Proceedings of the 2008 Oxford University Computing Laboratory student conference.

FAILY, S. and ŽIVNÝ, S. (eds.)

2008

Computing Laboratory

PROCEEDINGS OF THE OXFORD UNIVERSITY COMPUTING LABORATORY STUDENT CONFERENCE 2008

Programme Co-Chairs: Shamal Faily, Stanislav Živný
Conference Co-Chairs : Christo Fogelberg, András Salamon, Max Schäfer

CL-RR-08-10

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Table of Contents

Foreword

Holding a biennial student conference has been a recent tradition within the Computing Laboratory. For the first time this year, we have dropped *Programming Research Group* from the conference moniker. As these proceedings illustrate, the breadth of research interests are wide enough justify this name change.

This conference serves two purposes. First, the event is a useful pedagogical exercise for all participants, from the conference committee and referees, to the presenters and the audience. For some presenters, the conference may be the first time their work has been subjected to peer-review. For others, the conference is a testing ground for announcing work, which will be later presented at international conferences, workshops, and symposia. This leads to the conference's second purpose: an opportunity to expose the latest-and-greatest research findings within the laboratory. Conferences have traditionally been *meetings of mind*, where experts meet and discuss the research challenges they face. This conference will be no different from this norm. To reinforce the value of conferences, our keynote presentation will be given by Professor Tom Melham on how to attend conferences, and why.

The 14 abstracts within these proceedings were selected by the programme and conference committee after a round of peer-reviewing, by both students and staff within this department.

This conference would not have been possible without the hard work of many people. These include the referees who, despite many other commitments, managed to find the time to give valuable feedback on this year's submissions. The conference also would not have been possible without the generosity of the Computing Laboratory, who sponsored this event, and Keble College, who have generously provided us with a venue. Finally, a conference would not be a conference without this year's presenters, who have contributed such an interesting programme. On behalf of the whole committee - thank you!

Shamal Faily : Programme Co-Chair

Organisation

Programme Committee

Shamal Faily, Stanislav Živný

Conference Committee

Christo Fogelberg, András Salamon, Max Schäfer

Steering Committee

Richard Bird (Honorary Chair), Martin Xie (Past Chair)

Referees

Peter Boehm, Shamal Faily, Christo Fogelberg, Matthias Fruth, Jeremy Gibbons, Joe Loughry, Andrzej Murawski, Laura Rimmell, Toby Murray, András Salamon, Max Schäfer, Andrew Simpson, Mike Spivey, Nikos Tzevelekos, Yue Zhang, Stanislav Živný

A Process Semantics for BPMN

Peter Y.H. Wong and Jeremy Gibbons

Computing Laboratory, University of Oxford, United Kingdom
Peter.Wong@comlab.ox.ac.uk

1 Introduction

Modelling of business processes and workflows is an important area in software engineering. Business Process Modelling Notation (BPMN) [2] allows developers to take a process-oriented approach to modelling of systems. There are currently around forty implementations of the notation, but the notation specification developed by BPMI and adopted by OMG does not have a formal behavioural semantics, which we believe is crucial in behavioural specification and verification activities. The main contribution of our work is to provide a formal process semantics for a subset of BPMN, in terms of the process algebra CSP [3]. By using the language and the behavioural semantics of CSP as the denotational model, we show how the existing refinement orderings defined upon CSP processes can be applied to the refinement of business process diagrams, and hence demonstrate how to specify behavioural properties using BPMN. Moreover, our processes may be readily analysed using a model checker such as FDR [1]. This paper presents a simple example to demonstrate the application of our semantics. We have implemented a prototype for the semantic function in Haskell [1]; a full definition of the semantic function can be found in our technical report[2], and this work will appear in the proceedings of ICFEM 2008 [4].

2 Example

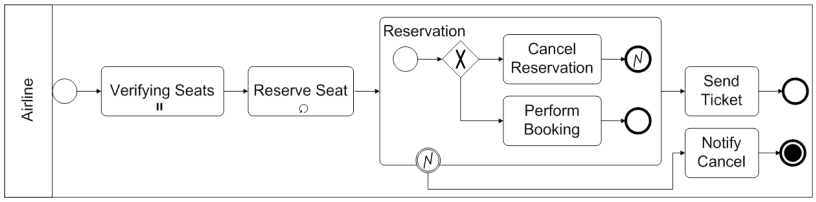


Fig. 1. A BPMN diagram describing the workflow of an airline reservation application.

Figure 1 shows a business process of an airline reservation system. It could be assumed to have been constructed during the development of the reservation system. We observe that the airline reservation business process is initiated by verifying seat availability, after which seats may be reserved. If the reservation period elapses, the business process will cancel the reservation automatically and notify the user. The user might decide to cancel

[1] http://www.comlab.ox.ac.uk/peter.wong/observation
[2] http://www.comlab.ox.ac.uk/peter.wong/pub/bpmnsem.pdf

her reservation, or proceed with the booking. Upon a successful booking, tickets will be issued.

By applying our semantic function to the diagram's syntactic description, we obtain the process corresponding to it,

$$Airline = (\ \|\| j : J \bullet \alpha P(j) \circ P(j)) \setminus \{|init, except|\}$$

where we define set J to index the processes corresponding to the states in the diagram. For each j in J, the process $P(j)$ describes the behaviour associated to the state indexed by j and $\alpha P(j)$ is the set of possible events performed by $P(j)$. For example the process $P(notify)$ defines the behaviour of the task *Notify Cancel*.

$$\begin{aligned} P(notify) = (((&init.notify \rightarrow starts.notify \rightarrow init.abort \rightarrow Skip) \\ &\triangle\ abt.1 \rightarrow Stop) \fatsemi P(notify)) \ \Box\ (fin.1 \rightarrow Skip) \end{aligned}$$

where the events $init.i$ and $starts.i$ represent transitions to and the execution of states i respectively, while events $fin.i$ and $abt.i$ are special events to denote completion and termination of the diagram respectively.

CSP's behavioural semantics admits refinement orderings under reverse containment, therefore a behavioural specification R can be expressed by constructing the most non-deterministic process satisfying it, called the characteristic process P_R. Any process Q that satisfies specification R has to refine P_R, denoted by $P_R \sqsubseteq Q$. For example, Figure 2 is a specification (an abstraction) of the diagram in Figure 1 We use the stable failures model

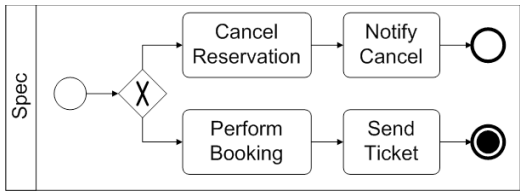


Fig. 2. A BPMN diagram of an abstraction of Figure 1.

to compare process *Airline* with *Spec*.

$$Spec \sqsubseteq_{\mathcal{F}} Airline \setminus (\alpha Airline \setminus \{\, starts.cancel, starts.send, starts.booking, starts.notify \,\})$$

This tells us that the diagram in Figure 1 behaves as specified by diagram in Figure 2.

This work is part of a bigger goal, which is to provide a formal framework allowing precise workflow specification by BPMN and consequently verification via model checking. This work is supported by a grant from Microsoft Research.

References

1. Formal Systems (Europe) Ltd. *Failures-Divergences Refinement, FDR2 User Manual*, 1998. `www.fsel.com`.
2. OMG. *Business Process Modeling Notation Specification 1.0*, Feb. 2006. `www.bpmn.org`.
3. A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1998.
4. P. Y. H. Wong and J. Gibbons. A Process Semantics for BPMN. In *Proceedings of 10th International Conference on Formal Engineering Methods*, volume 5256 of *LNCS*, 2008. To appear.

Getting Automated Refactorings Right

Max Schäfer*

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK.
Max.Schaefer@comlab.ox.ac.uk

Software refactoring is the process of improving the structure of existing code through behaviour-preserving transformations. Over the years, many different refactoring operations for different languages have been proposed, ranging from conceptually very simple operations such as Rename Variable to highly involved transformations such as Extract Slice. Some of the simpler refactorings have been automated and are available in many modern IDEs, but unfortunately even the simplest transformations are surprisingly hard to implement for realistic languages.

As an example, take the Java program in Fig. 1 on the left. Here we have a class `A` with a member field `x` initialised to `23`, and a method `m` with a local variable `y` initialised to `42`; the method returns the value of `x`, i.e. `23`. Assume we want to rename the local variable `y` to `x`.

To do this, we first need to change the declaration of the local variable, then update all references to the old `y` to point to the new `x` (in this case there are none). However, the result so far, depicted in the middle of Fig. 1, is not a correct refactoring: the `return` statement no longer returns the value of the field, but that of the local variable, i.e. `42`, changing the dynamic behaviour of the program.

From a static viewpoint, what has changed is the binding structure of the program: Whereas in the original program the name `x` in the `return` statement refers to the field, it now refers to the local variable. Instead, the refactoring should yield the program at the right, where the binding structure is the same, but the name in the `return` statement has been qualified.

