Proceedings of the Automated Reasoning Workshop 2016 Bridging the Gap between Theory and Practice ARW 2016

19th–20th May 2016
University of Liverpool
Liverpool, United Kingdom

Editor:
Ullrich Hustadt

Department of Computer Science
University of Liverpool

UNIVERSITY OF
LIVERPOOL

Preface

This volume contains the proceedings of ARW 2016, the twenty-third Workshop on Automated Reasoning, held 19th–20th May 2016, in Liverpool, England (UK). As for the previous events in this series, this workshop provides an informal forum for the automated reasoning community to discuss recent work, new ideas and current trends. It aims to bring together researchers from all areas of automated reasoning in order to foster links and facilitate cross-fertilisation of ideas among researchers from various disciplines; among researchers from academia, industry and government; and between theoreticians and practitioners.

These proceedings contain the abstracts of two invited talks, by Julian Padget (University of Bath), on "Deontic Sensors", and Ulrich Berger (Swansea University), on "Extracting nondeterministic programs", and eleven extended abstracts contributed by participants of the workshop.

The abstracts cover a wide range of topics including the use of SAT and SMT solvers for finite model finding with sorts; a model for intelligent agents (sensors) on wireless sensor networks to guard against energy-drain attacks; the probabilistic verification of an ant-based control algorithm for a swarm of robots; a method for forgetting of concept and role symbols in ontologies in an expressive description logic; a resolution-based approach to rewriting $\mathcal{ALC}$-TBoxes into Horn-$\mathcal{ALC}$-TBoxes of depth one for which conjunctive query answering can be done efficiently; translations from Metric Temporal Logic over the natural numbers to Linear Temporal Logic; ideas for the development of a universal algebra for syntax with bindings; the use of model-checking for the verification of autonomous systems that use rational agents to make decisions; a new normal form, calculus and prover for propositional basic multimodal logic; modal tableau systems with blocking and congruence closure; and natural deduction systems for classical and non-classical logics.

I would like to thank the members of the ARW Organising Committee for their advice. I would also like to thank all the colleagues who have helped with the local organisation, namely, Rebekah Martin, Dave Shield, Elaine Smith, and Lisa Smith.

Liverpool
May 2016

Ullrich Hustadt

Organising Committee

Alexander Bolotov	(University of Westminster)
Simon Colton	(Goldsmiths College, University of London)
David Crocker	(Escher Technologies)
Louise Dennis	(University of Liverpool)
Clare Dixon	(University of Liverpool)
Jacques Fleuriot	(University of Edinburgh)
Ullrich Hustadt	(University of Liverpool)
Mateja Jamnik	(Univerity of Cambridge)
Florian Kammüller	(Middlesex University)
Ekaterina Komendantskaya	(University of Dundee)
Alice Miller	(University of Glasgow)
Oliver Ray	(University of Bristol)
Renate A. Schmidt	(University of Manchester)
Volker Sorge	(University of Birmingham)

Workshop Website

`http://arw2016.csc.liv.ac.uk/`

Invited Talks

Contributed Abstracts

Deontic Sensors

Julian Padget

Department of Computer Science, University of Bath
j.a.padget@bath.ac.uk

Procedural programs do what we tell them, but not always what we want them to do, especially when presented with unexpected inputs (that we did not think about). We assume that such tight control reduces risk, whereas giving a program more autonomy is scary: "who knows what it might do?!?!".

In principle, some autonomy to choose an appropriate action ought to be at least as good and possibly better, by allowing greater resilience. But a program's capacity to understand its environment is limited to what the designer knows or can foresee, which is no better than where we started.

To provide up-to-date interpretation of (aspects of) the environment, we propose "deontic sensors", based on a formal model and realised through Answer Set programming. These sensors observe program actions and provide advice on what a program (agent!) can, ought and ought not to do and illustrate the concept with examples from a variety of socio-technical system demonstrators.

Extracting nondeterministic programs

Ulrich Berger

Department of Computer Science, Swansea University
u.berger@swansea.ac.uk

Program extraction exploits the Curry-Howard correspondence for the automatic synthesis of verified programs from formal constructive proofs. After an overview of the method the talk will focus on a particular application in exact real number computation. We introduce an extension of intuitionistic fixed point logic by a modal operator for nondeterminism and apply it to extract programs computing with Tsuiki's infinite Gray code for real numbers.

Using SAT and SMT Solvers for Finite Model Finding with Sorts

Giles Reger[1] Martin Suda[2]

[1] University of Manchester, Manchester, UK
giles.reger@manchester.ac.uk
[2] Institute for Information Systems, Vienna University of Technology, Austria
msuda@forsyte.tuwien.ac.at

Abstract: We report on a recent technique for finding finite models in multi-sorted first-order logic. The approach extends the MACE-style approach of encoding the check for the existence of a finite model of a certain size as a SAT problem. To deal with multiple sorts the space of possible assignments of sort domain sizes is searched. To efficiently navigate through this space, arithmetical constraints are produced and passed to an SMT solver which produces an answer describing the next domain size assignments to try. The technique is implemented in the Vampire theorem prover.

1 Introduction

One method for establishing satisfiability of a first-order formula is to search for a *finite model*. Such models are useful in a number of applications. There have been various approaches to finding finite models in first-order logic. The technique we are interested in was pioneered by the MACE model finder [4] and extended by the Paradox work [2]. The basic idea is to encode the check for a finite model of a certain size as a SAT problem.

We consider the extension of first-order logic with sorts. In this setting a finite model may need to use different domain sizes for different sorts (see the example below). Our new technique [5] explores the space of possible domain size assignments (one domain size for each sort) by producing *constraints* from each failed attempt and using an SMT solver (in our case Z3 [3]) to guide the search.

2 The Monkey Village Example

There is a much-used simple example for finite model finding with sorts. For a more interesting (similar) example see our paper [5]. The example involves a village of monkeys where each monkey owns at least two bananas. This can be captured by two formulas:

$$(\forall M : \mathit{monkey})(\mathsf{b}_1(M) \neq \mathsf{b}_2(M) \wedge \mathsf{owns}(M, \mathsf{b}_1(M)) \wedge \mathsf{owns}(M, \mathsf{b}_2(M)))$$
$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathsf{owns}(M_1, B) \wedge \mathsf{owns}(M_2, B) \rightarrow M_1 = M_2)$$

where the predicate owns associates monkeys with bananas and the functions b_1 and b_2 witness the existence of each monkey's minimum two bananas.

The smallest finite model for these formulas has a domain of size 1 for *monkey* and a domain of size 2 for *banana*. The two sorts cannot have the same size as there must be at least twice as many bananas as monkeys.

3 MACE-Style Finite Model Building

The MACE-Style approach produces a grounding of the first-order problem using a given n domain constants and then encodes this as a SAT problem. For this grounding to be sound the problem needs to be put into *flattened clausal* form where function and predicate symbols are only applied to variables, e.g. $\mathsf{owns}(M, \mathsf{b}_1(M))$ becomes $\mathsf{owns}(M, x) \vee \mathsf{b}_1(M) \neq x$. The flattened clauses are then instantiated with all mappings of variables to domain constants. One then needs to also encode information about *functionality* and *totality* of functions. For example, for the unary function b_1 and domain constants d_1 and d_2 for *banana* and c_1 for *monkey* we add the following:

Functionality	$\mathsf{b}_1(c_1) \neq d_1 \vee \mathsf{b}_1(c_1) \neq d_2$
Totality	$\mathsf{b}_1(c_1) = d_1 \vee \mathsf{b}_1(c_1) = d_2$

One can optionally add *symmetry breaking* information by ordering ground terms and making the smallest equal the first domain element, etc. This is necessary for efficiency.

The search for finite models then involves producing and checking SAT encodings for increasing domain sizes.

4 Adding Sorts

The previous encoding can be lifted to the multi-sorted setting by introducing a set of domain constants per sort and instantiating variables by constants from their sort. As previously noted, the number of domain constants per sort may (necessarily) vary. We utilise a simple breadth-first search algorithm of the possible domain size assignments. This search is driven by *constraints* derived from failed SAT proofs as summarised below.

Getting Constraints We update the above encoding so that a failed check for a model can give us some insight into why the check failed. We extend the encoding with two extra labels $|s| > n$ and $|s| < n$ for each sort s and for the concrete value n of the current size of the domain of s. Intuitively these stand for the size of sort s being too small or too big, respectively.

If we cannot satisfy a totality condition then the size of the sort is too small (we need more domain constants) and

Table 1: Experimental Results .

	CVC4	Paradox	iProver	Vampire		CVC4	Vampire
FOF+CNF: sat	1181	1444	1348	**1503**	UF: sat	764	**896**
FOF+CNF: unsat	-	-	1337	**1628**	UF: unsat	-	249

therefore we can extend the totality constraints to

$$\mathsf{b}_1(d_1) \neq d_1 \vee \mathsf{b}_1(d_1) \neq d_2 \vee |monkey| > 2$$

where the current size of $monkey$ is 2. Conversely, if we cannot satisfy a grounded input clause then the size of s may be too large and we can extend the grounding to

$$\mathsf{owns}(c_2, d_1) \vee \mathsf{b}_1(c_2) \neq d_1 \vee |monkey| < 2$$

using the previous clause from our example.

After extending the encoding we solve the SAT problem under the assumptions that we are using the correct sort sizes, i.e. we add the following for each sort s

$$\neg(|s| > 2) \wedge \neg(|s| < 2).$$

The mechanism for solving under assumptions, supported by many SAT solvers, provides a subset of these assumptions sufficient to show unsatisfiability of the SAT problem. This subset is our set of constraints that explains why this check failed.

Using Constraints To use these constraints to guide the search we use an SMT solver to find a model of the constraints. This model will assign a value to each domain size, giving us the next SAT problem to check. To ensure that this search grows appropriately we add the additional constraint that the sum of the sort sizes must initially equal the number of sorts. If no model can be found with this constraint then we add one to this value and try again.

5 More Fun with Sorts

We can do various other things in an attempt to improve this search for the right combination of sort sizes.

Monotonic Sorts A sort s is *monotonic* for a formula φ if adding another domain constant to s in a model of φ produces another model for φ (see [1]). Monotonic sorts can be easily detected and used in a number of ways.

Collapsing Sorts All monotonic sorts can be treated as a single sort. This reduces the size of the search space. However, if one of these sorts needs to be very large then all sorts will need to grow, potentially unnecessarily increasing the size of the SAT encoding.

Expanding Subsorts Alternatively, one can infer *subsorts* by identifying function and predicate positions that are disjoint with respect to variables. If these subsorts are monotonic then they can be treated as real sorts, with the constraint that they do not grow larger than their parent sort.

More Constraints It is possible to detect constraints *between* sorts due to (for example) an injective function from one sort to another. To detect such properties we adapt a standard saturation algorithm to use a single proof attempt to prove as many of these relationships as possible.

6 No Finite Model

There are cases where we can establish upper bounds on the size of a sort e.g. when it only uses constant symbols. These can be treated as additional constraints. If the resultant set of constraints is unsatisfiable without the previous restriction on sort sizes (that they sum to some number) then there is no model and the problem is unsatisfiable.

7 Experiments and Concluding Remarks

Table 1 gives a brief summary of the experimental results reported in [5]. Two sets of experiments are described. The first considers unsorted TPTP problems and applies sort expansion. The second considers SMT-LIB problems from the Uninterpreted Functions logic and applies monotonic sort grouping. The new techniques perform better than the other leading tools for finite model finding.

Future work involves introducing further heuristics for symmetry breaking and search. One option is to explore *incomplete* search strategies that skip parts of the search space; sacrificing finite model completeness for efficiency.

References

[1] Koen Claessen, Ann Lillieström, and Nicholas Smallbone. Sort it out with monotonicity - translating between many-sorted and unsorted first-order logic. In *CADE-23*, pages 207–221, 2011.

[2] Koen Claessen and Niklas Sörensson. New techniques that improve MACE-style model finding. In *CADE-19 Workshop: Model Computation - Principles, Algorithms and Applications*, 2003.

[3] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Proc. of TACAS*, volume 4963 of *LNCS*, pages 337–340, 2008.

[4] William Mccune. A Davis-Putnam Program and its Application to Finite First-Order Model Search: Quasigroup Existence Problems. Technical report, Argonne National Laboratory,, 1994.

[5] Giles Reger, Martin Suda, and Andrei Voronkov. Finding finite models in multi-sorted first-order logic. In *SAT-19*, 2016.

SENSOR INTELLIGENCE FOR TACKLING ENERGY-DRAIN ATTACKS ON WIRELESS SENSOR NETWORKS

Ekereuke Udoh, Vladimir Getov, Alexander Bolotov
Distributed and Intelligent Systems Research Group
University of Westminster, 115 New Cavendish Street, London W1W 6UW
w1562173@my.westminster.ac.uk, V.S.Getov@westminster.ac.uk, A.Bolotov@westminster.ac.uk

Abstract. In this paper we propose a model for intelligent agents (sensors) on a Wireless Sensor Network to guard against energy-drain attacks in an energy-efficient and autonomous manner. This is intended to be achieved via an energy-harvested Wireless Sensor Network using a novel architecture to propagate knowledge to other sensors based on automated reasoning from an attacked sensor.

Introduction. Wireless Sensor Networks (WSN) form part of the architecture of the Internet of Things (IoT) and are known particularly for their resource-constrained nature due to the fact that these sensors are usually powered by batteries alongside their low processing power. This makes the WSN prone to energy-drain attacks, one of which is known as denial-of-sleep attack [1]. A number of approaches exists which aim to tackle these attacks; however these approaches rarely take into consideration the future scale of the IoT as predicted to expand exponentially in the coming years [2]. The implication of this is that approaches would need to be, not just energy-efficient but, autonomous in nature in order to withstand the variety of attacks that may arise as a result of a larger network where there is a wider attack surface.

