Computing Science

PROCEEDINGS OF THE OXFORD UNIVERSITY COMPUTING LABORATORY STUDENT CONFERENCE 2010

Programme Committee: Sara-Jane Dunn, Joe Loughry, Ivan Lubenko (chair), Huy Vu

CS-RR-10-22

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

Contents

Session 1: Verification **5**

1.1 Cost Monadic Logic
Michael Vanden Boom 5

1.2 Alternating Timed Automata over Bounded Time
Mark Jenkins 7

1.3 Compositional Verification of Probabilistic Systems using Learning
Lu Feng 9

1.4 Automated Translation of Timed Automata to Tock CSP
Maneesh Khattri 11

Session 2: Computational Biology **13**

2.1 A model of cardiac defibrillation in the human heart
Miguel Bernabeu, Mikael Wallman and Blanca Rodriguez 13

2.2 Inference of Electrophysiological Parameter Values Based on Endocardial Mapping Data
Mikael Wallman 15

2.3 Towards a Colonic Crypt Model With Realistic, Deformable Geometry
Sara-Jane Dunn 17

2.4 Multi-Feature Identification in Abdominal CT Scans using Image Partition Forests
Stuart Golodetz, Irina Voiculescu and Stephen Cameron 19

Session 3: Software Engineering **21**

3.1 Complexity of Higher-Order Queries
Huy Vu 21

3.2 Principles for designing Out-Of-Band channels in Human-Interactive Security Protocols
Ronald Kainda 23

3.3 Local consistency and SAT-solvers on the example of chains of inequalities
Justyna Petke 25

3.4 ghttp: a Goroutine-based Web Server Toolkit
James Whitehead II 27

Session 4: Linguistics **29**

4.1 Rich morphology as a challenge to current statistical machine translation systems
Jan A. Botha 29

4.2 Can we learn from a text without understanding the words?
Pia-Ramona Wojtinnek 31

Session 5: Programming Languages **33**

5.1 Accelerating Dynamic Programming algorithms using Massively Parallel Hardware
Luke Cartey 33
5.2 Navigating the Deep Web with OXPath
Andrew Sellers 35

Foreword

The annual Oxford University Computing Laboratory student conference is a forum for the presentation of the graduate research work in the department. It is clear that the conference has now established itself as an annual institution and we are continuing to widen its scope to include the breadth and depth of the research in the Computing Laboratory. This year, in particular, submissions from the Computational Biology group justify this group having a dedicated session for the first time at the conference.

The conference is an increasing success, with 2010 featuring a large number of high-quality submissions from most research groups in the laboratory and a 100% increase in submissions from previous years. The 16 abstracts within these proceedings were selected from a very competitive round of peer-reviewing by both DPhil students and academic staff. The programme committee compiled this interesting programme, not without having to make some difficult decisions.

For some presenters, the conference offers an invaluable opportunity to preliminarily share results that will later be presented at international conferences, workshops and symposia. Conversely for others, having successfully navigated the peer review process for the first time, this will be a new experience. We hope that all the presenters, and you, the audience, will find this a useful and far reaching day.

This day would not have been possible without the assistance of many. Firstly, we extend our gratitude to our referees for giving up their time to contribute valuable feedback on all the submissions. Furthermore, we are grateful for the generosity of the Computing Laboratory in sponsoring the conference and Keble College for hosting us today. We would also like to thank Dr Mike Spivey for agreeing to deliver the conference keynote speech. Finally, on behalf of the whole committee, I would like to thank the presenters, whose work has allowed us to put together such a diverse and interesting programme.

Ivans Lubenko
Programme Chair

Organisation

Programme Committee

Ivan Lubenko (chair)
Sara-Jane Dunn
Joe Loughry
Huy Vu

Conference Committee

Christopher Broadbent
Sara-Jane Dunn
Shamal Faily
Tom Harper

Steering Committee

Marta Kwiatkowska (Honorary Chair)
Shamal Faily
John Lyle

Referees

Christopher Broadbent, Alastair Donaldson, Sara-Jane Dunn, Shamal Faily, Ivan Flechais, Matthew Hague, Ralf Hinze, David Hopkins, Peter Jeavons, Ronald Kainda, Joe Loughry, Ivan Lubenko, Thomas Lukasiewicz, John Lyle, Andrzej Murawski, Andrew Markham, Pras Pathmanathan, Mike Spivey, Jamie Vicary, Huy Vu, Jackie Wang, Jim Whitehead, Jonathan Whiteley, James Worrell

Cost Monadic Logic

M. Vanden Boom

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

A fundamental result in the theory of regular languages is the equivalence of monadic second-order logic (MSO) and finite automata. In the classical setting, a logical sentence is true or false, and an automaton either accepts or rejects the input. Recently, however, Colcombet [4] has proposed a quantitative extension to MSO called *cost monadic logic* (cost MSO). In this setting, a cost MSO sentence φ defines a function $[\![\varphi]\!]$ from some domain D under consideration (e.g. finite/infinite words or trees) to $\mathbb{N} \cup \{\infty\}$. Informally, this means that a cost MSO sentence can count some behaviour within the input structure, e.g. the number of positions labelled with a particular symbol.

Over finite words [4, 3] and finite trees [6], Colcombet and Löding showed that a sentence of cost monadic logic can be approximated by a *cost automaton*, a non-deterministic finite automaton enhanced with a finite set of counters which are initially assigned value 0 and are then updated at each transition using counter operations increment, reset, and no change. Unlike so-called "counter automata", the value of the counters cannot be used to affect control flow. Instead, the counters are used to assign a value to each structure. For instance, over finite words the value of a run of a cost automaton is the maximum value achieved by any counter during the course of the run, and the cost of the word is the minimal value across all accepting runs. If there are no accepting runs, then the word is assigned the value ∞. Historically, there have been a number of related automaton models including weighted automata [12], distance automata [7], nested distance-desert automata [9], R-automata [1], and ωBS-automata [2].

A result from [10] implies that it is undecidable whether two *regular cost functions* (functions definable by cost automata or cost MSO) define exactly the same function. However, Colcombet introduced in [3] an equivalence relation $\approx$ defined as follows: for functions $f, g : D \to \mathbb{N} \cup \{\omega\}$, we say $f \approx g$ iff for all $X \subseteq D$, $f(X)$ is bounded iff $g(X)$ is bounded. In other words $f \approx g$ means that f and g share similar boundedness properties (even if the particular values do not agree). For example, if D is the domain of finite words over the alphabet $\{a, b\}$, the function $f(u) = |u|_a$ which computes the number of a's in the word u is $\approx$-equivalent to $g(u) = 2 \cdot |u|_a$, but not $\approx$-equivalent to $g'(u) = |u|_b$ which computes the number of b's (this is witnessed by the set $X = \{b^n : n \in \mathbb{N}\}$ which is mapped to 0 by f and g, but has unbounded output via g'). For regular cost functions f, g over finite words or finite trees, Colcombet and Löding showed that it is decidable whether or not $f \approx g$. Thus, by weakening the relation from strict equality to this "boundedness preserving relation" $\approx$, we can regain decidability while still retaining some useful applications.

The applications of this logic and automaton model arise in problems related to the existence of bounds. Indeed, one of the earliest motivations for studying this type of automaton came from work on the "star-height problem": given some regular language L of finite words and natural number k, decide whether L can be represented by a regular expression which has at most k nestings of Kleene star operations. Hashiguchi [8] and then Kirsten [9] used automata with counters to prove this problem decidable. Colcombet and Löding [5] have used these cost automata to prove the decidability of the star-height problem for regular languages of finite trees as well. A challenging open question in this area is whether this result can be extended to the case of infinite trees. Thus, a greater understanding of cost MSO and cost automata over infinite trees is desirable.

In recent unpublished work, I have studied *weak cost monadic logic* over infinite trees. This logic is the natural restriction of cost MSO when second-order quantification ranges only over finite sets of nodes. Utilizing results from the finite tree case in [6], I have shown the following automata-logic connection:

Theorem 1. *A cost function over infinite trees is definable in weak cost MSO iff it is definable by an alternating weak cost automaton.*

Not surprisingly, an alternating weak cost automaton is an alternating weak automaton enhanced with counting features. The "weakness" requirement here implies a partition of the state-set and ordering of these partitions which are respected by the priority and transition functions (as in the classical case). The acceptance condition requires each path to stabilize in an accepting partition, and the value of an accepted tree is the maximum of the counter values achieved on any path through the run. Decidability of the $\approx$-relation for cost functions definable in weak cost MSO can then be shown using similar techniques as in [6].

Early work suggests that other classical results can also be adapted to weak cost MSO. For instance, Rabin [11] showed the separation of full MSO and weak MSO over infinite trees; a similar result holds for weak cost MSO.

Proposition 1. *There is a cost function over infinite trees definable in MSO but not definable in weak cost MSO.*

Future work includes comparing weak cost monadic logic and full cost monadic logic over infinite words. Another challenging extension would be to show the equivalence of alternating cost-parity tree automata and full cost MSO over infinite trees. Finally, it would be interesting to identify other problems (like the star-height problem) which can be reduced to questions of boundedness and consequently may admit solutions via cost MSO or cost automata.

References

1. Abdulla, P. A., Krcál, P., and Yi, W. R-automata. In *CONCUR* (2008), vol. 5201 of *Lecture Notes in Computer Science*, Springer, pp. 67–81.
2. Bojanczyk, M., and Colcombet, T. Bounds in ω-regularity. In *LICS* (2006), IEEE Computer Society, pp. 285–296.
3. Colcombet, T. Regular cost functions over words. Manuscript online, 2009.
4. Colcombet, T. Regular cost functions, part I: logic and algebra over words. Submitted, 2009.
5. Colcombet, T., and Löding, C. The nesting-depth of disjunctive mu-calculus. In *CSL* (2008), vol. 5213 of *Lecture Notes in Computer Science*, Springer, pp. 416–430.
6. Colcombet, T., and Löding, C. Regular cost functions over finite trees. In *LICS* (2010). To appear.
7. Hashiguchi, K. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci. 24*, 2 (1982), 233–244.
8. Hashiguchi, K. Relative star height, star height and finite automata with distance functions. In *Formal Properties of Finite Automata and Applications* (1988), J.-E. Pin, Ed., vol. 386 of *Lecture Notes in Computer Science*, Springer, pp. 74–88.
9. Kirsten, D. Distance desert automata and the star height problem. *RAIRO - Theoretical Informatics and Applications 39(3)* (2005), 455–509.
10. Krob, D. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation 4* (1994), 405–425.
11. Rabin, M. O. Weakly definable relations and special automata. In *Mathematical Logic and Foundations of Set Theory (Proc. Internat. Colloq., Jerusalem, 1968)*. North-Holland, Amsterdam, 1970, pp. 1–23.
12. Schützenberger, M. On the definition of a family of automata. *Information and Control 4* (1961), 245–270.

Alternating Timed Automata over Bounded Time*

Mark Jenkins**

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

One of the most critical problems in verification is that of *model checking*, which asks whether a particular system meets a given specification. The specification S is typically formalised in a *temporal logic*, which describes constraints on the desired behaviour of the system such as "resource R may only be accessed by one part of the system at a time" or "each request for access to R must eventually be granted". If the system I is formalised as a structure in which this logic may be interepreted, the problem is then to determine whether $I \models S$, i.e. whether the system I is a model of the specification S.

The simplest model of time for these purposes is the natural numbers $(0, 1, 2, ...)$, which allow us to capture the order of events in the system and discrete delays between them (for example as a number of clock ticks). Over this domain, there are efficient algorithms for model checking which reduce formulas of temporal logics to equivalent *finite state automata*. For a specification S, the language of its equivalent automaton $\mathcal{A}_S$ is the set of all possible behaviours allowed by the specification. The question of whether the system is a model of the specification is then the *language inclusion* problem – whether every behaviour word of the system is allowed by the specification.

