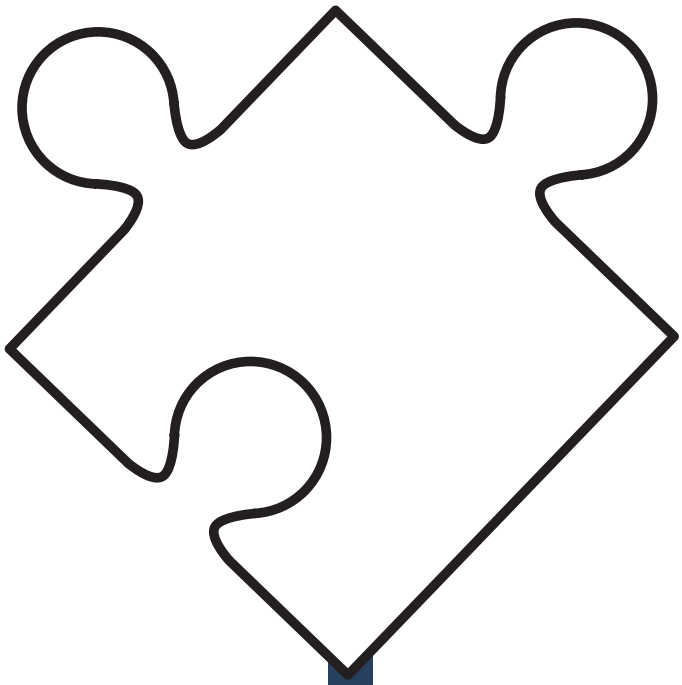
AUTOMATED REASONING WORKSHOP 2019

TECHNICAL REPORT



UNIVERSITY OF WESTMINSTER

Middlesex University London

Proceedings of the
26th Automated Reasoning Workshop 2019
Bridging the Gap between Theory and Practice
ARW 2019

2–3 September 2019
University of Middlesex
United Kingdom

Editors:
Alexander Bolotov and Florian Kammueller

UNIVERSITY OF WESTMINSTER

Middlesex University London

ARW2019 organisers:

Alexander Bolotov and Florian Kammueller

Program Commitee

Alexander Bolotov	Chair, (University of Westminster)
Jacques Fleuriot	Secretary/Treasurer (University of Edinburgh)
David Crocker	(Escher Technologies)
Louise Dennis	(University of Liverpool)
Ullrich Hustadt	(University of Liverpool)
Mateja Jamnik	(Univerity of Cambridge)
Florian Kammueller	(Middlesex University)
Ekaterina Komendantskaya	(University of Dundee)
Konstantin Korovin	(University of Manchester)
Alice Miller	(University of Glasgow)
Oliver Ray	(University of Bristol)
Giles Reger	(University of Manchester)

Workshop Website

`https://www.arw2019.org/`

Preface

This volume contains the proceedings of ARW 2019, the twenty sixths Workshop on Automated Reasoning (2nd–3d September 2019) hosted by the Department of Computer Science, Middlesex University, England (UK). Traditionally, this annual workshop which brings together, for a two-day intensive programme, researchers from different areas of automated reasoning, covers both traditional and emerging topics, disseminates achieved results or work in progress. During informal discussions at workshop sessions, the attendees, whether they are established in the Automated Reasoning community or are only at their early stages of their research career, gain invaluable feedback from colleagues. ARW always looks at the ways of strengthening links between academia, industry and government; between theoretical and practical advances. The 26th ARW is affiliated with TABLEAUX 2019 conference.

These proceedings contain forteen extended abstracts contributed by the participants of the workshop and assembled in order of their presentations at the workshop. The abstracts cover a wide range of topics including the development of reasoning techniques for Agents, Model-Checking, Proof Search for classical and non-classical logics, Description Logics, development of Intelligent Prediction Models, application of Machine Learning to theorem proving, applications of AR in Cloud Computing and Networking.

I would like to thank the members of the ARW Organising Committee for their advice and assistance. I would also like to thank the organisers of TABLEAUX/FroCoS 2019, and Andrei Popescu, the TABLEAUX Conference Chair, in particular, for the enormous work related to the organisation of this affiliation. I would also like to thank Natalia Yerashenia for helping in preparing these proceedings.

London
September 2019

Alexander Bolotov

Contributed Papers

Knowledge Sharing among Agents with Ontological Reasoning

David Toluhi Renate Schmidt

University of Manchester, UK
{david.toluhi, renate.schmidt}@manchester.ac.uk

Abstract: In Multi-Agent Systems a simplifying assumption is that agents represent the world using the same vocabulary. We relax this assumption and focus on scenarios where knowledge is distributed among agents with some overlap and investigate ways in which agents can restrict and adapt their knowledge with respect to their overlap in vocabulary to ensure that knowledge being shared is understood by their communication partners. Specifically, we focus on belief-desire-intention agents that make use of description logics to represent their knowledge of the world. This paper gives a brief overview of preliminary research and theory development: especially the approximation of concepts across subset vocabularies.

1 Introduction

The importance of knowledge approximation is highlighted in our day to day activities. Humans frequently approximate knowledge. For example, when communicating with toddlers or younger audiences, an adult restricts the used vocabulary to a level that can be understood. Other examples include experts communicating with people outside their domain of expertise: a medical doctor approximates knowledge when explaining a diagnosis to a patient or an IT expert explaining a system-diagnosis to a client.

Our work is motivated by recent advances in Description Logics (DLs): the development of forgetting tools for DLs, specifically those in [10] and [5]. Forgetting can be used to compute theory approximations [6] [1], part of our plan is to realize and demonstrate this in DLs. This approach is appealing because there is also an interest in the agent community in using ontologies/ontological-reasoning to model agents such as the efforts made in [3] and [8].

Ideas and realizations of the usefulness of knowledge approximation, particularly in the domain of *agent communication* can be found in [1] and [6].

Multi-Agent System (MAS) communication is mainly assertional[1] and is implemented using *speech acts* [2] which has been formalized and integrated into agent platforms such as that in [7]. Our aims are summarized as follows:

1. Study existing agent communication protocols.
2. Develop new generalized agent communication protocols and standards suitable for agents with different vocabularies.
3. Develop theory approximation techniques based on forgetting and definition generation.
4. Implement our ideas using LETHE/FAME as an extension to an existing MAS such as JASON [4].

We believe that definitions are mechanisms essential to tasks involving translation and approximation and also allow for two different agents to build and expand their common vocabulary. Agents may encounter unfamiliar symbols when sharing knowledge such as plans, descriptions or assertions. We look into building a *define* speech-act in order to solve this problem.

Our approach will attempt to develop ways of using the forgetting tools in [10] and [5] to automatically generate definitions. Our work draws on related work on definition generation in second-order logic presented in [9], and propositional logic briefly mentioned in [6]; we aim to implement tools to provide this functionality in DL. Also related is *terminological negotiation* [8] which uses an interpretation approach to compute translations and mappings between concepts in different agent's ontologies.

The necessity for sharing knowledge across disparate (but intersecting) vocabularies is highlighted in [8] and [1]. In [8] one of the listed motivations is the current inability to model agents in a heterogenous fashion which restricts the flexibility and application domain of the agents. Doherty et al in [1] highlight the requirements in large scale applications for agents to use different vocabularies.

2 Theory Approximation using Weakest Sufficient and Strongest Necessary Conditions

Our work focuses on disparate but intersecting sets of vocabularies. In order to communicate in such a setting, some sort of approximation is needed. In logic, *necessary conditions* and *sufficient conditions* [6] are two key established notions on the subject of approximating theories. More specifically, a theory α is a *strongest necessary* condition of a formula X over a signature Σ with respect to a background theory O_B denoted $SNC(X; O_B; \Sigma)$ iff for any theory α': $O_B,\ \alpha \models \alpha'$ where α' any other necessary condition [1]. A theory β is a *weakest sufficient* condition of a formula X over a signature Σ with respect to a background theory O_B denoted $WSC(X; O_B; \Sigma)$ iff for any theory β': $O_B,\ \beta' \models \beta$ where β' is any other sufficient condition [1]. Intuitively, one can think of the *strongest necessary* condition as the closest [2] upper bound of term and the *weakest sufficient* condition with respect to a given vocabulary that is a subset of the vocabulary of the background theory.

[1] composed of ground statements

[2] with respect to logical entailment ordering

It is suggested in [1] and [6] that *forgetting* plays a crucial role in computing both conditions. Our early attempts to implement these ideas suggest that this is not quite the case: we observe that forgetting can be used to compute theories/ontologies that entail the strongest-necessary or weakest-sufficient conditions. At the moment, it seems that more a appropriate method would be to use a modified version of forgetting modulo a background theory/ontology.

3 Definition generation

Following [1, 6, 9], we refined our goal for definition generation as follows:

Given: is an ontology O_B (background ontology/theory), a definiendum[3] X which is atomic, and a set of symbols Σ, where $\Sigma,\ X \subset signature(O_B)$.

The **goal** is to: find a definiens [4] ϕ for X such that:

1. $signature(\phi) \subseteq \Sigma$,
2. $O_B \models \phi \equiv X$ or
   $O_B \models \phi \sqsubseteq X$ or
   $O_B \models \phi \sqsupseteq X$, and
3. ϕ is a complex expression or logical axiom.

The second requirement uses one of three copulae: $\equiv$, $\sqsubseteq$ or $\sqsupseteq$. Assume we use the $\equiv$ copula.

Definability Following [6, 9], a theory O_B defines a symbol X w.r.t Σ where $\Sigma \subseteq signature(O_B)$ iff:

$$O_B,\ SNC(X; O_B; \Sigma) \models WSC(X; O_B; \Sigma)$$

Following [9] definitions are characterised as follows:
ϕ is a definiens of X in terms of Σ within O_B iff$_{def}$

$$O_B, SNC(X; O_B; \Sigma) \models \phi \text{ and}$$

$$O_B, \phi \models WSC(X; O_B; \Sigma)$$

As [9] points out, this captures the equivalence copula: $O_B \models \phi \equiv X$. Our proposed procedure is as follows:

Step 1: Compute the *strongest necessary* condition: $SNC(X; O_B; \Sigma)$ using forgetting modulo a background theory.

Step 2: Compute the *weakest sufficient* condition: $WSC(X; O_B; \Sigma)$ using forgetting modulo a background theory.

Step 3 (Definability Check): Confirm that

$$O_B,\ SNC(X; O_B; \Sigma) \models WSC(X; O_B; \Sigma)$$

Step 4 (Definiens): Compute a definiens ϕ such that :

$$O_B,\ SNC(X; O_B; \Sigma) \models \phi \text{ and}$$

$$O_B,\ \phi \models WSC(X; O_B; \Sigma)$$

Step 5: If $signature(\phi) \neq \{\bot, \top, X\}$ return ϕ else return *Failed*.

[3] Latin for *subject of definition*
[4] Latin for *definition*

4 Conclusion

We are currently exploring the procedure and researching methods to computing Craig-interpolants in DLs in order to compute a definition ϕ that satisfies the requirements of Step 4 above.

References

[1] Patrick Doherty, Witold Lukaszewicz, and Andrzej Szalas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In *Proc. IJCAI*, pages 145–154, 2001.

[2] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an agent communication language. In *Proc. CIKM*, pages 456–463. ACM, 1994.

[3] Artur Freitas, Alison R Panisson, Lucas Hilgert, Felipe Meneguzzi, Renata Vieira, and Rafael H Bordini. Integrating ontologies with multi-agent systems through cartago artifacts. In *Proc.IEEE/WIC/ACM*, volume 2, pages 143–150. IEEE, 2015.

[4] Thomas Klapiscak and Rafael H Bordini. Jasdl: A practical programming approach combining agent and semantic web technologies. In *International Workshop on Declarative Agent Languages and Technologies*, pages 91–110. Springer, 2008.

[5] P. Koopmann and R. A. Schmidt. LETHE: A saturation-based tool for non-classical reasoning. In M. Dumontier, B. Glimm, R. Goncalves, M. Horridge, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, G. Stamou, and G. Stoilos, editors, *Proc. ORE-2015*, volume 1387 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[6] Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128(1-2):143–159, 2001.

[7] Anand S. Rao. Agentspeak (l): BDI agents speak out in a logical computable language. In *Proc. MAAMAW*, pages 42–55. Springer, 1996.

[8] Marlo Souza, Alvaro Moreira, Renata Vieira, and John-Jules Ch. Meyer. Integrating ontology negotiation and agent communication. In *Proc. International Experiences and Directions Workshop on OWL*, pages 56–68. Springer, 2015.

[9] Christoph Wernhard. Second-order characterizations of definientia in formula classes. In *Proc. IJCAR Workshop*. IJCAR, 2014.

[10] Y. Zhao and R. A. Schmidt. FAME: An automated tool for semantic forgetting in expressive description logics. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, *Proc. IJCAR 2018*, volume 10900 of *Lecture Notes in Artificial Intelligence*, pages 19–27. Springer, 2018.

Towards an Under-Approximation Abstraction-Refinement for Reasoning with Large Theories

Julio Cesar Lopez Hernandez Konstantin Korovin

The University of Manchester, School of Computer Science
{lopezhej,korovin}@cs.man.ac.uk

Abstract: We present the main ideas of the under-approximation process, which is part of an abstraction-refinement framework for reasoning with large theories. The framework has the purpose of interleaving the axioms selection and reasoning phase, and it consists of two approximations: the over- and under-approximation, which can be combined to converge to a solution more quickly.

1 Introduction

Efficient reasoning with large theories is one of the main challenges in automated theorem proving. The main problem arises because of the enormous number of superfluous premises in the theories, and usually, a few of them are needed to prove a conjecture. Therefore it is desirable to select the most relevant axioms when proving a conjecture.

Current methods for axiom selection are based on the syntactic or semantic structure of the axioms, and conjecture formulas [2]. Other methods for axiom selection use machine learning to take advantage of previous knowledge about proving conjectures [6]. In [5], we present a framework based on abstraction-refinement for reasoning with large theories, which could use different methods to select suitable premises. In this abstract, we present the under-approximation, which is part of the abstraction-refinement framework.

2 Preliminaries

Let us consider a set of formulas $\mathcal{F}$ which we call a *concrete domain* and a set of formulas $\hat{\mathcal{F}}$ which we will call an *abstract domain*. For example, $\mathcal{F}$ can be the set of all first-order formulas and $\hat{\mathcal{F}}$ can be a fragment of first-order logic. Concrete and abstract domains can coincide.

An *abstraction function* is a mapping $\alpha : \mathcal{F} \mapsto \hat{\mathcal{F}}$. When there is no ambiguity we will call an abstraction function just an abstraction of $\mathcal{F}$. The identity function is an abstraction which will be called the *identity abstraction* α_{id}.

An abstraction α is called *under-approximating abstraction* (wrt. refutation) if for every $F \in \mathcal{F}$, $\alpha(F) \models \bot$ implies $F \models \bot$. An abstraction α is called *over-approximating abstraction* (wrt. refutation) if for every $F \in \mathcal{F}$, $F \models \bot$ implies $\alpha(F) \models \bot$.

We define an ordering on abstractions $\sqsubseteq$ called *abstraction refinement ordering* as follows: $\alpha \sqsubseteq \alpha'$ if for all $F \in \mathcal{F}$, $\alpha(F) \models \bot$ implies $\alpha'(F) \models \bot$. Two abstractions are *equivalent*, denoted by $\alpha \equiv \alpha'$ if $\alpha \sqsubseteq \alpha'$ and $\alpha' \sqsubseteq \alpha$. The strict part $\sqsubset$ of $\sqsubseteq$ is defined as $\alpha \sqsubset \alpha'$ if $\alpha \sqsubseteq \alpha'$ and $\alpha \not\equiv \alpha'$. An abstraction is *precise* if it is equivalent to the identity abstraction. An example of a non-trivial precise abstraction can be obtained by renaming function and predicate symbols. We have that every over-approximating abstraction α_s is above and every under-approximation abstraction α_w is below the identity abstraction wrt. the abstraction refinement ordering, i.e., $\alpha_w \sqsubseteq \alpha_{id} \sqsubseteq \alpha_s$.

