Proceedings of the Automated Reasoning Workshop 2017 ARW'17

3rd–4th April 2017
University of Bristol
Bristol, United Kingdom

Editor:
Oliver Ray

Department of Computer Science
University of Bristol

Preface

This volume contains the proceedings of ARW 2017, the twenty-fourth Workshop on Automated Reasoning, held on the 3rd–4th April 2017, in Bristol, UK. In common with previous events in this series, the purpose of the workshop was to provide an informal forum for the automated reasoning community to discuss recent work, new ideas and applications, and current trends. Its aim was to bring together researchers from all areas of automated reasoning in order to foster links among researchers from various disciplines; among theoreticians, implementers and users alike.

This year the workshop had a special focus on eXplainable Artificial Intelligence (XAI): a rapidly developing area that aims to re-establish an awareness of the importance of comprehensibility and communicability with respect to the outputs of learning and reasoning systems. We believe this is much a neglected topic to which logic-based approaches have the potential to make a great impact.

These proceedings contain the abstracts of two invited talks, by Maria Paola Bonacina (Università degli Studi di Verona), on 'Conflict-driven reasoning', and by Nello Cristianini (University of Bristol), on 'Why did the chicken cross the road? – Explanations and intelligent behaviour'.

The proceedings also contain twelve extended abstracts, contributed by the participants of the workshop, which cover a wide range of topics including Natural Deduction and Abstract Argumentation (Session 1), Logic-based Machine Learning (Session 2), Description Logics and Knowledge Bases (Session 3) and Temporal Logics and Diagrammatic Inference (Session 4).

I would like to thank the members of the ARW Organising Committee for their advice and my colleagues Helen Cooke and Kacper Sokol for helping with the local organisation.

Oliver Ray
Bristol, April 2017

Organising committee

Workshop Website

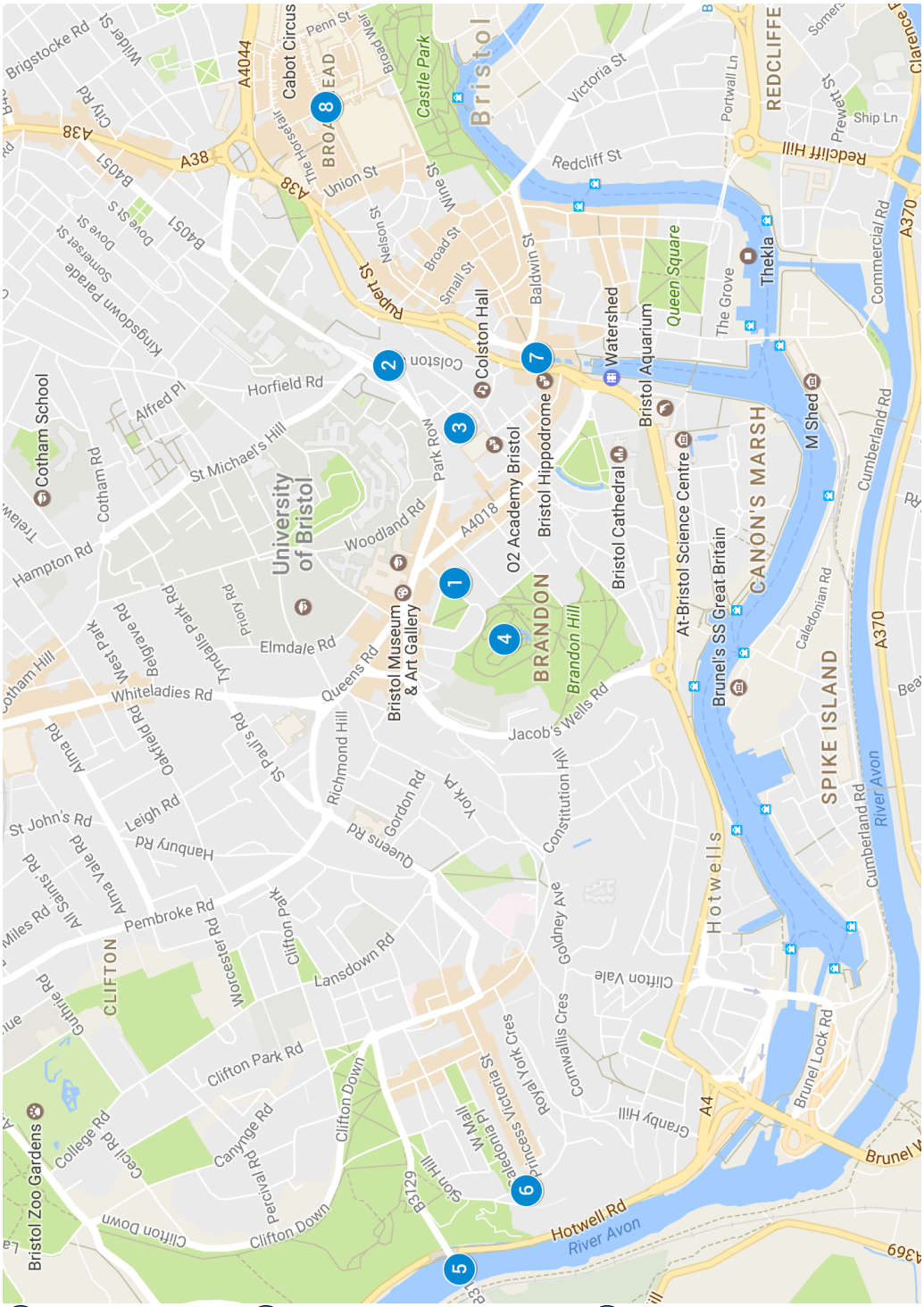
`https://www.cs.bris.ac.uk/~oray/ARW17/`