Proposed Approach. The intended approach is an improvement of existing approaches - Gateway Media Access Control (GMAC) and Hierarchical Collaborative Model (HCM). While GMAC [3] and the hash-based scheme [4] use centralized approach via cluster heads, HCM [5] and the distributed wake-up scheme [6] use a distributed architecture. Although these approaches seem very useful, they do not take into consideration the size of the network especially on a large scale. Our proposed architecture is based on a combination of both the centralized and the distributed approach. It would involve the use of intelligent agents whereby each sensor becomes an agent which can sense data and take responsive action with the workload dynamically distributed among them. However, this would not function optimally with the current battery-powered sensors, but rather an energy harvested IEEE 802.15.4 wireless sensor network [2]. This is necessary because the dynamic distribution would lead to an increase in processing power thereby consequently increasing energy costs. In [7], the concept of virtual clusters is introduced whereby nodes are grouped into the same subnet and presented as a single resource. The WSN will be dynamically divided into clusters with cluster heads appointed for each cluster. In this approach, if a sensor encounters or senses an attack, it immediately takes responsive action and also broadcasts the information to the rest of the appointed cluster heads via a "rumour" approach which may consume more bandwidth than processing power. The "rumour" approach is coined from the term "routing by rumour", which explains the semantics of distance-vector routing protocols whereby each router sends messages to its nearest neighbour until the information propagates to all the routers. In this case, the cluster heads send information to the nearest cluster head and it continues that way until the information gets to all the cluster heads which then pass the information to their clusters. The cluster heads then relay this information to the sensors in their clusters.

High Level Constituents of the Approach

- Automated reasoning via intelligent agents
  In [8] a management scheme based on automated reasoning whereby Bayesian reasoning is used during the learning phase, is proposed to help protect against intrusions and also enhance energy-efficiency on a wireless sensor network. Threshold analysis is also used prior to the reasoning. In [9], the BayesMob algorithm is used for self-healing in a case where one or more sensor nodes fail. In this paper, we consider the Bayesian equation for predictive reasoning by sensors as a way of anticipating an attack and preventing it beforehand. More specifically, our model is based on the following Bayesian equation:

  $$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)},$$

  where A and B are events
  - P (A) and P (B) are the probabilities of A and B without regard to each other.
  - P (A | B), a conditional probability, is the probability of observing event A given that B is true.
  - P (B | A) is the probability of observing event B given that A is true.

  In this context, for example, P (A) is the base rate or prior probability that a sensor is under attack. This could be based on a threshold value of the amount of energy being consumed by the sensor. P (B) could be the probability that the messages sent by the attacker have a certain size/frequency range. P (B|A) would then be the probability that a sensor under attack is receiving a certain message size/frequency range.

- Choice of WSN architecture
  A combination of centralized and distributed architecture is proposed. The centralized approach involves the use of clusters which are formed dynamically based on the location and proximity of sensors. Each cluster

has a cluster head which not only serves the other sensors but also acts as a proxy thereby hiding the identity of the sensors. At the cluster level, a single-hop architecture is used while a multi-hop architecture is used for communication between cluster heads. Because of its centralized approach, the single-hop architecture has low delay and a high channel capacity, while the multi-hop architecture which is distributed in nature has a high energy-efficiency and high signal-to-noise ratio [12].

- Resource availability (via energy-harvesting rather than battery-powered sensors)
  In [10], a relationship between autonomy and energy-efficiency is established whereby existing wireless sensor networks are limited by their battery power and therefore cannot be autonomous except more power is made available to them. Hence, energy-harvesting is proposed. In [11], the need for energy harvesting is also acknowledged considering that the existing battery-powered sensor nodes need periodic maintenance which contradicts with the characteristics of autonomous systems.

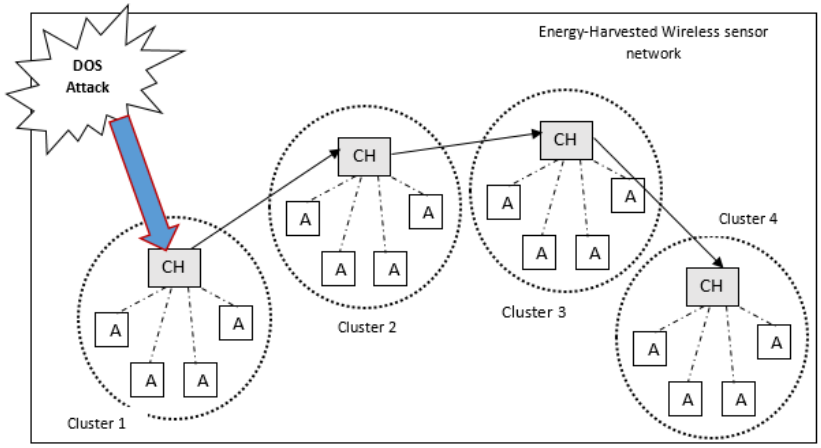

Figure 1: Proposed Wireless Sensor Network Architecture for Intelligent Agents (Sensors)

Figure 1 above shows an attack being directed at a cluster head. The cluster head (CH) is an intelligent agent and also acts as a proxy for the member-nodes of its cluster. The moment it realizes it is under attack, it appoints one of its members as a cluster head and isolates itself from the network thereby allowing communication to continue. The learned information is then passed to other cluster heads to enable them to easily prevent the attack, in case they become the new target.

Conclusions. Our novel architecture is intended to fit into the big picture of providing an energy-efficient and autonomous security on the IoT. Currently, the proposed approach is being tested on a simulator and the results will be analysed and discussed in the context of energy-efficiency and other existing approaches.

REFERENCES

[1] A. Merlo, M. Migliardi, and L. Caviglione, "A survey on energy-aware security mechanisms," *Pervasive Mob. Comput.*, vol. 24, pp. 77–90, 2015.

[2] B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "Energy Cost of Security in an Energy-Harvested IEEE 802.15.4 Wireless Sensor Network," pp. 198–201, 2014.

[3] M. I. Brownfield, M. I. Brownfield, and N. J. Davis, "Energy-efficient Wireless Sensor Network MAC Protocol Energy-efficient Wireless Sensor Network MAC Protocol," *Management*, 2006.

[4] M. Pirretti, S. Zhu, N. Vijaykrishnan, P. Mcdaniel, M. Kandemir, and R. Brooks, "The Sleep Deprivation Attack in Sensor Networks: Analysis and Methods of Defense," *Int. J. Distrib. Sens. Networks Int. J. Distrib. Sens. Networks Int. J. Distrib. Sens. Networks*, vol. 2, no. 03, pp. 267–287, 2006.

[5] T. Bhattasali, "Sleep Deprivation Attack Detection in Wireless Sensor Network," *Int. J. Comput. Appl.*, vol. 40, no. 15, pp. 19–25, 2012.

[6] F. J. Wu and Y. C. Tseng, "Distributed wake-up scheduling for data collection in tree-based wireless sensor networks," *IEEE Commun. Lett.*, vol. 13, no. 11, pp. 850–852, 2009.

[7] S. Isaiadis and V. Getov, "Integrating Mobile Devices into the Grid: Design Considerations and Evaluation," *Proc. of Euro-Par 2005 Conference, LNCS*, vol. 3648, pp. 1080-1088, Springer, 2005.

[8] H. Khalife and F. Krief, "Reasoning Services for Security and Energy Management in Wireless Sensor Networks," *Proc. 7th Int. Conf. Netw. Serv. Manag.*, pp. 520–524, 2011.

[9] M. Coles, D. Azzi, and B. Haynes, "A self-healing mobile wireless sensor network using predictive reasoning," *Sens. Rev.*, vol. 28, no. 4, pp. 326–333, 2008.

[10] R. Vullers, R. Schaijk, H. Visser, J. Penders, and C. Hoof, "Energy harvesting for autonomous wireless sensor networks," *IEEE Solid-State Circuits Mag.*, vol. 2, no. 2, pp. 29–38, 2010.

[11] A. Lambebo and S. Haghani, "A Wireless Sensor Network for Environmental Monitoring of Greenhouse Gases," p. 2010, 2014.

[12] A. K. Singh, S. Rajoriya, S. Nikhil, and T. K. Jain, "Design constraint in single-hop and multi-hop wireless sensor network using different network model architecture," *Int. Conf. Comput. Commun. Autom.*, pp. 436–441, 2015.

Probabilistic Verification of an Ant-Based Swarming Algorithm

Paul Gainer Clare Dixon Ullrich Hustadt

Department of Computer Science, University of Liverpool
Liverpool, L69 3BX – United Kingdom
{P.Gainer, CLDixon, U.Hustadt}@liverpool.ac.uk

Abstract: Control algorithms for robot swarms are often inspired by decentralised problem/solving systems found in nature. In this paper we conduct a formal analysis of an algorithm inspired by the foraging behaviour of ants, where a swarm of flying vehicles searches for a target at some unknown location. We give the results of checking probabilistic temporal properties that complement simulation results, and would facilitate the logistics of swarm deployment.

1 Introduction

A robot swarm is comprised of some number of simple, homogeneous robots, working together to achieve objectives in some environment without centralised control [5]. Coordination between members of the swarm is achieved through self-organisation and local interactions.

Swarm behaviours are generally analysed through simulation and observations of real implementations. The formal analysis of swarm behaviours can complement the design of swarm algorithms by revealing potential problems that may go unnoticed by empirical analysis [1].

A common approach in swarm robotics has been to develop control algorithms based on abstractions of natural systems. In particular, much work has been conducted to develop control algorithms based on the behaviours of social insects. In [3] a swarm of Micro Air Vehicles (MAVs) attempts to form a communication pathway between multiple ground users in a disaster area. The control algorithm for each MAV is inspired by the stigmergic foraging behaviour of army ants which maintain pheromone paths between their nest and food sources.

We have applied probabilistic temporal verification to the scenario presented in [3], generating paramaterized formal models for the probabilistic model checker PRISM, which we use to either exhaustively or statistically test probabilistic reachability and reward-based properties. We demonstrate how values pertaining to the logistics of deployments of swarms of MAVs, that would be unobtainable through simulation alone, can be calculated by exhaustively checking reward-based properties in the models.

This extended abstract gives an overview of the scenario and presents the results of checking probabilistic temporal logic properties in the models.

2 Scenario

The scenario to which we apply probabilistic model checking is presented in [3]. Here, a simulated swarm of MAVs is deployed in order to establish a robust emergency communication network between a *target user*, situated at some unknown location, and the base station wherefrom the swarm is launched.

Figure 1 shows a Y-junction grid consisting of possible positions that MAVs will ideally adopt in their search for the target user. MAVs are launched at regular intervals from position $(0, 0)$ on the grid, and are in the *exploring* state. In the exploring state a MAV navigates through the grid. When a MAV reaches a position in the grid where there is no other MAV it will change to the *node* state and remain at that position, acting as a platform upon which other MAVs can "deposit" virtual pheromone. When a MAV in the exploring state reaches a position in the grid where there is already a MAV in the node state, it continues moving outward and makes a probabilistic choice on which branch to take, determined by the levels of pheromone deposited at the next positions on the left and right branches. Pheromone levels dissipate gradually over time and when they are depleted a MAV in the node state changes to the *returning* state. It then navigates back through the grid towards the base node similarly to a MAV in the exploring state.

3 Modelling

Models of the scenario were constructed using the *probabilistic model checker* PRISM [4]. Given a probabilistic model of a system, PRISM can be used to analyse both temporal and probabilistic properties of the input model by exhaustively checking some logical requirement against all possible behaviours. Properties to be checked can be specified using *probabilistic temporal logics* such as Probabilistic Computation Tree Logic (PCTL). PCTL consists of clas-

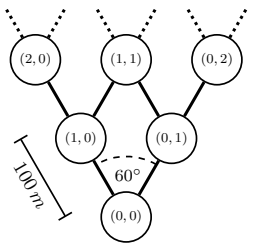

Figure 1: The Y-junction grid illustrating the ideal positions for MAVs.

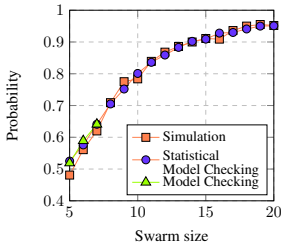

Figure 2: The mean probability of finding the user within 30 minutes.

sical logical operators, temporal operators,, and the probabilistic operator $\mathrm{P}_{\bowtie\gamma}(\phi)$ where $\bowtie \in \{<, \leq, >, \geq\}$ is a relational operator and γ is a probability threshold. PCTL can therefore be used to specify properties such as $\mathrm{P}_{\geq 0.5}(\Diamond\phi)$, meaning "$\phi$ holds at some future point with a probability of at least 0.5". PRISM allows properties to be expressed which evaluate to a numerical value, for instance $\mathrm{P}_{=?}(\Diamond\phi)$, "the probability of ϕ being true at some point in the future".

PRISM can be used to reason about other measurable aspects of model behaviours. Rewards can be associated with states and properties relating to expected values for these rewards can be checked in models. The R operator allows properties to be expressed such as the reachability reward property $\mathrm{R}_{=?}(\Diamond\phi)$, "what is the expected reward for reaching a state where ϕ is true".

In our models we consider only the moments in time where each MAV is exactly at some ideal position (i, j). Since MAVs are behaviourally identical we can use a counting abstraction to record the number of MAVs at each location. In addition we ignore the possibility of MAVs colliding or getting lost. We refer the reader to [2] for a comprehensive description of the model.

4 Experiments

To validate our model we applied statistical model checking using the PRISM discrete-event simulator and compared our results to those obtained from the simulations conducted in [3]. The mean probability of establishing contact with a user within 30 minutes was calculated over a series of 500 simulations for varying swarm sizes. The target user was located at some randomly determined location within a 60 degree arc in a known cardinal direction from the base node at a distance of $\approx 200/500\,m$.

Figure 2 compares the results of calculating the mean probability over all locations of a MAV establishing contact with the target user, to the results presented in [3]. Statistical model checking results were obtained using 500 discrete-event simulation samples with an average confidence interval of $\pm 2\%$ based on a 99.0% confidence level.

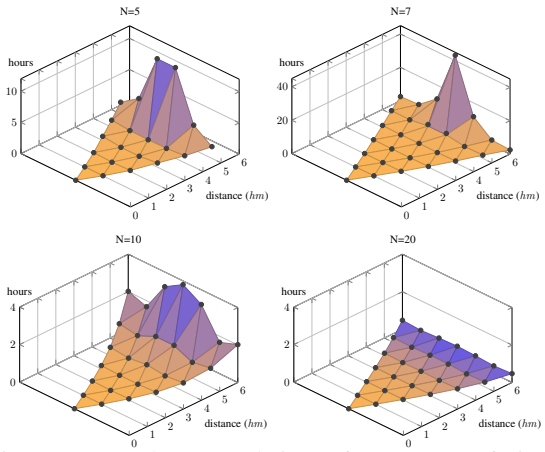

Figure 3: Total expected times for swarms of size N to establish communication with a user with probability 1.

