Modelling and verifying asynchronous systems in GROOVE

PROJECT PLAN

Master's thesis

Project period: 10.10.2014 - 10.04.2015

Student name: Claudio Corrodi

Status: 5th Semester, MSc in Computer Science

Email address: clcorrod@ethz.ch

Supervisor: Dr. Chris Poskitt, Chair of Software Engineering, ETH Zurich

External co-supervisor: Dr. Alexander Heußner, Software Technologies Research Group, University of Bamberg

1. PROJECT DESCRIPTION

Background

With the rise of multi-core and distributed architectures, demand is increasing for high-level programming interfaces that alleviate the notorious difficulty of writing efficient and portable parallel code. Two recent, contrasting developments include *Grand Central Dispatch (GCD)* [1] and *SCOOP* [2,3]. The former is a library present in Apple's operating systems (with ports existing for FreeBSD, Linux, and Windows), whereas the latter is a concurrency model for the object-oriented programming language Eiffel. Both provide concurrency at a higher level of abstraction than threads, through mechanisms for asynchronously dispatching "units" (or "blocks") of code – together with various dependencies between them. Concurrent programs expressed in such models can often exhibit complex, perplexing, and surprising behaviour, and there is a pressing need for tools and methods that facilitate their verification and analysis. In the case of GCD, a formal model has been proposed [15,16] and a translation to Petri nets prototyped [17]; in the case of SCOOP, a comprehensive operational semantics has been formalised in Maude [3,9] – but the model is too complex for many of the automatic analyses that the tool supports. For these and other kinds of asynchronous systems, a formalisation is needed that is natural, quick to prototype, supported by rigorous theory, and supported by a tool that can perform analyses of interest on the models.

Scope of the work

This project proposes to investigate, as such a formalism, the use of a *graph-based abstract semantics* – a visual, powerful, and rigorous modelling technique based on (algebraic) graph grammars [4]. By this, we mean a semantics in which program states are abstracted to graphs, and computational steps to applications of graph rewrite rules (akin to those of Chomsky string grammars, but lifted to graphs). Such a formalism – while unconventional – appears to be a natural choice for prototyping systems as complex as SCOOP, e.g. with objects, processors, and tasks all represented by nodes, and queues, locks, and handlers by edges. Furthermore, it benefits from a well-developed theoretical foundation [4] and support from a number of tools. The most notable of the latter is GROOVE [5], which is able to perform (bounded) model checking directly on graph grammars and programs, is equipped with state-pruning strategies that directly exploit the graph-based representation (e.g. symmetry reduction based on graph isomorphism [6]), and has had its maturity demonstrated through several encouraging case studies (e.g. modelling Java type graphs [13,14]).

This project, in particular, aims to develop automatic translations of SCOOP programs to inputs for the GROOVE model checker to verify, visualise, and analyse. To achieve this, a core (but expressive) subset of the SCOOP language will be formalised, in GROOVE, as a system of graph grammar rules. Then, an automatic translation of (this subset of) SCOOP programs to GROOVE inputs will be developed, and thoroughly evaluated on case studies. Both the formalisation and translation will be constructed as "parametrically" as possible, in order to allow an analogous future treatment of GCD and similar such asynchronous systems.

Objectives / intended results

- a review of the relevant literature (see Section 2)
- a formalisation of a core (but expressive) subset of SCOOP as a graph-based system of rules in GROOVE
- automatic translations of (this subset of) SCOOP programs to GROOVE inputs
- (informal) soundness arguments for the formalisation and translations
- case studies exploring and evaluating the use of GROOVE in analysing and verifying properties of SCOOP programs
- a critical evaluation, and a collection of "lessons learnt" for others wanting to model and analyse asynchronous systems (in particular, GCD) in a similar way

2. BACKGROUND MATERIAL

Reading list

This list comprises pointers to papers and resources that may be helpful for the project (larger resources, such as PhD dissertations, may serve better as definitive references than something to read in their entirety).

- ***SCOOP concurrency model***
  - general background: [2,7,8]
  - operational semantics: [3,9]

- ***GROOVE model checker***
  - general background and tutorials: [5,10]
  - case studies and best practices: [11]
  - comparison with SPIN: [12]
  - language translation (Java type graph): [13,14]

- ***formalisations of other asynchronous systems***
  - Queue-Dispatch Asynchronous Systems (QDAS) for GCD-like systems: [15,16]

3. PROJECT MANAGEMENT

Criteria for success

To be successful the project requires three deliverables. First of all, implementations achieving the objectives of Section 1 that are well-designed, engineered, and evaluated. Secondly, the project – from inception to conclusion – should be documented and evaluated to a high standard in a Master's thesis that adheres to ETH regulations. Finally, an assessed presentation of the work should be given (after the report submission) to the Chair of Software Engineering.