Programme

Monday, 3rd April	
09:00	Registration
10:00	Invited Talk: Conflict-Driven Reasoning by *Maria Paola Bonacina*
11:00	Coffee
11:30	Session 1 • On the Complexity of the Natural Deduction Proof Search Algorithm by *Daniil Kozhemiachenko, Alexander Bolotov, Vasilyi Shangin* • ABAplus: Implementing Attack Reversal in Structured Argumentation with Preferences by *Ziyi Bao, Kristijonas Čyras, Francesca Toni* • AA-CBR: Explaining Case-Based Reasoning via Argumentation by *Kristijonas Čyras, Francesca Toni*
13:00	Lunch
14:00	Session 2 • Automatically Discovering Human-readable Rules to Predict Effective Compiler Settings for Embedded Software by *Craig Blackmore, Oliver Ray, Kerstin Eder* • The role of textualisation and argumentation in understanding the machine learning process: a position paper by *Kacper Sokol, Peter Flach* • An Inductive Logic Programming approach for analysing Cyber-Attacks by *Oliver Ray, Sam Hicks, Steve Moyle*
15:30	Coffee
16:00	Business Meeting
17:00	Free Time
19:00	Dinner: Zerodegrees - Restaurant and Micro-Brewery

Tuesday, 4th April	
09:30	Session 3 • Forgetting Role Symbols in $\mathcal{ALCOQH}(\bigtriangledown)$-Ontologies by *Yizheng Zhao, Renate Schmidt* • Forgetting for Abduction in $\mathcal{ALC}$-Ontologies by *Warren Del-Pinto, Renate Schmidt* • A Framework for Axiom Selection in Large Theories by *Julio Cesar Lopez Hernandez, Konstantin Korovin*
11:00	Coffee
11:30	Session 4 • Mind the Gap: Metric Temporal Logic Translations by *Ullrich Hustadt, Clare Dixon, Ana Ozaki* • Verifying A Security Protocol for Secure Service Migration in Commercial Cloud Environments by *Gayathri Karthick, Florian Kammueller, Glenford Mapp, Mahdi Aiash* • Diagrammatic Reasoning for Ontology Debugging by *Zohreh Shams, Mateja Jamnik, Gem Stapleton, Yuri Sato*
13:00	Lunch
14:00	Invited Talk: Why did the chicken cross the road? - Explanations and intelligent behaviour by *Nello Cristianini*
15:00	Coffee Discussion (eXplainable AI)
16:00	Drinks: Avon Gorge Hotel (White Lion) Bar and Terrace

Local Map


Bristol
REDCLIFFE
CANON'S MARSH
SPIKE ISLAND
BRANDON
CLIFTON
Hotwells
University of Bristol
Colston Hall
Bristol Hippodrome
02 Academy Bristol
Bristol Cathedral
Watershed
Bristol Aquarium
At-Bristol Science Centre
Brunel's SS Great Britain
M Shed
Thekla
Bristol Museum & Art Gallery
Cotham School
Bristol Zoo Gardens
Cabot Circus
Castle Park
Queen Square
Brandon Hill
Clifton Down
Brunel Lock Rd
River Avon
Hotwell Rd
Redcliff Hill
Redcliff St
Victoria St
Union St
Park Row
Woodland Rd
Queens Rd
Jacob's Wells Rd
Whiteladies Rd
Elmdale Rd
Horfield Rd
St Michael's Hill
Cotham Rd
Hampton Rd
Pembroke Rd
Lansdown Rd
Clifton Park Rd
Granby Hill
Cumberland Rd
Commercial Rd
Caledonian Rd
A4
A38
A4044
A4018
A370
B4052
B4057
B3129
1 Workshop Venue
2 Zerodegrees Restaurant
3 Trenchard Street Car Park
4 Cabot Tower
5 Clifton Suspension Bridge
6 Avon Gorge Hotel
7 Harbourside
8 City Centre

Invited Talks

Contributed Abstracts

Conflict-driven reasoning

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
mariapaola.bonacina@univr.it

Abstract: The Conflict-Driven Clause Learning (CDCL) procedure works by guessing truth assignments to atoms, propagating their consequences through clauses, explaining conflicts by resolution, and learning new clauses, when assignments lead to conflicts. The success of CDCL in moving SAT-solving from theoretical hardness to practical success, suggests the challenge of generalizing the concept of conflict-driven reasoning. SGGS (Semantically-Guided Goal-Sensitive reasoning) lifts CDCL to first-order logic. CDSAT (Conflict-Driven Satisfiability) does the same for satisfiability modulo theories, introducing the more general problem of satisfiability modulo assignments. This talk gives an overview of the conflict-driven paradigm and then focuses on key features of CDSAT. (SGGS is joint work with David A. Plaisted; CDSAT is joint work with Stphane Graham-Lengrand and Natarajan Shankar.)