By associating a reward of one with each state in our models we can calculate the total time expected for the swarm to establish contact with a user at (i, j) with probability 1. In Figure 3 we show the total expected time in hours for a deployment of N MAVs to establish communication with the target user with a probability 1. Results where $N > 7$ were obtained using statistical methods over 4000 samples.

5 Conclusions and Further Work

We have constructed formal probabilistic models making some simplifying assumptions, and clearly shown a close correspondence between these models and the simulations conducted in the original scenario. We have checked both probabilistic and reward-based properties in our model, where the resultant calculated values could be used to plan the deployment of a swarm of MAVs where establishing contact with a user must be guaranteed, or achieved with a probability that exceeds some given threshold.

We aim to further abstract our approach so that the techniques that we have developed here can be applied to a broader range of swarm algorithms where stigmergic communication is used to coordinate the behaviour of the swarm.

Acknowledgments

The authors would like to thank the Networks Sciences and Technology Initiative (NeST) of the University of Liverpool for the use of their computing facilities. The first author would like to acknowledge the funding received from the Sir Joseph Rotblat Alumni Scholarship.

References

[1] Clare Dixon, Alan FT Winfield, Michael Fisher, and Chengxiu Zeng. Towards temporal verification of swarm robotic systems. *Robotics and Autonomous Sys.*, 60(11):1429–1441, 2012.

[2] Paul Gainer, Clare Dixon, and Ullrich Hustadt. Probabilistic Model Checking of Ant-Based Positionless Swarming (Extended Version). Technical Report ULCS-16-001, University of Liverpool, Liverpool, UK, 2016.

[3] Sabine Hauert, Laurent Winkler, Jean-Christophe Zufferey, and Dario Floreano. Ant-based swarming with positionless micro air vehicles for communication relay. *Swarm Intelligence*, 2(2-4):167–188, 2008.

[4] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. TACAS 2006*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

[5] Erol Şahin and Alan Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2):69–72, 2008.

Concept and Role Forgetting in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-Ontologies

Yizheng Zhao Renate A. Schmidt

School of Computer Science, The University of Manchester, UK

Abstract: We revise a recently-developed method for forgetting of concept and role symbols in ontologies expressible in the description logic $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$. Being based on an Ackermann approach, the method is one of only few approaches that can eliminate role symbols, that can handle role inverse and ABox statements, and is the only approach so far providing support for forgetting in description logics with nominals.

1 Introduction

Forgetting is a non-standard reasoning problem concerned with creating restricted views for ontologies relative to subsets of their initial signatures while preserving all logical consequences up to the symbols in the restricted views. It turns out to be a very useful technique in various tasks crucial for effective processing and management of ontologies. For example, forgetting can be used for ontology reuse, for creating ontology summaries, for information hiding, for computing logical difference between ontologies, for ontology debugging and repair, and for query answering.

Early work in the area primarily focused on forgetting concept symbols, as role forgetting was realised to be significantly harder than forgetting of concept symbols, because the result of forgetting role symbols often requires more expressivity than is available in the target logic.

The contribution of this work is a practical method for forgetting of concept and role symbols in expressive description logics not considered so far. The method accommodates ontologies expressible in the description logic $\mathcal{ALCOIH}$ and the extension allowing positive occurrences (μ^+) of the least fixpoint operator μ, the top role ∇ and role conjunction $\sqcap$. The added expressivity has the advantage that it reduces information loss; for instance, the solution of forgetting the role symbol r in the ontology $\{A \sqsubseteq \exists r.B, \exists r.B \sqsubseteq B\}$ is $\{A \sqsubseteq \exists \nabla.B, A \sqsubseteq B\}$, whereas in a description logic without the top role (or ABox axioms or nominals) the solution is $\{A \sqsubseteq B\}$, which is weaker.

Definition 1 (Forgetting in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$) *Let $\mathcal{O}$ and $\mathcal{O}'$ be $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-ontologies and let Σ be any subset of* $\mathrm{sig}(\mathcal{O})$. *$\mathcal{O}'$ is the* solution of forgetting the Σ-symbols *in $\mathcal{O}$, if the following conditions hold: (i) $\mathrm{sig}(\mathcal{O}') \subseteq \mathrm{sig}(\mathcal{O})$ and $\mathrm{sig}(\mathcal{O}') \cap \Sigma = \emptyset$, and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ Σ-equivalent to $\mathcal{I}$.*

2 The Forgetting Method

The forgetting process in our method consists of three main phases: the reduction to a set of $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-clauses, the forgetting phase and the definer elimination phase (see Figure 1). In the forgetting phase, an analyser may be used to generate forgetting orderings, $\succ^{\mathcal{C}}$ and $\succ^{\mathcal{R}}$, of the concept symbols and role symbols in Σ.

The input to the method is an ontology $\mathcal{O}$ of TBox and RBox axioms expressible in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$ (ABox axioms are assumed to be equivalently expressed as TBox axioms in our method), and a set Σ with the concept and role symbols to be forgotten. The first phase transforms the input ontology $\mathcal{O}$ into a set $\mathcal{N}$ of clauses.

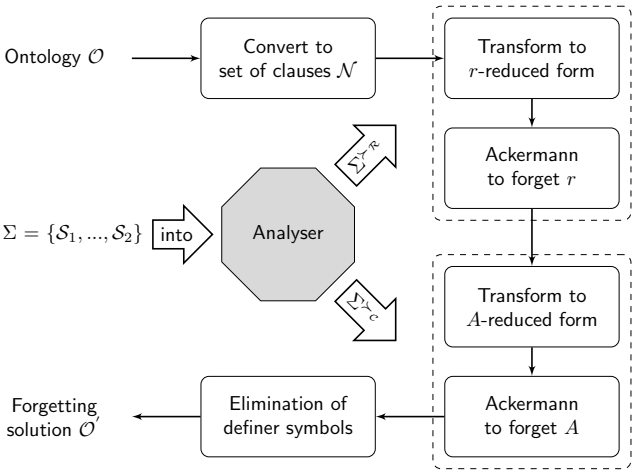

Figure 1: Overview of the forgetting method

The forgetting phase is an iteration of several rounds in which individual concept and role symbols are eliminated. An important feature of the method is that concept symbols and role symbols are forgotten in a focused way, i.e., the rules for concept forgetting and the rules for role forgetting are mutually independent; concept and role symbols can thus be forgotten in any order. In the forgetting phase, if $\mathcal{S} \in \Sigma$ is a symbol to be forgotten, the idea is to transform the $\mathcal{S}$-clauses into *$\mathcal{S}$-reduced form*, so that the forgetting rules (the Ackermann rules) can be applied to eliminate $\mathcal{S}$. The conversion to $\mathcal{S}$-reduced form is performed using the rewrite rules in the underlying calculi of our method, which can be found in [4]. To provide crucial control and flexibility in how the steps are performed, auxiliary concept symbols, called *definer symbols*, are introduced in the role forgetting rounds. The final phase attempts to eliminate these definer symbols via concept forgetting.

Previous research has shown that the success rates of forgetting depend very much upon the order in which the Σ-symbols are forgotten [1, 2, 3]. Our method either follows the user-specified ordering, or it uses a heuristic analysis based on frequency counts of the Σ-symbols to generate good orderings. We refer to the maximal symbol of Σ w.r.t. the forgetting ordering $\succ$ as the *pivot* in our method.

Non-Cyclic Ackermann$^{\mathcal{C}}$

$$\frac{\mathcal{N}, C_1 \sqcup A, \ldots, C_n \sqcup A}{\mathcal{N}^A_{\neg C_1 \sqcup \ldots \sqcup \neg C_n}}$$

provided: (i) A does not occur in the C_i, and (ii) $\mathcal{N}$ is negative w.r.t. A.

Cyclic Ackermann$^{\mathcal{C}}$

$$\frac{\mathcal{N}, C_1[A] \sqcup A, \ldots, C_n[A] \sqcup A}{\mathcal{N}^A_{\mu X.(\neg C_1 \sqcup \ldots \sqcup \neg C_n)[X]}}$$

provided: (i) the C_i are negative w.r.t. A, and (ii) $\mathcal{N}$ is negative w.r.t. A.

Purify$^{\mathcal{C}}$

$$\frac{\mathcal{N}}{\mathcal{N}^A_{(\neg)\top}}$$ provided: $\mathcal{N}$ is positive (negative) w.r.t. A.

Figure 2: Rules for forgetting pivot $A \in \mathsf{N_C}$

Theorem 1 *For any $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-ontology $\mathcal{O}$ and any set $\Sigma \subseteq \text{sig}(\mathcal{O})$ of symbols to be forgotten, the method always terminates and returns a set $\mathcal{O}'$ of clauses. If $\mathcal{O}'$ does not contain any Σ-symbols, the method was* successful*. $\mathcal{O}'$ is then a solution of forgetting the symbols in Σ from $\mathcal{O}$. If neither $\mathcal{O}$ nor $\mathcal{O}'$ uses fixpoints, $\mathcal{O}'$ is Σ-equivalent to $\mathcal{O}$ in $\mathcal{ALCOIH}(\triangledown, \sqcap)$. Otherwise, it is Σ-equivalent to $\mathcal{O}$ in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$.*

3 The Forgetting Rules

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be sets of *concept symbols* and *role symbols*.

Definition 2 (A-Reduced Form) *For $A \in \mathsf{N_C}$, a clause is in* A-reduced form *if it is negative w.r.t. A, or it has the form $A \sqcup C$, where C is an $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-concept that does not have any occurrences of A, or is negative w.r.t. A. A set $\mathcal{N}$ of clauses is in* A-reduced form *if every A-clause in $\mathcal{N}$ is in A-reduced form.*

The (Non-)Cyclic Ackermann$^{\mathcal{C}}$ rules and the Purify$^{\mathcal{C}}$ rule, given in Figure 2, are the *forgetting rules* that lead to the elimination of concept symbols in a set of clauses. For $A \in \mathsf{N_C}$ the pivot and C a concept expression, $\mathcal{N}^A_C$ denotes the set obtained from $\mathcal{N}$ by replacing every occurrence of A by C. The (Non-)Cyclic Ackermann$^{\mathcal{C}}$ rules are applied only if $\mathcal{N}$ is in A-reduced form. The Purify$^{\mathcal{C}}$ rule can be applied anytime provided A is *pure* in $\mathcal{N}$, i.e., every occurrence of A in $\mathcal{N}$ is positive or negative (under an even number of explicit and implicit negations or otherwise).

Definition 3 (r-Reduced Form) *For $r \in \mathsf{N_R}$, a clause is in* r-reduced form *if it has the form $C \sqcup \forall r.D$ or $C \sqcup \neg\forall r \sqcap Q.D$, where C and D are concepts that do not contain any occurrence of r and Q is a role that does not contain any occurrence of r; or it has the form $\neg S \sqcup r$ or $S \sqcup \neg r$, where S is a role symbol or an inverted role symbol. A set $\mathcal{N}$ of clauses is in* r-reduced form *if every r-clause in $\mathcal{N}$ is in r-reduced form.*

Finding the r-reduced form of a clause is not always possible, unless definer symbols are introduced. *Definer symbols* are specialised concept symbols that do not occur in the present ontology, and are introduced as follows: given a clause of the form $C \sqcup \forall r^{(-)}.D$ or $C \sqcup \neg\forall r^{(-)}.D$, with

Ackermann$^{\mathcal{R}}$

$$\frac{\mathcal{N}, \boxed{C^1 \sqcup \neg\forall r \sqcap Q^1.D^1, \ldots, C^k \sqcup \neg\forall r \sqcap Q^k.D^k}, \boxed{\neg T^1 \sqcup r, \ldots, \neg T^u \sqcup r}, C_1 \sqcup \forall r.D_1, \ldots, C_m \sqcup \forall r.D_m, \neg r \sqcup S_1, \ldots, \neg r \sqcup S_n}{\mathcal{N}, \boxed{\mathcal{T}\text{-}Block^{\mathcal{H}}(1,m), \ldots, \mathcal{T}\text{-}Block^{\mathcal{H}}(k,m)}, \boxed{\mathcal{R}\text{-}Block^{\mathcal{C}}(1,m), \ldots, \mathcal{R}\text{-}Block^{\mathcal{C}}(u,m)}, \boxed{\mathcal{R}\text{-}Block^{\mathcal{R}}(1,n), \ldots, \mathcal{R}\text{-}Block^{\mathcal{R}}(u,n)}}$$

provided: (i) r does not occur in $\mathcal{N}$, and (ii) $\mathcal{N}$ is in r-reduced form.

Purify$^{\mathcal{R}}$

$$\frac{\mathcal{N}}{\mathcal{N}^r_{(\neg)\triangledown}}$$ provided: $\mathcal{N}$ is positive (negative) w.r.t. r.

Notation in the Ackermann$^{\mathcal{R}}$ rule ($1 \le j \le k$, $1 \le l \le u$):

$\mathcal{T}\text{-}Block^{\mathcal{H}}(j,m)$ denotes the set

$$\{C^j \sqcup C^Y \sqcup \neg\forall \mathcal{H}^j.(D^j \sqcup D^Y) | Y \subseteq [m]\}, \text{ where}$$

$$C^Y = \begin{cases} \neg\top & \text{if } Y = \emptyset \\ \bigsqcup_{i \in Y} C_i & \text{otherwise,} \end{cases} \quad D^Y = \begin{cases} \neg\top & \text{if } Y = \emptyset \\ \bigsqcup_{i \in Y} \neg D_i & \text{otherwise,} \end{cases}$$

$$\text{and} \quad \mathcal{H} = \begin{cases} S_1 \sqcap \ldots \sqcap S_n \sqcap Q^j & \text{if } \mathcal{P}^-_{\mathcal{R}} \neq \emptyset \\ \triangledown \sqcap Q^j & \text{otherwise.} \end{cases}$$

$\mathcal{R}\text{-}Block^{\mathcal{C}}(l,m)$ denotes the set

$$\{C_1 \sqcup \forall T^l.D_1, \ldots, C_m \sqcup \forall T^l.D_m\}.$$

$\mathcal{R}\text{-}Block^{\mathcal{R}}(l,n)$ denotes the set $\{\neg T^l \sqcup S_1, \ldots, \neg T^l \sqcup S_n\}$.