Strengthening abstraction refinement of an under-approximating abstraction α is an abstraction α' which is above α and below the identity abstraction in the abstraction refinement ordering, i.e., $\alpha \sqsubseteq \alpha' \sqsubseteq \alpha_{id}$.

An *under-approximation abstraction-refinement process* is a possibly infinite sequence of strengthening abstraction refinements $\alpha_0, \ldots, \alpha_n, \ldots$ such that $\alpha_0 \sqsubseteq \ldots \sqsubseteq \alpha_n \sqsubseteq \ldots \sqsubseteq \alpha_{id}$.

3 Under-Approximation Abstraction-Refinement Process

We use ATP_S to denote an automated theorem prover, which is sound but possibly incomplete (wrt. refutation) [1]. The process starts by applying the under-approximating abstraction function to the set of concrete axioms A, $\hat{A}^w = \alpha_w(A)$. This set $\hat{A}^w$ of weaker axioms is used to prove the conjecture, using an ATP_S. If the conjecture is proved, the procedure stops and provides the proof. Otherwise, a model I of $\hat{A}^w$ and the negated conjecture is obtained. This model is used to refine the set of weaker axioms $\hat{A}^w$. During this refinement (strengthening abstraction refinement), the procedure tries to find a set of axioms $\breve{A}$ that turns the model into a countermodel but are still implied by A, i.e., $I \not\models \breve{A}$ and $A \models \breve{A}$. If the set of axioms $\breve{A}$ is empty, $\breve{A} = \emptyset$, the procedure stops and disproves the conjecture. Otherwise, the obtained set of axioms is added to the set of weaker axioms, $\hat{A}^w := \hat{A}^w \cup \breve{A}$. Using this new set of abstract axioms $\hat{A}^w$, another round for proving the conjecture starts. The process finishes when the conjecture is proved or disproved or the time limit for the quest of proof is reached.

3.1 Under-Approximating Abstraction Functions

In the case of under-approximation, we propose two under-approximating abstractions: *instantiation abstraction* and *deletion abstraction*. For the case of instantiation abstraction, abstraction function generates ground instances of the

concrete axioms as it is done in the Inst-Gen framework [3]. In the case of deletion abstraction, we delete certain concrete axioms from the theory. This particular abstraction can be used to incorporate other axioms selection methods (based on removing irrelevant axioms) into the under-approximation process. SInE algorithm is one example of this kind of methods. Also, approaches based on machine learning can be incorporated into the under-approximation process [6]. In practice, different abstractions can be recombined.

3.2 Strengthening Abstraction Refinement

In the case of deletion abstraction, refinement can be done by adding concrete axioms $\breve{A}$ that turn the model I, which is obtained form ATP_S, into a countermodel, $\breve{A} \subseteq \{\breve{a} \mid \breve{a} \in A, I \not\models \breve{a}\}$. In the case of instantiation abstraction, refinement can be done by generating a set of ground instances of axioms $A\sigma$ such that $I \not\models A\sigma$, $\breve{A} := A\sigma$.

4 Early Results

Currently, we are experimenting with a deletion abstraction based on SInE. The core idea of SInE is to use the generality of symbols. This generality is defined as the number of axioms where each of the symbols appears , i.e., symbols that appears in more axioms are more common (general). This generality establishes a relationship among formulas by linking the less common symbol in a formula (called trigger) with other formulas where the trigger appears. We consider different criteria for symbol to be a trigger based on its relationship to the conjecture. The premise selection is made by recursively following the links mentioned above, starting from the conjecture symbols.

In the implementation, the refinement of the deletion abstraction is performed by increasing the selection parameters: depth and tolerance. The first one controls the reachability depth of the selected axioms from the conjecture using links, and the second one the symbol generality to treat symbols with closer values of generality as equals.

In table 1 and 2, we show some comparative results of using: over/under-approximation and its combination [5] integrated with iProver. We used the set of problems from CASC-26 LTB category (1500 problems) for our experiments and turned off most of the iProver's preprocessing flags. Also, we use different values for iProver's option `--schedule`. In both tables, the column "SInE clausifier" indicates if a previous deletion of axioms was performed during the clausification using the given parameters to Vampire clausifier [4].

Table 1: **under**: under-approx abstraction based on SInE. Over-approx. strategy: `[arg_filter;sig]` restricted to split/Skolem symbols with refine until SAT (**over**)

Strategy	SInE clausifier	schedule	solutions
over	-sd 1 -st 1	default	1004
under		none	1016
under-over		none	996
under-over		abstr_ref	916
under-over	-sd 2 -st 4	abstr_ref	1008

Table 2: **under**: under-approx abstraction based on SInE. Over-approximating strategy: `[arg_filter]` restricted to split/Skolem symbols with refine until SAT (**over**)

Strategy	SInE clausifier	schedule	solutions
over	-sd 1 -st 1	default	986
under		none	1015
under-over		none	1019
under-over		abstr_ref	1000
under-over	-sd 2 -st 4	abstr_ref	1033

5 Conclusions

From the results showed in tables 1 and 2, we can see that using iProver with only under-approximation performs better than just using over-approximation. Overall the best result 2 was obtained when we combine over argument filtering approximation with SInE under-approximation with paramteres depth 2 and tolerance 4. It is not the case for results in 1 where the combination of approximations have a lower performance than using under- and over-approximation by separate. We need to experiment with more combinations of under and over-approximating strategies and investigate how the combination of over and under approximations could increase the performance.

References

[1] Kryštof Hoder, Giles Reger, Martin Suda, and Andrei Voronkov. Selecting the Selection. In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning*, volume 9706, pages 313–329, Cham, 2016. Springer International Publishing.

[2] Kryštof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6803 LNAI:299–314, 2011.

[3] Konstantin Korovin. Inst-Gen – A Modular Approach to Instantiation-Based Automated Reasoning. *Programming Logics: Essays in Memory of Harald Ganzinger*, 7797:239–270, 2013.

[4] Laura Kovács and Andrei Voronkov. First-order theorem proving and VAMPIRE. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8044 LNCS:1–35, 2013.

[5] Julio Cesar Lopez Hernandez and Konstantin Korovin. An Abstraction-Refinement Framework for Reasoning with Large Theories. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 663–679. Springer, 2018.

[6] Josef Urban. MaLARea: A metasystem for automated reasoning in large theories. *CEUR Workshop Proceedings*, 257:45–58, 2007.

Forgetting Relative to A Background Theory for The Description Logic $\mathcal{ALC}$

Mostafa Sakr Renate A. Schmidt

School of Computer Science, University of Manchester, Oxford Rd, Manchester M13 9PL

Abstract: Forgetting is the process of finding a restricted view of an ontology by removing a subset of the concept and role symbols that are present in the ontology while preserving all the entailments of the original ontology that can be represented by the remaining symbols. The contribution of our research is to develop a theory for forgetting relative to a background theory. The proposal is to consider another ontology (a.k.a. a background theory) as another input to the forgetting task. Here the aim is not to apply forgetting on the background theory, but rather to take support from it when forgetting symbols from the input ontology. An important application is abduction where there are two ontologies, observations and background theory. The aim of abduction is to generate hypothesis that if added to the background theory, would explain the observations. We define the problem forgetting relative to background theory, give an illustration to its application in abduction, and give an overview of related work and other related research.

1 Introduction

Knowledge is often represented as a set of axioms expressed in terms of concept and role symbols (also called signature) using logic $\mathcal{L}$. A set of axioms describing a particular domain is called an ontology.

Forgetting has been an active field of study recently. It is a non-standard reasoning task that takes as input an ontology and a subset of the concept and roles symbols that are present in the ontology (a.k.a. forgetting signature). The output of forgetting is an ontology that is subject to two conditions. The first condition is that the forgetting signature is not present in output ontology. The second is that any entailment of the original ontology that can be represented by the remaining signature is preserved in the output ontology. [7, 6, 4].

This problem has been mostly studied on single ontologies [5, 9] which is inadequate to real world applications in which knowledge tends to be represented by several interconnected ontologies. For example in Fig. 1, the SNOMED ontology, a very large and widely used medical ontology, is organized in several interconnected substructures. Each substructure can be considered as a separate ontology. The organism ontology for instance, is connected with other substructures, collectively called background theory. These connections are entailments established by the common vocabulary of the organism ontology and the background theory. For instance, concepts related joints may be defined in the organism ontology while procedures related to them are defined in the procedure ontology. When forgetting a set of symbols $\mathcal{F}$ from the organism ontology, the entailments over the remaining symbols together with the signature of the background theory should be preserved. For example, assume the following axioms in the organism and the procedure ontologies respectively: $Knee \sqsubseteq Joint$, $joint \sqsubseteq \exists applicableOperation.JointReplacement$. Forgetting the symbol $Joint$ from the organism ontology should preserve the entailment that $JointReplacement$ is an operation applicable to $Knee$ even though $JointReplacement$ is defined in the procedure ontology. This allows any reasoning application to execute on the forgetting solution together with the background theory. Additionally, since the background theory is available to the reasoning application, entailments that follow from the background theory only are irrelevant or redundant because they can be inferred when needed by the reasoning application. Therefore, such entailments should be avoided by the forgetting method.

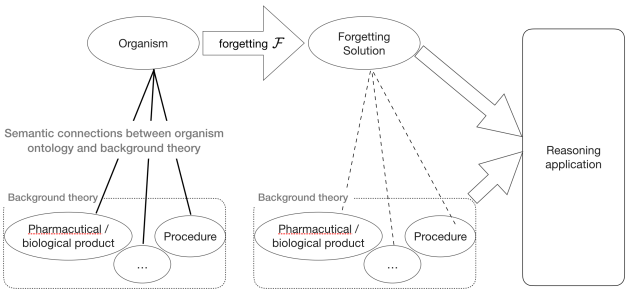


Figure 1: The SNOMED ontology is organized in interconnected ontologies. In order to use the forgetting solution of any ontology, for instance organism, the entailments (dotted lines) over the signature of the forgetting solution and the signature of the background theory must be preserved.

2 Contribution

Our research is concerned with developing the theory of a new forgetting method that takes into consideration a background theory. We also aim at producing a practical implementation that realizes this method.

The introduction of a background theory introduces conceptual and theoretical connections to the problem of *satisfiability modulo theories (smt)* [1]. smt, is the problem concerned with the satisfiability of a formula over one or more theories. In both applications, the interpretation of some

symbols that are present in the input formula/ontology is constrained by the background theory. The theoretical relations are yet to be investigated.

Another related topic is the problem of query reformulation with background theory[2, 8]. This problem is central for the database community where a database maintains two ontologies. The first represents a logical schema which is known to end user, and the second represents the physical schema which describes how the data is physically stored. The relation between both is described in a third ontology. Given a query, a database system reformulates it into another optimized query that uses the vocabulary of the physical schema. Similar to smt, the theoretical connections to this research are still not clear and are yet to be investigated.

3 Forgetting Relative to Background Theory

The introduction of a background theory requires a modification to the definition of forgetting. In this paper we consider the setting when the background theory is using the same description logic as the main ontology.

We define forgetting relative to background theory as follows: Let $\mathcal{O}_b$ and $\mathcal{O}$ be a background theory and an ontology written in a description logic $\mathcal{L}$, also let $\mathcal{F}$ be the forgetting signature, and $\Sigma = sig(\mathcal{O}_b \cup \mathcal{O}) \backslash \mathcal{F}$. We define $\mathcal{V}$ as the *forgetting of $\mathcal{O}$ relative to $\mathcal{O}_b$* iff:

1. $sig(\mathcal{V}) \subseteq \Sigma$, and
2. For every axiom α with $sig(\alpha) \subseteq \Sigma, \mathcal{O}_b, \mathcal{V} \models \alpha$ iff $\mathcal{O}_b, \mathcal{O} \models \alpha$
3. For every axiom β with $sig(\beta) \subseteq \Sigma$, $\mathcal{V} \models \beta$ only if $\mathcal{O}_b \not\models \beta$

4 Related Work

Forgetting relative to background theory was used to formalize the problem of finding the strongest necessary conditions (snc) and the weakest sufficient conditions (wsc) [6]. The problem can be formulated as: Let $\Sigma = sig(\mathcal{O}_b \cup \mathcal{O}) \backslash \mathcal{F}$ and let $\mathcal{V}$ be any formula where $sig(\mathcal{V}) \subseteq \Sigma$. Then $\mathcal{V}$ is snc/wsc iff:

$$snc : \mathcal{O}_b \models \mathcal{O} \rightarrow \mathcal{V} \quad \text{and,} \quad \mathcal{O}_b \models \mathcal{V} \rightarrow \mathcal{V}'$$
$$\text{for any necessary condition } \mathcal{V}',\ sig(\mathcal{V}') \subseteq \Sigma \qquad (1)$$
$$wsc : \mathcal{O}_b \models \mathcal{V} \rightarrow \mathcal{O} \quad \text{and,} \quad \mathcal{O}_b \models \mathcal{V}' \rightarrow \mathcal{V}$$
$$\text{for any sufficient condition } \mathcal{V}',\ sig(\mathcal{V}') \subseteq \Sigma \qquad (2)$$

where $\mathcal{O}_b, \mathcal{O}, \mathcal{F}$ are as in section 3. This problem was studied as a second order quantifier elimination problem [4]. The approach to compute snc/wsc is based on the following formulation:

$$SNC(\mathcal{O}, \mathcal{O}_b, \Sigma) \equiv \exists \mathcal{F}(\mathcal{O}_b \wedge \mathcal{O}) \qquad (3)$$
$$WSC(\mathcal{O}, \mathcal{O}_b, \Sigma) \equiv \neg \exists \mathcal{F}(\mathcal{O}_b \wedge \neg \mathcal{O}) \qquad (4)$$

This directly corresponds to forgetting since eliminating the existential quantifier in (3) and (4) means generating equivalent formula in which the quantified symbols $\mathcal{F}$ are not present. Equations 3 and 4 may generate solutions that follow only from $\mathcal{O}_b$. Notice however that if we removed these solutions from $\mathcal{V}$, equations 1 and 2 remain satisfied. In other words, these solutions are redundant w.r.t equations 1 and 2 and are not snc/wsc on $\mathcal{O}$ w.r.t. $\mathcal{O}_b$.

This observation became more visible in [3] where an ABox abduction method based on equation 4 was proposed. Here $\mathcal{O}$ is a disjunction of ABox observations and $\mathcal{O}_b$ is a background theory. The aim is to generate a hypothesis $\mathcal{H}$ that is when added to the background theory would explain the observation. The application of equation 4 generates many redundant formulas that follow only from $\mathcal{O}_b$. On one hand these formulas are not abductive solutions and a filtering post-process was required to exclude them[3]. On the other hand, these formulas are expensive to compute and should be avoided. Combining the background theory with the negated observations is forced by the fact that existing forgetting methods accept only one ontology. Our contribution solves this problem by generalizing the forgetting method to take the background theory separately. Condition 3 in our definition of forgetting (see section 3) then guarantees that these redundant solutions are avoided.

References

[1] C. Barrett and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Model Checking*, pages 305–343. Springer International Publishing, Cham, 2018.