Why did the chicken cross the road? – Explanations and intelligent behaviour

Nello Cristianini

Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
Nello.Cristianini@bristol.ac.uk

Abstract: This talk will informally discuss the elusive concept of explanation in the context of intelligent systems, giving a series of examples. We will focus on what we may consider as a valid explanation, and why in some situations we should expect one. This is part of ongoing work within the ThinkBIG project, that focusses on ethical and epistemological aspects of machine learning and intelligent technologies.

On the Complexity of the Natural Deduction Proof Search Algorithm

D.A. Kozhemiachenko,[1] A.E. Bolotov,[2] V.O. Shangin,[3]

[1] Lomonosov Moscow State University, kodaniil@yandex.ru
[2] University of Westminster, A.Bolotov@wmin.ac.uk
[3] Lomonosov Moscow State University, b.shngn@gmail.com

Abstract: We present our first account of the complexity of natural deduction proof search algorithms. Though we target the complexity for natural deduction for temporal logic, here we only tackle classical case, comparing the classical part of the proof search for temporal logic with the classical analytical tableau.

Rules of Natural Deduction System.

We commence with the review of the classical part of the natural deduction system for temporal logic [3] define below the sets of elimination and introduction rules, and where prefixes '*el*' and '*in*' abbreviate an elimination and an introduction rule, respectively.

Elimination Rules:

$$\wedge el_1 \frac{A \wedge B}{A} \qquad \wedge el_2 \frac{A \wedge B}{B} \qquad \neg el \quad \frac{\neg\neg A}{A}$$

$$\Rightarrow el \quad \frac{A \Rightarrow B, \quad A}{B} \qquad \vee el \quad \frac{A \vee B, \quad \neg A}{B}$$

Introduction Rules:

$$\vee in_1 \frac{A}{A \vee B} \quad \vee in_2 \frac{B}{A \vee B} \qquad \wedge in \frac{A, \quad B}{A \wedge B}$$

$$\Rightarrow in \quad \frac{[C], \quad B}{C \Rightarrow B} \qquad \neg in \quad \frac{[C], \quad B, \quad \neg B}{\neg C}$$

In '$\Rightarrow in$' and '$\neg in$' formula $[C]$ must be the most recent non discarded assumption occurring in the proof. When we apply one of these rules on step n and discard an assumption on step m, we also discard all formulae from m to $n-1$.

Searching Procedures

Searching Procedures update lists of formulae in the proof, list of goals (list_proof, list_goals) or both of them. Let $\bot$ abbreviate a dedicated goal, contradiction, and 'G_{cur}' abbreviate the current goal.

Procedure (1) simplifies structures of formulae in list_proof by an applicable elimination rule. **Procedure (2)** is fired when the current goal is not reached. Here we distinguish two subroutines. Procedure (2.1) applies when the current goal is not reached. Analysing the structure of the current goal we update list_proof and list_goals, respectively, by new goals or new assumptions. Subroutines (2.1.1)–(2.1.9) guide this process. The rules below have structure $\Gamma \Vdash \alpha \longrightarrow \Gamma' \Vdash \alpha'$ indicating that the rule modifies some given inference task $\Gamma \Vdash \alpha$ to a new inference task $\longrightarrow \Gamma' \Vdash \alpha'$.

(2.1.1) $\Gamma \vdash \Delta, A \longrightarrow \Gamma, \neg A \vdash \Delta, A, \bot$
(2.1.2) $\Gamma \vdash \Delta, \neg A \longrightarrow \Gamma, A \vdash \Delta, \neg A, \bot$
(2.1.3) $\Gamma \vdash \Delta, A \wedge B \longrightarrow \Gamma \vdash \Delta, A \wedge B, B, A$
(2.1.4.1) $\Gamma \vdash \Delta, A \vee B \longrightarrow \Gamma \vdash \Delta, A \vee B, A$
(2.1.4.2) $\Gamma \vdash \Delta, A \vee B \longrightarrow \Gamma \vdash \Delta, A \vee B, B$
(2.1.5) $\Gamma \vdash \Delta, A \Rightarrow B \longrightarrow \Gamma, A \vdash \Delta, A \Rightarrow B, B$

If applying Procedure (2.1.4) we could not reach goals $A,\ B$ then we delete these goals, leaving the current goal, $A \vee B$.

Procedure 2.2 is invoked when $G_n = \bot$. It searches for formulae in list_proof as sources for new goals. We abbreviate these designated formulae as Ψ. The idea behind this procedure is to search for "missing" premises to apply a relevant elimination rule to Ψ.