To achieve this preservation of binding structure, we need to identify all names (such as field and type references) in the program that might be influenced by the renaming, which requires a significant amount of name analysis. We can see that even a conceptually extremely simple refactoring like Renaming requires a potentially very involved analysis step to determine which transformations to perform.

```
class A {
  int x = 23;
  int m() {
    int y = 42;
    return x;
  }
}
```

```
class A {
  int x = 23;
  int m() {
    int x = 42;
    return x;
  }
}
```

```
class A {
  int x = 23;
  int m() {
    int x = 42;
    return this.x;
  }
}
```

Fig. 1. An example program, an incorrect refactoring, and a correct one

Traditionally, many refactoring engines have avoided implementing this analysis step, instead relying on an extensive set of preconditions to approximate it [5]. But since these preconditions have to be developed from scratch by reverse-engineering the language specification, it is hard to ensure that they are sufficient. Not surprisingly, even industrial strength refactoring tools come with their fair share of bugs, many of which are naming related [2].

However, many of the required analyses are already implemented (and thoroughly tested) in compilers. Hence our approach is to abandon preconditions, and instead build a refactoring engine on top of the

* This research was carried out in collaboration with other members of the ART [1] project.

JastAddJ compiler [3]. Due to the compiler's modular and extensible architecture we can easily reuse its static semantic analyses and build refactoring specific functionality based on them.

Like JastAddJ itself, our refactoring engine is implemented as an attribute grammar, which affords a very perspicuous and declarative implementation [6]. Extensive testing has shown that its performance is on par with industrial strength applications and it correctly refactors all examples we found to be problematic with other implementations. To gain more confidence in our approach, we have formalised central parts of our implementation in the theorem prover Coq, and proved correctness properties of the central analysis [4].

In the future, we will extend our work to encompass more complex refactorings and more advanced language features. Aspect oriented features, in particular, are a challenge, and we are not aware of any IDE support for automated refactoring of aspect oriented languages.

References

1. Aspect Refactoring Tools Project. Website at `http://progtools.comlab.ox.ac.uk/projects/refactoring/aspect-refactoring-tools`.
2. Torbjörn Ekman, Ran Ettinger, Max Schäfer, and Mathieu Verbaere. Refactoring bugs in Eclipse, IDEA and Visual Studio. `http://progtools.comlab.ox.ac.uk/refactoring/bugreports`, 2008.
3. Torbjörn Ekman and Görel Hedin. The JastAdd Extensible Java Compiler. In *OOPSLA*, 2007.
4. Max Schäfer and Torbjörn Ekman and Oege de Moor. Formalising and Verifying Reference Attribute Grammars in Coq. Available at `http://progtools.comlab.ox.ac.uk/projects/refactoring/formalising-rags`, 2008.
5. William F. Opdyke. *Refactoring Object-Oriented Frameworks*. PhD thesis, University of Illinois at Urbana-Champaign, 1992.
6. Max Schäfer, Torbjörn Ekman, and Oege de Moor. Sound and Extensible Renaming for Java. In *OOPSLA*, 2008.

Quantitative Verification of Software

Mark Kattenbelt*
mark.kattenbelt@comlab.ox.ac.uk

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

1 Introduction

The presence of faults in software can have catastrophic consequences. Unfortunately, it is very difficult to predict the general behaviour of real software. Using *software verification*, however, it is possible to obtain formal guarantees about particular aspects of this behaviour. Most research in software verification has focused on verifying *qualitative* properties such as '*the program never dereferences a null-pointer*' or '*the program always terminates*'. In the presence of probabilistic behaviour, for example through external failures (e.g. networking tools) or randomisation (e.g. randomised algorithms), it is often more appropriate to provide guarantees about *quantitative aspects* of the programs, such as '*the probability of program termination*' or '*the expected number of floating-point operations*'. In this paper, we describe a tool that automatically verifies quantitative properties of ANSI-C programs. More specifically, by marking a set of states of the program as goal states, this tool computes the minimum and maximum probability of reaching the goal, as well as the maximum expected cost incurred in reaching this goal.

2 Quantitative abstraction refinement

The complexity of software means that using abstraction is paramount. A common methodology is that of [5], where the data space of the program is represented increasingly precise finite Boolean abstractions induced by predicates. A single abstraction may or may not be sufficiently precise to result in a guarantee about the concrete program. Hence, if necessary, the abstraction is refined to obtain a more precise abstraction. As depicted in Figure 1 we employ a quantititave variant of this abstraction-refinement methodology in our tool, in which we use components from state-of-the-art tools such as GOTO-CC, SATABS and PRISM.

* Joint work with Marta Kwiatkowska, Gethin Norman and David Parker

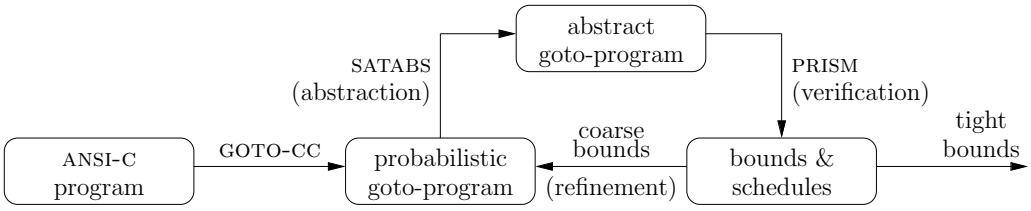


Fig. 1. The abstraction-refinement implementation.

We use GOTO-CC [1] to extract a *probabilistic goto-program* from ANSI-C code. We have defined special functions to model probabilistic and non-deterministic behaviour. Formally, the semantics of the program is described by a Markov decision process.

We use modified components from SATABS [3] to create *abstract goto-programs* through SAT-based predicate abstraction [4]. In our setting, the semantics of such abstractions are two-player stochastic games, which yield two-sided bounds on the probability (or expected cost) of the probabilistic goto-program [7,6]. We use components from PRISM [2] to efficiently compute these bounds. If the bounds are sufficiently tight (i.e. within some error ε), then the abstraction-refinement loop terminates, and otherwise we proceed with refinement.

The refinement procedure considers the optimal strategies of the abstract goto-programs generated by PRISM. The difference in the choices made by these strategies give rise to new predicates, which can be used to improve the abstraction in the next iteration.

3 Experimental results

We have used our tool to analyse two network utilities: a ping client (based on GNU INETUTILS 1.5) and a TFTP client (based on TFTP-HPA 0.48)[1]. Both programs are approximately 1kLOC in size and feature complex ANSI-C language features such as structures, functions, pointers and arrays. Our tool automatically (and efficiently) discovers abstractions that contain sufficient information to compute many useful quantitative properties of the program, such as '*the maximum expected number of echo requests required to establish connectivity*', '*the maximum probability of successfully transferring some file data*' and '*the maximum expected amount of data that is sent before timeout*'. Due to the complexity of these programs, the verification of these properties is far beyond the capabilities of conventional probabilistic model checkers.

4 Conclusion

We have described a tool for automatically computing quantitative properties of ANSI-C software. Our abstraction-refinement method performs well; it automatically and efficiently verifies quantitative properties of real network software, the verification of which is far beyond the capabilities of conventional probabilistic model checkers. We plan to further enhance the method with approximative abstractions, alternative abstract domains and an improved method for abstracting loops.

References

1. GOTO-CC web page. `http://www.verify.ethz.ch/goto-cc/`.
2. PRISM web page. `http://www.prismmodelchecker.org/`.
3. SATABS web page. `http://www.verify.ethz.ch/satabs/`.
4. Clarke, E., Kroening, D., Sharygina, N., and Yorav, K. SATABS: SAT-based predicate abstraction for ANSI-C. In *Proc.* TACAS *'05* (2005), vol. 3440 of LNCS, Springer, pp. 570–574.
5. Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., and Veith, H. Counterexample-guided abstraction refinement. In *Proc.* CAV *'00* (2000), vol. 1855 of LNCS, Springer, pp. 154–169.
6. Kattenbelt, M., Kwiatkowska, M., Norman, G., and Parker, D. A game-based abstraction-refinement framework for Markov decision processes. Tech. Rep. RR-08-06, OUCL, February 2008.
7. Kwiatkowska, M., Norman, G., and Parker, D. Game-based abstraction for Markov decision processes. In *Proc.* QEST *'06* (2006), IEEE CS, pp. 157–166.

[1] See also `http://www.prismmodelchecker.org/files/dphilconf08/`.

Evaluation of Existing IT Systems Under the Common Criteria at Levels Above EAL4

(a disaster story)

Joe Loughry

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Things don't always go the way we planned. Sometimes even well-engineered systems suffer devastating failures. This presentation describes the history and some important events leading up to the failure of a large computer system designed for handling classified information to achieve Common Criteria (CC) security evaluation in 2006.

