structural operational semantics[4] with an emphasis on correctness, readability and extendability. Having executable operational semantics, it should be possible, using the model checking facility of Maude, to answer questions about the operational semantics of SCOOP. For example, one question arose from the work on the implementation of SCOOP in Eiffel: If we defer the locking of the processors from the beginning of the feature application to the first separate call, do we loose any of the properties established with SCOOP? This could be generalized to questions concerning the impact of any change to the operational semantics of SCOOP.

This work can possibly lead to a publication with the title: "Using an executable operational semantics to finalize the design of a concurrent programming model".

Criteria for success

Complete all objectives listed under work packages.

2 BACKGROUND MATERIAL

Reading list

[4] describes the operational semantics, [5] is an example on how to implement different operational semantics and [1, 3] provide information on the Maude framework.

3 PROJECT MANAGEMENT

Components of the solution

1. Syntax: Maude features an integrated system for parsing. The Syntax of the simplified SCOOP model has to be defined in Maude.
2. Abstract data types: Implement the abstract data types in Maude.
3. Implementation of the inference rules: Use rewrite rules to implement the inference rules.
4. Simple Base Library (ANY, INTEGER, BOOLEAN, ...): Implement the base classes that are needed to write programs in SCOOP.
5. Scheduling: Figure out how to change the rewrite strategies Maude applies.
6. Tracing: Find possibilities to trace the program execution and document it.

7. Testing: Test the interpreter using examples.
    - Write example programs and check them with the interpreter.
    - Use provided Maude tools for testing and checking.
    - Write code documentation for every (non-trivial) function, data type etc.
8. Documentation:
    - Write a comprehensive user guide on how to use the interpreter.
    - The developer guide has to contain information on the architecture of the interpreter and how to extend the interpreter with custom schedulers.
9. Debugging: Add the possibility to pause the execution, go step by step and to inspect the state.

Project steps / Milestones

1. Implement Syntax without features added by Milestone 5 to 10
2. Implement ADT's without features added by Milestone 5 to 10
3. Implement inference rules without features added by Milestone 5 to 10
4. Create runtime system
5. Add support for separate calls
6. Add support for expanded types
7. Add support for once routines
8. Add support for post condition evaluation (only synchronous)
9. Add support for processor tags
10. Add support for invariant evaluation
11. Test whole interpreter
12. Inspect tracing and debugging
13. Finish documentation
14. Finish thesis

Schedule

Milestone	1	2	3	4	5	6	7	8	9	Weeks
1	X									3
2	X	X								1
3			X							1
4				X			X	X		2
5	X	X	X				X	X		2
6	X	X	X				X	X		2
7	X	X	X				X	X		2
8	X	X	X				X	X		2
9	X	X	X				X	X		1
10	X	X	X				X	X		1
11							X	X		3
12					X	X		X	X	2
13					X			X		1
14										1

Method of work

- Weekly meetings
- Weekly reports
- Online communication
- Maude for the implementation
- LaTeX for the documentation and thesis
- Iterative development, starting with a small subset of the SCOOP model (e. g. no expanded classes)
- Version Control System provided by the SCOOP research group

Deadline

Begin of April 2012

References

[1] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. Maude manual. Available from: `http://maude.cs.uiuc.edu/maude2-manual`.

[2] Maude homepage. Available from: `http://maude.cs.uiuc.edu`.

[3] Theodore McCombs. Maude primer. Available from: `http://maude.cs.uiuc.edu/primer`.

[4] B. Morandi, S. Nanz, and B. Meyer. A comprehensive operational semantics of the scoop programming model. *Arxiv preprint arXiv:1101.1038*, 2011.

[5] A. Verdejo and N. Martí-Oliet. Executable structural operational semantics in maude. *Journal of Logic and Algebraic Programming*, 67(1-2):226–293, 2006.