(2.2.1) $\Gamma, \neg A \vdash \Delta, \bot \longrightarrow \Gamma, \neg A \vdash \Delta, \bot, A$
(2.2.2) $\Gamma, A \vee B \vdash \Delta, \bot \longrightarrow \Gamma, A \vee B \vdash \Delta, \bot, \neg A$
(2.2.3) $\Gamma, A \Rightarrow B \vdash \Delta, \bot \longrightarrow \Gamma, A \Rightarrow B \vdash \Delta, \bot, A$

Applying the Procedure (2.2.1) we have $\neg A$ in the proof and are aiming to derive, A itself. If we are successful then this would give us a contradiction.

When we apply Procedures (2.2.2-2.2.3), our target is to derive formulae that being in the proof would enable us to apply a relevant elimination rule, $\vee_{el}, \Rightarrow_{el}$.

Procedure 3 checks reachability of the current goal in list_goals. If $Reached(G_n) =$ **true** then list_goals= list_goals- G_n and $G_{cur} = G_{n-1}$.

Procedure 4 guides the application of introduction rules. Any application of the introduction rule is completely determined by the current goal in list_goals. This property of our proof searching technique protects us from inferring by introduction rules an infinite number of formulae in list_proof.

Proof-Searching Algorithm [3]

Given a task $\vdash G$, we commence the algorithm by setting the initial goal, $G_0 = G$. Then for any goal G_{cur}, we apply Procedure 3, to check if it is reached. If G_i is not reached we apply Procedure 1. If G_{cur} is still not reached, then Procedure 2 is invoked which updates list_proof and list_goals dependent on the structure of G_{cur}. If G_{cur} is reached, then Procedure 4 is applied. Otherwise, which could only be in the case, when *current_goal* is set as $\bot$ and we do not have contradictory formulae in list_proof, we update list_goals looking for possible sources of new goals in list_proof. Continuing searching we may reach the initial goal, G_0, in which case we terminate having found the desired proof. Otherwise, we reach the stage when our search cannot update list_proofand list_goals any further. In the latter case we terminate, and no proof has been found and a counterexample can be extracted.

Marking technique introduces and eliminates special marks for formulae in list_proof and list_goals. Most of

these marks are devoted to prevent looping either in application of elimination rules or in searching. In particular, we mark: formulae that were used as premisses of the rules invoked in Procedure 1; goals $A \vee B$ in Procedure (2.1.4); those formulae in list_proof which were considered as sources of new goals in Procedure 2.2 and these new goals themselves to prevent looping in Procedure (2.1.1).

Let 'last(list_goals)' return the last element of list_goals, and list_goals — G_n deletes the last formula, G_n, from list_goals.

Now, based on the procedures (1)-(4) we introduce the proof search algorithms $\mathsf{NPComp}_{\mathsf{ALG}}$.

(0) list_proof(), list_goals(), **GO TO** (1)

(1) Given a task $\Gamma \Vdash G_0$, $G_{cur} = G_0$ $(\Gamma \neq \emptyset) \longrightarrow$ (list_proof $= \Gamma$, list_goals $= G_0$, **GO TO** (2)) else list_goals $= G_0$, **GO TO** (2).

(2) Procedure (3): Reached (G_{cur}) = **true** $\longrightarrow$ list_goals = list_goals $- G_{cur}$

$(G_{cur} = G_0) \longrightarrow$ **GO TO** $(6a)$ else $G_{cur} = last$(list_goals) **GO TO** (3)

Reached (G_{cur}) = **false** $\longrightarrow$ **GO TO** (4).

(3) Procedure (4): apply an introduction rule, **GO TO** (2).

(4) Procedure (1): elimination rules

(4a) Elimination rule is applicable, **GO TO** (2) else **GO TO** (5).

(5) Procedure (2): update list_proof and list_goals based on the structure of G_{cur}

(5a) Procedure (2.1): analysis of the structure of G_{cur}, **GO TO** (2) else

(5b) Procedure (2.2): searching for the sources of new goals in list_proof), **GO TO** (2) else

(5c) (if all compound formulae in list_proof are marked, i.e. have been considered as sources for new goals), **GO TO** (6b).

(6) Terminate ($\mathsf{NPComp}_{\mathsf{ALG}}$).

(6a) The desired ND proof has been found. EXIT,

(6b) No ND proof has been found. EXIT.

Complexity Analysis

We consider a family Σ_n of formulas introduced by Cook and Reckhow in [4].

$$\Sigma_n = \bigcup \{\pm A \vee \pm A_{\pm} \vee \ldots \vee A_{\pm(n-1)\pm}\}$$

Here $+A = p$ and $-A = \neg p$ are literals. One can exemplify this family with $\Sigma_1 = \{A, \neg A\}$ and $\Sigma_2 = \{A \vee A_+, A \vee \neg A_+, \neg A \vee A_-, \neg A \vee \neg A_-\}$. Informally, Σ_n is simply a family of all disjunctions of literals with n disjuncts. It is clear that $|\Sigma_n| = 2^n$. We will further designate each member of Σ_n with F_i^n $(1 \leqslant i \leqslant 2^n)$. Following Cook and Reckhow, analytic tableaux can show inconsistency of Σ_n in at least $2^{\Omega(2^n)}$ steps.