This problem can be solved by reformulating it as a simpler problem of *language emptiness* – is there any behaviour word which is possible in the system but forbidden by the specification? Such a translation requires us to find an automaton which captures the complement of the specification (those behaviours which are forbidden), but this is possible because even automata which deal with infinite behaviours (such as Büchi or Muller automata) are closed under complementation.

For many real-world systems, however, this model of time is insufficient. These systems have specifications which require that particular amounts of time elapse between certain events, regardless of the number of events which are triggered in between, such as the brakes of a car being applied within 10ms of the pedal being depressed. For such systems, we need a *dense* model of time such as the non-negative real numbers $\mathbb{R}_+$.

In order to support verification of such real-time systems, *timed automata* were developed by Alur and Dill [1]. These automata augment classical finite automata over infinite words with clocks which can capture timing conditions. These automata have been extensively studied ever since and one of the reasons for their wide adoption is the ability to determine language emptiness for timed automata in PSPACE. However, the dual problem of *universality* – given an automaton, does it accept every behaviour word? – is undecidable for timed automata. This implies that not only is language inclusion also undecidable, but that timed automata cannot be closed under complementation.

In one way, it is simple to rectify this deficiency by using an idea from classical automata known as *alternation*. The nondeterminism of timed automata means that there may be a choice or disjunction of potential next states following a transition; alternation generalises this by allowing conjunction as well as disjunction – some transitions lead to more than one simultaneous next state. *Alternating timed automata* can easily be complemented by exchanging conjunctions and disjunctions and complementing the acceptance condition. The

* The full version of this paper can be found at http://www.comlab.ox.ac.uk/people/mark.jenkins/
** Joint work with Joël Ouaknine, Alex Rabinovich and James Worrell

price to be paid for this generalisation is that language emptiness becomes undecidable, a fact which follows immediately from the undecidability of universality for timed automata.

There have been several attempts to recover from this undecidability in the literature; our approach follows a recent paper of Ouaknine, Rabinovich and Worrell [4] in considering language emptiness only over *bounded time*. This restriction still features dense time with arbitrarily many events and is useful for many real world applications – for example, a run of a security protocol will normally have an *a priori* time bound before it is abandoned. Our work shows that this time bounded language emptiness problem is decidable.

The run of an alternating timed automaton was characterised as a game by Lasota and Walukiewicz [3], and in this context the language emptiness question can be phrased as "Does there exist a winning strategy for one player in this game?". This formulation is very similar to Church's definition of the *synthesis problem*, which he introduced over 50 years ago [2]. The synthesis problem asks whether there exists a causal transformation of a bit-stream (i.e. the output stream has to be generated one bit at a time as the input is read) such that the relationship between the input and output streams satisfies some formula.

The game for alternating timed automata has two important differences to Church's game: the addition of real-time and the concept of parameters to the game (here, parameters represent the word input to the automaton). A game defined over real-time would seem to be novel and whilst the addition of parameters to the Church problem was considered in [5], the *parameter problem* our work solves was not considered. This problem is to determine for which parameter values the first player has a winning strategy for the game; it allows us to solve the language emptiness problem by ascertaining whether any such parameter values exist – these parameter values correspond to words accepted by the automaton.

It is interesting to note that the classical approach to solving temporal logic problems is to translate them into problems concerning automata for reasons of efficiency, yet our work takes the opposite direction. As a consequence of this, while our work shows that this problem is decidable, the complexity of our algorithm is extremely high: a tower of exponentials whose height is proportional to the time bound initially specified (thus the algorithm is nonelementary in general). It is not possible to improve on this, however, as we have established a matching hardness result using a reduction from language emptiness for star-free regular expressions [6].

In conclusion, we have shown that the problem of *Time-Bounded Model Checking* is decidable for alternating timed automata, in contrast to undecidability in the unbounded case. We believe it possible to implement a semi-algorithm which, on practical examples, can avoid the complexity blowup from our hardness result through careful use of abstraction.

References

1. Alur, R., and Dill, D. L. A theory of timed automata. *Theoretical Computer Science 126* (1994), 183–235.
2. Church, A. Applications of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic* (1957), vol. 1, pp. 3–50.
3. Lasota, S., and Walukiewicz, I. Alternating timed automata. In *Proceedings of FOSSACS 05, LNCS 3441* (2005), Springer, pp. 250–265.
4. Ouaknine, J., Rabinovich, A., and Worrell, J. Time-bounded verification. In *CONCUR* (2009), M. Bravetti and G. Zavattaro, Eds., vol. 5710 of *Lecture Notes in Computer Science*, Springer, pp. 496–510.
5. Rabinovich, A. M. Church synthesis problem with parameters. In *CSL* (2006), Z. Ésik, Ed., vol. 4207 of *Lecture Notes in Computer Science*, Springer, pp. 546–561.
6. Stockmeyer, L. J., and Meyer, A. R. Word problems requiring exponential time. In *Proceedings of STOC '73* (New York, NY, USA, 1973), ACM, pp. 1–9.

Compositional Verification of Probabilistic Systems using Learning ⋆

Lu Feng

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

Probabilistic model checking is a powerful formal method to analyse quantitative properties of systems that exhibit stochastic behaviour, such as probabilistic security protocols and randomised distributed algorithms. As with any formal verification technique, the principal challenge of probabilistic model checking is scalability. Faithful models of large-scale complex systems can quickly exceed the capabilities of today's techniques and tools.

A promising direction to improve scalability is the use of *compositional* techniques, where the verification process is decomposed into a separate analysis for each component of the system being verified. A popular approach to this is the *assume-guarantee* paradigm. This paper focuses on a recently proposed *asymmetric* assume-guarantee rule [3] as the following:

$$\frac{\langle true\rangle\ M_1\ \langle A\rangle_{\geq p_A} \qquad \langle A\rangle_{\geq p_A}\ M_2\ \langle G\rangle_{\geq p_G}}{\langle true\rangle\ M_1 \parallel M_2\ \langle G\rangle_{\geq p_G}}$$

where M_1 and M_2 are probabilistic automata [5]. Therefore, the verification of a *probabilistic safety property* $\langle G\rangle_{\geq p_G}$ on a two-component system $M_1 \| M_2$ can be decomposed into two separate problems: (i) checking that M_1 satisfies the assumption A with probability at least p_A; and (ii) checking that, under the assumption that property A holds with probability at least p_A, M_2 is guaranteed to satisfy G with probability at least p_G, denoted $\langle A\rangle_{\geq p_A}\ M_2\ \langle G\rangle_{\geq p_G}$. It is expected that the complexity of verification will be reduced, providing that A has a smaller state space than M_1.

A limitation of some previous work such as [3], however, is that it requires non-trivial manual effort to create appropriate assumptions. In this paper, we address this limitation by proposing a fully automated approach to the generation of such assumptions, based on the well-known L* learning algorithm [1]. Given components M_1, M_2 and probabilistic safety property $\langle G\rangle_{\geq p_G}$, our framework attempts to build an assumption $\langle A\rangle_{\geq p_A}$ that can be used to prove that $M_1 \| M_2$ satisfies $\langle G\rangle_{\geq p_G}$, without constructing the full model $M_1 \| M_2$.

Like in [4], this is done by generating a series of possible assumptions which are analysed by a (probabilistic) model checker. The following is an overview of the learning procedure. First, L* asks several *membership queries*, as to whether a given finite trace t should be included in the assumption A being generated; the responses are guided by checking whether $t \| M_2 \models \langle G\rangle_{\geq p_G}$ holds. Then, L* provides a *conjecture* for A. we use two oracles (corresponding to the two premises of the assume-guarantee rule) to analyse the conjecture: Oracle 1 executes one instance of multi-objective model checking $\langle A\rangle_{\geq ?}\ M_2\ \langle G\rangle_{\geq p_G}$ to determine a rational probability bound p_A for the probabilistic assumption $\langle A\rangle_{\geq p_A}$; Oracle 2 checks whether $\langle A\rangle_{\geq p_A}$ is satisfied on M_1. We use the results of oracles to determine whether it can verify or refute $M_1 \| M_2 \models \langle G\rangle_{\geq p_G}$. If neither is possible, the teacher generates and returns traces to L* based on counterexamples generated during model checking.

⋆ This is an extended abstract of [2], which is a joint work with Prof Marta Kwiatkowska and Dr David Parker.

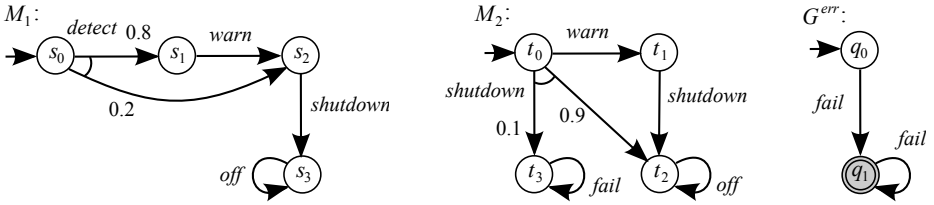


Fig. 1. Two probabilistic automata M_1, M_2 and the complement DFA G^{err} for safety property G .

Example 1. We execute the learning algorithm on probabilistic automata M_1, M_2 and property $\langle G \rangle_{\geq 0.98}$ (Fig. 1) and with alphabet α_A={warn, shutdown, off}. M_1 is a controller that issues a *shutdown* command to a device, modelled by M_2. Prior to this, M_1 tries to send a *warn* signal. If the attempt is not successful, there is a chance that M_2 will not shut down correctly, resulting in a failure.

The first conjectured assumption A_1 (Fig. 2) is passed to Oracle 1, which checks a quantitative query $\langle A_1 \rangle_{\geq ?}\, M_2\, \langle G \rangle_{\geq 0.98}$ yielding the result $\langle A_1 \rangle_{\geq 0.8}$. Intuitively, this means that, under the assumption that a *shutdown* action *never* occurs with probability 0.8 or more, M_2 would satisfy the property. Then, it proceeds to Oracle 2 and check whether $\langle A_1 \rangle_{\geq 0.8}$ holds on M_1. The result is false and a counterexample trace of $\langle warn, shutdown \rangle$ is returned to L*. This results in several more membership queries, after which a second conjecture A (Fig. 2) is produced. Oracle 1 again gives the result $\langle A \rangle_{\geq 0.8}$ but, this time, Oracle 2 confirms that $\langle true \rangle\, M_1\, \langle A \rangle_{\geq 0.8}$. Thus, the framework terminates concluding that $M_1 \| M_2 \models \langle G \rangle_{\geq 0.98}$.

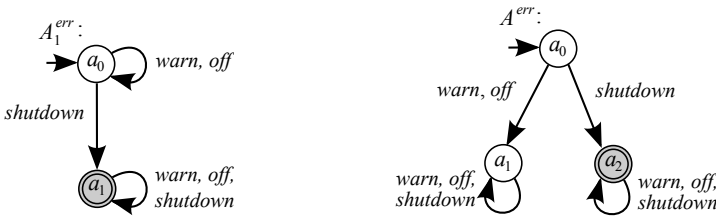


Fig. 2. The DFAs for two generated assumptions A_1, A

We have implemented our technique and successfully applied it to several large case studies (e.g., a randomised consensus algorithm and a sensor network). [1] This includes instances where the entire process of learning an assumption and then applying compositional verification is more efficient than conventional, non-compositional verification.

References

1. Angluin, D. Learning regular sets from queries and counterexamples. *Information and Computation 75*, 2 (1987), 87–106.
2. Feng, L., Kwiatkowska, M., and Parker, D. Compositional verification of probabilistic systems using learning. In *Proc. 7th International Conference on Quantitative Evaluation of Systems (QEST'10)* (2010), IEEE CS Press, pp. 133–142.
3. Kwiatkowska, M., Norman, G., Parker, D., and Qu, H. Assume-guarantee verification for probabilistic systems. In *Proc. TACAS'10* (2010), vol. 6105 of *LNCS*, pp. 23–37.
4. Pasareanu, C., Giannakopoulou, D., Bobaru, M., Cobleigh, J., and Barringer, H. Learning to divide and conquer: Applying the L* algorithm to automate assume-guarantee reasoning. *FMSD 32*, 3 (2008), 175–205.
5. Segala, R. *Modelling and Verification of Randomized Distributed Real Time Systems.* PhD thesis, Massachusetts Institute of Technology, 1995.