[2] M Benedikt, E. V. Kostylev, F. Mogavero, and E. Tsamoura. Reformulating queries: Theory and practice. In *Proc. IJCAI'26*, pages 837–843, 2017.

[3] W. Del-Pinto and R. A. Schmidt. ABox Abduction via Forgetting in $\mathcal{ALC}$. *Proc. AAAI'33*, 2019.

[4] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In *Proc. IJCAI*, pages 145–151. Morgan Kaufmann Publishers Inc., 2001.

[5] P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *FroCoS 2013*, volume 8152 of *LNAI*, pages 87–102. Springer, 2013.

[6] F. Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128(1):143 – 159, 2001.

[7] C. Lutz and F. Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. IJCAI'22*, 2011.

[8] L. Popa. *Object/relational query optimization with chase and backchase*. PhD thesis, University of Pennsylvania, 2001.

[9] Y. Zhao and R. A. Schmidt. Fame: An automated tool for semantic forgetting in expressive description logics. In *Proc. IJCAR 2018*, volume 10900 of *LNAI*, pages 19–27. Springer, 2018.

Querying Clique Guarded Existential Rules

Sen Zheng Renate A. Schmidt

University of Manchester, Manchester, UK

{sen.zheng,renate.schmidt}@manchester.ac.uk

Abstract: We investigate the Boolean conjunctive query answering problem, where from a given a Boolean conjunctive query q, a database $\mathcal{D}$ and a theory Σ, the aim is to check whether $\mathcal{D} \cup \Sigma \models q$. We show that Boolean conjunctive query answering can be decided using ordered resolution with dynamic selection when Σ is expressed as clique guarded existential rules. Clique guarded existential rules subsume rules commonly used in ontology-based query answering systems such as guarded existential rules.

1 Introduction

Query answering against decidable fragments is one of the fundamental problems behind ontology-based data access systems. This work is concerned with checking whether $\mathcal{D} \cup \Sigma \models q$, where $\mathcal{D}$ is a set of ground facts, Σ is a set of clique guarded existential rules and q is a Boolean conjunctive query. The class of clique guarded existential rules is a Horn fragment of the clique guarded fragment [5]. For the latter, the satisfiability is known as 2EXPTIME-complete [5]. This means the class of clique guarded existential rules is decidable. As far as we know, as yet there is no complexity result for (querying) clique guarded existential rules and there is no approach to decide the problem of query answering for clique guarded existential rules.

Our querying approach is based on ordered resolution with selection and can be embedded into the framework of [1]. Hence, the soundness and refutational completeness result from [1] applies immediately. We use the top variable technique to avoid term depth increase in the resolvents. This technique is inspired by the partial hyper-resolution technique from [3, 4], where it is used to decide the loosely guarded fragment [6]. We adapt this approach so that the top variable technique can be used for answering queries. The idea of the top variable technique is that we only apply resolution when the positive premises contains the potentially deepest terms.

2 Preliminaries

An *existential rule* R is a first order formula of the form $\forall X \forall Y \phi(X, Y) \rightarrow \exists Z \psi(X, Z)$ where $\phi(X, Y)$ and $\psi(X, Z)$ are conjunctions of atoms, and called the *body* and the *head* of R, respectively. X, Y and Z are variable sets. An existential rule R is *clique guarded* if each pair of free variables of the head co-occurs in at least one atom of the body. The definition of *clique guarded existential rules* (CGERs) is a strict extension of the definition of guarded existential rules [2] since free variables of the head do not require to occur in a single atom of the body. An example of a CGER is $\forall xyzv_1v_2v_3(A_1(x, y, v_1) \wedge A_2(x, z, v_2) \wedge A_3(y, z, v_3) \rightarrow \exists wB(x, y, z, w))$, which is not a guarded existential rule. The class of CGERs can be seen as a Horn fragment of the clique guarded fragment [5].

A *Boolean conjunctive query* (BCQ) q is a first-order formula of the form $q = \exists X \varphi(X)$ where φ is a conjunction of atoms containing only variables and constants. The rule set Σ denotes a set of clique guarded existential rules and the database $\mathcal{D}$ denotes a set of ground atoms. We answer BCQ satisfiability of $\mathcal{D} \cup \Sigma \models q$ by answering $\mathcal{D} \cup \Sigma \cup \neg q \models \bot$.

A term is *flat* if it is a variable or a ground term. A term is *simple* if it is a variable, a constant or a compound term $f(u_1, ..., u_n)$ where $n > 0$, such that $u_1, ..., u_n$ are variables or ground terms. A *flat* (*simple*) literal is a literal so that every term in it is flat (simple). A *flat* (*simple*) clause is a clause so that every literal in it is flat (simple). Assume a clause $\phi \rightarrow \psi$ where ϕ is a conjunction of flat atoms. *Chained variables* in ϕ are variables that occur in multiple atoms of ϕ. A compound term t is *weakly covering* if for every non-ground, compound subterm s of t, it is the case that $var(s) = var(t)$. A literal L is weakly covering if each argument of L is either a ground term, a variable, or a weakly covering term t, such that $var(t) = var(L)$. A clause C is weakly covering if each term t in C is either a ground term, a variable, or a weakly covering term such that $var(t) = var(C)$.

3 Decision Procedures

Now we describe the steps of our approach to deciding query answering for clique guarded existential rules.

Clausal Transformation. We use *CGER-Trans* to denote the clausal transformation for CGERs. There are three major steps to transform a clique guarded existential rule R into a set of clauses:

1. Rewrite implications by using negations and disjunctions, and transform R into negation normal form, obtaining the formula R_{nnf}.
2. Transform R_{nnf} into prenex normal form and apply outer Skolemisation: if $\forall X$ is the subsequence of all universal quantifiers of the ψ-prefix of subformula $\exists y \psi$ of ψ, then $\psi[y/f(X)]$ is the outer Skolemisation of $\exists y \psi$, obtaining the formula R_{sko}.
3. Drop all universal quantifiers and transform R_{sko} into its conjunctive normal form, denoted as a set of clauses, obtaining the Horn cliques guarded clauses.

By *Query-Trans* we denote the clausal transformation for Boolean conjunctive queries. One can obtain a *query clause* by simply negating a Boolean conjunctive query.

Clausal Normal Forms. A clause C is a *Horn clique guarded clause* if a condensed form of Horn clause C satisfies these conditions:

1. C is simple and weakly covering.
2. There is a set of negative flat literals $\mathcal{L}$ containing chained variables in C.
   (a) If there is no more than two chained variables, all variables in C occur in one literal of $\mathcal{L}$.
   (b) If there is more than two chained variables, all chained variables co-occur in at least one literal of $\mathcal{L}$. These literals are called *guards*. All variables in non-guard literals of C are chained variables in guards.

The definition of Horn clique guarded clause is a proper superset of the definition of the clique guarded existential rules because function symbols are allowed in negative literals. A ground fact is a Horn clique guarded clause and a *query clause* is a negative flat clause.

Resolution Refinement. We use an admissible ordering and a selection function making use of the top variable technique to restrict the application of resolution. Let *Query-Refine* denote the refinement using: A lexicographic path ordering $\succ_{lpo}$ based on a precedence $f > a > p$ for f denoting function symbols, a denoting constants and p denoting predicate symbols, and a selection function such that the following conditions all hold:

1. If a clause contains negative non-ground compound literals, then at least one of these literals is selected.
2. If there is no negative non-ground compound literal, but there are positive non-ground compound literals, then the maximality principle with respect to $\succ_{lpo}$ is applied to determine the eligible literals.
3. If a clause contains no non-ground compound literals, select all the negative literals containing top variables.

Condition 3 indicates that the selected literals are not set in advance. Instead, in each inference, these selected literals are determined before resolution by checking whether they contain top variables. Hence, we call this kind of selection the dynamic selection.

4 Termination

We use *Query-Res* to denote the calculus consisting of: the condensation rule, tautology elimination, ordered factoring and ordered resolution with selection defined by *Query-Refine*.

Claim 1 *In an application of* Query-Res*, the resolvents of a set of Horn clique guarded clauses and query clauses are query clauses.*

Claim 2 *In an application of* Query-Res*, the resolvents of a set of Horn clique guarded clauses are Horn clique guarded clauses.*

Claim 1 and Claim 2 show that *Query-Res* can guarantee that given a set of Horn clique guarded clauses and query clauses, all derived clauses are either Horn clique guarded clauses or query clauses.

Claim 3 *In an application of* Query-Res*, given a finite set of fixed-length Horn clique guarded clauses and fixed-length query clauses, each derived clause have a fixed length.*

Claim 3 shows that using *Query-Res*, a derived clause cannot be arbitrarily long. Claim 1, 2 and 3 show that:

Claim 4 Query-Res *decides query clauses and Horn clique guarded clauses.*

The following claim shows the main result of this work:

Claim 5 *The combination of the clausal transformations (*CGER-Trans *and* Query-Trans*) and resolution procedures* Query-Res *decide the Boolean conjunctive query answering problem for the clique guarded existential rules.*

5 Conclusion

We developed a decision procedure for answering Boolean conjunctive queries against clique guarded existential rules, based on ordered resolution and a sophisticated form of selection. Since this is still ongoing work, current work is focused on offering formal proofs to support our claims.

References

[1] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.

[2] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *Proc. LICS'10*, pages 228–242. IEEE, 2010.

[3] Hans de Nivelle and Maarten de Rijke. Deciding the guarded fragments by resolution. *J. Symb. Comput.*, 35(1):21–58, 2003.

[4] Harald Ganzinger and Hans de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. LICS'99*, pages 295–303. IEEE, 1999.

[5] Erich Grädel. Decision procedures for guarded logics. In *Proc. CADE'16*, pages 31–51. Springer, 1999.

[6] Johan van Benthem. Dynamic bits and pieces. Research Report LP-97-01, Univ. Amsterdam, 1997.

Experiments with Selection of Theorem Proving Heuristics

Edvard K. Holden Konstantin Korovin

The University of Manchester, UK

Abstract: Heuristics are crucial for the performance and success of a theorem prover. Selecting and scheduling heuristics has traditionally been handled by system experts. However, in recent years this challenge has been approached with machine learning [1, 3, 5, 6]. We describe multiple machine learning approaches for heuristic selection, then evaluate the techniques with iProver using the CASC-26 FOF problem set.

1 Introduction

Automatic Theorem Provers (ATP) utilise sophisticated algorithms to search for a proof in what can be an infinite search space. Numerous parameters tune the algorithms, where an efficient and successful parameter setting is known as a *heuristic*.

A robust set of heuristics is beneficial for the prover, but running the heuristics in an arbitrary order is computationally ineffective. Therefore, it is necessary to select a single optimal heuristic or to devise a custom heuristic schedule for a given problem.

In this paper, we compare prediction and scheduling approaches which utilise Machine Learning (ML), and syntactic properties of the problems. The different techniques are evaluated with iProver [4].

2 Predicting Heuristics

2.1 Supervised Heuristic Selection

Heuristic selection can be addressed with supervised machine learning, which requires features x and labels y. The features and labels form a dataset $D = [(x_1, y_1), ..., (x_n, y_n)]$, and the goal of the selection function is to learn the mapping function $f : x \to y$.

A problem p is a first-order problem in clausal form. A feature vector x_n represents the problem p_n, and the set of features utilised are counts of syntactic problem properties, as shown in table 1. The properties are computed after pre-processing in the prover. Log-scaling is further applied before learning.

Problem Properties	
Clauses	Conjectures
EPR Clauses	Horn Clauses
Unary Functions	Binary Functions
Literals	Equality Literals

Table 1: List of Features

The problem set is from the *CASC-26 FOF* dataset, and is combined with machine learnt heuristics generated in [3]. The minimum set cover of the heuristics is 7, which solves a total of 257 out of 500 problems. The heuristics in the minimum set cover are used for learning and selection.

2.2 Single and Scheduled Heuristics

There are two standard ways of selecting heuristics in ATP. The first method is to run a single optimal heuristic on a problem until a global timeout is met or a solution is found. The second procedure is to construct a scheduler. Here we construct a scheduler by imposing a ranking on the heuristics. A solving time is estimated for each problem-heuristic pair. The heuristics are then ran in order for the estimated time, until a solution is found or a global time-limit is met.

2.3 Prediction Approaches

Multiclass Prediction

In the multiclass setting, the model learns the mapping from a problem x_n to the heuristic y_n. The output of this model is a score on each heuristic which is used to impose a ranking, or a single selection by choosing the heuristic with the highest score. We use two different labelling approaches.

A1: Multiclass Fastest Label Prediction

In the fastest label prediction, every problem is labelled with the heuristic that solves it the fastest.

A2: Multiclass Temporal Label Prediction

In the temporal case, the labelling is carried out by assigning similar problems to the same heuristic. First, similar problems are clustered together. Then the problems in each cluster are labelled with the locally best heuristic, for the problems that are solved under this heuristic. Further, the locally best heuristic amongst the unassigned problems are computed and assigned to the unassigned problems that are solved under this heuristic. This process is repeated until all the problems are labelled.

Multi-Label Prediction

In multi-label setting, the model learns the mapping between a problem and a set of labels. We build the ML model by utilising a one-vs-the-rest strategy. To construct the decision rule, the probabilities for the heuristics being positive are ranked in order. The most probable heuristic is used as the single prediction, while the ranking constitutes the schedule. We experiment with 2 different multi-labelling methods.

Metric	Base	Static Heuristic					Heuristic Scheduler				
		A1	A2	B1	B2	C	A1	A2	B1	B2	C
Problems Solved	217	246	248	**253**	**253**	182	238	236	240	239	223
Average Time	19.7	7.0	12.9	6.2	15.9	**6.1**	10.5	16.4	9.4	17.6	8.8

Table 2: Practical Evaluation of the Approaches

B1: Multi-Label Fastest

In the fastest label case, a problem is labelled as positive if the heuristic is the fastest for the particular problem. Otherwise, it is labelled negative. This results in a single positive label per problem and is similar to the approach by Bridge et al. [1].

B2: Multi-Label Solved

In this approach, a problem-heuristic pair is labelled as positive if the heuristic solves the problem. Otherwise, it is labelled negative. This results in every problem being labelled with the heuristics that yields a proof of the problem.

C: Smallest Solving Time Prediction

For the last approach, the models learn the solving time y of a problem when ran over heuristic h. Thus, it learns the function $f_h : x_p \rightarrow time(h, p)$ by a regression model.

A regression model is constructed for every heuristic to learn the solving time for every solved problem. For the decision rule, the estimated solving times are ranked from smallest to lowest, where the heuristic with the smallest solving time is selected first. This is similar to the selection in MaLeS [5].

3 Experiments and Evaluation

We implemented the approaches above with XGBoost [2] as the ML model and iProver as the prover. XGBoost is an efficient random forest model which supports both regression and prediction. The baseline for the experiment is the top heuristic in the set cover, which solves 217 problems.

The models are trained and evaluated using 5-fold cross-validation. For the decision rules we evaluate their accuracy for the single heuristic according to the model's true labels. The test accuracies are shown in table 3.

The accuracy may indicate how well the models learn, but it does not necessarily provide information about how it performs at the task at hand. Therefore, the selection approaches are evaluated on the number of problems solved. They are also evaluated based on the average solving time per problem, for the problems in the intersection of all solved problems. The practical performance results are shown in table 2.

We evaluated the approaches both for the single heuristic selector and as a scheduler. The time allocation for each heuristic is allocated by a solving time estimator similar to the approach described in **C**.