We follow Massacci [5] (see also the discussion about Massacci's paper in [2]) and assume that literals are associated from left to right. Under these conditions Massacci showed that analytic tableaux can prove inconsistency of Σ_n in no more than $O(2^{n^2})$ steps which was exponentially shorter than lower bound provided by Cook and Reckhow.

We will further associate each Σ_n with two formulae:

$$\mathcal{F}_{\vee} = \bigvee_{i=1}^{2^n} F_i^n \quad \text{and} \quad \mathcal{F}_{\wedge} = \neg \bigwedge_{i=1}^{2^n} F_i^n$$

We assume that all F_i^n are associated and ordered arbitrarily in both cases.

It was shown that the proof searching algorithm for natural deduction is complete [1], i.e., that it can prove every classical propositional tautology. The algorithm has a remarkable property: it can delete steps of a derivation if it finds the current goal to be unreachable. This property means that there can be a difference between number of steps in the resulting inference and the number of formulas which were introduced to the inference.

The following theorems can be proved.

Theorem 1. *Proof searching algorithm can prove $\mathcal{F}_{\vee}$ in $O(2^n)$ steps including deleted ones.*

Theorem 2. *Proof searching algorithm can prove $\mathcal{F}_{\wedge}$ in $O(n \cdot 2^n)$ steps including deleted ones.*

References

[1] A.Bolotov, V. Bocharov, A. Gorchakov, and V. Shangin. Automated first order natural deduction. In *Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, India, December 20-22, 2005*, pages 1292–1311, 2005.

[2] N. Arai, T. Pitassi, and A. Urquhart. The complexity of analytic tableaux. *J. Symbolic Logic*, 71(3):777 – 790, 2006.

[3] A. Bolotov, O. Grigoriev, and V. Shangin. Automated natural deduction for propositional linear-time temporal logic. In *14th International Symposium on Temporal Representation and Reasoning (TIME 2007), 28-30 June 2007, Alicante, Spain*, pages 47–58, 2007.

[4] S.A. Cook and R. Reckhow. On the lengths of proofs in the propositional calculus. In *Proc. 6th ACM Symp.on Theory of Computing (STOC-74)*, pages 135–148, 1974.

[5] F. Massacci. The proof complexity of analytic and clausal tableaux. *Theoretical Computer Science*, 243(1):477 – 487, 2000.

ABAplus: Implementing Attack Reversal in Structured Argumentation with Preferences

Ziyi Bao Kristijonas Čyras Francesca Toni

Department of Computing, Imperial College London
{ziyi.bao14, k.cyras13, ft}@imperial.ac.uk

Abstract: ABA$^+$ is a recent extension of the well established structured argumentation formalism, Assumption-Based Argumentation (ABA). ABA$^+$ employs a novel way to account for preferences in structured argumentation, namely via attack reversal. We here present ABAplus, a system that implements ABA$^+$. ABAplus affords computation, visualisation and comparison of extensions under five argumentation semantics. It is available both as a stand-alone system and as a web application.

1 Introduction

Approaches to preferences in abstract argumentation (AA) [8] and structured argumentation (SA) [3] can be roughly classified as follows: 1. *discarding* attacks from attackers that are less preferred than attackees (e.g. see [1] for AA and ASPIC$^+$ [11] for SA); 2. *reversing* attacks from attackers that are less preferred than attackees (see [2] for AA and Assumption-Based Argumentation with Preferences (ABA$^+$) [6] for SA); 3. comparing extensions by aggregating preferences over their elements (e.g. see [2] for AA and [13] for SA); 4. incorporating numerical weights of arguments or attacks into the definition of semantics (e.g. see [5] for AA and [10] for SA). Implementations of several approaches in classes 1. and 4. exist (e.g. [4, 12]), but, to the best of our knowledge, implementations of approaches in classes 2. and 3. are lacking. In this paper we present an implementation of *attack reversal*, i.e. approaches in class 2., in SA with preferences.

Our system, *ABAplus*, implements the recently proposed formalism ABA$^+$, the only one (to the best of our knowledge) to reverse attacks in SA due to preferences. ABAplus uses a novel semantics-preserving mapping from ABA$^+$ to AA and employs off-the-shelf AA implementation ASPARTIX [9] for determining extensions. To this end, ABAplus implements the principle of *Weak Contraposition* (WCP)—a preference-dependent, relaxed form of contrapositive reasoning which distinguishes a class of ABA$^+$ frameworks that can be mapped to AA while preserving semantic correspondence. In particular, WCP guarantees a correct representation and implementation of ABA$^+$ via AA under five semantics, and allows ABAplus to provide concise graphical visualisation and comparison of ABA$^+$ frameworks. ABAplus is freely available as a stand-alone system (github.com/zb95/2016-ABAPlus) and a web application (www-abaplus.doc.ic.ac.uk).