[1] All case studies are available online: http://www.prismmodelchecker.org/files/qest10/

Automated Translation of Timed Automata to Tock CSP

Maneesh Khattri

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

Abstract. This paper describes the automated translation of timed automata to tock-CSP. This translation has been implemented in a translator. The tock-CSP output of the translator can be input to FDR for the automated verification of properties of the input timed automata. It has been shown, by the use of the *digitization* technique, that there are relationships between Timed Automata and tock-CSP. Preliminary experiments indicate that this technique is promising for the automated verification of real-time systems and should be further developed.

Real-time systems are commonly used in safety-critical systems. Hence, their correctness has to be guaranteed. Such systems can be analyzed by the application of *testing* or *formal verification methods*. The use of formal verification methods can show that errors are absent in the system and therefore can give the desired guarantee. Usually formal verification requires a specification, an implementation and a technique to determine that the implementation meets the specification. In explicit state *model checking* every reachable state of the model is explored and shown to meet the requirements. Techniques such as partial-order reduction, symmetry reduction, etc., help avoid the exploration of the whole state space. A prominent method to analyse real-time systems is by means of *timed automata* (Alur, Courcoubetis and Dill) and UppAal is a tool for model checking such timed automata.

CSP stands for *Communicating Sequential Processes* [Ros97]. The behavior of CSP processes can be verified using the model checker FDR (failures, divergence, refinement). *Timed CSP* [RR86] is an extension of CSP and is one of the prominent methods to analyse real-time systems. Timed CSP is a timed process algebra, with a continuous model of time. As timed CSP has a continuous model of time there is no direct way to apply the model checker FDR to such models. Hence a discrete timed model was presented in [Ros97] Chapter 14 which had a special event *tock*, which modeled the passage of one unit of time. The *digitization* technique was introduced in [HMP92]. It shows the conditions under which the analysis of a timed automaton with dense time can be performed using only the integral observations. Specifically, for some implementations, which are *closed under digitization*, and specifications, which are *closed under inverse digitization*, the check for correctness using the dense time observations is equivalent to the check for correctness using the integral observations. It has been shown that closed timed automata are closed under digitization and we can show that reachability properties are closed under inverse digitization. Hence we can apply the following translation from timed automata to tock CSP and use FDR to verify properties of timed automata.

Initially we thought that in the translation each clock variable in the timed automaton would be translated to a clock process. Each clock process had a parameter which varied from 0 to the maximum value to which that clock was compared. The value of the clock process could be read, reset to 0 and the clock process engaged in the tock event. A parallel composition of all the clock processes was made with synchronisation on the tock event. Later, an efficiency consideration was to have what we call *Guard Analyser Processes*. Instead

of having explicit clock processes, each one of the atomic clock constraints is analysed by one guard analyser process. The guard analyser process stores the value of a single clock and can communicate the value of the atomic clock constraint at any given time. Tocks can advance the clocks stored within the guard analyser processes and a write action resets the value of the clock in all the guard analyser processes which use that clock. The templates in the UppAal definition are translated to processes in tock CSP. The behavior of the process is modeled using **let** ... **within** (see [Ros97]). In such a CSP process each of the locations is modeled using two local processes where outgoing edges of the location are modeled as prefix processes guarded with the event. All the edges modeled as prefix processes are translated to an external choice (see [Ros97]) between the processes. This is the behavior of the local process (modeling a given location). Once a given event takes place the process behaves as the local process that models the target location. Finally, the local process modeling the initial location is specified using **within**.

As clocks in guard analyser processes are global and any process can reset the value of the clocks the consistency of clock values becomes very important. The reading of clocks should be atomic in the sense that if one has to read two clocks to analyse a guard it should not happen that after reading one clock a tock or write action occurs then the second clock is read. Two processes that synchronise over a channel should have a consistent view of the clock values. Once a visible action occurs the view of the clocks in all processes must be refreshed before any other action can take place. This is necessary as one or more clocks may be reset. Once a visible action occurs the invariants have to be reanalyzed.

Hence the translated tock CSP program has a three phase operation. Initially the whole system is in the *reading phase*. Then the whole system enters the *synchronisation phase*. After an event the system enter the *writing phase*. This is handled by special events *done_read*, *done_write*, and *reset*. All processes synchronise on these events except the clock processes. A monitor process ensures after a visible event the system does a reset so that a writing phase can follow before any other event can take place. UppAal uses a parallel composition similar to that in CCS in which timed automata synchronize on actions that are correspondingly named. Such parallel composition can be translated to CSP using the generalised parallel operator and some renaming. The complete translation is presented in [Kha09] where we have shown, using digitization, that the translation can be used to analyze reachability properties of the input timed automata. Further, we have shown how to handle location invariants, clock constraints, and other constructs in [Kha09]. Our experiments, on examples from the UppAal website, indicate that the FDR approach appears to have a better order of growth in handling more parallel processes, whereas performance of UppAal does not get worse with longer intervals. Preliminary experiments indicate that this technique is promising and should be further developed.

References

[HMP92] T. A. Henzinger, Z. Manna, and A. Pnueli. *What good are digital clocks?*. In Proceedings of the Nineteenth International Colloquium on Automata, Languages, and Programming (ICALP 92), volume 623, pages 545-558. Springer LNCS, 1992.

[Kha09] M. Khattri. *Automated translation of timed automata to timed CSP with clocks and to Tock-CSP*. PRS-DPhil transfer thesis, University of Oxford, 2009. (http://www.comlab.ox.ac.uk/people/Maneesh.Khattri/khattrithesis.ps)

[Ros97] A. W. Roscoe. *The theory and practice of concurrency*. Prentice Hall, 1997, ISBN 0-13-6774409-5.

[RR86] G. M. Reed and A. W. Roscoe. *A timed model for communicating sequential processes*. In proceedings of ICALP 86, pages 314-323. Springer LNCS, 1986.

A model of cardiac defibrillation in the human heart
Abstract

Miguel O. Bernabeu, Mikael Wallman, and Blanca Rodríguez

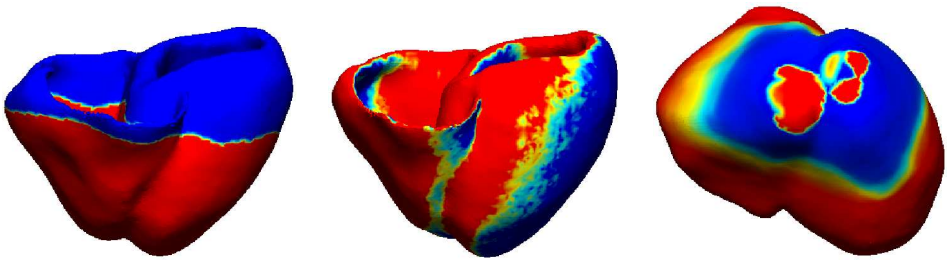
September 18, 2010

1 Background

Cardiac arrhythmic episodes such as ventricular fibrillation (potentially life-threatening) lead to chaotic rhythms and ventricular inability to fulfill the body's demand of oxygen. Electrical defibrillation through the application of strong electric shocks to the heart is the only effective therapy against lethal arrhythmias. Unfortunately, this treatment is not always successful and the mechanisms underlying its failure are still not well understood.

Over the last decades, a large body of theoretical and computational research has tried to unravel the mechanisms underlying shock-induced effects on the heart in an attempt to improve our understanding of defibrillation. However, in most cases, those studies have been performed using rabbit as an animal model and the difference in size between the rabbit and the human heart hampers extrapolation of the results of these studies to clinical practice [4, 1].

In the present work, we present simulations of shock-induced effects on the human heart using novel computational modelling and simulation approaches. A high-resolution anatomically-accurate human ventricular model which incorporates biophysically detailed representation of human membrane kinetics and realistic fibre architecture was constructed. Simulations of ventricular electrophysiology were performed with the open source simulation environment Chaste. Chaste has been developed using professional software engineering techniques and it is designed to be extensible, robust, accurate and maintainable as well as to achieve maximum efficiency in state-of-the-art High Performance Computer infrastructures.



(a) (b) (c)

Figure 1: Shock-induced arrhythmia protocol: (a) tissue after apical stimulus delivered at $t = 0$ ms, (b) tissue at shock-end $t_{coup} = 290$ ms, and (c) spiralwaves appear in the apex at $t = 363$ ms

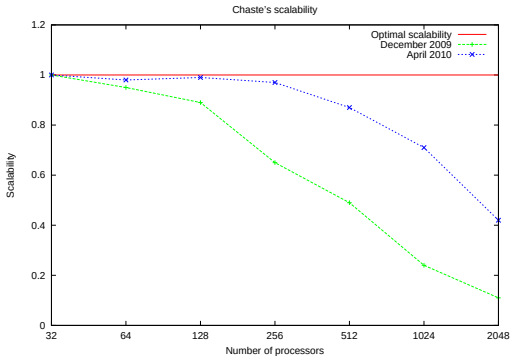


Figure 2: Chaste's parallel scalability for different number of processors. Linear scaling is assumed up to 32 processors.

2 Conclusions

In this paper we present novel modelling and simulation tools for the study of shock-induced arrhythmogenesis in human hearts. A semi-automated software pipeline was developed for the generation of cardiac geometries from volumetric descriptions. The software package Chaste was used for the automatic definition of tissue heterogeneities and fibre architecture, for the implementation of different stimulation protocols, and eventually for conducting the bidomain simulations presented in this work. As in previous studies ([4, 3, 2]), an apical stimulus was applied to elicit propagation, followed by a shock of strength $11 \times 10^{-3} A/cm^3$ delivered at different coupling intervals (t_{coup}). Fig. 1 shows the distribution of transmembrane potentials in the human heart model 166 ms after the apical stimulus was applied (a), at shock end ($t_{coup} = 290ms$) (b), and 65 ms postshock (c).

Improvements to the computational performance and scalability of the simulation software were key to enable the computationally-demanding simulations of shock-induced arrhythmogenesis in the human heart. Fig. 2 shows Chaste's parallel scalability. The parallel performance of the simulator is almost optimal for up to 256 processors, staying over 70% for 1024 processor. This implies that a simulation, like the one described above, taking around 12 hours in commodity desktop with four cores could be run in less than 5 minutes in a supercomputer equipped with 1024 cores.

References

[1] T. Maharaj, R. Blake, N. Trayanova, D. Gavaghan, and B. Rodriguez. The role of transmural ventricular heterogeneities in cardiac vulnerability to electric shocks. *Progress in Biophysics and Molecular Biology*, 96(1-3):321 – 338, 2008. Cardiovascular Physiome.

[2] B. Rodríguez, L. Li, J. Eason, I. Efimov, and N. Trayanova. Differences between left and right ventricular chamber geometry affect cardiac vulnerability to electric shocks. *Circ Res*, 97(2):168–75, 2005.

[3] B. Rodríguez, B. M. Tice, J. C. Eason, F. Aguel, and N. Trayanova. Cardiac vulnerability to electric shocks during phase 1a of acute global ischemia. *Heart Rhythm*, 1(6):695 – 703, 2004.

[4] B. Rodríguez and N. Trayanova. Upper limit of vulnerability in a defibrillation model of the rabbit ventricles. *Journal of Electrocardiology*, 36(Supplement 1):51 – 56, 2003.

Inference of Electrophysiological Parameter Values Based on Endocardial Mapping Data

Mikael Wallman

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

Heart failure is a disabling and potentially deadly condition. In the western world, the incidence of heart failure lies between 6% and 10% for persons over the age of 65. The cost for the health care related to heart failure in Great Britain alone was estimated to 905 million for the year 2000 [5, 4].