Approach	Accuracy
A1	16
A2	**85**
B1	78
B2	26
C	25

Table 3: 5-Fold Cross Validation Accuracy

4 Conclusion

We have evaluated multiple heuristic selection techniques with machine learning and syntactic problem features. From the results, we observe that the accuracy scores vary greatly and that the models with the best accuracies also tend to have the best performance results. However, as **B2** learns the set of solvable heuristics instead of optimal heuristics, the accuracy score is not necessaril a fair metric.

For four out of the five approaches, running a schedule has a negative impact. However, we have also experimented with neural networks, where the scheduler contributed positively. The neural networks generally had weaker results, probably due to little training data. This observation suggests that scheduling improves weakly performing ML models but obstructs strongly performing models.

The multi-label approaches solves the most problems overall, and **B1** both solves the most problems and solves them second fastest. The task in **B1** is fundamentally the same as **A1** but in the multi-label case. This suggests that the complexity of multiclass learning is challenging to model with the given features. It remains to be seen whether the results will persist over a larger dataset.

References

[1] J. P. Bridge, S. B. Holden, and L. C. Paulson. Machine learning for first-order theorem proving. *Journal of Automated Reasoning*, 53(2):141–172, Aug 2014.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[3] E. K. Holden and K. Korovin. Smac and xgboost your theorem prover. In *Proc. 4th Conference on Artificial Intelligence and Theorem Proving*, 2019.

[4] K. Korovin. iProver - an instantiation-based theorem prover for first-order logic (system description). In *IJCAR 2008. Proceedings*, pages 292–298, 2008.

[5] D. Kühlwein, S. Schulz, and J. Urban. E-males 1.1. In M. P. Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, 2013. Proceedings*, volume 7898 of *LNCS*, pages 407–413. Springer, 2013.

[6] M. Rawson and G. Reger. Dynamic strategy priority: Empower the strong and abandon the weak. In B. Konev, J. Urban, and P. Rmmer, editors, *6th Workshop on Practical Aspects of Automated Reasoning (PAAR)*, number 2162 in CEUR Workshop Proceedings, pages 58–71, Aachen, 2018.

Creating an Intelligent System for Bankruptcy Detection: Semantic data Analysis Integrating Graph Database and Financial Ontology

Natalia Yerashenia[1] Alexander Bolotov[2]

[1] University of Westminster, London, United Kingdom, w1578366@my.westminster.ac.uk
[2] University of Westminster, London, United Kingdom, A.Bolotov@westminster.ac.uk

Abstract: In this paper, we propose a novel intelligent methodology to construct a Bankruptcy Prediction Computation Model, which is aimed to execute a company's financial status analysis accurately. Based on the semantic data analysis and management, our methodology considers Semantic Database System as the core of the system. It comprises three layers: an Ontology of Bankruptcy Prediction, Semantic Search Engine, and a Semantic Analysis Graph Database.

1 Introduction

We propose a concept of an intelligent, analytical system to perform the prediction of the companies' bankruptcy. The system processes financial information of a company and undertakes a comprehensive investigation of companies' financial activities during a particular designated time period. We aim at creating a *Bankruptcy Prediction Computational Model (BPCM)* which is capable of the automated construction of an expert analytical report, where various data and information are presented reliably and objectively.

The main feature of the proposed system is the consolidation of the information management with the decision-making process to serve the prediction. This involves modern methods of searching, processing and storing potentially large amount of heterogeneous data together with advanced machine learning methods. In this paper, we define the process of the *Semantic Database System* construction, a novel development, which comprises *an Ontology of Bankruptcy Prediction (OBP)*, *a Semantic Search Engine (SDS)* and *a Semantic Analysis Graph Database (SAGRADA)*.

In Fig. 1 we present a flowchart of a BPCM, which is a visual representation of our research methodology.

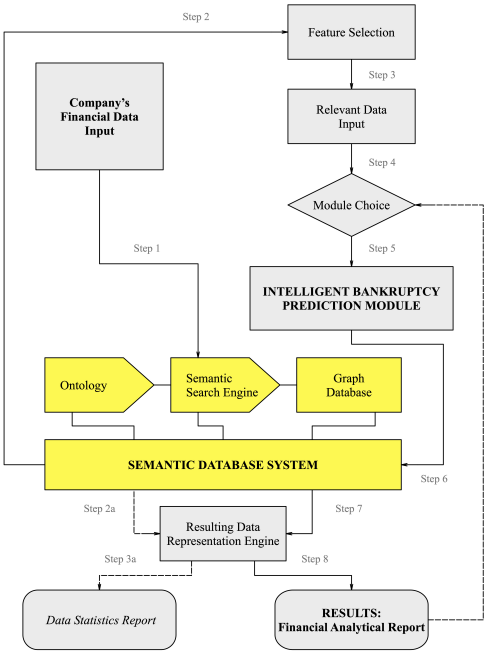


Figure 1: Bankruptcy Prediction Computational Model flowchart

2 Developing a Semantic Database System

We argue that the financial dataset to be analysed for our purposes, figuratively speaking, can be characterised by four 'V' and 'R'. It shares most (four out of five big 'V') of the qualities of *Big Data* – Variety, Velocity, Veracity and Value [4] being not dependent on the Volume. However, we underline the fifth, 'R', feature of these financial data – an extremely high level of Relationships. Indeed, similar to big data, in our case, we have heterogeneous data, coming from different sources. These components of a company's financial system can be (and usually this is the most common practice) described in the form of *relational tables*, e.g. it is easy to present a balance sheet or income statement in such a way. However, to show the interconnections between all elements of these tables, it is necessary to create a number of tables of a different structure containing thousands of objects. In this case, the efficiency of database management and search are substantially affected. For example, it becomes problematic to formulate a general query to several databases, because of the difference in objects and attributes of the domain or changes in objects over time. When the data are inserted, updated or deleted, the integrity constraints for the database with changing objects should be checked and assured that the data will be consistent after all modifications. Also, there is a problem of the integration of new nodes into the system. When adding a new node, it is essential to check the data and the data schema for consistency with the information already available in the system.

There are *NoSQL systems* that extend the capabilities of relational databases and tackle the requirements of the *'Big Four V + R'*.

Financial Ontology. The ontology presentation format defines the mechanisms to store concepts and their relationships in the library; it is a method of transmitting ontological descriptions to other consumers and a method of processing its concepts. Specific ontology presentation languages have been developed as ontological description formats (OWL, RDF, KIF) [1].

Ontologies are used as data sources for many software applications such as information retrieval, text analysis, knowledge extraction, and other information technologies, allowing more efficient processing of complex and diverse information. This way of representing knowledge enables applications to recognise those semantic differences that are obvious to people but not known to the computer.

The main and most crucial component of the financial risk management of a company is the knowledge base. Our approach to building an ontology describes the basic concepts of financial analysis, as well as the objects that serve as sources of knowledge for predicting a company's bankruptcy. It also contains the concepts and relationships required for the formation of a hierarchy of knowledge fields and the subsequent use of this hierarchy by various applications. In addition, expert rules and regulations can be described in terms of ontology, which significantly increases their level of succinctness and transparency for the users.

The structure and the content of the OBP are based on the experience of analysts specialising in the theory and practice of bankruptcy prediction. The hierarchy reflects a number of the most popular indicators used to conduct a financial analysis of a company, as well as their origin (documents and concepts to which they relate) and the relationship of these indicators to each other. Financial analytic factors form the penultimate row of the hierarchy, while the principal generalising object is the concept of Companys Financial Records. The last row in the hierarchy contains linguistic variables that will be later involved in the development of machine learning computational modules.

Graph Database. *Graph DB* (for instance, Neo4J) are an example of *NoSQL databases* aimed at representing semantical data [2]. Graph databases are used for storing, processing and automated visualisation of standard structural elements. A typical Graph DB usually contains some reference information regarding objects [3]. Therefore, the user/designer does not have to spend time searching for this information in the DB directories. It also reduces the number of possible human factor related errors. Graph DB enables to create standard elements automatically, which significantly reduces the design time.

Neo4j[1] is an open source Graph DB management system implemented in Java. This Graph DB environment stores data in a proprietary format specifically adapted for the presentation of graph information; this approach, in comparison with the modelling of a graph database, using a relational *Databases Management Systems*, allows for additional optimisation in the case of data with a more complex structure.

We emphasise that the OBP structure is an excellent basis for the Semantic Analysis Graph Database which is used as a repository of the financial data for BPCM. So, we intend to apply an existing solution of creating and managing GB and integrate it into our novel approach.

We have implemented a prototype Graph DB, SAGRADA, in Neo4j. The basic concepts in a Graph DB are nodes (an object of the database), relations (graph edges) and their properties. In our case, the nodes of the graph are financial ratios, financial indicators, and the documents containing them. Our graphical repository has 29 nodes divided into three categories – Ratio, Criteria (financial indicator), Statement, and 52 relationships between them (of two types – direct and inverse).

3 Conclusions

Based on the analysis of various modern approaches to the processing and storage of the heterogeneous data related to the financial analysis, we proposed a novel intelligent methodology to construct a BPCM. Our methodology is based upon the utilisation and integration of the semantic data management methods. Following this methodology, we have introduced a novel layered architecture for this BPCM, which integrates the Semantic Database System and a set of modern machine learning algorithms. We have implemented the principles of the new OBP and the Semantic Analysis Graph Database on the example of a company financial record.

Further, we will improve the structure of the OBP creating its formal conceptual representation through OWL/RDF languages. We will also work on further enhancement of the SAGRADA itself. We will also tackle a problem of the data exchange between the structural parts of the SDS finding a way to transfer data in various directions automatically.

References

[1] Helena Dudycz and Jerzy Korczak. Conceptual design of financial ontology. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1505–1511. IEEE, 2015.

[2] Venkat N Gudivada, Dhana Rao, and Vijay V Raghavan. Nosql systems for big data management. In *2014 IEEE World congress on services*, pages 190–197. IEEE, 2014.

[3] Jaroslav Pokornỳ. Graph databases: their power and limitations. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 58–69. Springer, 2015.

[4] Xizhao Wang and Yulin He. Learning from uncertainty for big data: Future analytical challenges and strategies. *IEEE Systems, Man, and Cybernetics Magazine*, 2(2):26–31, 2016.

[1] `https://neo4j.com/product/`

Reinforcement-Learned Input for Saturation Provers

Michael Rawson Giles Reger

University of Manchester, UK

Abstract: Many of today's best-performing automatic theorem provers (ATPs) for first-order logic rely on *saturation* algorithms. However, to date the most successful work on applying machine-learned guidance to ATPs relies on tableaux methods, as these are friendlier toward machine learning algorithms. We describe a reinforcement-learning system which selectively infers new clauses from an input problem. The system (in progress) is rewarded if generated clauses are subsequently used by the first-order saturation prover VAMPIRE in a proof. In this way, the system learns to generate new clauses which are important in the proof search, but might otherwise not be selected for some time in Vampire's proof search algorithm. The system is implemented via Q-learning, with a graph neural net processing the structure of the clause set acting as an approximator.

1 Saturation Provers

Saturation-based ATPs such as VAMPIRE [11] utilise an algorithm which attempts to produce a *saturated* set of first-order clauses from their input: that is, clauses closed under all possible inferences in their calculus (usually ordered resolution and superposition). If the empty clause is found, then this witnesses refutation and a proof may be given. Conversely, if a saturated set is produced, then the input is satisfiable and therefore not a theorem. On hard problems, neither case will occur and the prover will diverge and continue to generate a very large set of clauses until either time or memory runs out.

This approach lends itself to efficiency, but it is inherently unguided and ill-conditioned with respect to the input. In VAMPIRE and other similar ATP systems, a number of options and manually-programmed heuristics are available in order to try and delay divergence or find a proof faster on certain problem classes: these are made more useful by *portfolio modes* [9] in which many combinations of options are run in sequence in order to ameliorate the brittle performance of the underlying prover. Such provers have accumulated several pieces of experimentally-validated "folklore" [10, 1], notably

> *"There is (at least) one clause, which, once found, significantly accelerates finding a proof."*

A system which learns to find such a clause early would therefore be very useful.

2 Guidance for ATP systems

The general idea of guiding ATP systems with a machine-learned heuristic is hardly novel: experiments with premise selection [12] and internal guidance of tableau-based provers [4] have produced successful results. However, guiding saturation-based theorem provers has proved more difficult: there is a tradeoff between learning power [6] and throughput [2]: the better the heuristic, the more computationally expensive it is and therefore the greater the impact on prover throughput.

The system described falls somewhere between premise selection and internal guidance: while it can select clauses, it can also perform inferences which may help proof search directly (even acting as an poor man's theorem prover in its own right). However, it does not affect internal proof search routines and therefore does not impact prover throughput.

3 Reinforcement Learning Clausal Search

Reinforcement learning aims to produce agents which learn to perform actions in an environment while maximising a reward function [3]. Theorem proving with clauses can be viewed as performing actions (generating inferences) in an environment (the current clause set), aiming to find a reward (the empty clause). In principle, this would be a good way to build a learning theorem prover in its own right. However, this approach has a few problems:

1. The reward function here is very sparse, with only a single reward at the end of proof search.
2. Proofs may be very deep, which an agent at the start of its training may not be able to find in reasonable time.
3. The space of all possible inferences with all available clauses is very large, particularly in large knowledge bases.

In order to make this approach more friendly to reinforcement learning, a few modifications are introduced. An existing ATP system is employed after a fixed number of steps on the modified clause set: if the existing ATP can prove the modified problem making use of the generated clauses, a reward is applied. This helps with issues (1) and (2).

To reduce the number of possible inferences (3), the system is allowed to select which clauses from the input it wishes to use in generating inferences. We therefore partition the system into two disjoint clause sets:

selected: initially empty, the set the system has selected

available: input clauses, and inferences from *selected*

4 System Description

The system first uses the VAMPIRE ATP to clausify an input problem, obtaining a set of input clauses — those related to the conjecture are tracked and marked as such to enable the system to be goal-directed if it chooses. Any other reasonable clausifier could be used here.

Then, the system is allowed to choose one clause at a time from *available* and move it to *selected*: any inferences from the new clause with others in *selected* are then added to *available*. The agent's action-selection policy may be slightly randomised (e.g. ϵ-greedy or Boltzmann selection [8]): in training this gives an exploration effect, in testing this allows for multiple distinct clause sets to be produced: multiple proof attempts can then be run in parallel.

After a fixed number of clause selections, the process is halted and VAMPIRE runs in its default mode for a short time on all the original clauses, plus the generated clauses in *selected*. Clauses are ordered in such a way that VAMPIRE prefers those in *selected* before those in *available*. If clauses in *selected* are used in proof search, a reward signal is propagated, and the agent may learn from its experience. A mild punishment signal is propagated for clauses that were selected but not used.

The particular reinforcement algorithm used presently is Q-learning [13], with a graph convolutional network [5] applied to the entire clause set's structure as a Q-function approximator.

5 Future Work

As well as completing the implementation and evaluation of the system, there are several possible directions for future work:

- Different (neural) function approximators. The existing neural architecture is known to perform well on similar problems, but there is much to explore here.
- Other learning approaches: modern approaches such as A3C [7] claim significantly better performance on some domains.
- Suitable VAMPIRE modes for this environment, and the effect the underlying ATP algorithm has on learned policy.
- Modified reward functions: appearing in the proof is not the only way a clause may be useful within VAMPIRE. For example, if a clause subsumes a large section of search space, it may still increase proof search performance despite not appearing in the eventual proof. Possible future rewards might include the number of subsumed clauses, and the time taken for the ATP to run.
- There are a significant number of hyper-parameters within this system, all of which may be tuned to increase system performance on a given benchmark set.