Figure 3: Rules for forgetting pivot $r \in \mathsf{N_R}$

r being the pivot and occurring in $\mathcal{Q} \in \{C, D\}$, the definer symbols are used as substitutes, incrementally replacing C and D until neither contains any occurrences of r. A new clause $\neg\mathcal{D} \sqcup \mathcal{Q}$ is added to the clause set for each replaced subconcept $\mathcal{Q}$, where $\mathcal{D}$ is a fresh definer symbol.

Given a set $\mathcal{N}$ of clauses with $r \in \mathsf{N_R}$ being the pivot, once $\mathcal{N}$ has been transformed into r-reduced form, we apply the Ackermann$^{\mathcal{R}}$ rule given in Figure 3 to eliminate r. The Purify$^{\mathcal{R}}$ rule can be applied anytime provided r is pure.

References

[1] P. Koopmann and R. A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In *Proc. IJCAR'14*, volume 8562 of *Lecture Notes in Computer Science*, pages 434–448. Springer, 2014.

[2] R. A. Schmidt. The Ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.

[3] Y. Zhao and R. A. Schmidt. Concept Forgetting in $\mathcal{ALCOI}$-Ontologies Using an Ackermann Approach. In *Proc. ISWC'15*, volume 9366 of *Lecture Notes in Computer Science*, pages 587–602. Springer, 2015.

[4] Y. Zhao and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-ontologies. In *Proc. DL'16*, volume 1577 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. To appear in *Proc. IJCAI'16.*

Rewriting $\mathcal{ALC}$-TBoxes of Depth One via Resolution

Fabio Papacchini

Department of Computer Science, University of Liverpool, Fabio.Papacchini@liverpool.ac.uk

Abstract: Former studies show that there exists a class of $\mathcal{ALC}$-TBoxes of depth one (i.e., with no nested role restrictions) such that answering conjunctive queries is in PTIME in data complexity. We propose ideas on how to rewrite such TBoxes into Horn-$\mathcal{ALC}$-TBoxes of depth one via a resolution-based approach.

1 Introduction

Ontology-based data access (OBDA) has received particular attention from the research community in the last years. When enhancing query answering by the use of an ontology, one of the most natural questions arising is 'how does the ontology affect the complexity of query answering?'. Studies focusing on a wide variety of ontology languages and queries show that query answering becomes easily intractable. This is the case, for example, for the $\mathcal{ALC}$ language, where the worst-case data complexity it is known to be CONP-COMPLETE when answering conjunctive queries (CQs). It is, however, possible to identify $\mathcal{ALC}$-TBoxes for which CQ answering is tractable by adopting a fine-grained approach.

[2] shows that for $\mathcal{ALC}$-TBoxes of depth one (i.e., with no nested role restrictions) there exists a P/CONP dichotomy, and it presents a semantic characterisation for TBoxes where CQ answering is in PTIME. Specifically, [2] proves the equivalence between materialisability, unravelling tolerance and the ABox disjunction property for $\mathcal{ALC}$-TBoxes of depth one, and that CQ answering in the presence of these properties is in PTIME for data complexity. For the purpose of this abstract we give only the formal definition of the ABox disjunction property, and we refer to [2] for the other definitions. A TBox $\mathcal{T}$ has the *ABox disjunction property* if for all ABoxes $\mathcal{A}$ and $\mathcal{EL}$ concepts $C_1, \ldots, C_n$, it follows from $(\mathcal{T}, \mathcal{A}) \models C_1(a_1) \vee \ldots \vee C_n(a_n)$ that $(\mathcal{T}, \mathcal{A}) \models C_i(a_i)$ for some $i \leq n$. We recall that an $\mathcal{EL}$ concept is any concept built from $\exists$, $\sqcap$, $\top$, and any concept name A.

Recent results of our research show that for $\mathcal{ALC}$-TBoxes of depth one the ABox disjunction property can be made stronger by requiring the C_i to be of at most depth one, and that it is enough to check the ABox disjunction property over ABoxes that can be expressed as a depth one $\mathcal{EL}$ assertion. For example, the ABox $\{A(a), r(a,b), B(b), B'(b)\}$ can be represented as $(A \sqcap \exists r.(B \sqcap B'))(a)$.

2 Resolution Calculus

The idea of adopting a resolution-based method as a rewriting procedure is not new, e.g. [1].

We indicate literals by L, where $L := A \mid \neg A$, a possibly empty disjunction of literals by $\mathcal{C}_\sqcup$ or $\mathcal{D}_\sqcup$, a possibly empty conjunction of literals by $\mathcal{C}_\sqcap$ or $\mathcal{D}_\sqcap$. In this context, arbitrary concepts C and D are of the form $C := L \mid \forall r.\mathcal{C}_\sqcup \mid \exists r.\mathcal{C}_\sqcap$. Finally, we indicate the complement by an overline (e.g., $\overline{L}$ is the complement of a literal, $\overline{\mathcal{C}_\sqcap}$ is the complement of a conjunction of literals).

As the procedure is a resolution-based procedure, the input is a set of DL clauses. The set of clauses is computed using the common CNF transformation, with the proviso that $\forall r$ is distributed over conjunctions and $\exists r$ is distributed over disjunctions. Given the aforementioned definitions, a clause is just a disjunction of arbitrary concepts, and we indicate clauses by $\mathcal{C}$ and $\mathcal{D}$. This normal form and representation of clauses is possible because we assume the input to be an $\mathcal{ALC}$-TBox of depth one.

To simplify the calculus we assume that any clause in the calculus does not present any repetition, neither in a disjunction nor in a conjunction. Table 1 shows the rules of the resolution calculus, which we call Res.

Subsumption between clauses can be defined on a completely syntactic level. First, let us define an ordering between role restricted concepts. We say that $\forall r.\mathcal{C}_\sqcup \preceq \forall r.\mathcal{D}_\sqcup$ if $\mathcal{C}_\sqcup \subseteq \mathcal{D}_\sqcup$, and that $\exists r.\mathcal{C}_\sqcap \preceq \exists r.\mathcal{D}_\sqcap$ if $D_\sqcap \subseteq \mathcal{C}_\sqcap$. We indicate that a clause $\mathcal{C}$ *subsumes* a clause $\mathcal{D}$ by $\mathcal{C} \preceq \mathcal{D}$. $\mathcal{C} \preceq \mathcal{D}$ holds if either

- for all $L \in \mathcal{C}$ it is the case that $L \in \mathcal{D}$,
- for all $\exists r.\mathcal{C}_\sqcap \in \mathcal{C}$ there exists $\exists r.\mathcal{D}_\sqcap \in \mathcal{D}$ s.t. $\exists r.\mathcal{C}_\sqcap \preceq \exists r.\mathcal{D}_\sqcap$, and
- for all $\forall r.\mathcal{C}_\sqcup \in \mathcal{C}$ there exists $\forall r.\mathcal{D}_\sqcup \in \mathcal{D}$ s.t. $\forall r.\mathcal{C}_\sqcup \preceq \forall r.\mathcal{D}_\sqcup$,

or,

$\mathcal{C} = L_1 \sqcup \ldots \sqcup L_n$ and $\forall r.(L_1 \sqcup \ldots \sqcup L_m) \in \mathcal{D}$ where $m \geq n$.

A clause is *redundant* if it is subsumed or it is a tautology.

Clause simplification is applied during the saturation process, and we indicate the exhaustive application of simplification rules to a clause $\mathcal{C}$ by SIMP($\mathcal{C}$). Apart from the common simplification rules (e.g., removing $\exists r.\bot$ from a clause), two rules need to be illustrated. First,

$$(\mathbf{Cond})\ \frac{\mathcal{C} \sqcup \forall r.\mathcal{C}_\sqcup \sqcup \forall r.\mathcal{C}'_\sqcup}{\mathcal{C} \sqcup \forall r.\mathcal{C}'_\sqcup}\ \ \mathcal{C}_\sqcup \subseteq \mathcal{C}'_\sqcup\,.$$

The $(\mathbf{Cond})$ rule represents a condensation step. The first-order translation helps to illustrate the purpose of the rule.

$$(BR)\ \frac{\mathcal{C} \sqcup C \qquad \overline{C} \sqcup \mathcal{D}}{\mathcal{C} \sqcup \mathcal{D}} \qquad (\forall\bot)\ \frac{\mathcal{C} \sqcup \forall r.\bot \qquad \exists r.\mathcal{D}_\sqcap \sqcup \mathcal{D}}{\mathcal{C} \sqcup \mathcal{D}}$$

$$(BR\forall)\ \frac{\mathcal{C}_\sqcup \sqcup L \qquad \mathcal{D} \sqcup \forall r.(\overline{L} \sqcup \mathcal{D}_\sqcup)}{\mathcal{D} \sqcup \forall r.(\mathcal{C}_\sqcup \sqcup \mathcal{D}_\sqcup)} \qquad (BR\exists)\ \frac{\bigsqcup_i L_i \sqcup L \qquad \mathcal{C} \sqcup \exists r.(\overline{L} \sqcap \mathcal{C}_\sqcap)}{\mathcal{C} \sqcup \bigsqcup_i \exists r.(L_i \sqcap \mathcal{C}_\sqcap)}$$

$$(\forall\forall)\ \frac{\mathcal{C} \sqcup \forall r.(\mathcal{C}_\sqcup \sqcup L) \qquad \mathcal{D} \sqcup \forall r.(\overline{L} \sqcup \mathcal{D}_\sqcup)}{\mathcal{C} \sqcup \mathcal{D} \sqcup \forall r.(\mathcal{C}_\sqcup \sqcup \mathcal{D}_\sqcup)} \qquad (\forall\exists)\ \frac{\mathcal{C} \sqcup \forall r.(\bigsqcup_i L_i \sqcup L) \qquad \mathcal{D} \sqcup \exists r.(\overline{L} \sqcap \mathcal{D}_\sqcap)}{\mathcal{C} \sqcup \mathcal{D} \sqcup \bigsqcup_i \exists r.(L_i \sqcap \mathcal{D}_\sqcap)}$$

Table 1: Rules of the resolution calculus Res

Consider as an example the clause $\forall r.(A \sqcup B) \sqcup \forall r.(A \sqcup B \sqcup C)$, its first-order translation is as follows.

$$\neg r(x,y) \vee A(y) \vee B(y) \vee \neg r(x,z) \vee A(z) \vee B(z) \vee C(z)$$

A first-order prover would substitute y with z, and then apply factoring as much as possible. The resulting clause would be $\neg r(x,z) \vee A(z) \vee B(z) \vee C(z)$, which subsumes the original one. The $(\mathbf{Cond})$ rule performs exactly the same simplification.

Second,

$$(\mathbf{Taut})\ \frac{\mathcal{C} \sqcup \forall r.\mathcal{C}_\sqcup \sqcup \exists r.\mathcal{C}_\sqcap}{\top}\ \ \overline{\mathcal{C}_\sqcap} \subseteq \mathcal{C}_\sqcup\ .$$

The $(\mathbf{Taut})$ rule recognises particular tautologies, and it is necessary for the second step of our procedure. The idea behind this rule is that a clause such as $\forall r.(A \sqcup B) \sqcup \exists r.\neg A$ is equivalent to the TBox axiom $\exists r.(\neg A \sqcap \neg B) \sqsubseteq \exists r.\neg A$, where its tautological nature is more evident.

Given a set $\mathcal{N}$ of DL clauses, we indicate by $Res^*_\preceq(\mathcal{N})$ the saturation of $\mathcal{N}$ w.r.t. Res and the redundancy criterion.

Claim 1. *Let $\mathcal{N}$ be a set of DL clauses of an $\mathcal{ALC}$-TBox of depth one. Then $Res^*_\preceq(\mathcal{N})$ contains all non-redundant consequences of $\mathcal{N}$ with depth less or equal one.*

Soundness of the calculus can be easily proved, and refutational completeness of $Res^*_\preceq$ follows from Claim 1.

Claim 2. *$\mathcal{N}$ is unsatisfiable iff $\bot \in Res^*_\preceq(\mathcal{N})$.*

Claim 3. *$Res^*_\preceq(\mathcal{N})$ terminates for any set $\mathcal{N}$ of clauses. (There are only exponentially many non redundant consequences of depth one.)*

3 Procedure

Let us define a function POS as follows.

$$\text{POS}(C) = \begin{cases} \exists r.\bigsqcap A_i & \text{if } C = \exists r.\mathcal{C}_\sqcap \text{ and } A_i \in \mathcal{C}_\sqcap \\ C & \text{otherwise} \end{cases}$$

The POS function can be extended to clauses.

Let $\mathcal{N}$ be the set of DL clauses resulting from an $\mathcal{ALC}$-TBox of depth one. The procedure is divided into two parts. First, the resolution calculus Res is exhaustively applied to $\mathcal{N}$, resulting in the set $S = Res^*_\preceq(\mathcal{N})$. Second, let $S^+ = \{\mathcal{C} \mid \mathcal{C} = \text{SIMP}(\text{POS}(\mathcal{C}')), \mathcal{C}' \in S\}$, and let $S^+_\preceq$ be the result of applying redundancy elimination to S^+.

Claim 4. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox of depth one. $\mathcal{T}$ has the ABox disjunction property iff any clause $\mathcal{C} \in S^+_\preceq$ is a Horn-clause.*

The intuition for Claim 4 is as follows. First, any clause in $S^+_\preceq$ can be rewritten as a TBox axiom $C \sqsubseteq D$ where C is an $\mathcal{EL}$ concept. ($\Rightarrow$) Take a non-Horn clause in $S^+_\preceq$, build an ABox $\mathcal{A}$ from the left hand-side of its TBox representation, and prove that the ABox disjunction property fails on $\mathcal{A}$. ($\Leftarrow$) If the ABox disjunction property fails on an ABox $\mathcal{A}$, then there exists a non-Horn clause in $S^*_\preceq$.

Claim 5. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox of depth one. If $\mathcal{T}$ has the ABox disjunction property, then $S^+_\preceq$ is a Horn rewriting preserving CQ-answering.*

4 Conclusion

The ideas proposed in this abstract represent a possible syntactic characterisation for $\mathcal{ALC}$-TBoxes of depth one for which CQ answering is tractable. Due to the normal form transformation and the size of the resulting set of clauses, the procedure is double exponential in the size of the original TBox. This result agrees with the upper bound we obtained via a different approach. Evaluations of BioPortal ontologies[1], however, shows that the normal form transformation does not usually result in a substantial increase in size.