Over the recent decades, computer modelling and simulations of the heart have emerged as a complement to experimental investigations for understanding the complex mechanisms underlying cardiac function. Simulations allow virtual experiments that can aid researchers and clinicians in optimising clinical interventions in heart failure patients, such as pacemaker implantations [3]. By personalising these models, the benefits can be further increased.

In order to construct a patient specific model of the heart, parameters governing cardiac function need to be estimated on a case by case basis. The clinical data used to determine the electrical activation pattern of the heart is often sparse, typically consisting of point measurements of the electrical potential on the heart surface, along with a standard electrocardiogram (ECG). This presents two problems: available data must be utilized optimally in order to gain as much information as possible for model building, and a measure of the associated uncertainty must be obtained, since the cost of errors in a clinical setting is high.

In this work, the problems have been simultaneously addressed using Bayesian inference [2], coupled with a very fast method for approximating cardiac electrical activation times. The method for activation time approximation relies on modelling the cardiac tissue as a connected graph. Costs of traversing the edges of the graph are assigned based on the conduction parameters that are being estimated. Activation times are approximated by finding paths through the graph, from the point of initial activation to the points of measurement, using the A* algorithm [1]. The approximated activation times are used in the Bayesian inference to relate electrophysiological measurements to model parameters.

According to Bayes theorem, a probability distribution for a parameter value, s, given a set of measurements, $\{t_i\}$, $i = 1, \ldots, n$, can be expressed as

$$P(s|t_i) = \frac{P(\hat{t_i} - e|s)P(s)}{P(t_i)}, \tag{1}$$

where t is the measured activation time, $\hat{t}$ is the estimated activation time and $e = \hat{t} - t$ is the error of the model. The probability distribution of errors, $P(e)$, used when calculating the term $P(\hat{t_i} - e|s)$ in Equation (1), has been directly calculated from detailed simulations in this work, but will later be estimated using a holdout set of the measured data. By sequentially applying Equation (1) to the measured data, at each iteration substituting $P(s)$ for $P(s|t_i)$ from the previous iteration, probability distributions can be computed, describing the uncertainty about the parameter value of interest, given the data currently available.

For initial validation, several simulation studies have been performed, using increasingly realistic computational models of the cardiac tissue. Activation times calculated from the simulations have been used to validate the method for activation time approximation, as well as the Bayesian inference step. Results show good correlation between estimated and

true parameter values in several simulated scenarios. An example application on data from simulation of a 0.5×0.5 cm tissue sheet, using the method to infer the position of a low conductivity region from four point measurements, is shown in Figure 1. Additionally, the approach has been applied to the problem of optimal sample collection for point measurements on the cardiac surface, rendering results with potential benefits in clinical practice.

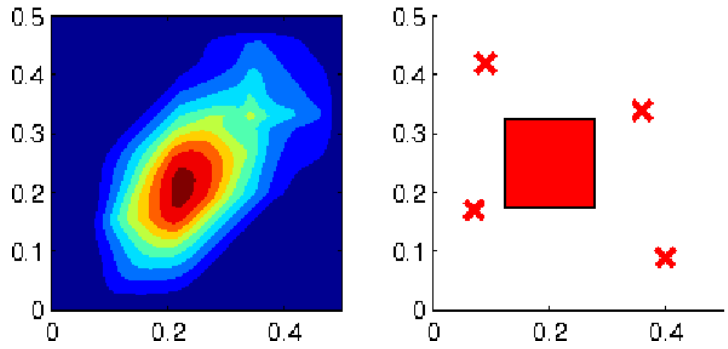


Fig. 1. Example application of the method for parameter inference. Right panel shows a low conductivity region marked by a square, and four point measurements marked by crosses. Left panel shows the Bayesian posterior probability distribution of the position of the low conductivity region, calculated from the four point measurements. All values are in centimetres.

References

1. Hart, P., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on 4*, 2 (1968), 100–107.
2. Jaynes, E. T. *Probability Theory: The Logic of Science*, 1st ed. Cambridge University Press, 2003.
3. Kerckhoffs, R. C., McCulloch, A. D., Omens, J. H., and Mulligan, L. J. Effects of biventricular pacing and scar size in a computational model of the failing heart with left bundle branch block. *Medical Image Analysis 13* (2009), 362–369.
4. McMurray, J. J. V., and Pfeffer, M. A. Heart failure. *Lancet 365* (2005), 1877–1889.
5. Stewart, S., Jenkins, A., Buchan, S., McGuire, A., Capewell, S., and McMurray, J. J. J. V. The current cost of heart failure to the national health service in the uk. *European Journal of Heart Failure 4* (2002), 361–371.

Towards a Colonic Crypt Model With Realistic, Deformable Geometry

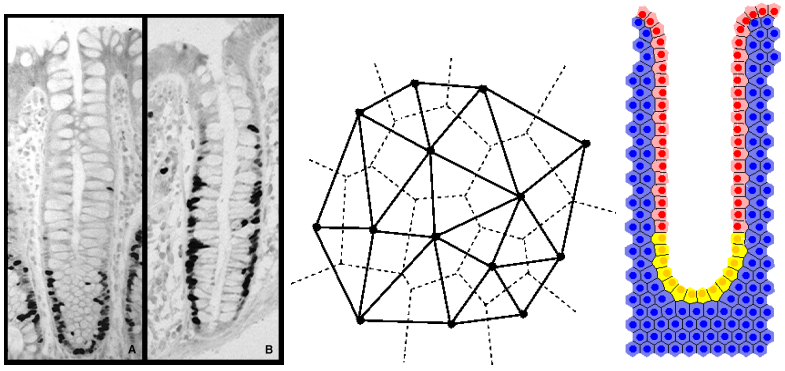
Sara-Jane Dunn

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

Colorectal cancer (CRC) is one of the leading causes of cancer-related death worldwide, demanding a response from scientists and clinicians to understand its aetiology and develop effective treatment. CRC is thought to originate via genetic alterations that cause disruption to the cellular dynamics of the crypts of Lieberkühn, test-tube shaped glands located in the large intestine, which are lined with a monolayer of epithelial cells (see Fig 1(a)) [4]. A delicate balance of cell division, migration and cell death is coordinated in the crypts to renew the epithelial layer every 5-6 days. It is believed that during colorectal carcinogenesis, cells accumulate genetic mutations that destabilise cell-cell contacts and disturb division and migration. The resulting abnormal behaviour can increase stress on the walls of the crypt, causing buckling. Such dysplastic crypts can allow the formation of a benign adenoma if mutated cells do not leave the crypt as they should, but rather persist and proliferate in a localised area. Over time and via accumulated mutations, these growths can progress to a malignant lesion that can break through to the underlying tissue stroma, and so aid metastasis.

The dynamic cell properties that are required to initiate crypt buckling are poorly understood, as it is difficult for biologists to experimentally observe, either *in vivo* or *in vitro*, the initial changes in this sequence of events. Performing *in silico* experiments using an accurate, predictive computational model of the crypt will highlight the necessary conditions required for buckling to occur, and so provide crucial insight into the tissue-level effects of genetic mutations that lead to CRC. Thus, a multi-scale model of the crypt with a realistic, deformable geometry is required. This talk concerns the progress and development of such a model, and its usefulness as a predictive tool to further the understanding of interactions across spatial scales within the context of colorectal cancer.

An off-lattice cell-centre modelling approach is adopted, with cell-cell connectivity defined by a Delaunay triangulation, and cell shape realistically prescribed by the dual Voronoi tessellation [3], concepts illustrated in Fig 1(b). As such, cell centres are defined by nodes that are free to move in space, which are connected to neighbouring cells along the lines of the triangulation. At the subcellular level, a cell cycle model that is dependent on extracellular signalling factors controls cell division, whilst at the cellular level, interactive forces between neighbouring cells are modelled as linear springs [1]. Two stages of model development will be presented, as the model is considered in both two and three dimensions. As a consequence of the results found by conducting simulations of the model in two dimensions, the limitations and possible errors that are introduced within this basic framework justify the advancement to three dimensions, and this will be discussed. The computational framework for this model is based within the Chaste environment [2], an open source software library written in object-oriented C++, created by a team based mainly within the Computational Biology group. Chaste is a general purpose simulation package aimed at multi-scale, computationally demanding problems arising in biology and physiology, developed using agile programming techniques and following software engineering practices. This computational modelling package will be introduced.



(a) (b) (c)

Fig. 1. (a) A histological image of the colonic crypt. (b) Cell-cell connectivity is defined between nodes by a Delaunay triangulation, indicated here by the solid lines, and the corresponding cell shapes are realistically prescribed by the dual Voronoi tessellation, indicated by the dashed lines. (c) An example of how this modelling framework is used to generate a realistic 2D cross-sectional model of the crypt. The epithelial cells are coloured yellow and pink, whilst the surrounding tissue cells are coloured blue.

The novel features of this model include a realistic, deformable geometry, permitting investigation of the mechanisms responsible for the destabilisation of the crypt structure. The realistic geometry is required to accurately track the mechanical restrictions that cells place on each other during division and migration. Further to this, the model takes into account the tissue structure, including the basement membrane and surrounding stroma, which provide crucial structural stability under healthy conditions but which deform as the crypt buckles – an example of this geometry in two dimensions is shown in Fig 1(c). In this talk, the model will be presented, and the conditions required for crypt homeostasis will be reported. With this established, examples of the effect of mutant cell phenotypes, such as reduced migration and altered cell-cell adhesion, will be demonstrated. These simulations illustrate how such a model can be used as a predictive tool. These results will be presented and discussed in the context of an established collaboration with biologists at the University of Dundee, whereby the results obtained enter a feedback loop that advances understanding of this important physiological system.

References

1. Meineke, F., Potten, C., and Loeffler, M. Cell migration and organization in the intestinal crypt using a lattice-free model. *Cell. Prolif. 34* (2001), 253 – 266.
2. Pitt-Francis, J., Pathmanathan, P., Bernabeu, M., Bordas, R., Cooper, J., Fletcher, A., Mirams, G., Murray, P., Osborne, J., Walter, A., Chapman, S., Garny, A., van Leeuwen, I., Maini, P., Rodriguez, B., Waters, S., Whiteley, J., Byrne, H., and Gavaghan, D. Chaste: a test-driven approach to software development for biological modelling. *Comp. Phys. Comm. 180* (2009), 2452 – 2471.
3. Shewchuk, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996, pp. 203–222.
4. Van Leeuwen, I. M. M., Byrne, H. M., Jensen, O. E., and King, J. R. Crypt dynamics and colorectal cancer: advances in mathematical modelling. *Cell Prolif. 39* (2006), 157 – 181.

Multi-Feature Identification in Abdominal CT Scans using Image Partition Forests

Stuart Golodetz, Irina Voiculescu and Stephen Cameron

Oxford University Computing Laboratory

For medics who must decide on the best way to treat patients, it is extremely helpful to be able to calculate the volumes of a patient's internal organs, and visualize the state of those organs in 3D (e.g. using the multiple material marching cubes (M3C) algorithm [5], as shown in Figure 1). However, both these tasks (and indeed numerous other applications in both the medical and other domains) depend on an ability to solve more fundamental problems in image analysis: those of *segmentation* and *feature identification*. These are the related challenges of how to partition an image into pieces in such a way as to ensure that those pieces have some meaning in a given domain, and how to then ascribe meaning to some or all of those pieces. Whilst a huge variety of techniques have been developed for these problems, there are many domains in which it is difficult to specify, at least in a sufficiently precise manner, what constitutes a meaningful piece and what does not. For that reason, both problems remain major research challenges.