2 ABAplus

In ABA$^+$, knowledge is represented through a deductive system $(\mathcal{L}, \mathcal{R})$ comprising of a formal language $\mathcal{L}$ and (a set of) inference rules $\mathcal{R}$, while a set $\mathcal{A} \subseteq \mathcal{L}$ of distinguished sentences, called *assumptions*, represents uncertain and/or incomplete information. Assumptions are also responsible for information conflicts: attacks arise among sets of assumptions whenever one set of assumptions deduces (via rules) the so-called *contrary* of some assumption in another set, where contraries are given by a total mapping $^- : \mathcal{A} \to \mathcal{L}$. The conflicts are resolved by taking into account *preferences* over assumptions, given as a transitive binary relation $\leqslant$ on $\mathcal{A}$.

In ABAplus, the user employs Prolog-like notation to specify: rules by `myRule(`h`, [`$b_1, \ldots, b_n$`]).` for a rule $h \leftarrow b_1, \ldots, b_n$; assumptions by `myAsm(`$a$`).` for an assumption a; contraries by `contrary(`a`, `a^c`).` for the contrary a^c of a; preferences by `myPrefLE(`b`, `a`).` and `myPrefLT(`b`, `a`).`, respectively for $b \leqslant a$ and $b < a$.

When processing the input, ABAplus checks and, if needed, enforces WCP [7] by adding certain 'contrapositive' rules to an ABA$^+$ framework. For example, an ABA$^+$ framework comprising three assumptions a, b, c with no preferences and a single rule $b^c \leftarrow a, c$, where $\overline{b} = b^c$ (the language and contraries are otherwise suppressed), satisfies WCP (as there are no preferences) and admits a unique extension, namely $\{a, c\}$, under all semantics. However, with the preference $a < b$ added, the framework does not satisfy WCP, whence ABAplus enforces it by adding, for instance, the rule $a^c \leftarrow b, c$ (where $\overline{a} = a^c$). The resulting framework satisfies WCP and has a unique extension, namely $\{b, c\}$, under all semantics.

After dealing with WCP, ABAplus maps the ABA$^+$ framework into an AA framework, called *assumption graph*, with arguments being assumption sets deducing contraries of assumptions and singleton sets of assumptions, and attacks determined by the ABA$^+$ attack relation. ASPARTIX solver is then used to determine the extensions of the assumption graph, which are then mapped into ABA$^+$ extensions. ABAplus uses the assumption graph to visualise and compare the reasoning outcomes too.

The (cropped) ABAplus output for our running example—assumptions a, b, c, preference $a < b$, rules $b^c \leftarrow a, c$ and $a^c \leftarrow b, c$ (for WCP)—is shown in Figure 1. ABAplus displays the assumption graph with nodes hold-