Acknowledgements

All the ideas in this abstract would have not been possible without the help and discussions with André Hernich, Boris Konev and Frank Wolter.

References

[1] M. Kaminski, Y. Nenov, and B. Cuenca Grau. Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In *Proc. of RR'14*, volume 8741 of *LNCS*, pages 76–91. Springer, 2014.

[2] C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. of KR'12*. AAAI Press, 2012.

[1] http://bioportal.bioontology.org

Metric Temporal Logic Translations over the Naturals

Clare Dixon Ullrich Hustadt Ana Ozaki

Department of Computer Science, University of Liverpool
Liverpool, L69 3BX United Kingdom
{CLDixon, U.Hustadt, anaozaki}@liverpool.ac.uk

Abstract: We study translations from Metric Temporal Logic (MTL) over the natural numbers to Linear Temporal Logic (LTL). In particular, we present two approaches for translating from MTL to LTL which preserve the EXPSPACE complexity of the satisfiability problem for MTL. Our translations, thus, allow us to utilise LTL provers to solve MTL satisfiability problems.

1 Introduction

Linear and branching-time temporal logics have been used for the specification and verification of reactive systems. In linear-time temporal logic [4] we can, for example, express that a formula ψ holds now or at some point in the future using the formula $\Diamond\psi$ (ψ holds eventually). However, some applications require not just that a formula ψ will hold eventually but that it holds within a particular time-frame for example between 3 and 7 moments from now. To express metric constraints, a range of Metric Temporal Logics (MTL) have been proposed, considering different underlying models of time and operators allowed [3]. Here we use MTL with pointwise discrete semantics, following [1], where each state in the sequence is mapped to a time point on a time line isomorphic to the natural numbers. In this instance of MTL, temporal operators are annotated with certain finite as well as infinite intervals, for example, $\Box_{[2,4]}p$ means that p should hold in all states that occur between the interval $[2, 4]$ of time, while $\Box_{[2,\infty)}p$ means that p should hold in all states that occur at least 2 moments from now. We provide two satisfiability preserving translations from MTL into LTL. Both translations are polynomial in the size of the MTL formula and the largest constant occurring in an interval (although exponential in the size of the MTL formula due to the binary encoding of the constants). Since the satisfiability problem for LTL is PSPACE [5], our translations preserve the EXPSPACE complexity of the MTL satisfiability problem [1].

2 Metric Temporal Logic Translations

We briefly state the syntax and semantics of LTL and MTL. Let $\mathcal{P}$ be a (countably infinite) set of propositional symbols. Well formed formulae in LTL are formed according to the rule: $\varphi, \psi := p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \bigcirc\varphi \mid (\varphi \mathcal{U} \psi)$ where $p \in \mathcal{P}$.

LTL Semantics. A *state sequence* σ over $(\mathbb{N}, <)$ is an infinite sequence of states $\sigma_i \subseteq \mathcal{P}$, $i \in \mathbb{N}$.

$(\sigma, i) \models p$	iff $p \in \sigma_i$
$(\sigma, i) \models (\varphi \wedge \psi)$	iff $(\sigma, i) \models \varphi$ and $(\sigma, i) \models \psi$
$(\sigma, i) \models \neg\varphi$	iff $(\sigma, i) \not\models \varphi$
$(\sigma, i) \models \bigcirc\varphi$	iff $(\sigma, i + 1) \models \varphi$
$(\sigma, i) \models (\varphi \mathcal{U} \psi)$	iff there is $k \geq i$ such that $(\sigma, k) \models \psi$ and for all $j \in \mathbb{N}$, if $i \leq j < k$ then $(\sigma, j) \models \varphi$

We denote by $\bigcirc^c$ a sequence of c next operators. Further connectives can be defined as usual: $p \vee \neg p \equiv \mathbf{true}$, $\mathbf{true} \equiv \neg(\mathbf{false})$, $\mathbf{true}\mathcal{U}\varphi \equiv \Diamond\varphi$ and $\Diamond\varphi \equiv \neg\Box\neg\varphi$. MTL formulae are constructed in a way similar to LTL, with the difference that temporal operators are now bounded by an interval I with natural numbers as end-points or ∞ on the right side. Note that since we work with natural numbers as end-points we can assume w.l.o.g that all our intervals are of the form $[c_1, c_2]$ or $[c_1, \infty)$, where $c_1, c_2 \in \mathbb{N}$. Well formed formulae in MTL are formed according to the rule: $\varphi, \psi := p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \bigcirc_I\varphi \mid (\varphi \mathcal{U}_I \psi)$ where $p \in \mathcal{P}$.

MTL Semantics. A *strict timed state sequence* $\rho = (\sigma, \tau)$ over $(\mathbb{N}, <)$ is a pair consisting of an infinite sequence σ of states $\sigma_i \subseteq \mathcal{P}$, $i \in \mathbb{N}$, and a function $\tau : \mathbb{N} \to \mathbb{N}$ that maps every i corresponding to the i-th state to a time point $\tau(i)$ such that $\tau(i) < \tau(i + 1)$.

$(\rho, i) \models \bigcirc_I\varphi$	iff $(\rho, i + 1) \models \varphi$ and $\tau(i + 1) - \tau(i) \in I$
$(\rho, i) \models (\varphi \mathcal{U}_I \psi)$	iff there is $k \geq i$ such that $\tau(k) - \tau(i) \in I$ and $(\rho, k) \models \psi$ and for all $j \in \mathbb{N}$, if $i \leq j < k$ then $(\rho, j) \models \varphi$

We omit propositional cases, which are as in LTL. Further connectives can be defined as usual: $\mathbf{true}\mathcal{U}_I\varphi \equiv \Diamond_I\varphi$ and $\Diamond_I\varphi \equiv \neg\Box_I\neg\varphi$. To transform an MTL formula into Negation Normal Form, one uses the constrained dual until $\tilde{\mathcal{U}}_I$ operator [3], defined as $(\varphi\tilde{\mathcal{U}}_I\psi) \equiv \neg(\neg\varphi\mathcal{U}_I\neg\psi)$. An MTL formula φ is in *Negation Normal Form (NNF)* iff the negation operator ($\neg$) occurs only in front of propositional variables. An MTL formula φ is in *Flat Normal Form (FNF)* iff it is of the form $p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \rightarrow \psi_i)$ where p_0, p_i are propositional variables or $\mathbf{true}$ and ψ_i is either a formula of propositional logic or it is of the form $\bigcirc_I\psi_1$, $\psi_1\mathcal{U}_I\psi_2$ or $\psi_1\tilde{\mathcal{U}}_I\psi_2$ where ψ_1, ψ_2 are formulae of propositional logic. The transformations into NNF and FNF are satisfiability preserving and can be performed in polynomial time. From now on assume that our MTL formulae are in NNF and FNF.

From MTL to LTL: encoding 'gaps' We translate MTL formulae for discrete time models into LTL using a new propositional symbol *gap*. $\neg gap$ is true in those states σ'_j of σ' such that there is $i \in \mathbb{N}$ with $\tau(i) = j$ and *gap* is true in all other states of σ'. As shown in Table 1, we translate for example $\bigcirc_{[2,3]}p$ into: $\bigvee_{2 \leq l \leq 3}(\bigcirc^l(\neg gap \wedge p) \wedge$

MTL	LTL Gap Translation
$(\bigcirc_{[c_1,\infty)}\alpha)^\sharp$	$(\bigwedge_{1\leq k<c_1} \bigcirc^k gap)$ $\wedge \bigcirc^{c_1}(gap\mathcal{U}(\alpha \wedge \neg gap))$
$(\bigcirc_{[c_1,c_2]}\alpha)^\sharp$	$\bigvee_{c_1\leq l\leq c_2}(\bigcirc^l(\neg gap \wedge \alpha)$ $\wedge \bigwedge_{1\leq k<l} \bigcirc^k gap)$
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$(\bigwedge_{0\leq k<c_1} \bigcirc^k(gap \vee \alpha))$ $\wedge \bigcirc^{c_1}((gap \vee \alpha)\mathcal{U}(\neg gap \wedge \beta))$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\bigvee_{c_1\leq l\leq c_2}(\bigcirc^l(\neg gap \wedge \beta)$ $\wedge \bigwedge_{0\leq k<l} \bigcirc^k(gap \vee \alpha))$

Table 1: MTL to LTL translation using 'gap' where α, β are propositional logic formulae and $c_1, c_2 > 0$.

$\bigwedge_{1\leq k<l} \bigcirc^k gap)$.

Theorem 1. Let $\varphi = p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \rightarrow \psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp = p_0 \wedge \bigwedge_i \Box(p_i \rightarrow (\neg gap \wedge \psi_i^\sharp))$ be the result of replacing each ψ_i in φ by $\psi_i^\sharp$ as in Table 1. Then, φ is satisfiable if, and only if, $\varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond \neg gap)$ is satisfiable.

From MTL to LTL: encoding time differences Let $C-1$ be the greatest number occurring in an interval in an MTL formula φ or 1, if none occur. W.l.o.g., we can consider only strict timed state sequences where the time difference from a state to its previous state is bounded by C [2]. Then, we can encode time differences with a set $\Pi_\delta = \{\delta_i^- \mid 1 \leq i \leq C\}$ of propositional variables where each δ_i^- represents a time difference of i w.r.t. the previous state (one could also encode the time difference to the next state instead of the difference from the previous state). We also encode variables of the form s_m^n with the meaning that 'the sum of the time differences from the last n states to the current state is m'. For our translation, we only need to define these variables up to sums bounded by $2 \cdot C$.

To simplify the presentation, we use two additional n-ary boolean operators $\oplus_{=1}$ and $\oplus_{\leq 1}$. If $S = \{\varphi_1, \ldots, \varphi_n\}$ is a finite set of LTL formulae, then $\oplus_{=1}(\varphi_1, \ldots, \varphi_n)$, also written $\oplus_{=1}S$, is a LTL formula. Let σ' be a state sequence and $i \in \mathbb{N}$. Then $(\sigma', i) \models \oplus_{=1}S$ iff $(\sigma', i) \models \varphi_j \in S$ for exactly one $\varphi_j \in S$, $1 \leq j \leq n$. Similarly, $(\sigma', i) \models \oplus_{\leq 1}S$ iff $(\sigma', i) \models \varphi_j \in S$ for at most one $\varphi_j \in S$, $1 \leq j \leq n$. Let S_C be the conjunction of the following:

1. $\bigcirc\Box \oplus_{=1} \Pi_\delta$, for $\Pi_\delta = \{\delta_k^- \mid 1 \leq k \leq C\}$;
2. $\Box(\delta_k^- \leftrightarrow s_k^1)$, for $1 \leq k \leq C$;
3. $\Box \oplus_{\leq 1} \Pi^i$, for $1 \leq i \leq 2 \cdot C$ and $\Pi^i = \{s_j^i \mid i \leq j \leq 2 \cdot C\}$;
4. $\Box(\bigcirc s_k^1 \wedge s_l^j \rightarrow \bigcirc s_{l+k}^{j+1})$, for $1 < j+1 \leq l+k \leq 2 \cdot C$.

Point 1 ensures that at all times, the time difference k from the current state to the previous (if it exists) is uniquely encoded by the variable δ_k^-. In Point 2 we have that the sum of the difference of the last state to the current, encoded by s_k^1, is exactly δ_k^-. Point 3 ensures that at all times we cannot have more than one value for the sum of the time differences of the last i states. Finally, Point 4 has the propagation of sum variables: if the sum of the last j states is l and the time difference to the next is k then the next state should have that the sum of the last $j+1$ states is $l+k$. As shown in Table 2, we translate, for example, $\bigcirc_{[2,3]}p$ into $\bigcirc((\delta_2^- \vee \delta_3^-) \wedge p)$.

MTL	LTL Time Difference Translation
$(\bigcirc_{[c_1,\infty)}\alpha)^\sharp$	$\bigcirc((\bigvee_{c_1\leq i\leq C} \delta_i^-) \wedge \alpha)$
$(\bigcirc_{[c_1,c_2]}\alpha)^\sharp$	$\bigcirc((\bigvee_{c_1\leq i\leq c_2} \delta_i^-) \wedge \alpha)$
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$\bigvee_{1\leq i\leq c_1}(\bigcirc^i((\bigvee_{c_1\leq j\leq c_1+C} s_j^i) \wedge \alpha\mathcal{U}\beta)$ $\wedge (\bigwedge_{0\leq k<i} \bigcirc^k \alpha))$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\bigvee_{1\leq i\leq c_2}(\bigcirc^i((\bigvee_{c_1\leq j\leq c_2} s_j^i) \wedge \beta)$ $\wedge (\bigwedge_{0\leq k<i} \bigcirc^k \alpha))$

Table 2: MTL to LTL translation where α, β are propositional logic formulae and $c_1, c_2 > 0$.

Theorem 2. Let $\varphi = p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \rightarrow \psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp = p_0 \wedge \bigwedge_i \Box(p_i \rightarrow \psi_i^\sharp)$ be the result of replacing each ψ_i in φ by $\psi_i^\sharp$ as in Table 2. Then, φ is satisfiable if, and only if, $\varphi^\sharp \wedge S_C$ is satisfiable.

3 Conclusion

We presented two translations from MTL to LTL. These translations provide a route to practical reasoning about MTL over natural numbers via LTL solvers. Our second translation and the MTL decision procedure presented in [1] are based on time differences and use the bounded model property. However, the translations using 'gap' do not require this property.

References

[1] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):35–77, 1993.

[2] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.

[3] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS '08, pages 1–13. Springer, 2008.

[4] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57. IEEE Computer Society, 1977.

[5] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985.

Universal Algebra for Syntax with Bindings

Lorenzo Gheri

Middlesex University, London, lg571@live.mdx.ac.uk

Abstract: This is a short outline of my Ph.D. project plans. This is a work in progress which aims at a general theory of Syntax with Bindings, which will come together with a formalization in Isabelle/HOL and some applications. The major contributions will be dealing with coinduction, infinite objects and generalized notions of bindings.

1 Introduction

The central aim of the project we are here presenting is to develop a framework for specification of, and reasoning about, *formal systems* of relevance in computer science and mathematical logic. We call "formal system" any logical theory comprehending a syntax and some semantics; these structures are heavily employed in modelling the semantics of programming languages, type systems and logical deduction.