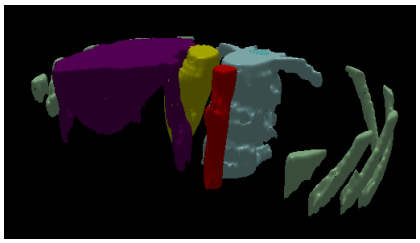
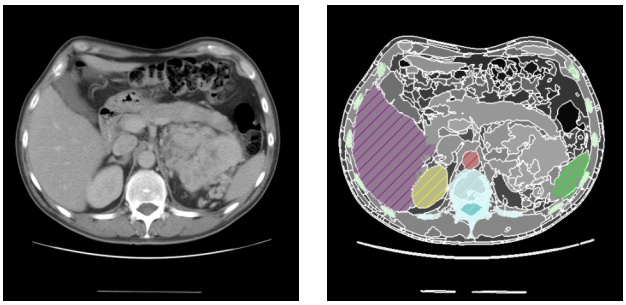


Fig. 1. Visualizing the results of our automatic feature identification algorithms using M3C

In this talk, we will present our methodology and the results of our work on tackling the segmentation and feature identification problems for a particular type of medical image: 3D computerised tomography (CT) scans of the abdomen (see Figure 2(a), which shows an axial (X-Y) slice through such a scan). Great progress has been made in recent years in this area, with some approaches [1, 2] able to identify a number of abdominal features automatically. However, because medical imaging is a domain in which such results are used in the treatment of real people, medics generally want to retain some control over the feature identification process, in particular by being able to view and edit the results. We have thus looked at how feature identification can be made fairly automated, whilst allowing users to make changes easily where necessary.

The approach we have taken is to represent 3D images hierarchically using a data structure we call an *image partition forest* (IPF). This is in essence a hierarchy of adjacency graphs, each of whose nodes represent regions that partition the image. One of these partitions of the image has been visualized in Figure 2(b). This representation is more appropriate for feature identification than the 'usual' pixel/voxel-based image representation, because it is able to represent higher-level structure in the image. It can also be used to facilitate editing of the results, using a set of novel algorithms we have developed for mutating a partition forest (an early version of these algorithms was presented at CSTST '08 [3]).

To identify abdominal features using IPFs, we use a range of techniques. Our method first identifies the spine (a major landmark) by filtering the IPF for nodes that satisfy spine-



(a) An axial slice through a 3D abdominal CT scan

(b) The corresponding slice through the labelled result

Fig. 2. An illustration of the desired results of the automatic feature identification process: (a) shows the type of image we're processing; (b) shows the features we hope to identify (aorta = red, liver = purple, ribs = light green, right kidney = yellow, spinal cord = blue, spine = light blue and spleen = green).

like properties (extends through every axial slice, reasonably white, reasonably large, etc.). Via testing on a number of image series, we have found this to be quite a robust approach. We next use this to identify the spinal cord, ensuring that e.g. it is well within the bounds of the spine, extends through every axial slice and has a reasonably low grey value and a reasonable size. We use a multi-layer region growing approach with post-processing to identify the ribs, again taking the location of the spine into account when doing so.

Identifying blood vessels and organs is more involved, because their grey value distributions are less distinct with respect to the rest of the image. For example, we identify the aorta by first filtering for reasonably-sized regions with a relatively high grey value towards the top-right of the spine. We then find the connected components of the results, pick the one which extends farthest down the image, remove any particularly dark regions and use a graph-based, conditional version of a technique called *morphological closing* [4] to fill in any holes. Our work on identifying soft-tissue organs is on-going, but we will nevertheless present a technique that is relatively successful at identifying the kidneys, and discuss our early approaches to liver and spleen identification.

We have implemented our feature identification algorithms as part of the development of a cross-platform segmentation, feature identification and 3D visualization tool. Whilst further work is still required to increase their overall robustness, we believe the initial results (as illustrated in Figures 1 and 2) are extremely promising. We aim to show our methods running on a real data set, courtesy of the Churchill Hospital, Oxford.

References

1. A Shimizu et al. Segmentation of multiple organs in non-contrast 3D abdominal CT images. *International Journal of Computer-Assisted Radiology and Surgery*, 2(3):135–142, 2007.
2. P Campadelli et al. Automatic Abdominal Organ Segmentation from CT images. *Electronic Letters on Computer Vision and Image Analysis*, 8(1):1–14, 2009.
3. S Golodetz, I Voiculescu, and S Cameron. Region Analysis of Abdominal CT Scans using Image Partition Forests. In *Proceedings of CSTST '08*, pages 432–7, Cergy-Pontoise, October 2008.
4. Rafael C Gonzalez and Richard E Woods. *Digital Image Processing.* Pearson Education, 2nd edition, 2002.
5. Ziji Wu and John M. Sullivan Jr. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, July 2003.

Complexity of Higher-Order Queries ⋆

Huy Vu

Computing Laboratory, Oxford University, Parks Road, Oxford, UK
huy.vu@comlab.ox.ac.uk

Higher-order functions play a fundamental role in computer science; they are critical to functional programs, and in object-oriented programming they play a key role in encapsulation. In database systems they have appeared in isolation at several points: query-transformation plays a role in numerous aspects of databases, including data integration [LRU96], access control [FGK07], and privacy [LHR+10]. In the context of nested-relational and functional query languages, the ability to create and pass functions as data plays a role; the role of higher-order functions is becoming more prominent within XML query languages, with the next version of the XQuery standard offering explicit support for higher-order features [RCDS09].

Still, the combination of database queries and higher-order functions has not been studied in its own right: in prior work within the database community it appears in conjunction with other language features and in restricted settings. Within finite model theory the ability to express database queries within higher-order functions has been studied, but not their combination. In [BPV10], a straightforward framework for combining relational algebra with higher-order functional languages is defined, which we refer to as λ-*embedded query languages*: it is exactly the simply-typed λ-calculus with database operators as "constants" (that is, as built-in functions).

Our prior work [BPV10] focuses on equivalence and containment of terms. In this paper, we give a full picture of the most basic problem concerning terms in higher-order query languages: evaluation of "order 0 terms": terms that evaluate to a database instance. We study this not only for positive relational algebra, but for *any* collection of relational operators, and also consider the impact of higher-order constants that give greater expressiveness, such as fixpoint operators.

We begin by defining the higher order language studied in this work.

- Types: We fix an infinite linearly-ordered set of *attribute names* (or *attributes*). The basic types are the *relational types* each given by a (possibly empty) set of attribute names, $\mathcal{T} = (a_1, \ldots, a_m)$. The order of any relational type is 0.
  We define *higher-order types* by using the functional type constructor: if $\mathcal{T}, \mathcal{T}'$ are types, then $\mathcal{T} \to \mathcal{T}'$ is a type whose order is $\texttt{order}(\mathcal{T} \to \mathcal{T}') = \max\big(\texttt{order}(\mathcal{T})+1, \texttt{order}(\mathcal{T}')\big)$. Order 1 types are often called *query types*.
- Constants: We will fix a set of constants of each type. Database instances are constants of relational type. The operators of Relational Algebra are constants of query type.
- Simply typed terms: Higher-order *terms* are build up from constants and variables by using the operations of abstraction and application:
  - every constant is a term of the constant's type;
  - if X is a variable of type $\mathcal{T}$ and ρ is a term of type $\mathcal{T}'$, then $\lambda X.\, \rho$ is a term of type $\mathcal{T} \to \mathcal{T}'$;
  - τ is a term of type $\mathcal{T} \to \mathcal{T}'$ and ρ is a term of type $\mathcal{T}$, then $\tau(\rho)$ is a term of type $\mathcal{T}'$.

⋆ This is a quick summary of joint work with my supervisor [VB11].

The semantics of terms can be found in [BPV10,VB11].

The *order* of a term τ is the order of its type. The *degree* of τ is the maximum order of its subterms.

After defining the terms, we consider the evaluation problem of terms of "degree 1": those that have variables ranging only over databases. For terms with only positive operators and only database variables, it was shown in [BPV10] that this language is essentially the same as non-recursive Datalog. Here we extend this to the more general setting of all relational operators, and to fixpoint operators, showing equivalence to variants of Datalog. This allows us to read off the complexity of evaluation via reduction to known results about Datalog.

We then extend to the first higher-order case: "degree 2", where terms can have variables ranging over queries. Here we get EXPTIME-complete complexity of evaluation through combining an analysis of the complexity of classical β-reduction with the results on degree 1.

Building on the degree 2 case and a technique inspired by Hillebrand and Kannellakis [HK96], we determine the complexity for general terms. The problem of evaluating degree k terms is m-EXPTIME-complete if $k = 2m$, and is m-EXPSPACE-complete if $k = 2m + 1$. Our results show that the complexity is non-elementary for the general calculus.

Having found the worst-case complexity for general terms, we turn to cases that have lower complexity. For example, we show that the complexity reduces drastically when we restrict the nesting of higher-order variables in terms.

All of these results are for a "strongly-typed" version of the λ-calculus, in which all operators and variables must be annotated with correct types. We look at a weaker "implicitly-typed" version, and show that the complexity does not change, even though we can now build queries in which the arity of the output can grow exponentially with the input database.

In the future we will find a more precise bound in terms of the nesting depth. We will explore the behavior of other built-in query transformation and database operations.

We have initiated a study of weakly-typed terms here; in future work we will consider more powerful operations on indices in weakly-typed terms, with the goal of matching the expressiveness of XML query languages. We will also study the relationship with polymorphic nested relational languages.

Our upper bounds rely on strategies where the interaction between β-reduction and query evaluation is fairly limited. We are not convinced that this is true for practical evaluation strategies, and will be looking at interleaved strategies in our implementation.

References

[BPV10] M. Benedikt, G. Puppis, and H. Vu. Positive Higher Order Queries. In *PODS*, 2010.

[FGK07] W. Fan, F. Geerts, and X. J. A. Kementsietsidis. Rewriting Regular XPath Queries on XML Views. In *ICDE*, 2007.

[HK96] G. Hillebrand and P. Kanellakis. On the expressive power of simply typed and let-polymorphic lambda calculi. In *LICS*, 1996.

[LHR+10] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, 2010.

[LRU96] A. Levy, A. Rajaraman, and J. Ullman. Answering Queries using Limited External Query Processors . In *PODS*, 1996.

[RCDS09] J. Robie, D. Chamberlin, J. Dyck, and J. Snelson. XQuery 1.1.: An XML Query Language , 2009.

[VB11] H. Vu and M. Benedikt. Complexity of higher-order queries. In *ICDT*, 2011.

Principles for designing Out-Of-Band channels in Human-Interactive Security Protocols

Ronald Kainda

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

1 Introduction

Human-Interactive Security Protocols (HISPs) require users to carry out security critical tasks with high degrees of accuracy. Users, however, are usually faced with competing tasks and security is not their primary goal. As a result users are unmotivated to do security tasks with required attention and accuracy. To minimise the possibility of security failures without requiring undue effort from users, Out-Of-Band (OOB) channels must be designed based on a number of principles. Currently proposed methods have inherent weaknesses including forcing designers to choose between security and usability, being limited to specific application contexts, and failing to ensure that users carry out security critical tasks with required attention [1]. This work proposes principles for designing secure and usable OOB channels that, despite being faced with competing tasks, users are able to carry out security critical tasks with required attention and accuracy.

2 Principles for designing OOB channels

Principle of commitment This principle can simply be stated as 'a user is committed to a particular value/action without knowing what the outcome of such a value or action will be'. The outcome of a users' value/action is only revealed after s/he is committed to it. This principle ensures that users do not get their desired outcome when it is not supposed to be the case. The *principle of commitment* requires that a user is provided only with partial information that allows him/her to commit to a final outcome. For example, *manual copying and entering* [2] reveals partial information to a user that he/she uses to make a commitment by entering it into other devices. The user at this stage does not know whether this information will be accepted by other devices but it is up to these devices to determine whether received information is correct or not. By doing so, a user cannot force a device to accept a value that does not match its own digest.