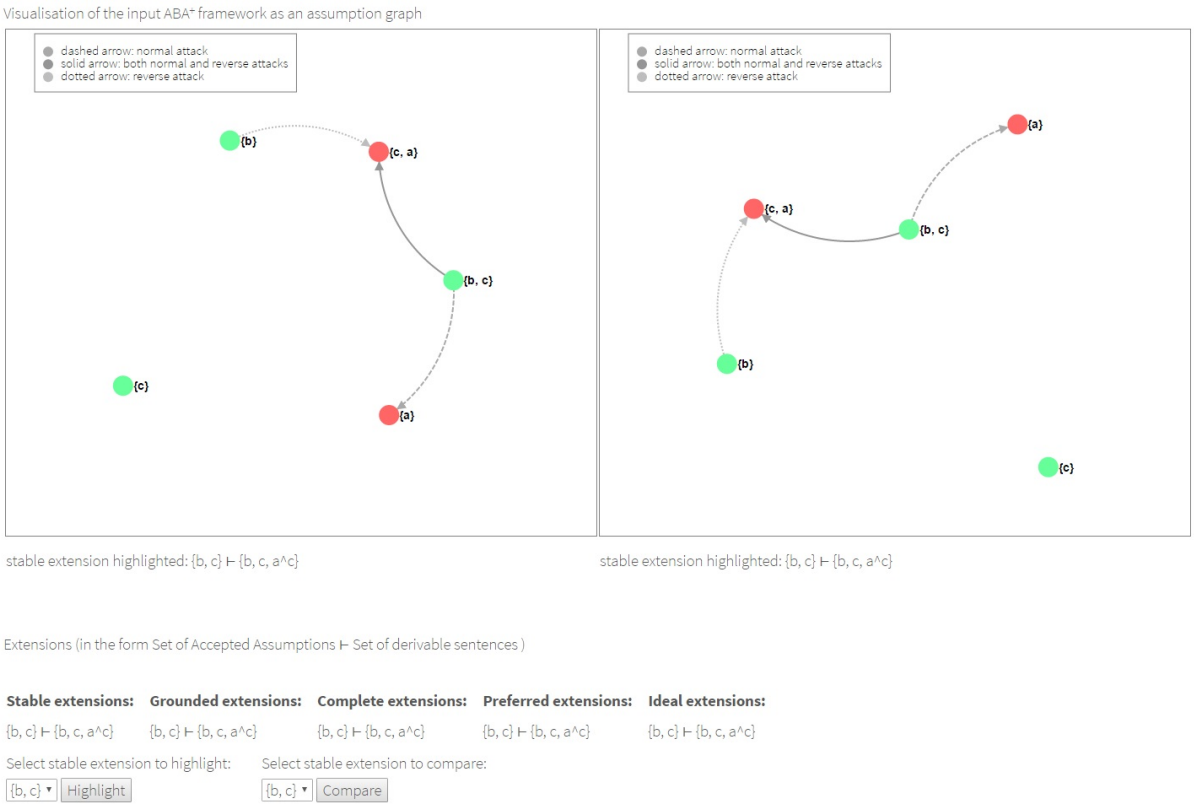


Figure 1: Screenshot (cropped) of the ABAplus reasoning outcome for the ABA$^+$ framework with assumptions a, b, c, preference $a < b$, and rules $b^c \leftarrow a, c$ and $a^c \leftarrow b, c$. Cropped are the editable input window with the specification of the ABA$^+$ framework, as well as the drop-down selection options to highlight and compare extensions under other semantics.

ing assumption sets and edges representing attacks. The latter are distinguished by type: reverse, normal (i.e. non-reverse), or both. ABAplus lists all extensions under five semantics (stable, grounded, complete, preferred, ideal) together with their conclusions (derivable sentences). Any extension can be highlighted in the assumption graph, whence nodes are coloured respectively green and red, indicating inclusion and non-inclusion in the extension of the relevant assumptions. Any two extensions (under possibly different semantics) can be compared, whence two accordingly highlighted assumption graphs are displayed. ABAplus also displays (cropped from Figure 1) an editable window with the input ABA$^+$ framework together with the rule(s) added to satisfy WCP, as well as (partially cropped from Figure 1) drop-down selection options to highlight and compare extensions under all (not only stable) semantics.

ABAplus is a proof-of-concept research tool under development, free to use and open to improvements.

References

[1] L. Amgoud and C. Cayrol. A Reasoning Model Based on the Production of Acceptable Arguments. *Ann. Math. Artif. Intell.*, 34(1-3):197–215, 2002.

[2] L. Amgoud and S. Vesic. Rich Preference-Based Argumentation Frameworks. *Int. J. Approx. Reason.*, 55(2):585–606, 2014.

[3] P. Besnard, A. García, A. Hunter, S. Modgil, H. Prakken, G. Simari, and F. Toni. Introduction to Structured Argumentation. *Arg&Comp*, 5(1):1–4, 2014.

[4] S. Bistarelli, F. Rossi, and F. Santini. ConArg: A Tool for Classical and Weighted Argumentation. In *COMMA*, 463–464, 2016.

[5] S. Bistarelli and F. Santini. A Common Computational Framework for Semiring-based Argumentation System. In *ECAI*, 131–136, 2010.

[6] K. Čyras and F. Toni. ABA+: Assumption-Based Argumentation with Preferences. In *KR*, 553–556, 2016.

[7] K. Čyras and F. Toni. ABA+: Assumption-Based Argumentation with Preferences. *CoRR*, 1610.03024, 2016.

[8] P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games. *Artif. Intell.*, 77:321–357, 1995.

[9] S. Gaggl, N. Manthey, . Ronca, J. Wallner, and S. Woltran. Improved answer-set programming encodings for abstract argumentation. *Theory Pract. Log. Program.*, 15(4-5):434–448, 2015.

[10] T. Gordon, H. Prakken, and D. Walton. The Carneades Model of Argument and Burden of Proof. *Artif. Intell.*, 171(10-15):875–896, 2007.

[11] H. Prakken. An Abstract Framework for Argumentation with Structured Arguments. *Arg&Comp*, 1(2):93–124, 2010.

[12] M. Snaith and C. Reed. TOAST: Online ASPIC+ implementation. In *COMMA*, 509–510, 2012.

[13] T. Wakaki. Assumption-Based Argumentation Equipped with Preferences. In *PRIMA*, 116–132, 2014.

AA-CBR: Explaining Case-Based Reasoning via Argumentation

Kristijonas Čyras Francesca Toni

Department of Computing, Imperial College London
{k.cyras13, ft}@imperial.ac.uk

Abstract: Case-based reasoning (CBR) is extensively used in AI for various applications, to assess a new situation (or case) by recollecting past situations (or cases) and employing the ones most similar to the new situation to give the assessment. We outline a recently proposed method for CBR, based on instantiated abstract argumentation (AA) and referred to as AA-CBR, for problems where cases are represented by abstract factors and (positive or negative) outcomes, and an outcome for a new case needs to be established. We also outline AA-CBR explanations of reasoning outcomes, which can be seen as dialogical processes between a proponent and an opponent seeking to justify and question, respectively, the AA-CBR outcomes.