Let us be clear: the software worked; what failed was the certification process. Although the product, the developer, and some of the participants cannot be named for legal and security reasons, it is believed that important lessons are waiting to be learnt from studying the catastrophic project. A look into the occult world of classified software development for national security systems is given, together with a quick overview of the Common Criteria for Information Technology Security Evaluation ('the Common Criteria') and why it's important.

The CC [2] is an international standard, descended from ITSEC and the Orange Book, for evaluating the security of Information Technology (IT) products. The CC applies equally well to software or hardware; a document called the Protection Profile (PP) is a pre-selected set of certain *Security Functional Requirements* (SFR)[1] that the Target of Evaluation (TOE)—the system under test—must meet. Protection Profiles exist for all sorts of things, ranging from operating systems to fingerprint readers, so end-users need only say, 'I want a product evaluated to the Postage Meter Protection Profile' and they have assurance that the product has been tested for certain functionality [10].

In addition to functional requirements, there are *Security Assurance Requirements* (SAR), that specify how the product should be designed and built. There are seven Evaluation Assurance Levels, ranging from EAL1 (minimal assurance) to EAL7—the most difficult level to achieve [3]. Level EAL2 corresponds approximately to 'minimum good commercial practise', while EAL4 requires the developer to use modern software engineering throughout. At EAL5, formal methods start to make an appearance. EAL6 is about the highest level that software has ever achieved. At EAL7, the entire thing must be formally designed and verified.

Science and engineering learn from studying failures; the businesses that develop systems often might prefer to hide the evidence of a failure and pretend that a disaster never happened. The author, who was a participant in some of these events, has obtained sanctioned access to the project records, and is currently trying to figure out exactly what happened. It may be that the Common Criteria itself is broken; if so, what were the intentions of the originators? If the Common Criteria has lost its way, how can it be fixed?

This was *not* new software. This was a successful system already in production for over a decade, written by competent software developers using exemplary processes and practises. The system is currently certified and accredited in hundreds of locations around the world, protecting classified information at Top Secret and above. It has withstood rigorous security evaluation many times. During the period 2006–2007, what happened?

[1] An example of an SFR is Audit.

Plenty of failures occur; not enough of them get published and studied [5–9]. This presentation will not tell you the answer. The answers are not known yet. But it is hoped that study of this failure, and wide publication of the anonymised results of this research, will benefit other systems seeking CC security evaluation, currently accepted by and shared among twenty-four countries in the mutual Recognition Agreement.

The author's thesis is a better way to do CC evaluation. Some interesting themes that are beginning to appear in the early phases of this research include:

- The effect of corporate email retention policies, originally intended to protect against litigation, on the short-term memory of organisations [12];
- Who are the licenced testing laboratories (CCTLs) and how much of a monopoly do they hold on information needed to complete an evaluation successfully?
- For national security applications, is EAL4 (medium assurance) even *close* to good enough [1, 4, 11]?
- The ethics of IT security evaluation.

I hope to give you a flavour in this presentation of what it's like to work in a classified environment, what Certification and Accreditation (C&A) is all about, and a sense of perplexity at what went wrong. There were some clear danger signs visible from the beginning; I will try to show what they were, and why they were disregarded. At this point, my thesis is more questions than answers, but the questions are interesting and the answers matter.

References

1. ASD(NII)/DoD CIO, United States Department of Defense. *Directive 8500.01E, Subject: Information Assurance (IA)*, October 24, 2002. Certified Current as of April 23, 2007.
2. Common Criteria Sponsoring Organizations. *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model*. Common Criteria Sponsoring Organisations, September 2006. Version 3.1, Revision 1, CCMB-2006-09-001.
3. Common Criteria Sponsoring Organizations. *Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components*. Common Criteria Sponsoring Organisations, September 2007. Version 3.1, Revision 2, CCMB-2007-09-003.
4. Defense Information Systems Agency. *Determining the Appropriate Evaluation Assurance Level for COTS IA and IA-Enabled Products*, March 15, 2004. DISA FSO GO434.
5. Drummond, H. *Escalation in Decision-Making: The Tragedy of* Taurus. Oxford University Press, Oxford, 1996.
6. Goldstein, H. Who killed the virtual case file? *IEEE Spectrum 42*, 9 (September 2005), 24–35.
7. Kletz, T. *Still Going Wrong!: Case Histories of Process Plant Disasters and How They Could Have Been Avoided*. Gulf Professional Publishing, Burlington, Massachusetts, 2003.
8. Kletz, T. A. *Lessons from Disaster—How Organisations Have No Memory and Accidents Recur*. Institution of Chemical Engineers, Rugby, UK, 1993.
9. Leveson, N. G. High pressure steam engines and computer software. *IEEE Computer 27*, 10 (October 1994), 65–73.
10. Postage meter approval protection profile, 30 April 2001. CLEF.25885.40.1.
11. Richelson, J. T. *The US Intelligence Community*, 5th ed. Westview Press, 2465 Central Avenue, Boulder, Colorado 80301-2877 USA, 2007.
12. Tipton, H. F., and Krause, M., Eds. *Information Security Management Handbook*, fifth ed., vol. 2. CRC Press, 2005.

The Weakest Link: Security in Human Interactive Security Protocols.

Ronald Kainda

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Ad hoc networks of mobile devices have posed challenges never seen before in conventional wired networks. In conventional wired networks, the use of trusted third parties or Public Key Infrastructure (PKI) has worked quite well. However, in ad hoc networks of mobile devices, there is no PKI that can cover all devices and scenarios at a reasonable cost[6]. In addition, these networks are usually created among devices that share no secrets nor know each other's identity.

The lack of a PKI that is both sufficiently universal and sufficiently available to cover all scenarios or presence of trusted third parties holding pre-shared secrets in ad hoc networks has forced researchers to look for alternative ways in which secure device association can be achieved in such environments. A common denominator among the proposed alternatives is the use of an out-of-band (OOB) or empirical channel [5].

The use of two channels has been proposed: one is a high bandwidth channel which is subject to the Dolev-Yao attacker model[1] [1] where messages that are public are exchanged between associating devices. The second channel is a low bandwidth out-of-band or *empirical* channel, which is not subject to the Dolev-Yao attacker model, through which messages that need integrity and or secrecy are exchanged.

On the normal channel, usually public keys are exchanged. These keys could be long term keys or ephemeral depending on the application. Assuming that the information exchanged on the normal channels are public keys, either one device whose key needs to be authenticated sends its public key to the other device(s) or all devices involved exchange their public keys.

The requirement on the empirical channel, that it should not be subject to the Dolev-Yao model of attack, has attracted much attention directed at finding ways in which it could be achieved. One proposal is based on the fact that two interacting human beings have a certain level of trust between them even though their devices have no previous association. In order to bootstrap security on the devices, the already existing human trust can be transferred to their devices. To this regard, humans are required to transfer information between devices that they identify as required to establish an association.

However, these proposals suffer from what will be termed here as the *weakest link problem*. Humans have, for some time now, been identified as the weakest link in a security chain [7, 8]. This is because, in many instances, security is not a primary goal for users [9] and any attempt to distract users from their primary goal in order to 'do security' may result in security being ignored or done only to get the primary goals achieved. Essentially, any security proposal that exerts mental and or physical workload on human users is likely to suffer from the weakest link problem.

The major challenge on the empirical channel is finding a trade off that achieves the required level of security for the amount of effort human users put into it. Examples of some of the proposals of implementing an empirical channels include comparing short strings [2, 3], and using an auxiliary device such as a camera phone [4] to transfer data between devices.

[1] Under the model, the attacker has control over the network; he can overhear, block, modify or insert messages on the channel.

However, these proposals not only demand human effort but also high degree of attention from users in order to achieve secure device associations. As a result some of these proposals are vulnerable to insecure actions from users, some demand too much effort from users such that they risk not being accepted in daily use while some are promising.

A usability study comparing some of the proposed methods has revealed some serious challenges to these proposals. These challenges are not only usability, but also security. They are as a result of the failure to design protocols around would be human users.

The study was conducted by recruiting participants to use a prototype of a mobile peer-to-peer payment system running on two mobile phones. Each participant interacted with all methods in the study. Objective data was collected by logging the number of errors and completion times while subjective data was collected through questionnaires and interviews.

Contrary to what most proponents of various methods claim, the results show that many of these methods are an 'added complication' that could result in usability problems and or security failures. Some of these methods, however, offer some strengths that cannot be ignored hence requires improvements and further investigations. Laboratory user studies have a weakness in that they may not reflect what is possible in a real world setting. However, this turned out to be one of the strengths of this study since in real world applications there are more challenges and if these methods can show weaknesses in an environment with fewer challenges then more could be found otherwise.

Future work will focus on developing a framework under which empirical channels that are both usable and secure may be developed. A number of factors that could help in developing this framework have already been identified. The major challenge is that the amount of data that has to be transmitted on the empirical channel entirely depends on the design of a particular protocol and as such no single empirical channel method may cover all protocols but rather a framework that would help decide what to use is necessary.