Principle of unpredictability Users tend to learn and master how a system can be used with least effort. Over time users end up achieving their primary goals without consciously engaging with the subtasks involved. This phenomena is known as *habituation* or *user conditioning*. Habituation occurs in activities that have non changing task sequences and following a specific course of actions results in the same outcome every time. Habituation is bad for security because we want users to carry out security tasks consciously and accurately. The principle of *unpredictability* simply states that 'a user should not be able to predict the sequence of actions that lead to a particular outcome'. This principle differs from the principle of commitment in that it focusses on making a sequence of actions unpredictable while the latter makes it difficult for a user to determine an outcome based on a particular action. For example, in web browser Secure Socket Layer (SSL) certificate warnings, users know that they have to take one of the two actions (principle of unpredictability

violated), either accepting or rejecting a certificate and they also know that accepting results in continuing to the intended website (principle of commitment violated).

Principle of single interaction path One secure design principle is to ensure that the path of least resistance is the most secure [3]. For example, Firefox 3.x web browser's implementation of allowing users to add exceptions of invalid, expired, or untrusted SSL certificates requires a user to single-click a rejection of the certificate and at least 4 clicks to accept the same certificate. The problem, however, is that in most instances users' desire to achieve their primary goals outweighs the effort required to accept a certificate. In HISPs, path of least resistance may mean an insecure route —for example, accepting a digest without comparing. Moreover, multiple paths to a single goal is likely to cause users difficulty in understanding an OOB method. The *principle of single interaction path* demands an implementation that provides a single path from start to end of a device association process.

Principle of design by context Systems must be designed to work within the context of operation. This is crucial because different environments pose different challenges on a system. Context may be classified as social, technological, and environmental. This principle refers to designing OOB channels within the context of an application in which they may operate. This requires thinking about specific user interactions, within a specific application context, that may hinder or help usability and security of OOB channels. Understanding the context in which an OOB method will operate is crucial to meeting human and contextual needs. There is a general consensus among researchers that device association process must be 'fast' to complete. However, fast usually means a user must spend as less time as she considers appropriate. This definition of fast neither provides useful information nor does it define the term itself. It may, however, be reasonable to think that a user is likely to measure the appropriateness of time spent on the association process in relation to the time spent on the primary task.

3 Summary and conclusion

The above principles were demonstrated by two proposed OOB channels. The channels were subjected to a usability experiment and results compared to studies of other methods. The comparison showed that the proposed methods are better in terms of effectiveness, efficiency, and user satisfaction. In addition proposed methods are applicable to a wide range of contexts without compromising security (See [1] for details on the methods and study). In summary, HISPs require users to carry out security critical tasks in order to establish secure device association. While device association is not usually a primary task for users, security critical tasks must be carried out correctly. Given users' lack of motivation and the demands of competing tasks, OOB channels must be designed such that users do not compromise the desired security.

References

1. R. Kainda, I. Flechais, and A. Roscoe. *Information Security Theory and Practice. Security and Privacy of Pervasive Systems and Smart Devices*, volume 6033 of *WISTP 2010, Lecture Notes in Computer Sciences*, chapter Secure and Usable Out-Of-Band Channels for Ad hoc Mobile Device Interactions, pages 308–315. Springer, April 2010.
2. R. Kainda, I. Flechais, and A. Roscoe. Two Heads are Better Than One: Security and Usability of Device Associations in Group Scenarios. In *SOUPS '10: Proceedings of the 6th symposium on Usable privacy and security*, 2010.
3. K.-P. Yee. User Interaction Design for Secure Systems. In *ICICS '02: Proceedings of the 4th International Conference on Information and Communications Security*, pages 278–290, London, UK, 2002. Springer-Verlag.

Local consistency and SAT-solvers
on the example of chains of inequalities

Justyna Petke

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

In the quest to find efficient problem solvers, two separate areas of research have developed, namely Constraint Programming and Boolean Satisfiability. The first area concentrates on solving instances of a *Constraint Satisfaction Problem* (CSP) which is defined by a triple (V, D, C) where V is a set of variables, D is a set of domain values for variables in V and C is a set of constraints that put restrictions on variable assignments in V. The problem of deciding whether there is a variable assignment that satisfies a propositional formula is called the *Boolean Satisfiability Problem* (SAT). Even though SAT is a subset of CSP, the usual approaches for solving those two kinds of problems differ [1].

SAT-solvers are considered by many users and researchers to be extremely efficient [1]. Hence some researchers have developed CSP to SAT translations and used a SAT-solver engine for dealing with general CSPs over large domains [4]. Such translations often produce huge SAT instances. (The direct encoding, for instance, introduces a Boolean variable for each possible variable assignment $v = a$.) However, SAT-based solvers still did quite well in CSP solver competitions [6]. In our recent paper [2] Peter Jeavons and I answer the question of why SAT-solvers do so well on families of CSP instances whose satisfiability can be decided by enforcing k-consistency for some fixed k.

Definition 1. *A CSP instance $P = (V, D, C)$ is* k-consistent *if for every valid assignment of any $k-1$ variables $\in V$ and any k^{th} variable $v \in V$, there exists a value for v in D that satisfies all the constraints.*

In particular, we show that modern clause-learning SAT-solvers will simulate the effect of enforcing k-consistency in expected polynomial time, for any fixed k.

A modern randomised SAT-solver works in the following way: it picks a variable at random and assigns it a value. If any clause becomes unit under that assignment, then another variable assignment is made to satisfy that clause - we call this process *unit propagation*. Variable assignments are made until all variables are assigned or some clause becomes false. In the second case, a new clause is added to the database that *explains* the conflict (for instance, the *decision learning scheme* adds a clause negating some current decision assignments) and a certain number of decision assignments is undone. A SAT-solver usually restarts after a certain number of conflicts - in this case all current decision assignments are undone.

We phrase our main result in an upper bound on the expected number of restarts of a randomised SAT-solver before it deduces unsatisfiability of the input instance. For the direct encoding and the decision learning scheme this bound is $O(n^{2k}d^{2k})$, where n is the number of variables in the original CSP instance and d is the maximum domain size.

In order to check how well our bound is met in practice, we considered a family of unsatisfiable CSP instances which we call *chains of inequalities*. We have previously shown that these instances are not solved efficiently by standard constraint solvers [3], but they can be shown to be unsatisfiable by enforcing $(2w-1)$-consistency.

An instance of the *chains of inequalities* CSP family is specified by two parameters, w and d. The variables are arranged in groups of size w, each with domain $0, \ldots, d-1$. A

constraint of arity $2w$ is imposed on each pair of successive groups, requiring that the sum of the values assigned to the first of these two groups should be strictly smaller than the sum of the values assigned to the second. We created $((d-1)*w+2)*w$ such groups to ensure that the instances generated are unsatisfiable. An instance with $w=2$ and $d=2$ is shown diagrammatically in Figure 1.

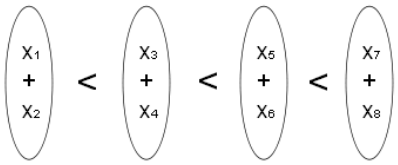


Fig. 1. An unsatisfiable chain of inequalities for w=2 and d=2.

We encoded the unsatisfiable chain of inequalities problem as a SAT formula using the direct encoding, which has a clause for each disallowed assignment specified by the constraints. We modified a state-of-the-art SAT solver called MiniSAT [5], so that it uses a random branching strategy and restarts after every conflict. We called this modified solver simple-MiniSAT. We ran the instances generated on both solvers. MiniSAT solved them very quickly. Simple-MiniSAT was significantly slower, but still produced much better runtimes than standard constraint solvers.

Next, we counted the number of restarts of simple-MiniSAT on small instances of the chains of inequalities family. Using those, we estimated the growth functions for the number of restarts for $w=2$, $w=3$ and $w=4$ and showed that they are in fact much smaller than our theoretical upper bounds.

We showed that without being explicitly designed to do so, current clause-learning SAT-solvers efficiently simulate k-consistency techniques, for all values of k. We gave some experimental results to show that this feature allows clause-learning SAT-solvers to efficiently solve certain families of CSP instances which are challenging for conventional constraint solvers.

References

1. L. Bordeaux, Y. Hamadi, and L. Zhang. Propositional satisfiability and constraint programming: A comparative survey. *ACM Computing Surveys*, 38(4), 2006.
2. J. Petke and P. Jeavons. Local consistency and SAT-solvers. In *Proceedings of CP 2010*, volume 6308 of *LNCS*, pages 398–413. Springer, 2010.
3. J. Petke and P. Jeavons. Tractable benchmarks for constraint programming. Technical Report RR-09-07, Computing Laboratory, University of Oxford, 2009.
4. N. Tamura, A. Taga, S. Kitagawa, and M. Banbara. Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272, June 2009.
5. MiniSAT constraint solver. Software available at http://minisat.se/MiniSat.html.
6. Third international CSP solver competition - http://cpai.ucc.ie/08/.

ghttp: a Goroutine-based Web Server Toolkit

James Whitehead II

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

We are investigating disciplined concurrent software design, where systems are composed of discrete processes with well-defined interfaces. As an initial case study, we have implemented a web server toolkit called `ghttp` that can be used to construct multi-purpose scalable web servers. Servers built using our web toolkit comprise single-purpose concurrent processes connected together to form a process network. An advantage of this structure is a direct correspondence between the code used to create the server, the resulting process network, and the manner in which the server handles web requests. As a result of this, these web servers are more understandable than the equivalent Apache or Lighttpd server configurations.

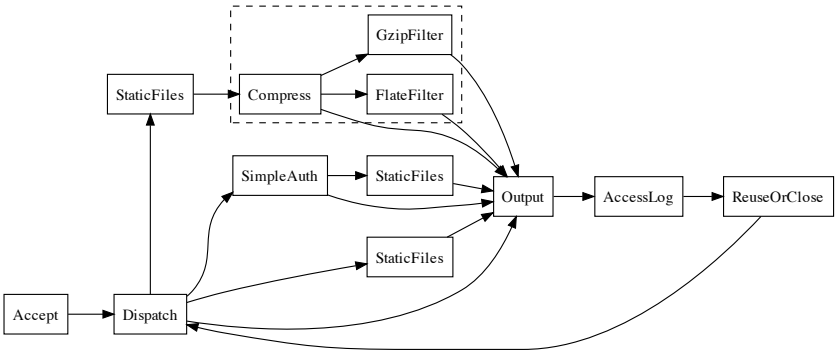


Fig. 1. Network topology for web server

Figure 1 shows the process network graph for an example web server built using our toolkit. Each node in this graph represents a single-purpose component that runs as an independent process in the resulting program. These processes communicate with each other solely via message passing over explicit channels, shown as edges between nodes. An abstraction of the HTTP connection, including the raw networking socket, incoming request, and HTTP response object is passed over these channels.

In its resting state, the process network closely mirrors the graph shown in Figure 1. Once the server starts receiving incoming requests, the system dynamically alters itself to respond. Each component, upon receiving a request, spawns a new goroutine (lightweight thread) to handle the request. In this way, these primary component serve as farm processes, becoming immediately available to receive another request. This helps to ensure that no single connection is able to monopolize server resources, while providing a high degree of scalability.

It should be noted that the compression component in this example web server is actually a collection of three sub-processes. Figure 1 shows these components, `Compress`, `GzipFilter`,

and `FlateFilter`, grouped together in a dashed box. As only some clients will be able to read a compressed response, the `Compress` component must first check the request headers to determine if the client supports any compression methods. If so, the connection can be forwarded to one of the compression components, otherwise it will bypass the compression step. Although a developer could configure their web server to include these components and logic independently, the composite component is able to transparently use sub-components to serve this purpose.

With the HTTP protocol, request and response headers must be output before the associated content. This allows both the client and server to examine the headers and make decisions about how to treat the incoming content. Accordingly, components using the `ghttp` toolkit must perform their work in two stages. The first stage involves examining the request and computing any necessary changes to the response, such as setting the status code or altering headers. In addition, if the component will need to produce or examine the content of the eventual response, it can request content readers or writers as appropriate. Once the headers are computed and any response resources are allocated, the component passes the connection object through to the next portion of the network. Afterwards, in the final stage, the component uses the content reader/writer to produce or transform the content of the response. Because operations on these content channels are synchronous, processes in the *content pipeline* will eventually block, waiting for another process to read from the pipeline.