References

[1] Kryštof Hoder, Giles Reger, Martin Suda, and Andrei Voronkov. Selecting the selection. In *International Joint Conference on Automated Reasoning*, pages 313–329. Springer, 2016.

[2] Jan Jakubův and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In *International Conference on Intelligent Computer Mathematics*, pages 292–302. Springer, 2017.

[3] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[4] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems*, pages 8822–8833, 2018.

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[6] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. *arXiv preprint arXiv:1701.06972*, 2017.

[7] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[8] Warren B Powell and Ilya O Ryzhov. Ranking and selection. *Chapter 4 in Optimal Learning*, pages 71–88, 2012.

[9] Michael Rawson and Giles Reger. Dynamic strategy priority: Empower the strong and abandon the weak. *AITP 2018*, 2018.

[10] Giles Reger and Martin Suda. Measuring progress to predict success: Can a good proof strategy be evolved? *AITP 2017*, 2017.

[11] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI communications*, 15(2, 3):91–110, 2002.

[12] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems*, pages 2786–2796, 2017.

[13] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Chained Strategy Generation: A Technique for Balancing Multiplayer Games Using Model Checking

William Kavanagh Dr Alice Miller

W.Kavanagh.1@research.gla.ac.uk, Alice.Miller@glasgow.ac.uk
University of Glasgow

Abstract: Game balancing is the task of ensuring that a game is both fair to its player and interesting to play. Many games offer players a choice of disparate game material (such as cars, characters or weapons) and ensuring that these materials are all *balanced* is a notoriously complex task. We introduce a new technique called Chained Strategy Generation (CSG) that uses probabilistic model checking and strategy synthesis to model high-level competitive play to represent games being played over a long period of time. We then apply CSG to a case study to show how it can be used to help describe a game that is cyclically balanced, ensuring players have a number of impactful decisions, which will lead to more interesting gameplay.

1 Introduction

An abstraction of a game known as the *metagame* [1], describes the evolving state of play between the strategies and materials offered to players. Games are often designed around a set of game material – such as heroes, cards, vehicles or weapons – which players can choose from at the start of the game. A game is considered well balanced if it has a *healthy* metagame, where a variety of strategies for different materials are popular over a long period of time, with new strategies becoming prominent when they are effective against the current *meta*. In order to achieve this, the ways of playing need to form an intransitive relationship, similar to Rock Paper Scissors. Rock Paper Scissors is frequently used as the foundation for game design where all materials are organised in small cycles with each material unit able to beat the next with a high probability. A relationship where one unit is designed to beat another may make a game seem imbalanced, but as part of a network of similar relationships where all of a unit's drawbacks are compensated by advantages the whole game is balanced. In a system like this quantifying which materials are best is not possible, one must consider what materials are best at a given point in time. Materials are judged based on what strategies are popular (or what is the current meta) and how well they can perform against them.

We have devised a mechanism for representing a full metagame representation which can be analysed to inform design decisions about which materials are too powerful and which are too weak. To do this, we use the probabilistic model checker Prism [3] to generate strategies using strategy synthesis [2]. By repeatedly doing this we inductively define a set of *effective strategies*, a process we call chained strategy generation (CSG). By studying these effective strategies, the materials that are used by them and how well other materials can perform against them we can make judgements on the comparative strengths of the materials. We also identify dominant strategies and dominated material, without the need to consider the full strategy sets of all material units.

2 Methodology

Let **G** be a two-player game between players $\{p1, p2\}$ and M the set of material. We define a strategy in terms of players and material such that $\theta(p, m)$ is a strategy for player p using material $m \in M$. Define a particular strategy $\theta_{naive}(p, M)$ which chooses material randomly with a uniform distribution at the start of the game and chooses actions randomly with a uniform distribution of all actions available. Let $\theta_{p \to p'}$ be a function that maps a strategy for player p into the equivalent strategy for p' and let $P_{win(1)}(\theta(p, m), \theta'(p', m))$ be the probability that strategy $\theta(p, m)$ beats $\theta'(p', m)$.

By modelling the game as a Markov Decision Process where one player has a fixed strategy and the other is represented as having a nondeterministic choice of actions, we can use model checking to generate the strategy with the highest probability of winning against the fixed strategy, i.e. the optimal *counter*. For example, given some strategy $\theta(p, m)$, we can use model checking to find the strategy $\theta'(p', m)$ for which $P_{win(2)}(\theta(p, m), \theta'(p', m))$ is a maximum. This is equivalent to finding the adversary which maximises the probability of reaching a state in which $p2$ wins. We denote the strategy generated by this process $counter_{p'}(\theta(p, m))$.

CSG is described by:

1. **For** $m \in M$:
   $\theta^*(p1, m) := counter_{p1}(\theta_{naive}(p2, M))$
2. *The meta* at iteration 0 (θ^0) is $\theta^*(p1, m)$ for which $P_{win(1)}(\theta^*(p1, m), \theta_{naive}(p2, M))$ is maximum
3. $k := 1$
4. **For** $m \in M$:
   $\theta^*(p1, m) := counter_{p1}(\theta^{(k-1)}_{p1 \to p2})$
5. $\theta^*(p1, m)$ for which $P_{win(1)}(\theta^*(p1, m), \theta^{(k-1)}_{p1 \to p2})$ is maximum is *the meta* at iteration k (θ^k)
6. **If** $\theta^k \neq \theta^j$ for all $j < k$: k++; **Goto** step 4
   Else: **Quit**

CSG terminates under one of two conditions, either a dominant strategy has been identified – a strategy best

played against by itself – or a cycle of effective non-dominant strategies have been found. A dominant strategy suggests that a game is poorly balanced and easily solved, players would soon converge upon the dominant strategy and repeated plays of the game are likely to involve both players employing the dominant strategy. In a well-designed game it would take a long time for CSG to terminate and most material units would at some point be used by a meta strategy.

3 Results

We implemented CSG on a simple case-study of a 2-player turn-based game where players choose a pair of characters from a choice of 5: a Knight, an Archer, a Wizard, a Rogue and a Healer. These characters are all qualitatively differentiated. The Archer can target multiple opponents, the Wizard can stun an opponent, preventing them from acting on their next turn, the Rogue can execute low-health opponents and the Healer will increase either their current health value or that of an ally upon successfully attacking an opponent. All characters have have a maximum health value, an accuracy value and a damage value, the healer also has a heal value – the value by which they increase an ally's health – and the rogue also has an execute value – the value at which they can do damage equal to the opponent's current health. These 17 attributes are the *configuration* of the game. A coin is flipped to decide who goes first then players take it in turns to attempt one action from any of their alive characters targetting any of their opponent's alive characters. Our aim is to assess how balanced the materials are for a given configuration and, if needed, to suggest what to change.

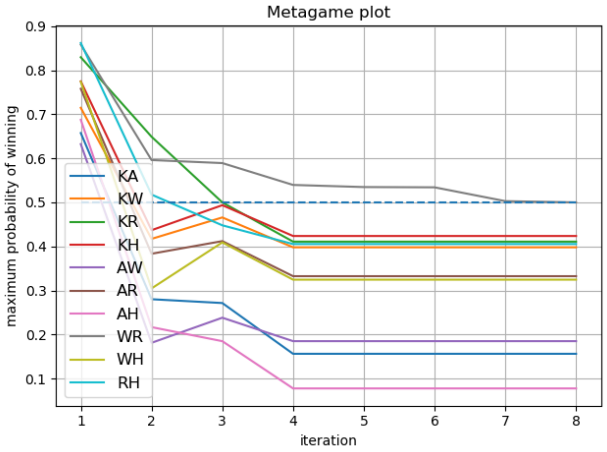


Figure 1: CSG performed on a poorly balanced game.

Character initial	K	A	W	R	H
Health	9	6	7	7	7
Accuracy	0.5	0.85	0.75	0.65	0.7
Damage	4	2	2	3	2
Execute/Heal	-	-	-	6	2

Table 1: A configuration for the case study

One way the results of CSG can be analysed is to plot the probabilities of each material played maximally against the meta strategies. This is shown in fig. 1 for the configuration in table 1. A dominant strategy is clearly identified for a Wizard-Rogue pair (WR). Because our case study uses *teams* of material, we study the aggregate win chance for each character when all 10 pairs play against each other using their final strategies. Ideally these values would be close to 0.5 to indicate that each pair is as likely to win as each other overall. The results for each character are shown in table 2. It is clear that the Archer is too weak and should be made stronger whilst the Rogue is too strong and should be brought more in line with the other characters. We reconfigured the Archer to have 7 health and 0.9 accuracy and the Rogue to have an execute range of 5 and accuracy of 0.6. The result of this change is that average win rates for all characters are more uniform as shown in table 2 and no dominant strategy is identified. This shows how effectively CSG informs game design in spite of the sensitivity of metagame development.

Char. initial	K	A	W	R	H
Former	0.497	0.404	0.5	0.602	0.497
Updated	0.478	0.521	0.484	0.514	0.504

Table 2: Table comparing the aggregate win probabilities for all characters between the two configurations to 3dp.

4 Conclusion

We have shown how CSG allows for analysis of a game's balance and can be used to inform better game configurations. By predicting the direction of the metagame, CSG allows game designers to compare material units in terms of how viable they are throughout the game's lifespan, rather than at a single point in time.

Future work will involved the development of an automated tool that analyses and reconfigures games itself, until it finds the optimal configuration within user defined bounds. This tool has the potential to be highly useful for game development and to furthering understanding of complex game systems. Ultimately, by ensuring the material is developed to be fair, CSG will help designers to make games which are more interesting and more fun to play.

References

[1] M. Debus. Metagames: on the ontology of games outside of games. In *Proceedings of the International Conference on the Foundations of Digital Games, 2017.*

[2] Ruben Giaquinta, Ruth Hoffmann, Murray Ireland, Alice Miller, and Gethin Norman. Strategy synthesis for autonomous agents using PRISM. pages 220–236, 2018.

[3] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. Int. Conf. Computer Aided Verification (CAV'11).*

Using model checking in the design of a sensor network protocol

Ivaylo Valkov Alice Miller

University of Glasgow

Abstract: We investigate how the PRISM and Alloy model checkers can be used in the design of a sensor network communication protocol. We introduce the two model checkers and illustrate how Alloy can be used to specify and analyse an existing communication protocol WirelessHART. We then propose how PRISM and Alloy will be used in the *design* of a new protocol, Ctrl-MAC, which is currently being developed as part of an EPSRC funded program grant, S4: Science of Sensor System Software. The aim is to exploit the strengths of each approach to allow us to select parameters and configurations to optimise the protocol.

1 Introduction

Testing is the most commonly used method for validation of software systems. However testing alone can not provide guarantees of complex system behaviour. Concurrent systems, such as communication protocols, are particularly hard to verify using testing. In such systems we want to prove temporal properties such as: "when a message is sent it will eventually arrive at its destination", or "if a message is sent from a component then it will receive an acknowledgement before the timeout period has elapsed".

Formal methods are commonly used for the verification of software and hardware systems. They comprise a range of techniques based on mathematics and logical reasoning, and are important in the creation of more robust and reliable systems. One such technique is model checking. We propose using model checking in the *design* of a new sensor network protocol. We can identify and prove properties of the protocol as it is developed - and adjust parameters accordingly. This differs from the common use in which properties of an existing protocol are verified, with no option to modify the protocol.

2 Model checking

Model checking is the process of creating a formal model of a software or hardware system and then using a software tool, called a model checker, to automate the search for proofs of or counterexamples to some properties of the system. The syntax and structure of the model that is being created depends on the choice of model checker, as each model checker relates a model to the underlying logical reasoning and logical concepts in a different way.

PRISM [5] is a probabilistic model checker that allows for the verification of a number of Markov chain variants, like Discrete Time Markov Chains (DTMCs) and Markov Decision Processes (MDPs). It has been used to formally verify quantitative properties of many network protocols including the device discovery phase of Bluetooth [2] and the CSMA/CA mechanism of the 802.15.4 based Zigbee standard [3].

The Alloy Analyzer (Alloy) is a model checker that uses a simple and powerful first-order logic language for specifying models which are then analysed with off-the-shelf SAT solvers. This allows models to be created in an iterative and incremental manner: they can be verified, inspected, evaluated and modified during multiple iterations. Furthermore, Alloy allows the configuration of the setting on which properties are being verified to be easily changed. For example, in the context of protocol analysis, models are often confined to a small fixed number of devices, which are placed in a particular configuration which should best exhibit the property under verification. Using Alloy a family of configurations can be analysed simultaneously.

Alloy has been used in the past to provide formal proofs for a variety of network protocols and to find security flaws in others. For example it has been used to formally verify five web security mechanisms that relate to user-supplied information [1]. In [4] Alloy is applied directly to model web protocols in a novel security analysis technique. In Section 3 we illustrate the use of Alloy for modelling an existing wireless protocol and in Section 4 we propose its use, along with PRISM, in the design of a new protocol.

3 An example: WirelessHART

We have investigated the use of Alloy for protocol analysis within the context of an existing protocol, namely the WirelessHART protocol, based on the IEEE 802.15.4 protocol standard.

WirelessHART is a short-range network protocol whose main goal is to perform low-cost communications over a network in such a way as to preserve battery life. It is a centralised protocol with one device acting as the personal area network (PAN) coordinator for the network. The protocol distinguishes between reduced function devices (RFDs), that are only able to gather and send data, and full function devices (FFDs), that are capable of transferring data from other nodes. All of the data is gathered at the PAN coordinator, which must be an FFD. Fig 1 shows an example of such a network.

As an illustration, we present below a small fragment of Alloy in which we declare the basic entities (atoms) that will be used in the model. These are referred to as signatures (*sig*).

```
// There isn't a device that is not
// a RFD or FFD
abstract sig Device { }
// Devices are either reduced function
// or full function
sig RFD, FFD extends Device { }
```

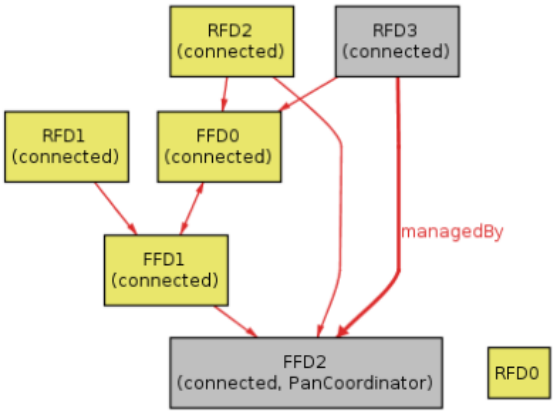


Figure 1: Network consisting of 3 FFDs and 4 RFDs. One device is not connected to the network.

After specifying the types of devices that exist we need to define how they relate in a network. To do so we create a Network signature which contains a number of nested relations. The *connected* relation is used to specify which devices belong to a given network. This is particularly useful when modelling scenarios where devices join or leave a network.

```
sig Network {
  connected: set Device,
  PanCoordinator: FFD & connected,
  managedBy: (connected - PanCoordinator)
                -> (FFD & connected),
  // RFDs cannot receive data
  receiveFrom: FFD -> Device
}{
  // connected devices are reachable
  // from the PanCoordinator
  connected in PanCoordinator.*receiveFrom
  // managedBy: inverse of
  // receiveFrom
  managedBy = ~receiveFrom
}
```

The *PanCoordinator* relation is used to specify a single device which acts as a central device for the network. The relations: *managedBy* and *receiveFrom* denote immediate connections between two devices. Finally, the *connected* relation defines that to be connected to a network a node means to be reachable from the central node.