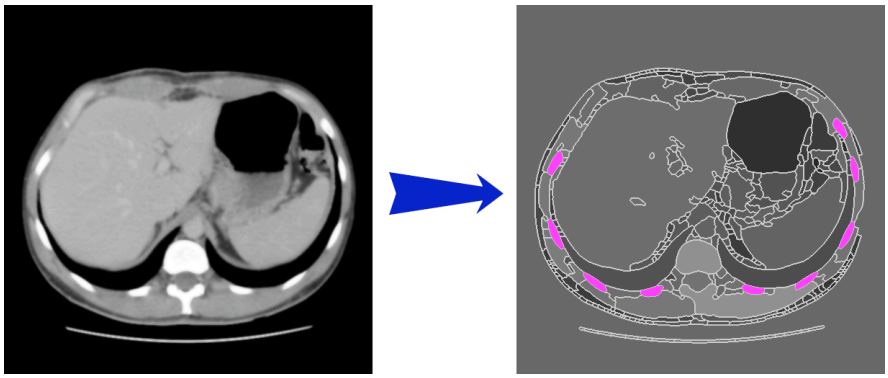
References

1. Dolev, D., and Yao, A. On the security of public key protocols. *Information Theory, IEEE Transactions on 29*, 2 (Mar 1983), 198–208.
2. Gehrmann, C., Mitchell, C. J., and Nyberg, K. Manual authentication for wireless devices. In *RSA Cryptobytes* (Spring 2004), vol. 7(1), RSA Security, pp. 29–37.
3. Goodrich, M., Sirivianos, M., Solis, J., Tsudik, G., and Uzun, E. Loud and clear: Human-verifiable authentication based on audio. In *Proc. 26th IEEE International Conference on Distributed Computing Systems ICDCS 2006* (04–07 July 2006), pp. 10–10.
4. McCune, J., Perrig, A., and Reiter, M. Seeing-is-believing: using camera phones for human-verifiable authentication. In *Proc. IEEE Symposium on Security and Privacy* (8–11 May 2005), pp. 110–124.
5. Roscoe, A. W. Human-centred computer security. Unpublished draft, 2006.
6. Sadie Creese, Michael Goldsmith, R. H. P. W. W. A. R., and Zakiuddin, I. Exploiting empirical engagement in authentication protocol design. In *Proceedings of SPPC* (2005).
7. Sasse, M. A., Brostoff, S., and Weirich, D. Transforming the 'weakest link' — a human/computer interaction approach to usable and effective security. *BT Technology Journal 19*, 3 (2001), 122–131.
8. Schneier, B. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc., New York, NY, USA, 2000.
9. Whitten, A., and Tygar, J. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th USENIX Security Symposium, August 1999, Washington* (1999), pp. 169–183.

The Use of Image Partition Forests in the Analysis of Abdominal CT Scans

Stuart Golodetz, Irina Voiculescu and Stephen Cameron

Oxford University Computing Laboratory



Recent advances in the ability to sense the inside of the human body have provided doctors with a plethora of useful information with which to make their decisions. In particular, techniques such as those based on ultrasound imaging, magnetic resonance imaging (MRI) and computerised tomography (CT) are capable of providing high-resolution images that differentiate tissue density, and so despite the high cost of the equipment such scans are becoming routine (at least in the developed world). These, however, provide only a limited amount of information regarding the 'state' of a particular patient's situation, because only a few scans are taken for each patient, at discrete points in time, and it is difficult to infer how a tumour evolved in the time elapsed between the scans.

We are currently focusing on images of patients with renal (kidney) tumours, but we hope that our work may ultimately have applications to other types of tumour as well. The long-term goal of our work is to investigate the extent to which it is possible to interpolate between, and extrapolate from, these 'snapshots' into a patient's timeline, with the intentions of (a) better understanding what is happening in the periods when no scans are being taken and (b) trying to predict how a tumour may grow in the future, something which may ultimately have implications for the treatment decisions doctors make for individual patients.

There are a number of key technical processes necessary for this project, which can loosely be partitioned under the two main headings of image analysis and growth modelling. Growth modelling involves making predictions about the state of a tumour at points in time for which scans are not available. Image analysis provides the input to the growth modelling process, theoretically allowing generic models of tumour growth to be adapted to individual patients. Our initial work has been focused on image analysis, whilst we acquire the data necessary for growth modelling. In order to provide useful input to the growth modeller, a key first task is to segment the images. Segmentation (when coupled with feature identification) is the process of finding regions with semantic meaning in an image; for instance, we might identify a particular image region as showing a kidney. Our goal is to partially automate this

process to help the users, while at the same time recognising their need to be in charge of the process. Segmentation is difficult to automate in the case of medical images because the shapes of interesting features can vary significantly from one image to the next. Moreover, the boundaries between features are often unclear: for example, whilst a radiologist can easily tell the difference between normal parenchyma and cancerous tissue, automatically tracing the precise boundary is a much more difficult task.

One approach to the problem initially segments the image using the watershed transform, which is a common image processing technique from the field of mathematical morphology. This works by treating the image as a 3D surface, with the (x,y) coordinates of the image pixel giving its landscape 'grid reference' and the grey value giving its height. The landscape is divided into valleys (one per local minimum in the image), thereby partitioning the original image. The watershed tends to over-segment images, however, so a standard approach is to then use region merging to reduce the region count to the desired level.

The waterfall transform is a multi-pass, hierarchical algorithm designed for expressly this purpose. It naturally produces a partition hierarchy of the image, in which each node represents a region of the image. This is variously known as a partition tree, picture tree or semantic segmentation tree approach (although these differ somewhat in the details, they are all based on the same basic principle). Each node of the tree can be annotated with useful properties for its corresponding region, and these properties can then be used when searching the tree to classify a region as being a particular feature of interest.

This is an interesting overall approach which would work well provided that the features we're interested in really did correspond to individual regions in the tree; unfortunately this is not always the case. The problem is in how the tree is constructed: it is common to merge together regions which satisfy some similarity criterion (e.g. difference of mean grey levels within a suitable tolerance). This is a reasonable first approximation, but provides no opportunity to bring crucial anatomical knowledge to bear on the problem.

Our novel contribution in this area is in the development of algorithms which allow anatomically-based modifications to be made to the hierarchy at a later stage, i.e. after the original segmentation is completed, thus providing a potential solution to the feature identification problem. In particular, we have devised algorithms which maintain a valid partition hierarchy whilst allowing the following changes to be made:

- Disconnecting/deleting a subtree (used, for instance, to remove parts of the hierarchy from further consideration when identifying a region as being a feature of interest)
- Moving a subtree from being the child of one parent to being that of another node in the parent's layer
- Splitting a region
- Merging a contiguous group of regions into a larger region

Our work on using these algorithms to help identify features in the image automatically is on-going, but we have achieved some promising initial results for the automatic identification of the ribs. These are an excellent initial target for a region-based segmentation algorithm, because the regions corresponding to them in the tree have several distinguishing properties (very high grey values, medium elongatedness and smallish size). We use a Bayesian classifier to quantify the extent to which a region resembles a rib, and hope to solidify this method (the probabilities involved are currently defined empirically, whereas we would prefer to use a training set of images) and extend it to other features of interest in the near future, potentially using different subsets of relevant region properties in each case. We will present some results of our method on a number of different CT slices, courtesy of the Churchill Hospital, Oxford. Our images will show the results of automatically highlighting the ribs using the methods described above.

Belief Propagation in Fuzzy Bayesian Networks
A Worked Example*

Christopher Fogelberg

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Fuzzy Bayesian networks (FBN) are a graphical machine learning model representation with variables which are simultaneously fuzzy and uncertain[2].

Bayesian networks (BN) are commonly used in machine learning. This is due to their statistical rationality, capacity for rigorous causal inference, and robustness in the face of noisy, partially missing and realistic data. They are also more easily human-interpretable than other machine learning representations such as neural networks, and experts can specify prior knowledge in a principled manner to guide the machine learning search. A wide range of search algorithms have been developed for structural and parameter inference, including *structural EM*[3] and *MCMC*. Classically, Bayesian networks use continuous (Gaussian) or multinomial variables.

Similarly, a fuzzy model has a wide range of advantages. Fuzzy models are also robust in the face of noise-corrupted data. The use of linguistic terms aids human comprehension of the learnt model, and they are particularly useful when the data is insufficient to formulate a precise model. The need to specify membership functions also forces the designer to consider the semantic interpretation of the model parameterisation and construction more explicitly.

For these reasons, FBN (which combine these advantages) may be useful. Theoretical analysis in current research[1] indicates that fuzzy variables can be more expressive than multinomial or continuous variables. FBN may also be used as part of an integrated sequence of machine learning techniques that include reversible dimensionality reduction techniques such as fuzzy cover clustering algorithms. This may allow larger problems to be addressed with FBN than with classic BN.