The connection will eventually arrive at the `Output` component, which serializes the response meta-data, such as status code and headers, and transmits them to the client. It then begins reading the content of the response from the content pipeline, with those components being re-activated as necessary. Figure 2 illustrates the process network for the components that server compressed files, along with the content pipeline that is constructed. The content pipeline is indicated with dashed arrows labelled with paired writers and readers.

Fig. 2. Server processing of compressed static files

The synchronous nature of the content pipeline and the delay in writing the body of the response ensures content is only generated as it is ready to be transmitted. This avoids the need to store large amounts of response data in memory or overcommitting resources to a potentially unreliable client connection. If a client disconnects prior to transmission, the associated sub-network gracefully terminates.

The construction of the `ghttp` web server toolkit has provided us with an example of a concurrent system that is composed of discrete processes that interact via well-defined interfaces. The processes are connected to create processing pipelines for various types of HTTP requests. Content generation and transformation is accomplished by constructing dynamic pipelines, enabling lazy commitment of resources when servicing requests. The semantic properties of these pipelines is enforced using the statically-typed nature of the Go programming language. Future work includes designing a systems-level server that manages high contention of shared resources.

Rich morphology as a challenge to current statistical machine translation systems

Jan A. Botha

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

Morphological variation among different natural languages is often ignored in Statistical Machine Translation (SMT) research. Instead, translation is treated as a mapping of words and phrases from one language to another [4], without taking cognisance of word-internal structure. This approach has worked reasonably well where languages exhibit sufficient similarity, but is ill-suited for language pairs with substantial morphological divergence, e.g. Turkish and English.

The training data used to estimate the probabilistic translation models is inherently sparse. This sparsity is even more acute for a language with complex morphology, where a very large number of different word forms may be derived from the same base word, e.g. the German *fahren* (to drive/ride) gives rise to *fahre, fährst* and *fahrt*, corresponding to I, you (sing.) and you (pl.) drive. In practice, this means it is hard or impossible to produce correct translations in cases where a particular word form was observed rarely or not at all in the training data. For languages with sufficiently complex morphology, this problem cannot be solved by simply using more training data.

By modelling morphological structure, we have something to fall back on when called upon to translate a previously unseen word. Ideally, the morphemes in observed words should be leveraged to produce novel words that were not observed in training, as shown in this French example:

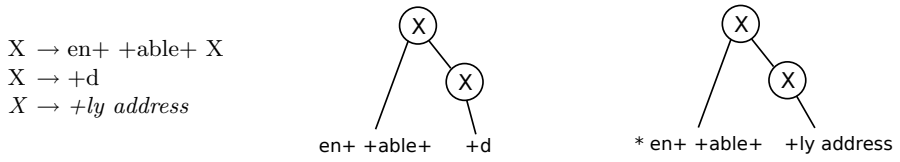
Observed during training			***Generated (not observed)***	
(I hear)	j'**entend**s			
(we reply)	nous répond<u>ons</u>	$\Longrightarrow$	nous **entend**<u>ons</u>	*(we hear)*

As a starting point, we can use an off-the-shelf tool to split words into morphemes, e.g. [3]. This gives us a basic model of morphology. We will treat this as a black box for the moment and consider the challenges that arise when integrating such a model into an existing translation system.

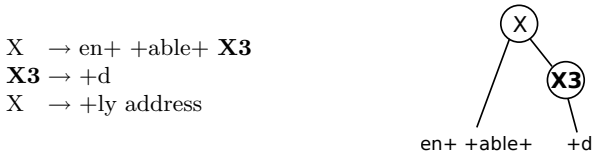
One can view translation approximately as a two-step process: First, given an input sentence in the source language, the system generates multiple candidate translations in the target language. The goal here is translational equivalence, i.e. that the candidate words and phrases convey the same meaning as those in the source sentence. No attention is paid to how fluent the full sentences are in terms of the target language. This is the focus of the second step, where a best candidate is chosen according to fluency. This judgement is made by scoring candidate sentences against a so-called language model (LM) of the target language. The LM, typically an n-gram model, is oblivious to the source language.

It should be clear that the integration of morphology affects both of these steps. We briefly describe two approaches, one applied to each step.

Firstly, we applied a clustering method [1] to refine a popular translation grammar [2] such that it is capable of modelling the kind of word formation exemplified in the example above. This allows novel words to be output in step 1. In particular, added non-terminal categories constrain the combination of morphemes so that illegitimate words do not form, which happens under the unrefined grammar:



(a) Non-sensical words can be produced when using a single non-terminal category.



(b) Constraining the grammar with labelled rules can avoid the problem.

Fig. 1. Grammar fragments and possible derivations they allow when using a single category (top) vs. multiple categories (bottom).

The main results from an evaluation on Dutch→French were that this approach successfully generated words outside of the training vocabulary, that the clustering method learnt extensive and interesting fragments of Dutch morphology, but that translation quality was worse than that of the original system.

A major source of error was the fact that the segmentation of all text into morphemes introduced errors into the bilingual word alignments from which the grammars are induced.

But bad alignments are only part of the story. Another factor is how the integration of morphology affects step 2 of the translation process. This motivates us to study the role of the LM in isolation of the alignment problem. The isolation is achieved by extracting a grammar from unsegmented text, after which the resulting grammar rules are morphologically segmented.

As in the previous case, the candidate sentences generated under these conditions are sequences of morphemes. Instead of simply scoring the sentences against an LM trained on segmented text, we are seeking better ways of measuring the fluency. We expect a back-off approach to be a useful starting point for this: we allow fluency to be measured in terms of words (obtained by rejoining the appropriate morpheme sequences), but for rare or previously unseen words, we fall back onto measuring fluency in terms of the actual morpheme sequences. This way, the appropriate level of representation can be chosen dynamically.

References

1. Blunsom, P., Callison-Burch, C., Cohn, T., Dyer, C., Graehl, J., Lopez, A., Botha, J. A., Eidelman, V., Nguyen, T., Wang, Z., Weese, J., Buzek, O., and Chen, D. Models of Synchronous Grammar Induction, final report, Language Engineering Workshop. Tech. Rep. (in preparation), Johns Hopkins University, 2010.
2. Chiang, D. Hierarchical Phrase-Based Translation. *Computational Linguistics 33*, 2 (June 2007), 201–228.
3. Creutz, M., and Lagus, K. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing 4*, 1 (2007), 1–34.
4. Lopez, A. Statistical machine translation. *ACM Computing Surveys 40*, 3 (2008), 1–49.

Can we learn from text without understanding the words?

Pia-Ramona Wojtinnek

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

Whichever AI or natural language processing task we start with - we always run into the same problem. If only we could put more of all that knowledge that we humans have into the computer - then they could do some wonderful things! It could search the web for the most suitable mobile contract, translate some French publication, summarize an elaborate text or even converse with us! If only GOOGLEtranslate! knew that German "Flügel" can mean "grand piano" as well as "wing", and that a piano is something you play on, then it wouldn't suggest that I "played on the wing" yesterday! Consequently, a lot of effort has been put into building lexical and world-knowledge resources. However, manual acquisition cannot scale up - the most widely used lexical resource WordNet, for example, contains word senses and subclass or part-of relations of about 120,000 concepts and took more than 10 years to build. Therefore, our ultimate goal should be to automatically acquire knowledge from text.

We aim to approach this bottleneck by automatically building knowledge networks directly from text using state-of-the-art linguistic analysis tools. The basic idea is simple: say you were given a huge collection of text about *zoodies* and *tollties* in the style of following and you were asked to represent all you can gather about this domain in a structure.

> *A zoodie roonged tushly. The goan tollties then soufed into the reef. That made the zoodie fretch the tollties foudly and it mooted the reef, which glousted the freet.*

You don't know what is going on, but you will gather that there are things (*zoodie, tollties, reef, freet*), relations (*soufed into, fretch etc.*) and attributes (*tushly, goan, foudly*). To get a good overall overview, you might draw a network with the objects as nodes, connected by the relations and modified by the attributes, which gives you a good starting point to answer questions about this world. And this is how we build our structure: the linguistic analysis tools do not understand the words, but the grammar and we use their automatic analysis to build such a knowledge network on a domain that we are interested in. Fig 1 shows a sample network based on few M.Sc. Handbooks paragraphs. This type of knowledge network can be built for any domain (as the only prerequisite is a text collection) and also in any language as long as equivalent linguistic analysis tools are available. For details of the building method see [1].

We evaluate our knowledge networks on two tasks - semantic relatedness and text similarity, the first of which is described below. As a starting point, we have built large, general-purpose networks using the British National Corpus (BNC) as text collection. It is one of the largest English corpora and contains about 5.8 million sentences from a wide range of sources and topics, ensuring broad coverage of our resulting networks. Our currently largest network is based on 4 million sentences of the BNC, contains 50 million nodes, and took approximately 10 days to build including the linguistic analysis.

One of the tasks that we evaluate our knowledge network on is semantic relatedness and similarity. The objective is to give a numeric measure of the closeness of the meaning of terms. For example, *pound* and *quid* are synonmous, *pound* and *dollar* are similar and *pound* and *profit* are topically related. The core assumption of any statistical approaches to

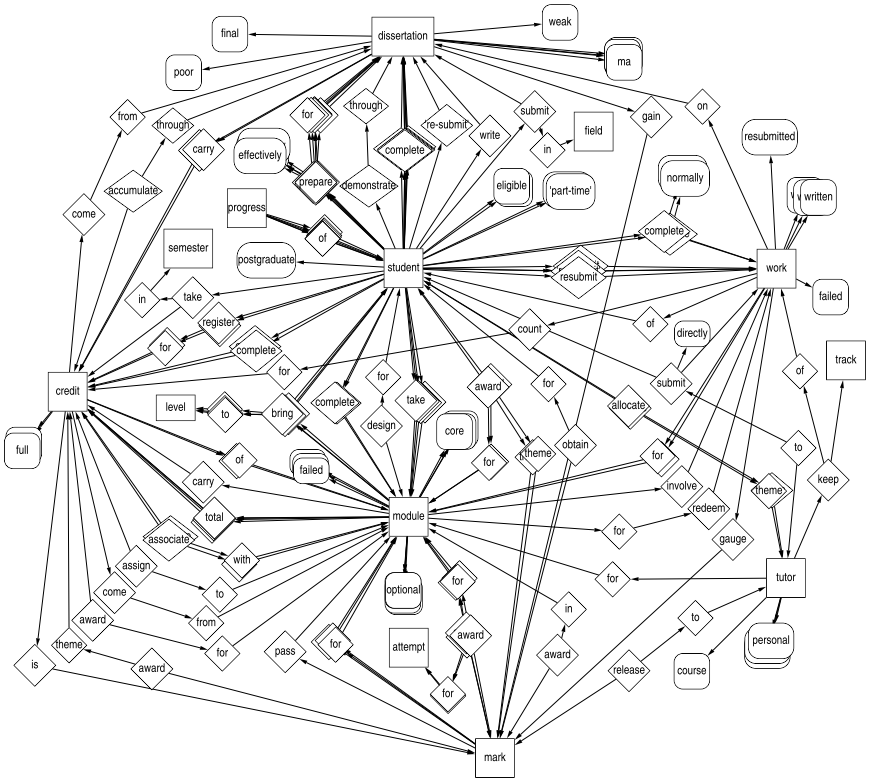


Fig. 1. Subgraph displaying a few concepts and relations from sample network.

this task is that words with similar meaning tend to occur in similiar contexts such as that they tend to co-occur with the same words. For example, as *pound* and *quid* would both occur with *spend* or *pay*. In general, the more fine-grained similarities we want to discover, the smaller and more specific the context need to be. In our case, we measure the similarity of terms by the similarity of their local networks around the corresponding nodes in the network. In other words, we assume that nodes which are semantically similar overlap in the nodes they are strongly connected to. The way in which we measure this overlap consists two steps. In the first step we use a technique called spreading activation to determine the local network around the target words and measure the importance of each contained node to the word. This means we put a certain amount of activation into each target node and let this activation spread over the links [1]. In this way, the nodes around the target nodes receive different levels of activation, giving a measure of their importance. In the second step we construct vectors from the nodes and their activation levels and use a vector similarity measurement to retrieve the score of their context similarity. Our initial results on gold standard collection of word pairs and their similarity scores are promising and our approach may be able to compete with methods based on manually acquired knowledge resources such as WordNet.