Method of work

There will be regular meetings, telephone conferences, and email discussions. A desk will be provided in the CAB building for the duration of the thesis, and a hot desk will be available – if desired – in the RZ building on Fridays. The supervisors will provide their early, preliminary prototypes of SCOOP / GCD programs in GROOVE for the student to learn from and build upon.

Quality management

The documentation will consist of the final report. Implementation work should be version controlled, with meaningful commit messages throughout. If requested by the student, an optional code review session can be organised, towards the latter part of the project, with members of the Chair of Software Engineering.

4. PLAN WITH MILESTONES

Project steps

Due Date	**Duration**	**Description**
October 29	3 Weeks	Literature review.
November 5	1 Week	Software research (finding existing software and infrastructure that can be used).
December 3	4 Weeks	Formalising graph representation of SCOOP programs.
January 14	6 Weeks	Automatic translation of SCOOP programs to GROOVE models.
January 21	1 Week	Evaluate possibility of extending approach to GCD.
February 18	4 Weeks	Case studies, verification, and evaluation.
March 18	4 Weeks	Writing the report.
April 10 (deadline)	3 Weeks	Reserved for unexpected delays.

REFERENCES

[1] *Grand Central Dispatch Reference.* https://developer.apple.com/library/mac/documentation/Performance/Reference/GCD_libdispatch_Ref/Reference/reference.html, accessed October 2014

[2] Piotr Nienaltowski: *Practical framework for contract-based concurrent object-oriented programming*. Ph.D. dissertation, ETH Zurich, 2007

[3] Benjamin Morandi, Mischael Schill, Sebastian Nanz, Bertrand Meyer: *Prototyping a Concurrency Model.* ACSD 2013: 170-179

[4] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, Gabriele Taentzer: *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series, Springer 2006

[5] Amir Hossein Ghamarian, Maarten de Mol, Arend Rensink, Eduardo Zambon, Maria Zimakova: *Modelling and analysis using GROOVE*. International Journal on Software Tools for Technology Transfer 14(1): 15-40 (2012)

[6] Arend Rensink: Isomorphism Checking in GROOVE. ECEASST 1 (2006)

[7] Bertrand Meyer and Sebastian Nanz: *Concepts of Concurrent Computation: Chapter 9*. Book draft, http://se.inf.ethz.ch/courses/2014a_spring/ccc/reading_materials/book/, accessed October 2014

[8] *SCOOP: Project Homepage.* http://cme.ethz.ch/scoop/, accessed October 2014

[9] Benjamin Morandi: *Prototyping a concurrency model*. Ph.D. dissertation, ETH Zurich, 2014

[10] Arend Rensink, Iovka Boneva, Harmen Kastenberg and Tom Staijen: *User Manual for the GROOVE Tool Set.* http://groove.cs.utwente.nl/wp-content/uploads/usermanual1.pdf, accessed October 2014

[11] Eduardo Zambon, Arend Rensink: *Solving the N-Queens Problem with GROOVE - Towards a Compendium of Best Practices*. ECEASST 67 (2014)

[12] Giorgio Delzanno, Arend Rensink, Riccardo Traverso: *Graph- versus Vector-Based Analysis of a Consensus Protocol*. GRAPHITE 2014: 44-57

[13] Arend Rensink, Eduardo Zambon: *A Type Graph Model for Java Programs*. FMOODS/FORTE 2009: 237-242

[14] Eduardo Zambon: *Abstract Graph Transformation: Theory and Practice*. Ph.D. thesis, University of Twente, 2013

[15] Gilles Geeraerts, Alexander Heußner, Jean-François Raskin: *Queue-Dispatch Asynchronous Systems*. ACSD 2013: 150-159

[16] Gilles Geeraerts, Alexander Heußner, Jean-François Raskin: *On the Verification of Concurrent, Asynchronous Programs with Waiting Queues*. ACM Transactions on Embedded Computing. *To appear.*

[17] Julien Meulemans: *G2Q2P*. GitHub repository, https://github.com/JulienMe/G2Q2P. Accessed October 2014