In order to ensure that devices cannot send data to themselves, we add an additional constraint to the model:

```
fact {all nw: Network | all d: Device |
        d->d not in nw.receiveFrom }
```

We can now use Alloy to generate an instance of this model for a defined number and type of devices. Fig 1 is an example of such an instance. Manual inspection demonstrates that there are no self-related devices and that connected devices are appropriately marked.

4 Model checking for sensor network protocol design

PRISM is an obvious formalism for modelling communication protocolols and our wirelessHART example demonstrates the suitability of Alloy in this context. We propose to model and analyse a wireless communications protocol that is currently under development, in both PRISM and Alloy. Ctrl-MAC is a sensor network communication protocol that is being developed as part of the Science of Sensor Systems Software (S4) project. This is an EPSRC-funded project held by the University of Glasgow with the Universities of St Andrews and Liverpool and Imperial College. Ctrl-MAC is similar to WirelessHart in that they both use time division multiple access governed by a central gateway node. Its main goal is to provide reliable communication within a given time constraint for Cyber Physical Systems (CPS) such as water distribution systems and electric grids. By using model checking throughout the development of the protocol we will inform its design by choosing parameters and configurations to optimise performance.

References

[1] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song. Towards a formal foundation of web security. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 290–304, 2010.

[2] M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A formal analysis of Bluetooth device discovery. *Int. Journal on Software Tools for Technology Transfer*, 8(6):621–632, 2006.

[3] M. Fruth. *Formal Methods for the Analysis of Wireless Network Protocols*. PhD thesis, Oxford University, 2011.

[4] A Kumar. A lightweight formal approach for analyzing security of web protocols. In *Proc. RAID 2014*, pages 289–298.

[5] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV 2011*, pages 585–591.

Exploring Secure Service Migration in Commercial Cloud Environments

Gayathri Karthick, Dr.Florian Kammueller, Dr.Glenford Mapp, and Dr.Mahdi Aiash

School of Science and Technology, Middlesex University, Email: gk419s@live.mdx.ac.uk, F.kammueller@mdx.ac.uk,, G.Mapp@mdx.ac.uk, M.Aiash@mdx.ac.uk

Mobile users are making more demand of future networks. They want to access the applications such as Video and audio which involved the allocation of network resources. In addition, Commercial Providers are actively advertising their resources such as CPU, Memory, Storage, and Network which must be allocated appropriately according to the capacity of Cloud Servers. The advent of virtual machine technology for example, VMware, Container technology, such as Docker and Kubernetes, have made the migration of services between different Cloud systems possible. This enables the development of mobile services that can ensure low latency between servers and their mobile clients resulting in better Quality of Service. Though there are many mechanisms in place to provide support for mobile services, a key component that is missing is the development of security protocols that allow the safe transfer of servers to different Cloud environments. This lack of security has resulted in technical failures, service disruption, and unavailability that directly affect the Cloud Provider's profit. This paper proposes a new Resource Allocation Security Protocol for Secure Service Migration. The new protocol has been verified in Avispa and Proverif and is being implemented in a new Service Migration Prototype in order to securely manage, allocate resources in Commercial Cloud Environments.

Mobile services, Security protocol, Vehicular Cloud, Avispa and ProVerif.

1. Introduction

Cloud computing facilitates the migration of data and services. This gives many incentives for data owners to migrate their data and services to Cloud Storage Platforms at low cost. One aspect of the research on mobile services that has been inadequate is support for security. In particular, it is important that servers do not end up being hosted on unsafe Cloud infrastructure which can hamper service delivery to mobile clients and also Clouds do not end up hosting malicious servers which can damage Cloud infrastructure. In our first attempt, using Avispa tool, we showed that the proposed protocol is safe under normal operation(1); In our second attempt, using ProVerif, we showed that the proposed protocol succeeds in three significant security properties namely: Secrecy, Authentication and Key exchange(2). In addition, we are building a Service Migration Prototype for service migration systems using Docker and Kubernetes to verify the proposed protocol.The rest of the paper is organized as follows. Section II describes background methodologies for our solution approach. Section III details Resource Allocation Algorithm while Section IV shows the design for the Service Migration Prototype. The paper concludes with Section V.

2. Solution Approach

In our new approach, we are working on secure service migration between Commercial Cloud infrastructures. The Resource Allocation Security Protocol (RASP) has been developed to support mobile services that allow the safe transfers of resources to different Cloud environments. The key components are Server SA which is presently located on Cloud CA. SA receives an advertisement from Commercial Cloud CB and therefore needs to consider whether the migration is feasible and can be done securely. The final key component is a Registry which contains all the records of Cloud Resources.The protocol is broken into four stages to clarify the necessary operations involved in secure migration. (3).

1. **Stage1: Advertisement**: Cloud CB actively advertises its resources which is picked up by server SA on Cloud CA.
2. **Stage2: Authentication of SA and CB as well as migration request and response**: Server SA first requests the Registry to authenticate Cloud CB and the resources it holds. Once it receives the approval from the Registry, it sends a migration request to Cloud CB. Cloud CB then requests the Registry to authenticate server SA and the resources it requires. Once this is verified, Cloud CB sends a positive migration response to server SA.
3. **Stage3: Migration transfer**: Server SA sends a message to Cloud CB to begin the migration transfer. Cloud CB begins the transfer and signals server SA when the transfer is completed.
4. **Stage4: Update of New service location to the Registry**: The new service SB is now running on Cloud CB and informs the Registry that it has been successfully migrated.

2.2 Results using Proverif. In our second attempt, by using symmetric session key(Ksc), the requested service is transferred to the new location CB. As we mentioned in the first attempt, nonces are used to protect the session between SA and CB. However,in the second attempt(Fig.12), the Ksc is used to do the actual transfer.Hence, this is a more secure mechanism than using nonces for the actual data transfer .

4.Resource Allocation Algorithm

The RASP algorithm has two key components for each Cloud Systems. one is total resources in terms of CPU, Storage,

Fig. 1. Proverif shows the results that RASP can preserve the secrecy and key exchange.

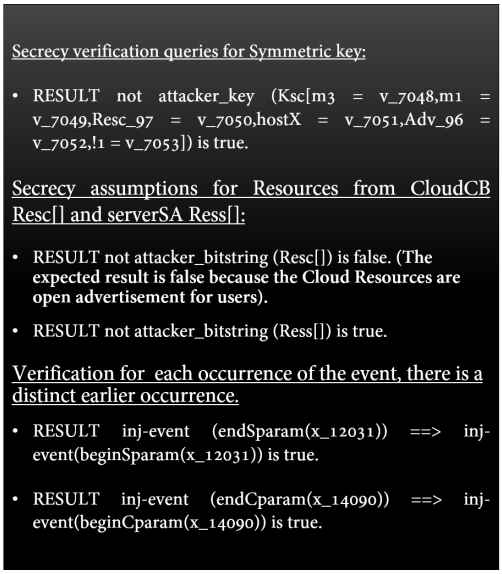


Fig. 2. Proverif shows the results that RASP can preserve the authentication and key exchange.

Network and Memory which is fixed accordingly to the capacity of the system. The second component is available resources which can be allocated to migrating services. Each Cloud Systems advertise their total and available resources to mobile services. In order to efficiently migrate the services, the first step is to verify whether its feasible to migrate based on the server requirements and the second step is to ensure that the service migration to the new Cloud Systems can be securely achieved. Once the server has verified that it is feasible to migrate the service, then RASP protocol is invoked to do the transfer. In the RASP protocol, the Registry also contains information about the capacity and available resources on the Cloud and thus can verify that the new Cloud System is a Valid Cloud and has the resources to host the server. In addition, according to the RASP protocol the server running on

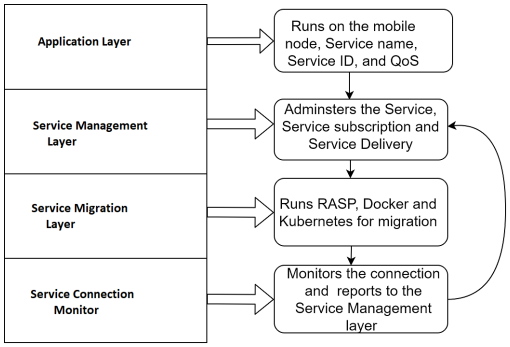


Fig. 3. Prototype Design shows the new Service Migration Prototype.

the new Cloud, will inform the Registry that the service migration is completed and therefore the Registry will update its Resource Database for the previous and the new Cloud Systems.

5.Service Migration Prototype

The new prototype has 4 layers and is an implementation of the service framework developed by Sardis (4). The first layer is an Application Layer which runs on the mobile node and invokes the service through the Service Management Layer(SML) giving the service name, service id and the required QoS. The SML administers the service and is responsible for Service Subscription and Service Delivery. The SML runs RASP and uses Docker and Kubernetes to do the migration. The Service Connection Layer monitors the connection between the mobile node and the server and reports to the SML.This is shown in Fig.2.

The first application that will use the framework is to provide storage for mobile applications using the FUSE system. However, the new system will divide the FUSE system in two parts. The file system which runs on the mobile node and the storage system which will be implemented on network memory service.Hence, as the mobile node moves around its data is moved to the nearest network memory server to maintain low latency and provide a good QoS.

6.Conclusions and Future work

Based on the protocol presented, which was verified using Proverif, VANET systems would be the best option on which to implement the Resource Allocation Security Protocol. Hence, the protocol will be tested on a VANET test bed developed by Middlesex University on its Hendon Campus.

Bibliography

1. Gayathri Karthick, Glenford E Mapp, Florian Kammueller, and Mahdi Aiash. Exploring a security protocol for secure service migration in commercial cloud environments. 2017.
2. Gayathri Karthick, Glenford Mapp, Florian Kammueller, and Mahdi Aiash. Formalization and analysis of a resource allocation security protocol for secure service migration. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 207–212. IEEE, 2018.
3. K Kammueller, Glenford Mapp, Sandip Patel, and S Abubaker. Engineering security protocols with model checking-radius-sha256 and secured simple protocol. In *Proceedings of the 7th International Conference on Internet Monitoring and Protection*, 2012.
4. Fragkiskos Sardis, Glenford Mapp, Jonathan Loo, Mahdi Aiash, and Alexey Vinel. On the investigation of cloud-based mobile media environments with service-populating and qos-aware mechanisms. *IEEE transactions on multimedia*, 15(4):769–777, 2013.

$\mathbf{UBiSKt}$-Prolog: an automated theorem prover for a bi-intuitionistic modal logic with universal modalities

Giulia Sindoni[1] Brandon Bennett[2]

[1] School of Computing, University of Leeds, Leeds, LS2 9JT, UK scgsi@leeds.ac.uk
[2] School of Computing, University of Leeds, Leeds, LS2 9JT, UK B.Bennett@leeds.ac.uk

Abstract: We present a tableau-style automated theorem-prover for the bi-intuitionistic modal logic with universal modalities, $\mathbf{UBiSKt}$. The prover has been implemented using Prolog.

1 Introduction

We present an automated tableau-style theorem-prover for the logic $\mathbf{UBiSKt}$, implemented within Prolog. $\mathbf{UBiSKt}$ is a bi-intuitionistic modal logic, which extends the logic $\mathbf{BiSKt}$ [4] with universal modalities [3, 2]. The semantics of $\mathbf{UBiSKt}$ is based on a pre-ordered set (U, H). Four modalities $\Box$, $\blacklozenge$, $\blacksquare$, $\Diamond$ are interpreted with respect to an additional relation $R \subseteq U \times U$. The universal modalities A ("everywhere") and E ("somewhere"), are interpreted with respect to the universal relation $U \times U$. Under a certain interpretation, (U, H) can be seen as an undirected graph, or, more generally, as a hypergraph, and formulae in the logic can be regarded as referring to subgraphs. $\mathbf{UBiSKt}$ has been considered in the context of qualitative spatial reasoning over discrete space. Indeed, a variety of spatial relations between subgraphs can be defined within $\mathbf{UBiSKt}$, such as external connection between two subgraphs, or a subgraph being a core part or a peripheral part of a second one [3, 2]. Given the connection between this logic and the discipline of mathematical morphology, these spatial relations are considered with respect to change in level of detail [2]. Various extensions of $\mathbf{UBiSKt}$ can be obtained, as $S4$-$\mathbf{UBiSKt}$, where R is a pre-order, and $S5$-$\mathbf{UBiSKt}$ that can be seen as a logic for graphs and hypergraphs partitions, following the approach from [1].

2 The tableau calculus and its implementation

$\mathsf{Tab}\mathbf{UBiSKt}$ is a signed tableau calculus with internalised Kripke semantics, extending $\mathsf{Tab}\mathbf{BiSKt}$ [4] with rules for universal modalities. Expressions in the calculus can be of the form $w : S\varphi$, $\bot$, wHv, wRv, where S denotes a sign, either T for true or F for false, and w, v are labels from a fixed denumerable set Label in the tableau language, whose intended meanings are elements of U. A pair of rules is associated with each logical connective and operator, one handling truth and the other handling falsity. As any tableau-style calculus, $\mathsf{Tab}\mathbf{UBiSKt}$ is a refutation procedure. If we want to know whether a formula φ is a theorem, then the input to the calculus is the signed labelled formula $w : F(\varphi)$. If a model can be build from this then φ is not a theorem in $\mathbf{UBiSKt}$. Otherwise a closed tableau will be built, meaning that there is no model of the initial assumption $w : F(\varphi)$.

Rules are of five different kinds: closure rules, deriving contradiction and closing a branch; non-creating and non-branching rules, adding a new leaf to the branch and adding truth (or falsity) of a formula at a world; branching rules, splitting the branch in two different branches; creating rules, adding leaves with new labels, names for worlds that are new in the tree; and relational rules, describing the properties of H and R. Table 1 provides with some examples of tableau-rules in $\mathsf{Tab}\mathbf{UBiSKt}$ and the corresponding implementation in Prolog.

The proof procedure is implemented by a recursive algorithm. Rules will be applied until no "active" formula with a logical connective or operator remains. For some of the rules, we adopt a non-destructive-tableau approach: once a formula (or group of formulae) that matches the premise of a rule has been analysed by that rule and the corresponding conclusion has been added to the branch, the formula-premise is kept in the branch. However the formula will not be analysed again by the same rule, as the conclusion is already present it the branch. A non-destructive approach is preferable for the rules handling truth of a box, and falsity of a diamond. For example, consider the rule handling truth of the universal box A. It needs to be applied every time a new label for a new world is added to the tree. The fact the rule will be blocked if the conclusion of the rule is already present in the branch, will stop the program from applying the rule over again, as this might cause the program to loop.

3 Conclusion

Our initial work has indicated many possibilities for enhancing tableau-based reasoning in this kind of modal calculus, by constraining the ordering of rule applications and by special handling of formulae relating to the relational structure of possible worlds. A further challenge is to automate a theorem-prover for the $S5$ extension of $\mathbf{UBiSKt}$, where graphs and hypergraphs partitions can be represented. We are also interested in comparing our Prolog implementation with an alternative implementation of $\mathsf{Tab}\mathbf{UBiSKt}$ in MetTel [5], which we are also working on.