References

1. Wojtinnek, P.-R., Harrington, B., Rudolph, S., and Pulman, S. Conceptual knowledge acquisition using automatically generated large-scale semantic networks. Tech. rep., Oxford University Computing Laboratory, April 2010.

Accelerating Dynamic Programming algorithms using Massively Parallel Hardware

Luke Cartey

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

1 Introduction

Dynamic programming is a popular and simple technique for improving performance in algorithms that adhere to two key principles: a) **optimal substructure** and b) **overlapping sub-problems**. An excellent example of dynamic programming is the *Levenshtein distance* algorithm, a solution to the *edit-distance problem* for determining the number of operations required to transform one string to another. It is typically defined using the following recursion equations:

$$d(x,y) = \begin{cases} x & \text{if } y = 0 \\ y & \text{if } x = 0 \\ d(x-1,y-1) & \text{if } s[x] = t[y] \\ min(d(x-1,y), d(x,y-1), d(x-1,y-1)) + 1 & \text{otherwise} \end{cases}$$

Each result $d(x,y)$ is used a number of times; applying dynamic programming we can either *tabulate* the results or *memoise* the results as they are computed.

Dynamic programming is typically used where performance is an important factor; for example, many applications in scientific computing make use of the improvements offered by dynamic programming. An important set of examples are the statistical applications in Bioinformatics, including the *Smith-Waterman* algorithm for alignment and the *Viterbi* algorithm, associated with Hidden Markov Models.

2 Automatic parallelisation of Dynamic Programming

GPUs are increasingly accessible for general purpose computing, with extensive tool-chains now available [2, 1, 3]. However, accessibility does not imply usability, and the intricacies of the custom hardware caused by the drive for improved graphics performance, the differences between GPUs and the paradigm shift of programming for massively parallel hardware all contribute to a notoriously difficult platform for direct development.

Recent attempts have been made to stem this complexity through use of *embedded languages* [5]. Our approach is slightly different; rather than trying to integrate some measure of parallelism into an existing language or attempting to build a general purpose language, we are researching the value of *domain-specific languages* for the GPU. By limiting the scope of our language, we can take a much more reasoned approach to parallelisation.

Our domain for this paper is optimisation problems; in particular those formulated with the optimal sub-structure condition typical of dynamic programming algorithms. Not all algorithms with this condition are amenable to parallelisation; the problem must split into a sufficient number of sub-problems that can be solved concurrently.

Consider again our example of the Levenshtein distance - an example with a Bioinformatics counterpart, the Smith-Waterman algorithm, used widely for sequence alignment,

and typically needed to analyse millions of sequences, each consisting of thousands of characters. If we compare the strings "kitten" and "knitting", the recursions can be *tabulated*, where the arrows represent the dependencies between cells:

		k	**n**	**i**	**t**	**t**	**i**	**n**	**g**
	0	1	2	3	4	5	6	7	8
k	1	0	1	2	3	4	5	6	7
i	2	1	1	1	2	3	4	5	6
t	3	2	2	2	1	2	3	4	5
t	4	3	3	3	2	1	2	3	4
e	5	4	4	4	3	3	2	3	4
n	6	5	4	5	4	4	4	2	3

In a serial computation, we would compute each cell in turn, in an order that appropriately resolves the marked dependencies. In a massively parallel environment we must find a degree of concurrency that supports the large number of processors available. In this application, we can observe that cells along the diagonal are independent of each other, and can therefore be evaluated concurrently.

We have developed a prototype that determines for a recursion, using static analysis, a suitable candidate for GPU implementation from a fixed set of parallel schemes. A candidate is described by the *partitions* which the tabulation for the function can be split into, where partitions are defined by a *partition function* on parameters. The partition function gives each partition a unique *partition value*; partitions may only depend on partitions with lower values, which can be used with a base case analysis to prove termination.

To confirm that the equation only depends on lower valued partitions, we compare the partition value of the original call against the partition value of each recursive call. It is then safe to evaluate any cells with matching partition values concurrently, on the condition that all cells with lower partition values have already been computed. In the example a suitable partition function is $P_f(x, y) = x + y$; it describes the diagonals where cell computation is independent and encodes that a diagonal depends on the diagonals to the left.

This analysis is possible because we posit a simple language and one where we interpret rather than compile the recursions, to allow ourselves the luxury of performing the analysis with known, fixed bounds on each variable. The overhead of interpretation in this approach is negligible for the long execution-time of desirable applications for the GPU.

3 Future Work

As part of our wider research we are developing domain-specific languages for the support of statistical algorithms in Bioinformatics. We have developed HMMingbird [4], a language and compiler for Hidden Markov Models which contains a fixed set of algorithms; we hope to use this research to extend the language to allow custom algorithms.

References

1. ATI Stream Technology. http://www.amd.com/stream.
2. *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*, 2010.
3. Khronos. OpenCL. http://www.khronos.org/opencl/.
4. Luke Cartey. Domain Specific Languages for Massively Parallel Processors, Transfer Report. http://www.hmmingbird.co.uk/downloads/transferreport.pdf, 2010.
5. Tarditi, D., Puri, S., and Oglesby, J. Accelerator: using data parallelism to program GPUs for general-purpose uses. In *ASPLOS-XII* (2006).

Navigating the Deep Web with OXPath

Andrew Sellers

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

The interactive nature of modern web interfaces exacerbates an unfortunate problem: the dynamic nature of these user interfaces, driven by client and server-side scripting (e.g. AJAX), creates challenges for automated processes to access this information with the techniques developed for extracting static web content. The deep web, the part of the web accessible only through such web interfaces, contains significant amounts of useful information; while each individual piece of information is readily available on some webpage, their manual extraction and aggregation is often impractical due to the number of possible combinations—forcing us to accept far from optimal solutions.

Extracting and aggregating web information is not a new challenge. Previous approaches, in the overwhelming majority, either (1) require service providers to deliver their data in a structured fashion (e.g. the Semantic Web); or, (2) "wrap" unstructured information sources to extract and aggregate relevant data. The first case levies requirements that service providers have little incentive to adopt, which leaves us with wrapping. Wrapping a web site, however, is often tedious, since many AJAX-enabled web applications reveal the relevant data only through user interactions. Previous work does not adequately address web page scripting. Even when scripting is considered, the simulation of user actions is neither declarative nor succinct, but rather relies on imperative scripts.

XPath is the language of choice to query node sets in static XML or HTML trees. However, many current Web applications, such as GMail or FaceBook, extensively rely on scripting and HTML events to implement complex user interactions that are not adequately expressible in XPath.

Contribution. We introduce OXPath, an extension of XPath 1.0, to allow the declarative specification of user interactions with (scripted) web applications. We show that just four concise extensions over XPath enable OXPath to deal with scripted web applications while retaining XPath's declarativity and succinctness. Underlying these extensions is OXPath's ability to access the *dynamic DOM trees of a current browser engine*, reflecting all changes caused by scripting: (1) The *simulation of user actions*, such as filling form fields or hovering over a details button, enables interaction with AJAX applications which modify the DOM dynamically. (2) *Selection based on dynamically computed CSS attributes* allows navigation based on presentation, e.g. to the first green section title. (3) For expressing the interaction with forms, *navigation exclusively relying on visible fields* is essential. (4) Extraction markers allows *identification of relevant pieces* for extraction.

Example. The following gives a flavour of OXPath, displaying an expression for finding the cheapest flight information on the popular travel site `kayak.com`:

```
{http://www.kayak.com/flights /}//field()[5]/{$origin}
  /following::field()[@type='text'][1]/{$destination}
  /following::field()[last()]/{click /}
  //tbody[@class~='flightresult'][1]:<flight>/tr[2]
    [td[2]/a:<price=string(.)>] [td[4]:<airline=string(.)>]
```

In this example, we simulate user events with actions, such as `{click /}`, select only visible fields with the node-test `field()`, and identify data to be extracted with extraction markers such as `<flight>`. Next, we explain these in more detail.

Language. Every XPath expression is also valid OXPath. We extend XPath with the notion of Kleene star over path expressions from [2] for increased expressibility in page navigation. Further, we add the following features:

1. **a new kind of location step for actions and form filling** For explicitly simulating user actions such as clicks or mouse-overs, OXPath introduces two types of *action steps*: *contextual action steps* such as `{click}` and *absolute action steps* such as `{click /}` with a trailing slash. Actions may modify the current DOM or replace it entirely. Absolute actions return the root nodes of the DOMs resulting from executing the action. Contextual actions return those nodes in the resulting DOMs that are matched by the *action-free prefix* of the expressions, which is built by removing all intermediate contextual actions from the segment starting at the previous absolute action. Further, OXPath supports an absolute page action `{uri /}` that returns the root of `uri` as well as the use of XPath variables (such as `$x`) in form actions whose bindings are provided by the environment.
2. **a new axis for selecting nodes based on visual attributes** The `style` axis is comparable to the `attribute` axis, but navigates the (computed) CSS properties of a node rather than its properties.
3. **a new node-test for selecting visible fields** To speed-up and simplify the navigation of visible form fields, we introduce the node-test `field()` in order to navigate only over the *visible fields* of a page.
4. **extraction markers** Navigation and form filling are ultimately means to data extraction. In XPath, only a single node set can be returned, while data extraction may require related attributes within records. Attributes may be atomic or records themselves, thus we facilitate record nesting in OXPath. Records and attributes are annotated with extraction markers, such as `:<flight>` in the previous example.

Architecture. Figure 1 illustrates OXPath's architecture for simulating user interactions: We use the web testing framework HtmlUnit [1] as an automated browser. First, an HTML document is parsed, and every inline or on-load script is executed. On the resulting DOM, we evaluate the expression up to the first action, trigger all relevant events, and obtain a set of new (or modified) DOMs. The expression up to the next action is evaluated against the new DOMs and so on.

Future. Our current work focuses on defining a formal semantics of OXPath as well as extending our prototype to a cloud-based platform.

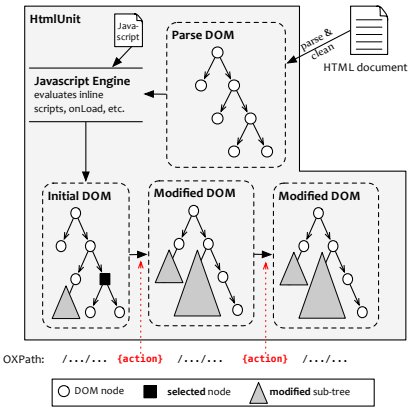


Fig. 1. OXPath Event-Actions

Acknowledgements. Joint work with T. Furche, G. Gottlob, G. Grasso, and C. Schallhart. The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 246858. The views expressed in this article are solely those of the author.

References

1. HtmlUnit. `http://htmlunit.sourceforge.net/` Online only. Retrieved at 14/09/2010.
2. M. Marx. Conditional XPath. *ACM TODS*, 30(4), 2005.

Author Index

Andrew Sellers, 35

Blanca Rodriguez, 13

Huy Vu, 21

Irina Voiculescu , 19

James Whitehead II, 27
Jan A. Botha, 29
Justyna Petke, 25

Lu Feng, 9
Luke Cartey, 33

Maneesh Khattri, 11
Mark Jenkins, 7
Michael Vanden Boom, 5
Miguel Bernabeu, 13
Mikael Wallman, 13, 15

Pia-Ramona Wojtinnek, 31

Ronald Kainda, 23

Sara-Jane Dunn, 17
Stephen Cameron, 19
Stuart Golodetz, 19