It is important to note that fuzziness and probability are distinct. Consider the difference between a pool of water which is half boiling water and half ice water (fuzziness), and a pool which has a 50% chance of being boiling water and a 50% chance of being ice water (uncertainty). Somebody might dive into the first pool, but no-one would dive into the second. Previous research which has combined fuzziness and BNs has arbitrarily mixed them, using fuzziness as an approximation so that intractable *hybrid Bayesian networks* can be tractably approximated[4, 5]. A hybrid Bayesian network is one which has both multinomial and continuous distributions over its variables.

Although other mathematical research has also considered the rigorous unification of fuzziness and probability[6], ours appears to be unique in that it explicitly maintains the distinction while considering them simultaneously.

Our formalism consists of two key elements. First, there is the notation and conceptual unification. This is based on a fundamental conceptualisation of variables as entities which specify dimensional spaces. This leads to a semantics of fuzzy uncertainty.

Secondly, there are the mechanics of belief propagation in FBN. In most respects, FBN are identical to classic BN. In particular, they have the same structure and parameterisation. This means that structural EM[3], the junction-tree algorithm (for belief propagation

* The long version of this paper is available online at: http://syntilect.com/cgf/pubs:fbn-example

on graphs which aren't polytrees) and other algorithms can be used with only trivial modifications which take the differences in FBN belief propagation into account.

Probabilistic FBN belief propagation is often algorithmically intractable. However, except for a few graphs, propagating the fuzzy expected value (calculated using *fuzzy integration*) is only a small factor less efficient than probabilistic propagation in classic BN. Excluding the complexity on these few graphs (which classic BN are also intractable for), propagating the fuzzy expected value does not appear to have any disadvantages when compared to classic probabilistic propagation.

This paper describes the FBN formalisation, and contrasts belief propagation in classic and fuzzy BN. By doing this it reifies FBN belief propagation and makes the formalism easier to understand. Further, it illustrates the advantages and expressivity of FBN in certain situations. We briefly extend this contrast to discuss the interpretation of fuzzy variables, and what it means for a variable to be simultaneously fuzzy and uncertain. Readers are referred to the long version of this paper, the original paper[2] and current research[1] for a more detailed discussion of these questions.

References

1. Fogelberg, C. Fuzziness, probability and fuzzy probability: A conceptual analysis. Current theoretical research., August 2008.
2. Fogelberg, C., Palade, V., and Assheton, P. Belief propagation in fuzzy Bayesian networks. In *1st International Workshop on Combinations of Intelligent Methods and Applications(CIMA) at ECAI'08* (University of Patras, Greece, 22 July 2008), I. Hatzilygeroudis, Ed., pp. 19–24.
3. Friedman, N., Murphy, K., and Russell, S. Learning the structure of dynamic probabilistic networks. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)* (San Francisco, CA, 1998), vol. 14, Morgan Kaufmann, pp. 139–147.
4. Heng, X.-C., and Qin, Z. FPBN: A new formalism for evaluating hybrid Bayesian networks using fuzzy sets and partial least-squares. In *ICIC (2)* (Hefei, China, August 23–26 2005), D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds., vol. 3645 of *Lecture Notes in Computer Science*, Springer, pp. 209–217.
5. Pan, H., and Liu, L. Fuzzy Bayesian networks - a general formalism for representation, inference and learning with hybrid Bayesian networks. *IJPRAI 14*, 7 (2000), 941–962.
6. Zadeh, L. A. Generalized theory of uncertainty (GTU) — principal concepts and ideas. *Computational Statistics & Data Analysis 51*, 1 (2006), 15–46.

Translucent Abstraction

Safe Views through Bidirectional Transformation

Meng Wang and Jeremy Gibbons

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

The *view-update* problem [1] found in database research has been a topic of study for decades. It is essentially about mapping updates on *views* of data back to the original data in a legitimate manner. As an example of the view-update problem, we might have some tree-structured source data, which we project into a linear abstract value through a *get* function. Now we might want to update the abstract value; the challenge then is to come up with a backward function *put*, as the inverse of *get*, to put the updates in the abstract value back into the source value.

The keys to solutions to the view-update problem are *bidirectional transformation techniques*, which allow computations to be carried out in both directions [5, 3, 4, 6, 2]. We could try to derive a backwards version of a transformation that is written in a more general language. Very often, a *get* function from a source to an abstract value

$$get :: S \to V$$

is an abstraction that loses information. It is generally impossible to reconstruct the source with an inverse function. Thus, almost all bidirectional systems try instead to come up with a backward transformation of the form

$$put :: (V, C) \to S$$

where C (known as a *complement*) supplies information that is lost during the transformation and is used to rebuild the source.

A *well-behaved* bidirectional transformation must satisfy a few laws: given that s and (v, s) are in the domains of *get* and *put* respectively,

$$\begin{array}{ll} put\ (get\ s, s) = s & \text{(GetPut)} \\ get\ (put\ (v, s)) = v & \text{(PutGet)} \end{array}$$

Intuitively, the (GetPut) law states that if we *get* an abstract value from a source s and *put* it back without modification, we should get exactly s. The (PutGet) law states that if we *get* from an unmodified saved abstract value, we should *get* back the original abstract value.

There is another law found in the view-update literature, namely the (PutPut) law:

$$put\ (v', put\ (v, s)) = put\ (v', s) \qquad \text{(PutPut)}$$

This states that saved updates can be cancelled by updates on abstract values. However, this law is less relevant to our application of programming language design and, therefore, considered optional in this work.

Independently, a concept of *views* [7] has developed in the area of programming language design (for example, by Wadler). A view is an abstraction of data's actual implementation, which provides a programming interface that is more convenient to use and more robust to changes. With a pair of conversion functions, data can be converted *to* and *from* a view.

Views provide a powerful mechanism, which reconciles abstraction and pattern matching. We term this kind of abstraction *translucent*, in contrast to the transparent nature of algebraic datatypes and the opaque nature of abstract datatypes.

Conventionally, view implementers are required to come up with a pair of conversion functions that are each other's inverses, a condition that is difficult to check and maintain. This non-machine-checkable condition is a serious weakness of the original design of views, which was considered impractical and was never implemented. 'Safe' variants of views have been proposed by many authors since; this is still an active research.

We argue that the problem of views in programming languages is closely related to the view-update problem. Traditionally, research into the view-update problem focuses on preserving data and its associations rather than on structure. This is because most database data is stored in simple and loosely specified structures, the preservation of which is either simple or less important. However, the situation is different in functional languages, where we frequently make use of rich datatypes and write functions that transform between them. As a result, we propose a bidirectional system that focuses on structure preservation.

Specifically, we:

- extend previous bidirectional transformation frameworks by accepting a wider range of programmer-defined *get* functions on general datatypes; a *put* function is automatically generated, so as to be well-typed and law-abiding;
- enlarge the domain of the backward transformation to deal with arbitrary updates on abstract values that fall within the range of the *get* function, through extending the framework with a function $create : V \to S$, in addition to *put*, which is used when the complement value is invalidated by some updates on the abstract value;
- use our bidirectional transformation technique in the field of programming language design, specifically views of datatypes, resulting in a system that is convenient (programmers only need to specify a one-directional transformation), correct (equational reasoning on views can be justified), and safe (poorly designed views are detected at the earliest stage).

A draft paper is available at `http://web.comlab.ox.ac.uk/people/Meng.Wang/`.

References

1. Bancilhon, F., and Spyratos, N. Update semantics of relational views. *ACM Trans. Database Syst. 6*, 4 (1981).
2. Bohannon, A., Foster, J. N., Pierce, B. C., Pilkiewicz, A., and Schmitt, A. Boomerang: Resourceful lenses for string data. In *Principles of Programming Languages* (Jan. 2008).
3. Foster, J. N., Greenwald, M. B., Moore, J. T., Pierce, B. C., and Schmitt, A. Combinators for bi-directional tree transformations: a linguistic approach to the view update problem. In *ACM symposium on Principles of programming languages* (2005).
4. Hu, Z., Mu, S., and Takeichi, M. A programmable editor for developing structured documents based on bidirectional transformations. In *ACM Symposium on Partial Evaluation and Program Manipulation* (2004).
5. Matsuda, K., Hu, Z., Nakano, K., Hamana, M., and Takeichi, M. Bidirectionalization transformation based on automatic derivation of view complement functions. In *ACM International Conference on Functional Programming* (2007).
6. Mu, S., Hu, Z., and Takeichi, M. An algebraic approach to bi-directional updating. In *Asian Symposium on Programming Languages and Systems* (2004).
7. Wadler, P. Views: a way for pattern matching to cohabit with data abstraction. In *Principles of Programming Languages* (1987).

On Global Model Checking Trees generated by Higher Order Recursion Schemes

Christopher Broadbent – joint work with Luke Ong

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

Higher-order recursion schemes are systems of re-write rules on typed *non-terminal symbols*. An *order-n* recursion scheme is one whose non-terminals are all given a type of order less than or equal to n. As such schemes unfold, they leave behind a 'trail' of *terminal symbols* from some set Σ. This trail takes on a tree structure. By considering what the trail looks like as a recursion scheme G unfolds *ad infinitum* we see that G defines a (possibly infinite) tree $[\![G]\!]$ with nodes labelled in Σ.