Notably, the area of programming languages has seen an explosion of formal systems aimed at capturing a wide range of computational and logical analysis aspects. For example, there is research in making programming languages more secure and specification-conforming, reflected in a variety of type systems (see Pierce's monograph [5]) and formal analysis tools, such as proof assistants , model checkers , J-Flow and other security tools, CSP-based tools etc. All this research relies on methodologies for rigorous, formal reasoning and computation.

Even though formal systems come in a wide variety, there are some fundamental mechanisms that seem common to all (or most) of them. These include the notions of *binding* and *scoping* aimed at dealing with locality: a program variable can be bound in a certain submodule (such as a function or a procedure) or a logic variable can be bound in the scope of a quantifier. Binding typically comes in pair with the notion of instantiation or substitution. For example, when a programming function is applied to a value, its execution proceeds by instantiating its formal parameter with that value and then evaluating the body of the function. In formal logic, this corresponds to the notion of substituting a term for a variable. Often, the operational behaviour of programs can be understood in terms of the interplay between bindings and instantiation/substitution. The laws that govern this interplay obey certain patterns that can be conceptualized in isolation from the particular language.

Another important aspect in formal systems is dealing with infinite behaviour. A finite syntactic object may produce an infinite semantic object by unfolding (as in operational semantics) or by interpretation (as in denotational semantics). The syntactic world is guided by bindings and substitution, whereas, in mirror, the semantic world is guided by functional objects and application. Likewise, the syntactic world can be specified *inductively*, whereas the semantic world is best describable *coinductively*.

Our work will contribute to the identification of such patterns and their presentation as a general theory, in the style of Universal Algebra, widely applicable for many formal systems. This work will be guided by concrete applications, including well-known challenges in formal reasoning and cases of notoriously difficult or tedious constructions. We plan to validate and illustrate our results on concrete case studies conducted in the proof assistant Isabelle/HOL [1].

2 The General Framework

The starting point of this project is that formal systems share a common syntactic structure. With the development of a unifying theory for syntax with bindings we aim at building a formalized framework, which can capture many formal systems and constructions as particular instances. In this section we sketch an embryonic version of the syntactic part of this framework.

First we need to be given a set of *sorts*, which are specified when the framework is instantiated. The intuition is that each object of the system is of a certain sort; if a sort is allowed to contain variables, it is said a *variable sorts*.

Example. In simply typed lambda calculus there are just two sorts, one for terms and one for types. As we need term variables, terms (but not types) are elements of a certain variable sort.

The next step is to populate our system. For every sort we give some elements, *terms*, which will form the respective *syntactic category*. Usually some auxiliary objects are given, as a base case; we call these the *context*. For example, the context must contain a countably infinite set of *variables* for every variable sort. The way we build every other element is with a set of *constructors*, given for the specific formal system. Every constructor comes with its *arity*; here we must also declare which are the *bound* variables in which term.

In short, we have a set, the syntactic category, for every sort, the elements of which are called *terms* and are defined inductively by means of constructors.

Example. Let us consider the constructors of the simply typed lambda calculus. The syntactic category Type of types has one constructor b for each type constant and another constructor $\rightarrow$ which takes two types and builds a type

(the usual type of functions). The syntactic category Term of terms contains as a subset a countably infinite set of variables. Its constructor are:

- a constructor for every term constant c;
- a constructor for applications, which from two terms M and N builds a third one MN;
- a constructor for abstractions, which binds a variable x of a certain type σ in the term M, building the term $\lambda x : \sigma.M$.

As for inference systems, the theory of Nominal Logic has already taken care of inductive ones [6], by means of swapping, equivariance and freshness. For the coinductive case however, such a general theory still need to be developed.

To conclude, it is worth mentioning that from this presentation it is possible to move a first automatic step into the semantics of the formal system. First we need a *domain* (a set) for every sort, in which we will interpret the terms of that sort. If the sort is a variable sort we also need a *valuation*, namely a function that maps variables in elements of the appropriate domain. To conclude we must have a function for every constructor, in a way that the arity of the function matches the arity of the constructor. Once we are given these objects we can define the *interpretation* for every term by structural recursion, with a standard treatment for bindings.

3 Objectives

The central objective of our project is to develop a Universal Algebra for Syntax with Bindings, that improves on the state of art.

The first step will be to develop a solid and unitary theory, which addresses notoriously problematic features, especially substitution in terms modulo alpha equivalence, and integrates syntax and semantics. Here the progress will be in terms of generality, rigour and a formalization in Isabelle/HOL.

The universal theory I am proposing will also allow flexible bindings, including recursively specified bindings (e.g. *records* and *patterns* [2]).

While Nominal Logic [6] covers largely the theory for inductive objects, there are no such comprehensive works for coinductively defined objects involving bindings. Moreover our work will take care of codatatypes, and in general objects with no finite support. This means that for these it is not always possible to generate a fresh variable (i.e. different from every other in the object) just because their syntactic structure involves a finite number of variables. A significant example of these structure are infinite trees, as Bohm trees in Barendregt [3] or as the semantics naturally associated to infinite terms by repeatedly unfolding.

In short, here the main goal is to achieve a formal general treatment of infinite objects modulo α-equivalence and of corecursion and coinduction for binding structures.

Our project will also provide some general result for the denotational semantics of a generic formal system. In particular it will capture in a general way the substitution lemma and the fact that the interpretation of the term does not depend on the valuation for the non-free variables in that term.

Moreover, the framework is intended to grasp a more complex notion of bindings, for example the recursive binding of the System F (as in the POPLmark document [2] and in Pierce [5]). Here techniques from category theory will be heavily involved, such as Bounded Natural Functors [7].

In terms of applications, for example this framework will be suitable for a formalization of a real programming language. Moreover it will give a tool for a general treatment of adequacy in Higher Order Abstract Syntax, in contrast with the current literature characterized by different solutions to particular examples.

A deeper theory of binders could allow to faithfully represent the behaviour of actual bindings in languages such as Java, and also provide a mathematical model to the phenomenon of entanglement in quantum computing: we could imagine a quantum system and indeed a quantum program as a term of a certain syntax, in which the entanglement link between two particles is represented by an appropriate binding of two objects in the term.

References

[1] Isabelle home page, `https://isabelle.in.tum.de/`.

[2] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic, *Mechanized Metatheory for the Masses: The POPLMark Challenge*, `http://www.seas.upenn.edu/~plclub/poplmark/`, 2005.

[3] Henk Barendregt, *The Lambda Calculus: its Syntax and Semantics*, revised ed., North-Holland, 1984.

[4] John C. Mitchell, *Foundations for Programming Languages*, MIT Press, 1996.

[5] Benjamin C. Pierce, *Types and Programming Languages*, MIT Press, 2002.

[6] Andrew M. Pitts, *Nominal Logic: A First Order Theory of Names and Binding*, Fourth International Symposium in Theoretical Aspects of Computer Software, TACS 2001, Sendai, Japan, October 29-31, 2001, Proceedings, volume 2215 of Lecture Notes in Computer Science, pages 219–242. Springer-Verlag, 2001.

[7] Dmitriy Traytel, Andrei Popescu, Jasmin Christian Blanchette, *Foundational, Compositional (Co)datatypes for Higher-Order Logic: Category Theory Applied to Theorem Proving*, LICS 2012, 596-605, IEEE.

Verifiable Autonomy using Rational Agents

Louise A. Dennis[1] Maryam Kamali[1] Michael Fisher[1]

Department of Computer Science, University of Liverpool, L.A.Dennis@liverpool.ac.uk

Abstract: We are interested in the verification of autonomous systems that use a *rational agent* to make decisions. We discuss the use of model-checking to provide guarantees about the behaviour of such systems.

1 Introduction

Hybrid autonomous systems combine low-level control algorithms with high level reasoning techniques from artificial intelligence. These systems control machines such as cars, drones and robots that move and act on the physical world. Governments, industry and the public anticipate rapid advances in these areas over the coming decades, particularly in the technologies for driverless cars, assistive robots and unmanned aircraft and it is hoped that these technologies will enrich the lives of many people, improve safety and help alleviate the problems of an ageing population. However the public is also anxious about such systems and, in particular, the safety of such systems and their potential capacity to make *stupid* decisions. For this reason, the verification and validation of autonomous systems is an area of active research.

In control engineering, autonomous systems are typically described in terms of their *sensors* and their *actuators*. Sensors provide the system with information about the state of the external world such as temperature, speed, the distance to any obstacle and so on. Actuators control the system's motors and, ultimately, its behaviour in the physical world. Control engineering has developed many algorithms which allow information from sensors to be used to determine the behaviour of the actuators, often when controlling difficult physical behaviours such as allowing a drone to hover in position, or a robot to ride a bicycle. Symbolic Artificial intelligence techniques are used when the system needs to expand beyond single activities, to situations that involve making choices or combining sequences of behaviours.

One technology for achieving this is rational agent programming in which a decision problem is framed in terms of the *beliefs* and *desires* of the system [9]. A rational agent selects programmer supplied *plans* for execution based on these beliefs and desires. In an autonomous system, the beliefs are derived from the information supplied by its sensors, the desires are goals supplied by its users or programmers and the plans are described in terms of sequences of *actions* which generally relate to algorithms in the underlying control system – for instance following a flight path.

Fig. 1 provides a high level view of such a system.

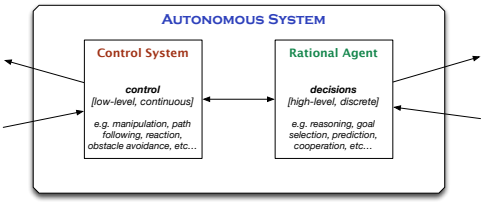

Fig. 1: A Rational Agent based Hybrid Autonomous System

2 Verification of a Single Decision Making Component

Given a system with an architecture similar to that in Fig. 1, we can seek to verify the operation rational-agent based decision making component in isolation [7, 4]. This is motivated partly by the difficulty involved in precisely reasoning about continuous behaviour but also by the observation that the decision-making component is the novel part of many of these systems.

We are primarily interested in the use of model-checking to verify that all executions of this component obey any safety parameters. We have explored the use of *program model-checking* using the AJPF system [5]. This lets us verify the actual code used to program the system which is a particularly attractive option when dealing with certification issues.

In order to restrict our verification to just the decision-making component we consider all possible sets of beliefs/perception that the agent may hold at each point in time. We show that these always lead to the selection of appropriate action by the agent. What we can not verify is that the beliefs were a correct representation of the real world, nor that the selected action has the desired effect. In effect we verify that correct decisions are made given the information available, but we do not verify the results of those decisions nor the veracity of the information.

3 Verifying Interacting Decision Making Components

While the decision-making components often contains the main novelty of an autonomous system it is important not to underestimate the effect this may have on overall system behaviour both in terms of a single autonomous system and in situations where multiple autonomous systems interact.

To investigate this we have considered a vehicle platooning scenario in which several autonomous cars attempt to form and maintain a platoon behind a car controlled by a human driver. As well as formally verifying individual agent/vehicle decisions, we also represented this system in the UPPAAL model-checker [2] which, in particular, allows the user to explore the real-time properties of a system. To do this we abstract away from the code that programs the

rational agents and represent their behaviour in a simple protocol-like form which assumes the correct execution of the individual agent programs.

We are then able to investigate whether the whole convoy is able to meet various timing requirements. For instance, given assumptions about the time taken to change lane, and the time for requests to be made and acknowledgments to be received we can verify that the time between a vehicle requesting entry to a platoon and it assuming its correct place within the platoon falls within acceptable time bounds [8].

4 Verification of Ethical Governors

Model-checking does not scale well as systems and choices increase. This is of concern in applications involving planning and scheduling (and, potentially, learning). Here we may prefer to have a smaller tractable rational agent based component concerned only with reasoning about parts of the execution which have an ethical dimension.

For this we look at the idea of *ethical governors* [1]. We view an ethical governor as a component that can act to filter, prioritise or modify the plans or actions proposed by an underlying autonomous system. It does this in order to conform to ethical considerations. This type of architecture is shown in Fig. 2.

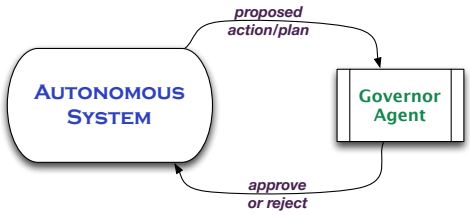

Fig. 2: A Governor Agent monitoring an Autonomous System

In this work the underlying system generates sets of plans or actions and passes these to an Ethical Governor. The governor evaluates the ethical outcomes of these plans or actions and returns either the most ethical or some set of ethically acceptable choices. We model the ethical governor as a rational agent and this allows us to use model-checking to verify the logic used by the ethical governor in order to ensure that, for instance, it only chooses an option in which a human is hurt if all other options had ethically worse outcomes [3, 6].

5 Conclusion

This abstract has surveyed work on the verification of autonomous systems. It has focused on the verification of systems which use a rational agent to make key decisions either in general, or specifically as part of ethical reasoning. We have focused primarily on the verification of these rational agents considered separately from the wider autonomous system but have also discussed preliminary work on how properties of overall system behaviour can be verified.

Acknowledgments

The work in this paper was funded by EPSRC grant Verifiable Autonomy, EP/L024845/1. `http://wordpress.csc.liv.ac.uk/va/`

Data

The case studies described in this abstract can be found at `http://mcapl.sourceforge.net` and `github.com/VerifiableAutonomy`.

References

[1] R.C. Arkin, P. Ulam, and A.R. Wagner. Moral decision making in autonomous systems: Enforcement, moral emotions, dignity, trust, and deception. *Proceedings of the IEEE*, 100(3):571 –589, 2012.

[2] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL — a Tool Suite for Automatic Verification of Real–Time Systems. In *Proc. of Workshop on Verification and Control of Hybrid Systems III*, number 1066 in Lecture Notes in Computer Science, pages 232–243. Springer–Verlag, October 1995.

[3] Louise Dennis, Michael Fisher, Marija Slavkovik, and Matt Webster. Formal verification of ethical choices in autonomous systems. *Robotics and Autonomous Systems*, pages –, 2015.

[4] Louise A. Dennis, Michael Fisher, Nicholas K. Lincoln, Alexei Lisitsa, and Sandor M. Veres. Practical verification of decision-making in agent-based autonomous systems. *Automated Software Engineering*, pages 1–55, 2014.

[5] Louise A. Dennis, Michael Fisher, Matthew Webster, and Rafael H. Bordini. Model Checking Agent Programming Languages. *Automated Software Engineering*, 19(1):5–63, 2012.