$\dfrac{w : T(\varphi), \quad w : F\varphi}{\bot}\ (\bot)$	`refute(Formulae, [tf_close]) :-`
	`  select(W:(Phi=t), Formulae, Rest),`
	`  member(W:(Phi=f),  Rest ), !.`

$\dfrac{w : T(\mathsf{A}\,\varphi), \quad v : S\psi}{v : T\varphi}\ (T\,\mathsf{A})$	`refute(Formulae, [t_ubox| Rules]) :-`
	`  select( _W:(ubox(Phi)=t),`
	`  Formulae, Rest),`
	`  member(V:(_), Formulae),`
	`  \+(member(V:(Phi=t), Rest)),!,`
	`  refute([V:(Phi=t) | Formulae],`
	`  Rules).`

v fresh on the branch $\dfrac{w : F(\neg\varphi)}{H(w,v) \quad v : T\varphi}\ (F\neg)$	`refute(Formulae, [f_nneg | Rules]) :-`
	`  select(W:(nneg(Phi)=f),`
	`  Formulae, Rest),`
	`  !,`
	`  V= @(nneg(Phi),S),`
	`  refute( [h(W,V), h(V,V),`
	`  V:(Phi=t) | Rest],`
	`  Rules ).`

Table 1: The top row shows the branch closing rule used to derive contradiction. The middle row gives the rule handling the truth of the universal box, A. The bottom row gives the rule handling the falsity of the intuitionistic negation $\neg$.

References

[1] T. Shaheen and J. G. Stell. Graphical partitions and graphical relations. *Fundamenta Informaticae*, 165(1):75–98, 2019.

[2] G Sindoni, K. Sano, and J. G. Stell. Axiomatizing discrete spatial relations. In *International Conference on Relational and Algebraic Methods in Computer Science*, pages 113–130. Springer, 2018.

[3] G. Sindoni and J. G Stell. The logic of discrete qualitative relations. In *COSIT'17 proceedings*, volume 86, pages 1:1–1:15. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2017.

[4] J.G. Stell, R.A. Schmidt, and D. Rydeheard. A bi-intuitionistic modal logic: Foundations and automation. *J. Logical and Algebraic Methods in Programming*, 85(4):500–519, 2016.

[5] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. Mettel2: Towards a tableau prover generation platform. In *PAAR@ IJCAR*, pages 149–162, 2012.

Using Contexts in Tableaux for PLTL: An illustrative Example

Alex Abuin[1] Alexander Bolotov[2] Unai Diaz de Cerio[1] Montserrat Hermo[3] Paqui Lucio[3]

[1] Ikerlan Technology Research Centre, Mondragon, Spain {aabuin,udiazcerio}@ikerlan.es
[2] University of Westminster, London, UK. A.Bolotov@westminster.ac.uk
[3] University of the Basque Country, San Sebastián, Spain.
{montserrat.hermo,paqui.lucio}@ehu.es

Abstract: The use of contexts in the one-pass tableau eliminates the required test that two-pass tableau make in order to check whether all eventualities are fulfilled or not. In this paper, we provide a succinct description of both: two-pass and context-based (one-pass) tableau, and illustrate the role that contexts play via an example.

1 Introduction

Various tableaux techniques have been proposed for *Propositional Linear-Time Temporal Logic* (PLTL) (see [3]).
Formulae of PLTL are constructed over a set of propositional symbols $Prop$, the constants $\mathbf{T}$ and $\mathbf{F}$, by using the classical connectives ($\neg, \wedge, \vee$) and the temporal operators: $\circ$, $\Box$, $\Diamond$, $\mathcal{U}$ and $\mathcal{R}$. Formulae of the type $\Diamond\varphi$ and $\varphi\mathcal{U}\psi$ are called *eventualities*, and formulae of the type $\Box\varphi$ are called *always-formulae*. Semantic structures are $\mathcal{M} = s_0, s_1, s_2, s_3, \ldots$, i.e. discrete, linear sequence of states, isomorphic to natural numbers. Each state, $s_i, 0 \leq i$, is the set of atoms that are true at the i-th moment of time. The relation $\models$, where $\langle \mathcal{M}, i\rangle \models \varphi$ means that φ is true in the model $\mathcal{M}$ at the (state) index $i \in \mathbb{N}$, is inductively defined in the usual way for classical constants, atoms and connectives. For the two basic temporal connectives: (1) $\langle \mathcal{M}, i\rangle \models \circ\varphi$ iff $\langle \mathcal{M}, i+1\rangle \models \varphi$ and (2) $\langle \mathcal{M}, i\rangle \models \varphi\mathcal{U}\psi$ iff there exists $j \geq i$ such that $\langle \mathcal{M}, j\rangle \models \psi$ and $\langle \mathcal{M}, k\rangle \models \varphi$ for every k, $i \leq k < j$. The semantic of temporal connectives $\Diamond$,$\mathcal{R}$ and $\Box$ is derived based on the abbreviations: $\Diamond\varphi \equiv \mathbf{T}\mathcal{U}\varphi$, $\Box\varphi \equiv \neg\Diamond\neg\varphi$ and $\varphi\mathcal{R}\psi \equiv \neg((\neg\varphi)\mathcal{U}(\neg\psi))$. If, for a formula φ, there exists a model, M, such that $\langle M, 0\rangle \models \varphi$ then φ is satisfiable.
The core idea of tableaux methods for PLTL is to generate all possible pre-models of the input set of formulae and to check whether eventualities are fulfilled on them. Two-pass tableaux techniques, based on [4], require two phases to perform this test. In the first phase, a graph of states, which represents all possible pre-models, is constructed and loaded in memory. In the second phase, for each state s that contains some eventuality $\varphi\mathcal{U}\psi$, a graph-theoretic algorithm should look for a state, reachable from s, that satisfies ψ. Two-pass tableaux methods fail to maintain the classical correspondence between tableaux and sequents that associates a sequent proof with the closed tableau. To avoid the second phase and, hence, to keep the ability of generating (sequent) proofs from tableaux, in [1, 2], dual systems of tableaux and sequents were presented. The tableau system in [2] is proved to be sound, refutationally complete, and terminating.

2 Two-pass and Context-based Tableaux

The one-pass tableau method [2] adds to the standard tableau rules (that are used in the first pass of the two-pass approach [4]) the so called β^+-rules. These use the context to expand a branch where the fulfilment of the selected eventuality is delayed. The standard tableau rules can be written as decomposition rules that map a set of formulae into a set of sets of formulae. Some examples are

$(\vee)$ $\Phi \cup \{\varphi \vee \psi\} \longrightarrow \{\Phi' \cup \{\varphi\}, \Phi' \cup \{\psi\}\}$

$(\Box)$ $\Phi \cup \{\Box\varphi\} \longrightarrow \{\Phi' \cup \{\varphi, \circ\Box\varphi\}\}$

$(\mathcal{U})$ $\Phi \cup \{\varphi\mathcal{U}\psi\} \longrightarrow \{\Phi' \cup \{\psi\}, \Phi' \cup \{\varphi, \circ(\varphi\mathcal{U}\psi)\}\}$

where the Φ's are, depending on the approach, equal to Φ or $\Phi \cup \{\chi\}$ where χ is the decomposed formula in each case. In [4] these rules are applied to the set of formulae, starting with the initial set of sets $\{\Phi_0\}$, where Φ_0 is the set of formulae to test for satisfiability. The decomposed formulae are marked, but not discharged, because they are needed for the fulfilment test (hence, in rules, the decomposed formula is kept in Φ'). The process, for each set Φ, stops when either Φ contains a contradiction (i.e. $\mathbf{F}$ or $\{\phi, \neg\phi\}$ for some ϕ), or Φ or every non-marked formula is *elementary* i.e. of type $\circ\varphi$ or literals (atoms or negated atoms). Contradictory sets do not have successors. Sets of elementary formulae are called pre-states, to which the rule $(\circ)$ $\Phi \longrightarrow \{\varphi \mid \circ\varphi \in \Phi\}$ is applied for jumping to the next state, hence producing the successors of the initial state. In [2] the same process is applied except for discharging the decomposed formula (hence $\Phi' = \Phi$ in the above rules, and for using the additional $(\mathcal{U})^+$ rule:
$\Delta \cup \{\varphi\mathcal{U}\psi\} \longrightarrow \{\Delta \cup \{\psi\}, \Delta \cup \{\varphi, \circ((\varphi \wedge \neg\Delta')\mathcal{U}\psi)\}\}$
where Δ is the so-called *context* and Δ' is the conjunction of all elements of Δ except the always-formulae. The context of an eventuality is simply the set of formulae that 'accompanies' the eventuality in the label of the state, and the rule use it to force its fulfilment. When $(\mathcal{U})^+$ is applied to a node labelled by a set of formulae $\Delta \cup \{\varphi\mathcal{U}\psi\}$, then the context is Δ. Therefore, if ψ is not satisfied, then $\neg\Delta'$ also belongs to the labels of that state. This means that the context, Δ, of the previous label is not repeated. As Δ' is a

finite set/conjunction of formulae and $\neg\Delta'$ is the finite disjunction of the negations of the formulae in Δ', the $(\mathcal{U})^+$ rule forces at least one formula in Δ to be falsified during the transition from the previous state to the subsequent one (whenever ψ is not satisfied). The $(\mathcal{U})^+$ rule allows us to avoid the construction of an auxiliary graph that is used (in the second pass in two-pass tableau) to determine whether all eventualities are satisfied or not.

3 The Role of Contexts

In the construction of all possible models of a (set of) PLTL formula(e) –which could be either a graph (as in [4]) or a tree-shaped structure (as in [2])– there are two crucial operations: the first is to generate all the successors (or next-states) of a given state; and the second is to check the fulfilment of eventualities. In this section we illustrate, through a representative example, how contexts are used in the one-pass tableau for (1) guiding the generation of the successors and (2) replacing the fulfilment test by a fair selection of eventualities. For clarity, and also for comparison, we first briefly explain the graph construction in [4] for an illustrative example. Then, we explain how contexts guide the construction of successors based on the same example. From now on, we call $\mathsf{Succ}(s)$ to the set of all successors of a given state s.

Let us consider some fixed $n > 0$ and the following set A of always-formulae:

$$\{\Box(\neg x_i \vee \bigcirc x_{i+1}) \mid 0 \leq i < n\} \cup$$
$$\{\Box(\neg x_n \vee \bigcirc x_0)\} \cup \{\Box(\neg x_i \vee \bigcirc p) \mid 0 \leq i < n\}$$

Our concern is to test if the set $\Phi = A \cup \{p, x_0, \Diamond\neg p\}$ is satisfiable. We use X to denote the set $\{x_0, x_1, \ldots, x_n\}$. For that, the construction of the graph ([4]) for Φ starts with the initial state (a node of the graph) $s_0 = \Phi$[1]. Then, the calculation of the set $\mathsf{Succ}(s_0)$ produces one state of the form $s_Z = A \cup Z \cup \{p, \Diamond\neg p\}$[2], for each non-empty $Z \subseteq X$ such that $x_1 \in Z$. Hence, $\mathsf{Succ}(s_0)$ consists of $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n$ sets. Next, for each $s_Z \in \mathsf{Succ}(s_0)$, and for all $0 \leq i < n$, if $x_i \in s_Z$ then x_{i+1} is included in all elements of $\mathsf{Succ}(s_Z)$, and if $x_n \in s_Z$ then x_0 is included in all elements of $\mathsf{Succ}(s_Z)$. Hence, again, $\mathsf{Succ}(s_Z)$ contains exponentially many sets. Many of them are repeatedly generated. When a generated state is already in the graph, a new successor-edge is added to the graph. The exact number of (different) states in the graph is $2^{n+1} - 1$, while the number of successor-edges is $2^{O(n)}$. At the end, the fulfilment test shows that the eventuality $\Diamond\neg p$ (which belongs to every state) is never fulfilled.

The context-based tableau construction starts with initial state $s_0 = \Phi$. Then, $\mathsf{Succ}(s_0)$ consists on one state $s_{Z_1} = A \cup Z_1 \cup \{p, (\neg p \vee \neg x_0)\mathcal{U}\neg p\}$ for each non-empty $Z_1 \subseteq X$ such that $x_1 \in Z_1$. Then, $\mathsf{Succ}(s_{Z_1})$ is empty for each Z_1 such that $x_0 \in Z_1$. Otherwise, if $x_0 \notin Z_1$, then successors of s_{Z_1} are of the form

$$s_{Z_2} = A \cup Z_2 \cup \{p, (\neg p \vee (\neg x_0 \wedge \neg Z_1))\mathcal{U}\neg p\}$$

where $Z_2 \subseteq X$ and $\neg Z_1$ is the disjunction of all $\neg x_j$ such that $x_j \in Z_1$. Since, $x_0 \notin Z_1$, if $x_j \in Z_1$ then $j \neq 0$. Therefore, $\mathsf{Succ}(s_{Z_2})$ is empty for each $Z_2 \subseteq X$ such that $\{x_0, x_j\} \cap Z_2 \neq \emptyset$ for some $j \neq 0$. The value of j depends on the elements in Z_1, and could be different for each state in $\mathsf{Succ}(s_{Z_2})$. Consider any fixed $j \neq 0$ such that $x_j \in Z_1$ for each Z_2 such that $\{x_0, x_j\} \cap Z_2 = \emptyset$, then the successors of s_{Z_2} are of the form

$$s_{Z_3} = A \cup Z_3 \cup \{p, (\neg p \vee (\neg x_0 \wedge \neg Z_1 \wedge \neg Z_2))\mathcal{U}\neg p\}$$

Since $\{x_0, x_j\} \cap Z_2 = \emptyset$, then $\mathsf{Succ}(s_{Z_3})$ is empty for each $Z_3 \subseteq X$ such that $\{x_0, x_j, x_k\} \cap Z_3 \neq \emptyset$ for some $k \neq j$ and $k \neq 0$ Therefore, at each step, half of the successors (in the previous graph) are not generated in the tableau, and the context in the eventuality is increased. We construct a tree whose breadth is reduced in a half at each level. And, what is more important, each branch has at most depth n. Indeed, in at most $n + 1$ steps, the node

$$s_{Z_{n+1}} = A \cup Z_n \cup \{p, (\neg p \vee (\neg x_0 \wedge \neg Z_1 \wedge \cdots \wedge \neg Z_n))\mathcal{U}\neg p\}$$

has the empty set of successors. We construct this tree in depth and prune each branch, where the successor of a state has been previously generated. The tree has at most $2^{O(n)}$ nodes as the number of different contexts coincides with the number of different states. When the tree construction ends no fulfilment test is needed, since the eventuality has been selected in every branch. Since the tableau is closed, Φ is unsatisfiable. SAT-solvers are currently used for calculating successors, preventing repetitions of successors, and the formulae like $\neg p \vee (\neg Z_0 \wedge \cdots \wedge \neg Z_j)$ are included in the set passed to the SAT-solver.

[1] Nodes/states are labelled by set of formulae, we intentionally use the same name for a node and its label.

[2] For clarity, we omit the negative literals $\neg x_j$, and the subformulae of formulae in A

References

[1] Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. A cut-free and invariant-free sequent calculus for PLTL. In Jacques Duparc and Thomas A. Henzinger, editors, Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, volume 4646 of Lecture Notes in Computer Science, pages 481–495. Springer, 2007.

[2] Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. Dual systems of tableaux and sequents for PLTL. The Journal of Logic and Algebraic Programming, 78(8):701–722, 2009.

[3] Rajeev Goré. Tableau methods for modal and temporal logics. In Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors, Handbook of Tableau Methods, pages 297–396. Springer Netherlands, Dordrecht, 1999.

[4] Pierre Wolper. The tableau method for temporal logic: An overview. Logique Et Analyse, 28(110-111):119–136, 1985.

Uniform Interpolation in Modal Logic

Ruba Alassaf Renate Schmidt

The University of Manchester
{ruba.alassaf,renate.schmidt}@manchester.ac.uk

Abstract: Uniform interpolation is a reasoning technique that seeks to compute a weaker theory by restricting its signature. We are interested in uniform interpolation in modal logic which is an under explored area. This paper gives a high-level description of an automated approach to compute uniform interpolation based on Ackermann's Lemma together with modal resolution.