Owing to the fact that recursion schemes can be viewed as terms in a higher-order functional language, such (infinite) trees are useful in that they can be used to model the behaviour of higher-order programs. This motivates investigating which properties of these trees can be algorithmically verified. The logic known as *the modal μ-calculus* is considered a 'gold standard' for expressive power and Ong has shown that given a recursion scheme G and a μ-calculus sentence ϕ it can be decided whether ϕ holds *at the root* of $[\![G]\!]$ [7]. This problem is referred to as *the Local μ-calculus Model Checking Problem* for $[\![G]\!]$.

A natural extension of this problem is to ask for a finite representation of the *set* of *all* tree nodes at which a μ-calculus sentence holds; this is known as the *Global μ-calculus Model Checking Problem* for $[\![G]\!]$. Our present work focuses on extending Ong's algorithm for the local problem to the global case.

The Local Problem

Theorem 1 (Ong 2006). *Let G be a recursion scheme of order-n. The local μ-calculus model checking problem for $[\![G]\!]$ is n-EXPTIME complete.*

The problem can be reduced to determining whether a so-called *alternating parity tree automaton* (APT) $\mathcal{B}$ accepts $[\![G]\!]$ [3]. This can then be thought of as determining the winner of a game known as a *parity game*.

Ong's proof makes use of a novel method that employs *game semantics* [5]. A rooted finite graph $\mathrm{Gr}(G)$ can be constructed that describes the 'inner workings' of the game semantics of the recursion scheme. An APT $\mathcal{C}$ can then be constructed that *simulates* an 'evaluation' of the game semantics represented by the graph. It does so in such a way that it accepts if and only if $\mathcal{B}$ accepts $[\![G]\!]$. Since $\mathrm{Gr}(G)$ is finite, the model checking problem is thus reduced to solving a finite parity game, which can

be achieved using standard methods. The proof provides a pleasing collaboration between games as in 'game semantics' and games as in 'model checking games'.

The Global Problem We claim that it is always possible to construct a so-called *order-n collapsible pushdown automaton* (n-CPDA) [4] that represents this set.

Theorem 2. *Given an order-n recursion scheme G and a μ-calculus sentence ϕ, we can effectively construct in n-EXPTIME an n-CPDA that accepts precisely those nodes of $[\![G]\!]$ at which ϕ holds.*

The finite parity games used to solve the Local Model Checking Problem are of interest here since their finiteness allows us to compute their 'winning region' for a given player. However, recall that these games are based on an automaton that *simulates* the game semantics of the recursion scheme. This simulation is adequate for the local problem but on its own does not extrapolate to the global question.

On the other hand it is known that an n-CPDA can *actually compute* the requisite game semantics [4]. The problem with these automata is that we do not (yet) know how to compute a suitable finite representation of the 'winning regions' of parity games played over their transition graph. We can nevertheless adapt the 'game semantics computing n-CPDA' such that it 'plays' one of our finite games. This automaton turns out to be precisely what we are looking for.

Our presentation will predominantly focus on explaining the content of Theorem 2. We introduce higher-order recursion schemes and higher-order (collapsible) pushdown automata and their symbiotic relationship [6, 4], particularly in the context of the Global Model Checking Problem [1, 2] and our result. This overview should enable us to conclude by giving a brief account of the main idea behind the proof of Theorem 2.

References

1. A. Carayol, M. Hague, A. Meyer, C.-H. L. Ong, and O. Serre. Winning regions of higher-order pushdown games. In *Proc. LICS*, pages 193–204. IEEE Computer Society, 2008.
2. A. Carayol and M. Slaats. Positional strategies for higher-order pushdown parity games. In *Proc. MFCS*, volume 5162 of *LNCS*, pages 217–228. Springer, 2008.
3. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS*, pages 368–377. IEEE computer society, 1991.
4. M. Hague, A. S. Murawski, C.-H L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *Proc. LICS*. IEEE Computer Society, 2008.
5. M. Hyland and C.-H L. Ong. On full abstraction for PCF: I, II and III. *Information and computation*, 163(2):285–408, 2000.
6. T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In *Proc. FoSSaCS*, volume 2303 of *LNCS*, pages 205–222. Springer, 2002.
7. C.-H L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. LICS*. IEEE Computer Society, 2006.

Timed Logics Meet Process Algebras

Christoph Haase

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
christoph.haase@comlab.ox.ac.uk

Abstract. Ensuring functional correctness of computer systems has always been a holy grail in Computer Science. In this presentation, I am going to give an introduction on how systems whose functional correctness depends on meeting timing constraints can automatically be verified. In this context, I will present Timed Automata as a formalism for the modelling of timed systems and Metric Temporal Logic (MTL) as a representative of a timed logic for specifying properties of timed systems. I will moreover show recent results of my work on the effects on the decidability of MTL extended with operators known from process algebras.

1 Model Checking Timed Systems

As computer systems become ubiquitous, it is more important than ever to ensure that they work as intended. Especially when used in embedded systems, they often take direct responsibility for the health and safety of humans and thus producers of such systems have to guarantee that they contain no errors. Many of these systems can be seen as *timed systems*, as their correct functional behaviour heavily depends on meeting timing constraints. *Model Checking* is a well established technique for ensuring functional correctness of finite state systems. In contrast to other verification techniques, it can *prove* the absence of errors with respect to a specification, *and* checking whether or not a specification holds is performed completely *automatically*. Traditionally, a system is modeled by a finite state automaton whose language defines all possible runs of the system and a specification is given in terms of a formula of some temporal logic. It is then checked whether or not the automata fulfils the formula, i.e., if every run is consistent with the formula. However, the traditional Model Checking formalisms only allow for making qualitative statements about properties of systems, despite the need to reason about quantitative behaviour like meeting timing constraints. To overcome this limitation, *timed automata* have been introduced in [1] by Alur and Dill for the modelling and verification of timed systems, and have shown to be suitable for the verification of real-world applications, see e.g. [4]. Nevertheless, their biggest lack is that the language inclusion problem becomes undecidable if the underlying time model is dense. In recent years however, subclasses of timed automata with a decidable language inclusion problem, as well as decidable timed logics have been studied [3], [6]. They have shown to provide sufficient expressive power for writing and checking commonly used specifications. Having these decidability results as a basis, one can find two directions in current research. On the one hand, useful fragments of timed logics with better computational properties have been identified. On the other hand, it is investigated whether there are extensions of timed logics that add to their expressive power, but still allow for a decidable satisfiability problem.

The interested reader is referred to [1], [2] and [5] for surveys on timed automata and real-time logics.

2 Content of the Presentation

In this presentation, I will start with showing how the traditional Model Checking approach is naturally extended to timed systems in terms of timed automata and timed logics. I will then continue summing up important results on these automata and logics. Lastly, I am going to present results of my recent work on the decidability of Metric Temporal Logic extended with operators known from Process Algebras. The latter are a popular formalism from both the literature and practice that allow for specifying and reasoning about systems. I will show that MTL extended with hiding, renaming or asynchronous interleaving renders the satisfiability problem undecidable. On the positive side, MTL remains decidable when synchronized interleaving or fixpoint operators are added to the logic.

References

1. Alur, R., and Dill, D. L. A theory of timed automata. *Theoretical Computer Science 126*, 2 (1994), 183–235.
2. Bellini, P., Mattolini, R., and Nesi, P. Temporal logics for real-time system specification. *ACM Comput. Surv. 32*, 1 (2000), 12–42.
3. Bouyer, P., Markey, N., Ouaknine, J., and Worrell, J. The cost of punctuality. In *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 109–120.
4. David, A., and Yi, W. Modelling and analysis of a commercial field bus protocol. In *Proceedings of the 12th Euromicro Conference on Real Time Systems* (2000), IEEE Computer Society, pp. 165–172.
5. Henzinger, T. It's about time: Real-time logics reviewed, 1998.
6. Ouaknine, J., and Worrell, J. On the decidability and complexity of metric temporal logic over finite words. *CoRR abs/cs/0702120* (2007).

A Tale of Two Parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search

Yue Zhang
Mansfield College
yue.zhang@mansfield.ox.ac.uk

Dependency graphs describe natural language sentences by the binary relationships between words. Figure 1 illustrates the dependency graph for the sentence "I like playing table-tennis with her". Each arc in the figure represents the relationship between a pair of words, where the arrow points from the modifier word to the head word it modifies. For example, the word "I" is linked to the word "like", showing that it is the subject that modifies the verb. In a dependency graph, there is one and only one word that does not have a head word, and it is called the root word. Every non-root word must have exactly one head word. A dependency graph that does not have any loops is also a dependency tree. The vast majority of sentences in both English and Chinese can be represented by dependency trees.