[6] Louise A. Dennis, Michael Fisher, and Alan F. T. Winfield. Towards verifiably ethical robot behaviour. In *AAAI Workshop on AI and Ethics (1st International Conference on AI and Ethics)*, Austin, TX, January 2015.

[7] Michael Fisher, Louise A. Dennis, and Matthew Webster. Verifying Autonomous Systems. *ACM Communications*, 56(9):84–93, 2013.

[8] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres. Formal Verification of Autonomous Vehicle Platooning. *ArXiv e-prints*, February 2016. Under Review.

[9] A. S. Rao and M. P. Georgeff. An Abstract Architecture for Rational Agents. In *Proc. International Conference on Knowledge Representation and Reasoning (KR&R)*, pages 439–449. Morgan Kaufmann, 1992.

KSP: A resolution-based prover for multimodal K

Cláudia Nalon[1] Ullrich Hustadt[2] Clare Dixon[2]

[1] Department of Computer Science, University of Brasília, Brasília, Brazil
nalon@unb.br
[2] Department of Computer Science, University of Liverpool, Liverpool, UK
{U.Hustadt, CLDixon}@liverpool.ac.uk

Abstract: We briefly present a new normal form and calculus for the propositional basic multimodal logic K_n. We introduce the prover KSP that implements that calculus and methods for normal form transformation. We present an experimental evaluation that compares KSP with a range of existing reasoners for K_n.

1 Introduction

Automatic theorem proving for the basic multimodal logic K_n, the logic that extends propositional logic with unary operators $[a]$ and $\langle a\rangle$ for each index a in a finite set A, has been intensely studied as it is able to express non-trivial problems in Artificial Intelligence and other areas. For instance, the description logic ALC, which has been applied to terminological representation, is a syntactic variant of K_n. Problems in Quantified Boolean Propositional Logic can also be translated into K_n. In this paper we briefly describe a new normal form and calculus for K_n, the prover KSP implementing the calculus, and present an experimental evaluation of KSP.

2 A Normal Form and Calculus for K_n

In [6] we have presented a novel resolution-based calculus for K_n. The calculus takes advantage of the following well-known properties of basic modal logic: (i) if a modal formula φ is satisfiable, then it is satisfiable in a Kripke model where the union of the accessibility relations is a tree; and (ii) in tree models, checking the local satisfiability of φ can be reduced to checking the local satisfiability of its subformulae at the depth of a model with corresponds to the modal level where those subformulae occur in φ.

The calculus operates on labelled clauses of the form $ml : C$ (literal clause), $ml : l \to [a]l'$ (positive a-clause), $ml : l \to \langle a\rangle l'$ (negative a-clause) where $ml \in \mathbb{N} \cup \{*\}$, C is a propositional clause, and l, l' are propositional literals. Intuitively, the label ml indicates the depth of the Kripke model at which the clause is true. The special label $*$ is used if a clause is true at all depths/worlds in a model and it normally only occurs in the normal form if we want to check a formula for global satisfiability instead of local satisfiability. Any formula of K_n can be transformed into an equi-satisfiable set of labelled clauses.

The calculus uses a simple form of unification. We define a partial function σ on sets of labels as follows: $\sigma(\{ml, *\}) = ml$; and $\sigma(\{ml\}) = ml$; otherwise, σ is undefined. The rules of the calculus, shown in Table 1, can only be applied if the unification of the labels involved is defined.

$$[\mathsf{LRES}]\ \frac{ml_1 : C \vee l \qquad ml_2 : D \vee \neg l}{ml : C \vee D} \qquad [\mathsf{MRES}]\ \frac{ml_1 : \ l_1 \to [a]l \qquad ml_2 : \ l_2 \to \langle a\rangle \neg l}{ml : \neg l_1 \vee \neg l_2}$$

$$[\mathsf{GEN1}]\ \frac{\begin{array}{rl} ml_1 : & l'_1 \to [a]\neg l_1 \\ & \vdots \\ ml_m : & l'_m \to [a]\neg l_m \\ ml_{m+1} : & l' \to \langle a\rangle \neg l \\ ml_{m+2} : & l_1 \vee \ldots \vee l_m \vee l \end{array}}{ml : \ \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'}$$

$$[\mathsf{GEN2}]\ \frac{\begin{array}{rl} ml_1 : & l'_1 \to [a] l_1 \\ ml_2 : & l'_2 \to [a]\neg l_1 \\ ml_3 : & l'_3 \to \langle a\rangle l_2 \end{array}}{ml : \ \neg l'_1 \vee \neg l'_2 \vee \neg l'_3}$$

$$[\mathsf{GEN3}]\ \frac{\begin{array}{rl} ml_1 : & l'_1 \to [a]\neg l_1 \\ & \vdots \\ ml_m : & l'_m \to [a]\neg l_m \\ ml_{m+1} : & l' \to \langle a\rangle l \\ ml_{m+2} : & l_1 \vee \ldots \vee l_m \end{array}}{ml : \ \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l'}$$

Table 1: Inference rules, where $ml = \sigma(\{ml_1, ml_2\})$ in LRES, MRES; $ml = \sigma(\{ml_1, \ldots, ml_{m+1}, ml_{m+2} - 1\})$ in GEN1, GEN3; and $ml = \sigma(\{ml_1, ml_2, ml_3\})$ in GEN2.

3 KSP

KSP is an implementation, written in C, of the calculus in Table 1. It also incorporates a range of methods to transform modal formulae into the clausal normal form that the calculus operates on. The main loop is based on the given-clause algorithm implementation in Otter, a variation of the set of support strategy.

Besides the basic calculus KSP uses three more inference rules including unit resolution. The user can restrict LRES by choosing ordered (clauses can only be resolved on their maximal literals with respect to an ordering chosen by the prover in such a way to preserve completeness), negative (one of the premises is a negative clause, i.e. a clause where all literals are negative), positive (one of the premises is a positive clause), or negative + ordered resolution (both negative and ordered resolution inferences are performed). The completeness of some of these refinements depends on the

particular normal form chosen. For a comprehensive description of KSP see [4], the prover itself is available at [5].

4 Evaluation

We have compared KSP with BDDTab, FaCT++ 1.6.3, InKreSAT 1.0, Spartacus 1.0, and a combination of the optimised functional translation [2] with Vampire 3.0 (OFT + Vampire).

Our benchmarks, available at [5], consist of three collections of modal formulae:

1. Modalised random QBF (MQBF) formulae [3] (1016 formulae, 617 known to be satisfiable and 399 known to be unsatisfiable).
2. LWB basic modal logic benchmark formulae, further subdivided into 18 classes [1] (1008 formulae, half are satisfiable and half are unsatisfiable by construction of the benchmark classes).
3. Randomly generated 3CNF_K formulae [7] over 3 to 10 propositional symbols with modal depth 1 or 2 (1000 formulae, 457 are known to be satisfiable and 464 are known to be unsatisfiable).

Figure 1 shows the performance of the various provers on these three collections of benchmark formulae, in particular, the graphs show how many formulae a prover can solve if given a certain amount of CPU time to solve each. KSP performs significantly better than any of the other provers on the MQBF collection. On the LWB collection overall, KSP performs about as well as BDDTab, FaCT++ and InKreSAT, while Spartacus performs best. A more detailed analysis shows that BDDTab and InKreSAT are the best performing provers on one LWB class each, OFT + Vampire on two, KSP on six, and Spartacus on eight classes. Finally, on the 3CNF_K collection, InKreSAT is the best performing prover and KSP the worst performing one.

In general, KSP performs best on formulae with high modal depth where atomic subformulae are (evenly) spread over a wide range of modal levels. The benchmarks indicate that KSP is a useful addition to the collection of provers that are already available for K_n.

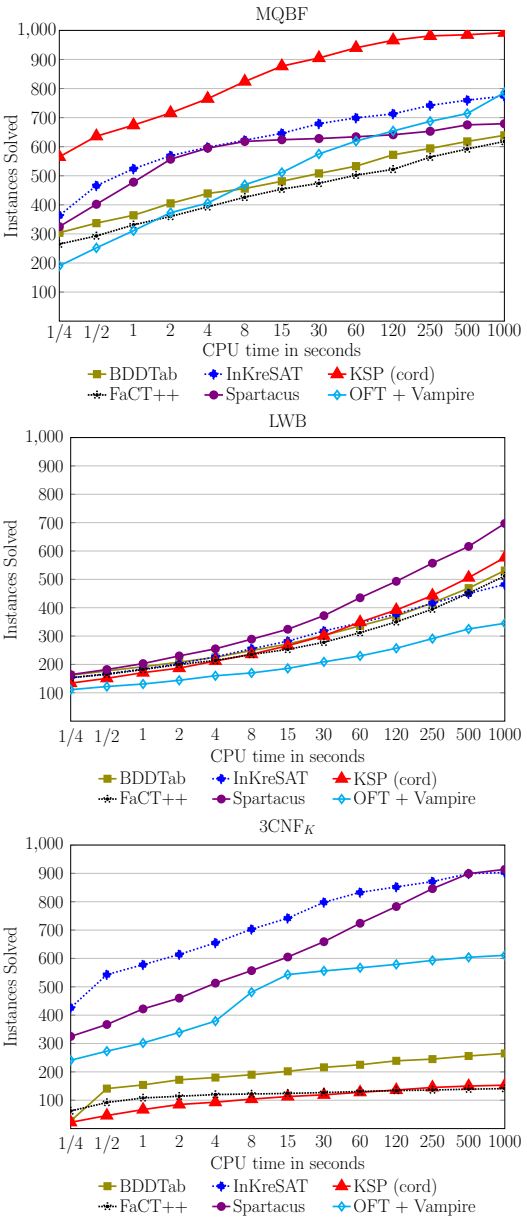

Figure 1: Benchmarking results

References

[1] Peter Balsiger, Alain Heuerding, and Stefan Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *J. Autom. Reasoning*, 24(3):297–317, 2000.

[2] Ian R. Horrocks, Ullrich Hustadt, Ulrike Sattler, and Renate Schmidt. Computational modal logic. In *Handbook of Modal Logic*, pages 181–245. Elsevier, 2006.

[3] Fabio Massacci and Francesco M. Donini. Design and results of TANCS-2000 non-classical (modal) systems comparison. In *Proc. TABLEAUX 2000*, volume 1847 of *LNCS*, pages 52–56. Springer, 2000.

[4] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. KSP: A resolution-based prover for multimodal K. To appear in Proc. IJCAR 2016.

[5] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. KSP: sources and benchmarks. `http://www.cic.unb.br/~nalon/#software`.

[6] Cláudia Nalon, Ullrich Hustadt, and Clare Dixon. A modal-layered resolution calculus for K. In *Proc. TABLEAUX 2015*, volume 9323 of *LNCS*, pages 185–200. Springer, 2015.

[7] Peter F. Patel-Schneider and Roberto Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *J. Artif. Intell. Res. (JAIR)*, 18:351–389, 2003.

Modal Tableau Systems with Blocking and Congruence Closure

Renate A. Schmidt[1] Uwe Waldmann[2]

[1] School of Computer Science, The University of Manchester, UK
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract: The interest of our research are semantic tableau approaches closely related to bottom-up model generation methods. Using equality-based blocking techniques these can be used to decide logics representable in first-order logic that have the finite model property. Many common modal and description logics have these properties and can therefore be decided in this way. This paper integrates congruence closure, which is probably the most powerful and efficient way to realise reasoning with ground equations, into a modal tableau system with equality-based blocking. The system is described for an extension of modal logic **K** characterised by frames in which the accessibility relation is transitive and every world has a distinct immediate predecessor. We show the system is sound and complete, and discuss how various forms of blocking such as ancestor blocking can be realised in this setting.

Tableau systems provide a natural and powerful form of reasoning widely used for non-classical logics, especially modal, description, and hybrid logics. In this work the focus is on semantic tableau systems closely related to bottom-up model generation methods [2]. Using unrestricted blocking [12], which is an equality-based blocking technique, these can decide logics with the finite model property, representable in first-order logic [13, 14]. Many common modal and description logics have these properties and can therefore be decided using semantic tableau systems with equality-based blocking.

For many common modal and description logics there are ways to avoid the explicit use of equality in the tableau system. For more expressive logics, with nominals as in hybrid modal logics and description logics (nominals are distinguished propositional variables that hold at exactly one world), it becomes harder to avoid the explicit handling of equality (though not impossible [7]). For modal logics where the binary relations satisfy frame conditions expressible as first-order formulae with equality, explicit handling of equations is the easiest and sometimes the only known way to perform equality reasoning. Single-valuedness of a relation is an example of a frame condition expressed using equality. Another example is the following

$$\forall x \exists y \forall z \big(R(y,x) \wedge x \not\approx y \wedge \big((R(y,z) \wedge R(z,x)) \rightarrow (z \approx x \vee z \approx y) \big) \big), \tag{1}$$

where $\approx$ denotes equality. Provision for explicit equality reasoning is also necessary for tableau systems with equality-based blocking.

In semantic tableau systems explicit equality handling has been realised in a variety of ways. Using standard equality rules is conceptually easiest and most general, and is often used [4, 5, 12]. This approach leads to a combinatorial explosion of derived formulae to ensure all elements in the same equivalence class have the same information content. Many of these formulae are unneeded and fewer formulae are derived when using paramodulation-style rules, where the central idea is replacement of equals by equals [3, 5]. Ordered rewriting presents a further refinement and is significantly more efficient because equations are oriented by an ordering and then used to simplify the formulae. Ordered rewriting is used, e.g., in a semantic tableau system of [9] for the description logic $\mathcal{SHOI}$. Different equality reasoning methods have also been integrated into non-ground tableau and related approaches, e.g. [3, 5, 6].

We require efficient handling of *ground equations*. For this purpose congruence closure algorithms provide probably the most efficient algorithms [10]. The Nelson-Oppen congruence closure method has been incorporated with Smullyan-type tableau system for first-order logic by [8]. Congruence closure algorithms have also been very successfully combined with the DPLL approach and are standardly integrated in SMT-solvers as theory reasoners for the theory of equations with uninterpreted function [11].