1 Introduction

Uniform interpolation is a nonstandard automated reasoning task that aims to restrict the symbols used to describe a theory whilst preserving logical consequences up to the remaining symbols. More formally, the uniform interpolant of a logical formula ϕ with respect to a signature Σ is a formula ϕ' so that for any formula ψ where ϕ' and ψ only contain symbols from Σ, we have that $\phi \models \psi$ iff $\phi' \models \psi$.

Uniform interpolation is an under explored area in modal logic. Ghilardi constructively proved that the basic modal logic K has the uniform interpolation property [7]. A syntactical approach to compute uniform interpolation for modal logics K and T was presented in [2]. A result that found that modal logic $S5$ has the uniform interpolation property was given in [11]. The modal logics $K45$ and $KD45$ were shown to have the uniform interpolation property in [4] and an algorithm to compute it was given for the modal logics K, D, T, $K45$ and $KD45$. On the other hand, it is known that $S4$ and $K4$ do not have the uniform interpolation property [7].

The main shortcoming with these methods is that they are not designed for application purposes. Herzig and Mengin [8] proposed a practical method that is based on resolution to compute uniform interpolant for modal logic K which is, as far as we know, the only practical method available. Therefore, the focus of this research is *to give practical methods to compute uniform interpolants for the modal logics in which the problem is known to be solvable.* We seek to achieve this by analysing the ideas that were recently proposed for uniform interpolation in description logic (see e.g. [9, 10, 12, 13]), and studying if they could be adapted for modal logic. Although the impact of our research is expected to be mainly theoretical, we recognise that our work may indirectly impact other more applied research areas. In the following, we discuss how this could be possible.

Modal logic is widely used in agent systems to model and formalise the cognitive behaviours of an agent such as knowledge and belief. Agents become more interesting when they are integrated into a multi-agent system. The goal of multi-agent systems is to build agents that are able to cooperatively interact with a community of other agents in order to achieve a certain goal, such as sharing knowledge.

In game applications, it is sometimes required that agents communicate and share some of their knowledge. An agent that aims to win the game is more likely to be interested in sharing information regarding certain variables without giving any information about other variables, i.e., without giving other agents more information than required. This can be achieved by applying uniform interpolation to restrict the signature in order to compute a weaker view of the agents knowledge that does not expose any possibly secret information.

In most agent-based applications, it is often assumed that agents communicate using the same language. Uniform interpolation becomes very useful when this assumption is relaxed. The idea is to use uniform interpolation to allow an agent to express knowledge about a certain topic by computing a view that only uses some terms. By this method, an agent is equipped to deliver their knowledge in a way that can be understood by other communicating agents. Depending on the expertise of the receiving agent, the sending agent would share its knowledge by applying a uniform interpolation algorithm to the terms that the receiving agent can understand. In this way, the agents will be able to share their knowledge with other agents who specialise in different domains.

2 Related Work

Uniform interpolation has been studied in various other logics such as first-order logic [3, 5, 6] and description logic [9, 10, 12, 13]. The problem is investigated under other names such as second-order quantifier elimination [5, 6] since uniform interpolation can be viewed as the problem of eliminating second-order existential quantifiers [1]. It is also studied as forgetting [10, 12, 13] which is the dual notion of uniform interpolation.

Generally, the known practical approaches to compute uniform interpolation fall within two categories:

- Resolution-based approaches.
  The idea behind these approaches is to exhaustively generate consequences and subsequently eliminate the ones that contain symbols outside the restricted signature. Examples of methods that are based on this approach include [5, 6, 8, 9].

- Ackermann-based approaches.
  These methods rely on a monotonicity theorem due

to Ackermann [1]. The idea is to eliminate the symbols outside the restricted signature by attempting to compute a definition for a selected symbol and subsequently replacing the symbol with its computed definition. This idea was used in methods such as [3, 12].

3 Current Work

Motivated by the recent advancements for uniform interpolation in description logic and its relationship to modal logic, we are currently studying these approaches and ideas in order to see if they are applicable to the solvable normal modal systems. Generally, it was noticeable that approaches that use Ackermann's Lemma had better performance in practice but in some cases, a few symbols outside the signature Σ remained in the output. On the other hand, approaches that were based on resolution computed uniform interpolants, sometimes with definer symbols, but did not always terminate within an acceptable time-frame. Hence, we believe that it may be beneficial to combine the two approaches into one method; since in this way, the procedure benefits from the good performance of Ackermann-based approaches and the high success rate of resolution-based approaches.

The general idea behind our approach to compute a uniform interpolant is outlined by the following steps. Let ϕ be an logical formula and Σ be a signature.

1. Normalise ϕ into a set of clauses N.
2. Apply Ackermann's lemma to eliminate the symbols that do not appear in Σ iteratively until it could not be applied anymore. Let the result be N_{ack}.
3. If symbols outside Σ still remain, proceed to the next step; otherwise return the uniform interpolant $\phi'(= N_{ack})$.
4. Apply the modal resolution rules to N_{ack} until it can not be applied anymore. Let the result be N_{res}.
5. Eliminate the clauses that contain symbols outside Σ from N_{res} to produce the uniform interpolant ϕ'.

Furthermore, it would be interesting to study the possibility of incorporating orderings and selection functions from automated reasoning to reduce the computational complexity space, and potentially guarantee termination.

References

[1] W. Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110:390–413, 1935.

[2] Marta Blkov. Uniform interpolation and propositional quantifiers in modal logics. *Studia Logica: An International Journal for Symbolic Logic*, 85(1):1–31, 2007.

[3] Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336, 1997.

[4] Liangda Fang, Yongmei Liu, and Hans Van Ditmarsch. Forgetting in multi-agent modal logics. In *Proc. IJCAI 2016*, pages 1066–1073. IJCAI/AAAI Press, 2016.

[5] D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, 2008.

[6] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, KR'92, pages 425–435, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[7] Silvio Ghilardi and Marek Zawadowski. Undefinability of propositional quantifiers in the modal system s4. *Studia Logica*, 55(2):259–271, Jun 1995.

[8] Andreas Herzig and Jérôme Mengin. Uniform interpolation by resolution in modal logic. In *JELIA*, 2008.

[9] P. Koopmann. *Practical Uniform Interpolation for Expressive Description Logics*. PhD thesis, University of Manchester, 2015.

[10] P. Koopmann and R. A. Schmidt. Implementation and evaluation of forgetting in $\mathcal{ALC}$-ontologies. In *Proc. WoMO 2013*. CEUR-WS.org, 2013.

[11] Frank Wolter. Fusions of modal logics revisited. In *In Advances in modal logic*, pages 361–379. CSLI, 1998.

[12] Y. Zhao and R. A. Schmidt. Concept forgetting in $\mathcal{ALCOI}$-ontologies using an Ackermann approach. In *Proc. ISWC 2015*, volume 9366 of *Lecture Notes in Computer Science*, pages 587–602. Springer, 2015.

[13] Yizheng Zhao and Renate A. Schmidt. Fame: An automated tool for semantic forgetting in expressive description logics. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning*, pages 19–27, Cham, 2018. Springer International Publishing.

Experimenting with superposition in iProver

André Duarte Konstantin Korovin

University of Manchester, {andre.duarte,konstantin.korovin}@manchester.ac.uk

Abstract: In this work we extend iProver with support for the superposition calculus. Then, we develop a flexible simplification setup that subsumes and generalises common architectures such as Discount or Otter. This also includes the concept of "immediate simplification", wherein newly derived clauses are more aggressively simplified among themselves, which can make the given clause redundant and thus its children discarded.

1 Introduction

iProver [1] is an automated theorem prover for first-order logic. It implements primarily the Inst-Gen calculus, but it also implements resolution and supports running them in combination. In this work we extend iProver with support for the superposition calculus.

Superposition is a set of inference rules that is complete for first-order logic with equality predicates only (and therefore for all first-order logic via an embedding in the former fragment). We do not present it here (see e.g. [2]).

The calculus is performed in a conventional given clause loop [3]. In iProver, it can either be run standalone, or in combination with the main instantiation calculus. In the latter mode, superposition is run simultaneously with instantiation to generate clauses for simplifications in the instantiation loop (but not to perform instantiation inferences).

2 Simplifications

Apart from the generating inferences, necessary for completeness, we can add *simplification inferences*. These are inferences where some or all of the premises are deleted. They are not required for completeness but are crucial for performance. In this work, we use the following rules (where a ~~crossed out~~ premise indicates that it can be deleted after adding the conclusion):

$$\text{Tautology deletion} \quad \frac{\cancel{l \vee \bar{l} \vee C} \qquad \cancel{t = t \vee C}}{} \tag{1}$$

$$\text{Syntactic eq. res.} \quad \frac{\cancel{t \neq t \vee C}}{} \tag{2}$$

$$\text{Subsumption} \quad \frac{\cancel{C\theta \vee D} \quad C}{} \tag{3}$$

$$\text{Subset subsumption} \quad \frac{\cancel{C \vee D} \quad C}{} \tag{4}$$

$$\text{Subsumption res.} \quad \frac{p \vee C \quad \cancel{\bar{q} \vee D}}{D} \tag{5}$$

where there exists θ such that $(p \vee C)\theta \subseteq q \vee D$.

$$\text{Demodulation} \quad \frac{l = r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]} \tag{6}$$

where $l\theta \succ r\theta$ and $\{l\theta = r\theta\} \prec C[l\theta \mapsto r\theta]$.

Light normalisation In addition, we introduce the following rule:

$$\text{Light normalisation} \quad \frac{l = r \quad \cancel{C[l]}}{C[l \mapsto r]} \tag{7}$$

which is a special case of the demodulation rule. It's advantageous to formulate this separately because it may be implemented much more efficiently than demodulation (simple replacement, no instantiation), and as such we may want e.g. to apply light normalisation wrt. all clauses but demodulation only wrt. active clauses

Simplification scheduling How these simplifications are performed can greatly impact the performance of the solver, so care is needed, and tuning this part of the solver can pay off significantly. We can choose to perform some simplifications at different times, or not at all. Additionally, some of these simplifications require auxiliary data structures (here referred to generally as 'indices') to be done efficiently, and some indices support several rules. Therefore we also need to choose which clauses to add to each indices at which stages.

For example, Otter-style loops [3] perform simplifications on clauses before adding them to the passive set. The problem with that is that the passive set is often orders of magnitude larger than the active set, therefore performance will degrade significantly as this set grows, and the system will spend most of its time performing simplifications on clauses that may not even end up being used. On the other hand, Discount-style loops [4] perform simplifications only with clauses that have been added to the active set. This has the benefit of reducing the time spent in simplifications, at the cost of potentially missing many valuable simplifications wrt. passive clauses. It is not clear where the "sweet spot" is, in terms of these setups, so we want a flexible configuration to experiment with and compare different approaches.

Immediate simplification In addition, we also introduce the idea of "immediate simplification". The intuition is as follows. Clauses that are derived in each loop are "related" to each other. It may be beneficial to keep the set of immediate conclusions inter-simplified. Also,

throughout the execution of the program the set of generated clauses in each loop remains small compared to the set of passive or active clauses. Therefore, we can get away with applying more expensive rules that we don't necessarily want to apply on the set of all clauses (e.g. only "light" simplifications between newly derived clauses and passive clauses, but more expensive "full" simplifications among newly derived clauses). Finally, during this process, it is possible that the given clause itself becomes redundant (e.g. subsumed by one of its children). If this happens, we can add the responsible clauses to the passive set, remove the given clause from the set, and then throw away this iteration's newly generated clauses and abort the iteration and proceed to the next given clause. This may speed things up if many iterations are thus aborted.

Also, we may want to apply a distinct set of (more expensive) simplifications among the input clauses. We also take this into consideration.

Simplification setup We propose a general and flexible framework to specify how these simplifications are performed. This lets us experiment with and evaluate many different configurations. In pseudocode:

```
input_set = ∅
for i in input_clauses:
  simplify(i wrt input_set via input)
  add(i to indices_input)
for i in input_set:
  add(i to indices_passive)

main_set = ∅
loop:
  immed_set = ∅
  given = take(clause from passive)
  simplify(given wrt main_set via
      rules_active)
  add(given to indices_active)
  for i in all generating inferences between
      given and active:
    simplify(i wrt immed_set via rules_immed)
    if given was eliminated in immed_set
        by clauses:
      add(clauses to indices_passive)
      goto loop
    simplify(i wrt main_set via rules_passive)
    add(i to indices_immed)
  for i in immed_set:
    add(i to indices_passive)
```

where `add(clause to indices)` adds a `clause` to some simplification `indices`, and `simplify(clause wrt set via rules)` simplifies a `clause`, via a set of `rules`, by some clause(s) in `set`.

This general scheme gives great flexibility for the user to specify which simplifications are done at which stages. Namely, we can specify: to which indices are clauses added after generation (`indices_passive`), after adding to passive (`indices_active`), during immediate simplification (`indices_immed`), during input preprocessing (`indices_input`); also which simplifications are done before activation (`rules_active`), after generatuion, wrt. the main set (`rules_passive`), and wrt. the immediate set (`rules_immed`), and among input clauses (`rules_input`).

An Otter loop would be

$$\texttt{indices\_passive} = \text{all} \qquad \texttt{rules\_passive} = \emptyset$$
$$\texttt{indices\_active} = \emptyset \qquad \texttt{rules\_active} = \text{all}$$

while a Discount loop would be

$$\texttt{indices\_passive} = \emptyset \qquad \texttt{rules\_passive} = \emptyset$$
$$\texttt{indices\_active} = \text{all} \qquad \texttt{rules\_active} = \text{all}$$

with the rest $=\emptyset$. In our experiments we will test several distinct setups.

Simultaneous superposition Another improvement is the usage of "simultaneous superposition" [5]. Recall the superposition rule:

$$\frac{l = r \vee C \quad t[s] \doteq u \vee D}{(t[s \mapsto r] = u \vee C \vee D)\theta} \tag{8}$$

where $\theta = \text{mgu}(l, s)$, $l\theta \not\preceq r\theta$, $t\theta \not\preceq u\theta$, and s is not a variable. The conventional rule is that by $t[s]$ and $t[s \mapsto r]$ we mean resp. "a distinguished occurrence of s as a subterm of t" and "replacing that subterm at that position by r. We call the variant *simultaneous superposition* where we mean instead "replacing all occurrences of s in t by r". This variant is still refutationally complete.

3 Results

We integrated the simultaneous superposition calculus into iProver and evaluated it over 15 168 first-order problems in TPTP-v7.2.0. The superposition loop can solve 7375 (49%), the instantiation loop (on the previous version of iProver) 7884 (52%), and their combination can solve 8708 (57%). Therefore we can conclude that combination with superposition improved the performance of iProver over the whole TPTP.

References

[1] K. Korovin, "Inst-Gen — A Modular Approach to Instantiation-Based Automated Reasoning," in *Programming Logics* (A. Voronkov and C. Weidenbach, eds.), vol. 7797, pp. 239–270, Springer Berlin Heidelberg.

[2] J. A. Robinson, *Handbook of automated reasoning*. Elsevier MIT Press, 2001.

[3] W. McCune, "OTTER 3.3 reference manual," *CoRR*, vol. cs.SC/0310056, 2003.

[4] J. Denzinger, M. Kronenburg, and S. Schulz, "DISCOUNT — A distributed and learning equational prover," *Journal of Automated Reasoning*, vol. 18, pp. 189–198, Apr 1997.

[5] D. Benanav, "Simultaneous paramodulation," in *10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, pp. 442–455, 1990.