Dependency parsing is the task of producing the dependency graph for an input sentence. A common simplified form of dependency parsing is projective dependency parsing, which only studies dependency trees. Dependency parsing has recently gained much attention in the natural language processing community. Dependency trees are used by NLP applications such as machine-translation (Shen et al., 2008)

Graph-based (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras et al., 2006) and transition-based (Yamada and Matsumoto, 2003; Nivre et al., 2006) parsing algorithms offer two different approaches to data-driven dependency parsing. Given an input sentence, a graph-based algorithm finds the highest scoring parse tree from all possible outputs, scoring each complete tree, while a transition-based algorithm builds a parse by a sequence of actions, scoring each action individually.

The MSTParser (McDonald and Pereira, 2006) and MaltParser (Nivre et al., 2006) are representative of each approach and gave comparable accuracies in the CoNLL-X shared task (Buchholz and Marsi, 2006). However, they make different types of errors, which can be seen as a reflection of their theoretical differences (McDonald and Nivre, 2007). The graph-based MSTParser has the strength of exact inference, but its choice of features is constrained by the requirement of efficient dynamic programming. The transition-based MaltParser is deterministic, yet its comparatively larger feature range is an advantage. By comparing

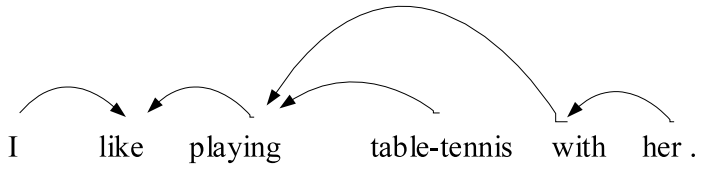


Figure 1: An example dependency tree

the two, three interesting research questions arise: (1) how to increase the flexibility in defining features for graph-based parsing; (2) how to add search to transition-based parsing; and (3) how to combine the two parsing approaches so that the strengths of each are utilised.

We study these questions under one framework: beam-search. Beam-search has been successful in many natural language processing tasks (Koehn et al., 2003; Collins and Roark, 2004), and can achieve accuracy that is close to exact inference. Moreover, a beam-search decoder does not impose restrictions on the search problem in the way that an exact inference decoder typically does, such as requiring the "optimal subproblem" property for dynamic programming, and therefore enables a comparatively wider range of features for a statistical system.

We develop three parsers. Firstly, using the same features as MSTParser, we develop a graph-based parser to examine the accuracy loss from beam-search compared to exact-search, and the accuracy gain from extra features that are hard to encode for exact inference. Our conclusion is that beam-search is a competitive choice for graph-based parsing. Secondly, using the transition actions from MaltParser, we build a transition-based parser and show that search has a positive effect on its accuracy compared to deterministic parsing. Finally, we show that by using a beam-search decoder, we are able to combine graph-based and transition-based parsing into a single system, with the combined system significantly outperforming each individual system. In experiments with the English and Chinese Penn Treebank data, the combined parser gave 92.1% and 86.2% accuracy, respectively. Both accuracies are comparable to the best dependency parsing results for these data sets, while the Chinese accuracy outperforms the previous best reported by 1.8%. In line with previous work on dependency parsing using the Penn Treebank, we focus on projective dependency parsing.

The work has been done with my supervisor Dr. Stephen Clark. For more information about this talk, please refer to our paper (Zhang and Clark, 2008).

References

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164, New York City, USA, June.

Xavier Carreras, Mihai Surdeanu, and Lluis Marquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of CoNLL*, New York City, USA, June.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118, Barcelona, Spain, July.

Philip Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL/HLT*, Edmonton, Canada, May.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP/CoNLL*, pages 122–131, Prague, Czech Republic, June.

R McDonald and F Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *In Proc. of EACL*, pages 81–88, Trento, Italy, April.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, Michigan, June.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225, New York City, USA, June.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*, Nancy, France, April.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*, Hoululu, Hawaii, October.

Query Rewriting under Description Logic Constraints

Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks

Computing Laboratory
University of Oxford
Oxford, UK
{hector.perez-urbina,boris.motik,ian.horrocks}@comlab.ox.ac.uk

1 Introduction

Query answering under constraints is the problem of computing the answers to a query over an incomplete database w.r.t. a set of constraints [7]. Such a problem is relevant in a variety of scenarios, such as information integration [5] and data exchange [4]. Query answering under constraints can be solved via query rewriting: given a query Q over a set of constraints R, one computes a query $\mathsf{rew}(Q, R)$ (which depends on Q and R) such that, for every database instance I, the answers of Q over R and I coincide with the answers of $\mathsf{rew}(Q, R)$ over I.

We consider Description Logic (DL) constraints. DLs are a family of knowledge representation formalisms that represent a given domain in terms of concepts (unary predicates), roles (binary predicates), and individuals (constants) [1]. DL constraints express inclusion dependencies between concepts and roles, and their expressive power is due to the possibility of using complex expressions in the specification of the dependencies. Notably, it is not necessary to impose acyclicity restrictions on DL constraints for query answering to remain decidable. Conjunctive query rewriting under DL constraints has been considered by Calvanese et al. for DLs for which query answering is in LogSpace w.r.t. data complexity [3]. Similarly, Rosati presented a rewriting algorithm for DLs for which query answering is PTime-complete w.r.t. data complexity [6].

Contribution. We present a conjunctive query rewriting algorithm for constraints modeled with the DL $\mathcal{ELHIO}$—one of the most expressive DLs for which query answering remains tractable w.r.t. data complexity. Notably, our technique is *optimal* for the full spectrum of DLs with data complexity of query answering ranging from PTime-complete to LogSpace: if R is expressed in a PTime-complete DL, then $\mathsf{rew}(Q, R)$ is a datalog query; if R is expressed in an NLogSpace-complete DL, then $\mathsf{rew}(Q, R)$ consists of a union of conjunctive queries and a linear datalog query; finally, if R is expressed in a LogSpace DL, then $\mathsf{rew}(Q, R)$ is a union of conjunctive queries. Therefore, our algorithm *generalizes* and *extends* the rewriting techniques of [3] and [6]. Moreover, given the possible forms of the rewritten query, its evaluation can be delegated to existing (deductive) database systems, thus taking advantage of the query optimization strategies provided by such systems.

2 The Algorithm in a Nutshell

We transform Q and R into $\mathsf{rew}(Q, R)$ using Resolution with Free Selection (RFS) [2]. We first obtain a *logic program* LP from Q and R; then, we compute LP_∞ by saturating LP using $\mathcal{R}^{DL}$—our suitably parameterized RFS calculus; afterwards, we transform LP_∞ into a suboptimal *datalog program* D by getting rid of redundant clauses; and finally, we compute an optimal datalog query $\mathsf{rew}(Q, R)$ by unfolding certain types of clauses in D.

Challenges. The expressive power of $\mathcal{ELHIO}$ allows one to state that two different constants denote the same individual. Hence, computing $\mathsf{rew}(Q, \mathbf{R})$ requires reasoning with equality. It is well known that encoding equality using the so-called axiomatization of equality is highly inefficient [2]; therefore, we make use of our own *approximation* of equality, which is part of the logic program LP. Since LP may contain cyclic clauses with functional terms, we must ensure that the saturation of LP by $\mathcal{R}^{DL}$ terminates in general. Moreover, we need to ensure that removing clauses with functional terms or equality in order to obtain D does not affect completeness. Finally, we must show that the last unfolding step always produces an optimal rewriting.

3 Current and Future Work

We have implemented our algorithm in a prototype system called ReQuIEM (Rewriting Queries In Expressive Models). We are currently working on the development of optimization techniques to reduce the size of $\mathsf{rew}(Q, R)$, and reduce/eliminate recursion when possible. We plan to evaluate ReQuIEM in an Information Integration setting.

References

1. F. Baader and W. Nutt. *Basic Description Logics*, chapter 2, pages 47–100. Cambridge University Press, 2003.
2. L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 2, pages 19–100. North Holland, 2001.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. of Automated Reasoning*, 2007.
4. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *ICDT*, volume 2572 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2003.
5. M. Lenzerini. Data Integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM Press.
6. R. Rosati. On conjunctive query answering in EL. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, CEUR-WS, 2007.
7. R. van der Meyden. Logical Approaches to Incomplete Information: A Survey. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer, 1998.

Robotic Search-and-Rescue

An integrated approach

Julian de Hoog[1], Stephen Cameron[1], and Arnoud Visser[2]

[1] Oxford University Computing Laboratory, Parks Road, Oxford OX1 3QD, UK
[2] Intelligent Systems Laboratory, University of Amsterdam, 1098 SJ Amsterdam, the Netherlands

1. Introduction

Recent advances in robotics and computer science mean that it is now possible to use teams of robots for many real-world applications. One very important application is robotic search-and-rescue. Robots are highly suitable for search-and-rescue tasks since they may be deployed in dangerous and toxic environments without putting human responders at risk.

Initial uses of rescue robots, such as during rescue efforts following the 2001 collapse of the World Trade Centre in New York, have highlighted the fact that many improvements are still required if robots are to provide extensive assistance in search-and-rescue scenarios. While much effort is going into development of more robust and mobile robot platforms, it is also necessary to develop advanced methods for multi-robot coordination and practical user interfaces for humans to control the robots.