The motivation of the present work is to combine congruence closure with semantic tableau systems for modal, description, and hybrid logics. Since it presents a general framework in which many existing congruence closure algorithms can be described (and in order to achieve more generality), we combine the *abstract congruence closure system* of [1] with our semantic tableau system. Our ultimate goal is to provide a general framework with general soundness and completeness results for developing and studying equality reasoning and blocking in semantic tableau systems. The tableau system we consider has been obtained in the tableau synthesis framework of [13], but in this framework equality is accommodated by the standard equality rules. We have shown how these can be replaced by abstract congruence closure rules.

The most closely related work is the aforementioned [8], because the flavour of the tableau systems we are concerned with is similar to that of Smullyan-type tableau systems for first-order logic. The key difference is the way in which we use the congruence closure algorithm: In [8], the congruence closure component is essentially a black box that is queried to check entailed equalities. In contrast, we use the

convergent term rewrite system produced by the abstract congruence closure algorithm also systematically to normalise the remaining tableau formulae. This means that duplication of formulae is avoided and that restrictions of the search space that depend on normalisation can be applied easily. In addition, we show that the ideas are not limited to a fixed set of the well-known tableau rules for first-order logic, but can be combined with special-purpose tableau systems of other logics having other kinds of tableau rules. Also related is [9] and the implementation of equality reasoning in METTEL-generated tableau provers [16], where ordered rewriting is used. This work does however not have the same level of generality as abstract congruence closure.

We present an abstract semantic tableau system with abstract ways of handling both blocking and equality. The focus is on showing how the abstract congruence closure system of [1] can be combined with a semantic tableau system for a modal logic. In contrast to earlier work, we use a "white box" integration, so that the abstract congruence closure is not only used to check entailed equalities, but also to normalize tableau formulae, so that logically equivalent formulae are eliminated. The particular modal tableau system was chosen to illustrate the most important ideas of integrating congruence closure so that the integration can be extended to other tableau systems for other modal, description, and hybrid logics. We believe the case study is general enough to work out how to combine congruence closure with Smullyan-type tableau rules for first-order logic, or incorporate it into bottom-up model generation and hypertableau methods. The ideas are also applicable in tableau systems obtained in the tableau synthesis framework of [13].

The paper in which this work is publieshed is [15].

References

[1] L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *J. Automat. Reason.*, 31(2):129–168, 2003.

[2] P. Baumgartner and R. A. Schmidt. Blocking and other enhancements for bottom-up model generation methods. In *Proc. IJCAR'06*, vol. 4130 of *LNAI*, pages 125–139. Springer, 2006.

[3] B. Beckert. Semantic tableaux with equality. *J. Logic Comput.*, 7(1):39–58, 1997.

[4] T. Bolander and P. Blackburn. Termination for hybrid tableaus. *J. Logic Comput.*, 17(3):517–554, 2007.

[5] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In *Handbook of Automated Reasoning*, pages 611–706. Elsevier, 2001.

[6] M. Giese. Superposition-based equality handling for analytic tableaux. *J. Automat. Reason.*, 38(1-3):127–153, 2007.

[7] M. Kaminski. *Incremental Decision Procedures for Modal Logics with Nominals and Eventualities*. PhD thesis, Universität des Saarlandes, Germany, 2012.

[8] T. Käufl and N. Zabel. Cooperation of decision procedures in a tableau-based theorem prover. *Revue d'Intelligence Artificielle*, 4(3):99–126, 1990.

[9] M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. A refined tableau calculus with controlled blocking for the description logic $\mathcal{SHOI}$. In *Proc. TABLEAUX'13*, vol. 8123 of *LNCS*, pages 188–202. Springer, 2013.

[10] R. Nieuwenhuis and A. Oliveras. Fast congruence closure and extensions. *Inform. and Computat.*, 205(4):557–580, 2007.

[11] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.

[12] R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In *Proc. ISWC'07 + ASWC'07*, vol. 4825 of *LNCS*, pages 438–451. Springer, 2007.

[13] R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Logical Methods in Comput. Sci.*, 7(2):1–32, 2011.

[14] R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *ACM Trans. Comput. Log.*, 15(1), 2014.

[15] R. A. Schmidt and U. Waldmann. Modal tableau systems with blocking and congruence closure. In *Proc. TABLEAUX 2015*, vol. 9323 of *LNCS*, pages 38–53. Springer, 2015.

[16] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeL2. In *Proc. JELIA'12*, vol. 7519 of *LNCS*, pages 492–495. Springer, 2012.

Towards Generalised Proof Search for Natural Deduction Systems for logics $I_{\langle\alpha,\beta\rangle}$

Alexander Bolotov[1] Vasily Shangin[2*]

[1] University of Westminster, Department of Computer Science A.Bolotov@wmin.ac.uk
[2] Lomonosov Moscow State University, Faculty of Philosophy, Department of Logic, shangin@philos.msu.ru

Abstract: We continue our investigation of the proof searching procedures developed for natural deduction calculus for classical and a variety of non-classical logics. In particular, we deal with natural deduction systems for propositional logics $I_{\langle\alpha,\beta\rangle}$, where $\alpha, \beta \in 0, 1, 2, 3, \ldots \omega$ such that $I_{\langle 0,0\rangle}$ is classical logic, proposed by Vladimir Popov. We aim at generalising the concept of an inference for these systems that is fundamental to proof searching technique for these logics.

1 Introduction

In [6], a logic $I_{\langle\alpha,\beta\rangle}$ is proposed as Hilbert-style calculi $HI_{\langle\alpha,\beta\rangle}$, where $\alpha, \beta \in 0, 1, 2, 3, \ldots \omega$ such that $HI_{\langle 0,0\rangle}$ is classical logic, to deal with a generalization of Glivenko theorem [5]. We present natural deduction calculi $ND_{\langle\alpha,\beta\rangle}$, for these logics. We show that A is a theorem of $HI_{\langle\alpha,\beta\rangle}$ iff A is a theorem of $ND_{\langle\alpha,\beta\rangle}$. Moreover, we present a generalised proof search technique for each natural deduction calculus in question. The propositional language L over the alphabet $p, p_1, p_2, \ldots, (,), Bool$ ($Bool = \wedge, \supset, \vee, \neg$) and a notion of a formula of language L are defined in the standard way. A formula is said to be quasi-elementary iff no logical connective of $Bool$ occurs in it [6]. Let $|A|$ abbreviate the length of A, the number of all occurrences of the logical connectives of L in A. Letters A, B, C, D, E with lower indices run over arbitrary formulae. Letters α, β with upper and lower indices run over arbitrary finite sets of formulae. Letters α, β run over $0, 1, 2, 3, \ldots \omega$.

2 Hilbert-style systems $HI_{\langle\alpha,\beta\rangle}$

In [6], V. Popov presents a Hilbert-style calculus $HI_{\langle\alpha,\beta\rangle}$ with the following axioms:

(I) $(A \supset B) \supset ((B \supset C) \supset (A \supset C))$

(II) $A \supset (A \vee B)$

(III) $B \supset (A \vee B)$

(IV) $(A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C))$

(V) $(A \wedge B) \supset A$

(VI) $(A \wedge B) \supset B$

(VII) $(C \supset A) \supset ((C \supset B) \supset (C \supset (A \wedge B)))$

(VIII) $(A \supset (B \supset C)) \supset ((A \wedge B) \supset C)$

(IX) $((A \wedge B) \supset C) \supset (A \supset (B \supset C))$

(X) $(((A \supset B) \supset A) \supset A)$

(XI) $\neg D \supset (D \supset A)$, where D is not a quasi-elementary formulae and $|D| < \alpha$

(XII $(E \supset (\neg A \supset A)) \supset E$, where E is not a quasi-elementary formulae and $|E| < \beta$.

Modus ponens is the only inference rule of the calculus. Definitions of an inference and proof in $HI_{\langle\alpha,\beta\rangle}$ are standard as well as notions of their length.

3 ND systems $ND_{\langle\alpha,\beta\rangle}$

$$\wedge\, el_1 \quad \frac{A \wedge B}{A} \qquad \wedge\, el_2 \frac{A \wedge B}{B} \qquad \wedge\, in \frac{A,\ B}{A \wedge B}$$

$$\vee\, in_1 \quad \frac{A}{A \vee B} \qquad \vee\, in_2 \frac{B}{A \vee B} \qquad \supset\, el \frac{A \supset B,\ A}{B}$$

$$\neg\, in_{1_\alpha} \quad \frac{D, \neg D}{E}$$ where D is not a quasi-elementary formula with $|D| < \alpha$.

$$\supset\, in \quad \frac{[A]B}{A \supset B} \qquad \supset_p \frac{[A \supset B]\ A}{A}$$

$$\neg\, in_{2_\beta} \frac{[E]\ \neg(B \supset B)}{\neg E}$$

Formulae in the square brackets are the last in the list of assumptions. Additionally, in $\neg\ in_{2_\beta}$, formula E is not a quasi-elementary formula with $|E| < \beta$.

An inference is said to be a non-empty finite linearly ordered sequence of formulae $C_1, C_2, \ldots, C_k$, satisfying the following conditions:

- Each C_i is either an assumption or is inferred from the previous formulae via an ND rule;
- In applying $\supset\ in$, each formula, starting from the last assumption $[A]$ up to (but not including) $A \supset B$, the result of this rule, is discarded from the inference;

*This research was supported by Russian Foundation for Humanities, (Logical-Epistemological Representation of Knowledge), project 16-03-00749,

- In applying $\supset_P$, each formula, starting from the last premise $[A \supset B]$ up to (but not including) A, the result of this rule, is discarded from the inference;
- In applying $\neg\, in_{2_\beta}$,each formula, starting from the last premise $[E]$ up to (but not including) E, the result of this rule, is discarded from the inference.

Given an inference $C_1, C_2, \ldots, C_k$ with $A_1, A_2, \ldots, A_n$ being non-discarded assumptions and with the last formula $C_k = B$, we have an inference of B from assumptions $A_1, A_2, \ldots, A_n$. If the set of formulae Γ contains $A_1, A_2, \ldots, A_n$ and there is an inference of B from $A_1, A_2, \ldots, A_n$ then we say there is an inference of B from a set of formulae Γ [1].

MAIN THEOREM

$\Gamma \vdash_{HI_{\langle\alpha,\beta\rangle}} A \iff \Gamma \vdash_{ND_{\langle\alpha,\beta\rangle}} A$, for each $\alpha, \beta \in \{0, 1, 2, 3, \ldots \omega\}$.

Proof Idea. Left to Right:

We need to show that if there exists an inference of A from Γ in $HI_{\langle\alpha,\beta\rangle}$ then there exists an inference of A from Γ in $ND_{\langle\alpha,\beta\rangle}$. For the proof we define a notion of a "height of an inference" (similar to the definitions of heights of proofs, so that the height of the inference of an axiom is 1, etc) and then we prove this direction of the theorem by mathematical induction on the height of the inference of an arbitrary formula A from Γ in $HI_{\langle\alpha,\beta\rangle}$. The base case would require to prove all axioms in the ND system. The proof for the inductive step is of course more involved and it uses the structural similarity of the modus ponens rule in the axiomatics and an ND $\supset\ el$ rule.

Right to Left:

We need to show that if there exists an inference of A from Γ in $ND_{\langle\alpha,\beta\rangle}$ then there exists an inference of A from Γ in $HI_{\langle\alpha,\beta\rangle}$. We note that for the base case the ND inferences are "trivial" and it is easy to construct corresponding axiomatic proof. For the inductive step, the proof is complex and is based on the identification of the cases of the applications of the ND rules.

4 Towards Generalised Proof Search for $ND_{\langle\alpha,\beta\rangle}$

Here we draw a route to formulating this generalised proof search for natural deduction calculi. We first note that proof search for various logics is based on the notion of algo-derivation that is served to establish inferences in an automated way (for decidable logics).

Algo-derivation in $ND_{\langle\alpha,\beta\rangle}$, abbreviated as $ND_{\langle\alpha,\beta\rangle_{\mathsf{ALG}}}$, is a pair (*list_proof*, *list_goals*) whose construction is determined by the searching procedure outlined below.

Below we give a very short insight into the searching procedures referring the reader to [4], [3], [2] for more detailed description of various searching techniques that formed "classical" propositional reasoning in natural deduction representations of linear-time temporal logic, paraconsistent logic PCont and paracomplete logic PComp.

Searching Procedures.

Procedure (1). Here we search for an applicable elimination ND-rule in order to update *list_proof*.

Procedure (2). We look at the structure of the current goal and update *list_proof* and *list_goals*, respectively, by new goals or new assumptions. If no updates are possible and the current goal is not reached we analyse compound formulae in *list_proof* in order to find sources for new goals.

Procedure (3). This checks the reachability of the current goal in the sequence *list_goals*.

Procedure (4). Procedure (4) results in finding a relevant introduction rule to be applied. As we have already noted, the specifics of our searching technique is complete determination of the application of the introduction rules. Any application of such a rule is strictly determined by the current goal in *list_goals*.

CONJECTURE

Abbreviating an algo-proof of A from Γ in $ND_{\langle\alpha,\beta\rangle}$ by $\Gamma \vdash_{ND_{\langle\alpha,\beta\rangle_{\mathsf{ALG}}}} A$, we aim to establish the following:

For for each $\alpha, \beta \in \{0, 1, 2, 3, \ldots \omega\}$, $\Gamma \vdash_{ND_{\langle\alpha,\beta\rangle}} A$ if, and only if, there exists $\Gamma \vdash_{ND_{\langle\alpha,\beta\rangle_{\mathsf{ALG}}}} A$.

References

[1] V. Bocharov and V. Markin Introduction to logic. Moscow, 2011 (in Russian).

[2] A. Bolotov and V. Shangin. Tackling Incomplete System Specifications Using Natural Deduction in the Paracomplete Setting. IEEE 38th Annual Computer Software and Applications Conference, COMPSAC 2014, Vasteras, Sweden, July 21-25, 2014, pages, 91–96.

[3] A. Bolotov and V. Shangin. Natural Deduction System in Paraconsistent Setting: proof search for PCont. Journal of Intelligent Systems, Vol. 21, N1, 2012, pp 1-24.

[4] A. Bolotov, O. Grigoriev, and V. Shangin. Automated natural deduction for propositional linear-time temporal logic. Proceedings of the Time-2007, International Symposium on Temporal Representation and Reasoning, June, 2007.

[5] V. Glivenko Sur quelques points de la logique de M. Brouwer Bull. Soc. Math. Belg. 15, 183-188.

[6] V. Popov. *On one generalization of Glivenko theorem.* Logical investigations. V. 21(1). 2015. pages. 100-121, (in Russian).