Here we describe an approach being developed jointly by Amsterdam's Intelligent Systems Laboratory and the Oxford University Computing Laboratory that integrates advanced techniques from a variety of fields such as mapping, localization, exploration, communication, navigation and human-robot interface design. All research reported here is performed using the USARSim simulator[3].

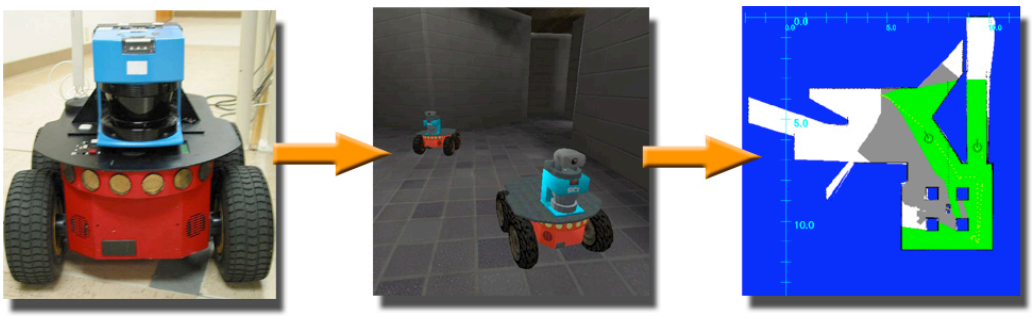


Fig. 1. The simulation process: a real P2AT robot is modeled in the simulator. Teams of simulated P2AT's are then used to test exploration algorithms. The operator only sees the gradually built-up map on the right.

[3] Available for download at http://sourceforge.net/projects/usarsim. Based on a commercial game engine, this simulator uses state of the art techniques for simulating physics and rendering graphics. It is fully configurable, allowing for independent development of simulated robot models, sensors and environments.

2. The Integrated Approach

Mapping and Localization. Our mapping is based on a manifold approach that uses scanmatching and stores the map as a graph structure. Displacements are initially estimated using the inertial navigation sensor, but subsequently corrected by applying a localization and mapping algorithm based on the accurate laser range measurements. Current laser scans are matched with recent laser scans. Once the differences exceed a particular threshold, a new node is created in the graph along with a new link representing displacement. The graph structure may easily be converted into an occupancy grid with standard rendering techniques.

Exploration and Communication. We use a frontier-based exploration approach. As a robot explores, it simultaneously maintains two occupancy grid based maps. The first is based on the maximum sensing range r_{max} of the laser range scanner and the second is based on a more conservative safety distance r_{safe}. "Frontiers" are the boundaries between the safe space and the open space. Robots can then choose which frontier is most suitable for exploration by evaluating a combination of distance to that frontier (i.e. the *movement cost*), potential area to be explored beyond that frontier (i.e. the *information gain*), and likelihood of being able to communicate with the human operator from the frontier (i.e. the *communication likelihood*). This latter value is estimated by maintaining a table of distance - signal strength value pairs and extrapolating to the location of interest.

Navigation. Once an optimal assignment of robots to frontiers has been determined, a path may be planned for each robot to its assigned frontier. This is performed by convoluting the obstacles in the occupancy grid with the shape of the robot using a Gaussian convolution kernel, and using breadth-first search to determine whether a path exists. Way-points along the calculated path are used to guide the robot to its goal.

Human-Robot Interface Design. Control of even a single robot can require extensive operator resources, so proper interface design is crucial to a successful robotic search-and-rescue effort. In our approach the human operator may dynamically assign individual robots a variety of behaviours, from complete teleoperation to complete autonomous exploration. As a result attention may be paid to areas or situations that require it most.

3. Future research

The probability of communication success is built into the exploration algorithm. However, in certain situations significant areas of interest may be out of range of the human operator so communication remains a major issue. To solve this we hope to implement a *territorial exploration* approach: while some robots explore the far reaches of the environment, others are dynamically assigned particular territories within which they behave as information relays. Multi-hop communication has been applied with success elsewhere, but generally not to more than four robots at a time. We hope to experiment with greater numbers of robots.

Simulation research must proceed in step with hardware development and engineering research. As more robust and mobile robot sensors and models become available, control interfaces for these must be developed and incorporated into the already existing framework. In the long term, simulation research reported here is only of use if it is similarly successful in the real world. Applying our integrated approach to a team of real robots would provide an excellent opportunity for validation and evaluation.

Model-Driven Development of Relational Databases

Chen-Wei Wang

Oxford University Computing Laboratory

Methodology of *Object-Orientation* (*OO*) facilitates the systematic classification and modelling of complex business entities, as well as the specification of their associated business rules, i.e. semantic integrity constraints, and behaviours ([8]). On the other hand, technology of *Relational Database Management System* (*RDBMS*) characterises a computerised and mathematical repository ([3]) that promises the data independence and persistence, as well as the second storage and transaction management. Building a safe and stable bridge between these two has been of both industrial and academic interest, as stated recently in [9]: "*Modern applications are built using two very different technologies: object-oriented programming for business logic; and relational databases for data storage. ...ORM (object-relational mapping) is a bridge between the two that allows applications to access relational data in an object-oriented way.*".

An *Information System* (IS) may be characterised by a huge collection of data subject to business rules represented as a set of integrity constraints, and operations upon an IS do not require complicated algorithmic steps; instead, they are a list of primitive updates and retrievals, requiring the preservation of all the existing business rules. My research is to tackle the problem of converting an existing engine, called *Booster*([4][5][6][7][10]) capable of automatically generating a working information system into a *verifying compiler* [2] for SQL relational databases. This conversion is formally justified by proving that the transformation from domain-specific object models into its relational counterparts is a process of *data refinement*[1].

Semantic integrity constraints are specified at the interface Booster object model level as class invariants, and user-specified methos are autoamtically expanded by taking these semantic constraints as well as the association invariants into consideration, through the calculation of the weakest precondition, with respect to the domain-specific implementation strategies from the postconditions. That is, the expanded method preconditions determine the availability of methods. On the one hand, since semantic integrity can be ensured by using Booster as the user interface, RDBMS is used merely as the storage mechanism, skipping all its supported run-time integrity checking before and after each transaction execution, but taking advantage of its runtime ability of handling concurrent transactional executions. And on the other hand, since RDBMS is not proprietary standard and all the semantic integrity rules have been already pushed/decoupled into the generated SQL database, the Booster interface can be turned off on demand. As a result, the database can be used as a stand-alone application, as if it were developed by a native SQL database programmer, until the next time of using Booster to generate new guards, upon evolving requirements. This certainly makes the rich amount of RDBMS facilities all applicable.

Critical part of this research is the formal justification of the model transformation from object into relational. Each instance Booster object model is formalised in the Abstract Machine Notation (AMN) [1], and a restricted form of SQL database is also formalised in AMN,

and hence it is possible to prove, within the same formal framework, that the transformation from object into relation is a process of data refinement. Furturemore, to avoid proving transformations of similar nature each time an instance object model is supplied, another two B machines are specified to describe the generic structures of the object and relaitonal data models. As a result, the *linking invariant* between the these two machines of meta-structures specifies how the transformation should be implemented, and upon proving its preservation by each primitive transaction that intends to modify any association in system, it is then automatically proved of correctness for each instance relational model that the Booster engine generates.

References

1. J.-R. Abrial. *The B-book: assigning programs to meanings*. Cambridge University Press, 1996.
2. J. C. Bicarregui, C. A. R. Hoare, and J. C. P. Woodcock. The verified software repository: a step towards the verifying compiler. *Formal Aspects of Computing*, 18(2):143 – 151, 2006.
3. E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.
4. J. Davies, C. Crichton, E. Crichton, D. Neilson, and I. H. Sørensen. Formality, evolution, and model-driven software engineering. In *Proceedings of the Brazilian Symposium on Formal Methods (SBMF 2004)*, volume 130, pages 39–55, May 2005. Electronic Notes in Theoretical Computer Science.
5. J. Davies, D. Faitelson, and J. Welch. Domain-specific semantics and data refinement of object models. In *Brazilian Symposium on Formal Methods (SBMF)*, 2006.
6. J. Davies, J. Welch, A. Cavarra, and E. Crichton. On the generation of object databases using booster. In *ICECCS '06: Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems*, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.
7. D. Faitelson, J. Welch, and J. Davies. From predicates to programs: The semantics of a method language. *Electronic Notes in Theoretical Computer Science*, 184:171–187, 2007.
8. B. Meyer. *Object-Oriented Software Construction*. Prentice Hall PTR, 2nd edition, March 2000.
9. C. Russel. Bridging the object-relational divide. *ACM Queue*, 6(3), May/June 2008.
10. J. Welch, D. Faitelson, and J. Davies. Automatic maintenance of association invariants. *Software and Systems Modeling*, 2008.

Index