1 Introduction

Case-based reasoning (CBR), as overviewed in [8], is extensively used in various applications of AI (see e.g. [4, 8]). At a high-level, in CBR a reasoner in need to assess a new situation, or *new case*, recollects past situations, or *past cases*, and employs the ones most similar to the new situation to give the assessment. Several approaches to CBR use (forms of) argumentation, e.g. [1, 7] and, more recently, the AA-CBR approach of [2, 3].

AA-CBR instantiates abstract argumentation (AA) [5] to resolve conflicts amongst most similar past cases with diverging outcomes. It provides: 1) a method for computing outcomes for new cases, given past cases and a *default* outcome; and 2) *explanations* for computed outcomes, as dialogical exchanges between a proponent, in favour of the default outcome for the new case, and an opponent, against the default outcome.

As common in the literature (see e.g. [8]), in AA-CBR past cases are represented as sets of *factors* (also known as features or attribute-value pairs, cf. [9]) together with an outcome, which may be positive ($+$) or negative ($-$). AA-CBR then relies upon the *grounded extension* [5] of an AA framework with, as arguments, a default case (with an empty set of factors and the default outcome), past cases (with their outcomes) and a new case (with unknown outcome). Roughly, a past case attacks another past case or the default case if they have a different outcomes, the former is more specific than the latter and at least as concise as any other similarly more specific, conflicting past case. The following example illustrates AA-CBR.

Example 1. Suppose Bob wishes to rent his spare room to get between £800 and £900 per month, and decides to use an online AA-CBR system to determine whether this amount is reasonable and why, based on similar lodgings being rented. Let N, the new case, represent the set of features of Bob's room, e.g. $N = \{\mathsf{S}, \mathsf{E}, \mathsf{O}, \mathsf{G}\}$ (the room is Small, with an En-suite bathroom in an Open-plan flat with a Gym in the building). Here the default outcome is $+$, indicating Bob's bias for the price range £800–£900. The past cases are either of the form $(X, +)$, for lodgings in the desired price range, or $(Y, -)$, for lodgings in different (lower or higher) price ranges, with X, Y the feature sets of these lodgings. For example, suppose the past cases are $(\{\mathsf{S}\}, -)$ (Small rooms go for lower prices), $(\{\mathsf{S}, \mathsf{E}\}, +)$ (En-suite compensates for Small room), $(\{\mathsf{S}, \mathsf{O}\}, +)$ (Open-plan flat compensates for Small). Then, the corresponding (instantiated) AA framework [2], consisting of a set of *arguments* (as cases) and a binary *attack* relation over arguments, is depicted in Figure 1 (where nodes hold arguments—with $(\emptyset, +)$ the argument for the default case, and $(\{\mathsf{S}, \mathsf{E}, \mathsf{O}, \mathsf{G}\}, ?)$ the argument for the new case—and arrows denote attacks).

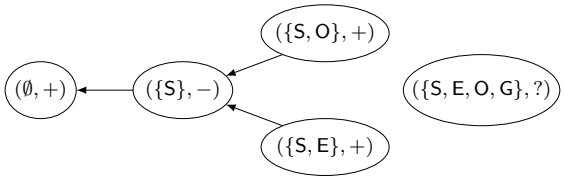


Figure 1: The AA framework from Example 1.

The grounded extension of this AA framework—roughly, obtained by iteratively accepting the unattacked arguments and discarding the arguments attacked by the accepted ones—is $\mathbb{G} = \{(\{\mathsf{S}, \mathsf{E}, \mathsf{O}, \mathsf{G}\}, ?), (\{\mathsf{S}, \mathsf{E}, \}, +), (\{\mathsf{S}, \mathsf{O}, \}, +), (\emptyset, +)\}$. Since $(\emptyset, +) \in \mathbb{G}$, the outcome for the new case—called *AA outcome*—determined by AA-CBR is $+$, with two possible explanations $\mathcal{T}_\mathtt{P}$ and $\mathcal{T}'_\mathtt{P}$ depicted in Figure 2 (with P standing for proponent and O standing for opponent).

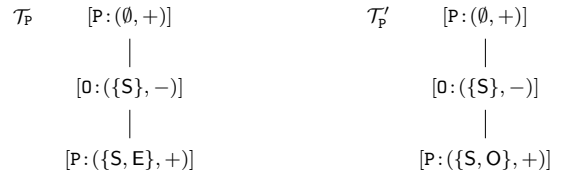


Figure 2: Explanations of the AA outcome in Example 1.

Thus, for example, $\mathcal{T}_\mathtt{P}$ explains the recommendation $+$ dialectically as follows: the default outcome $+$ needs to be defended against the objection posed by the past case $(\{\mathsf{S}\}, -)$, and this can be achieved by using the past case $(\{\mathsf{S}, \mathsf{E}\}, +)$, that cannot be objected against.

2 Properties of AA-CBR Explanations

The notion of explanation is deemed crucial for CBR in many settings, but is inherently hard to define formally (see e.g. [9]). A common form of explanation in CBR amounts to displaying the most similar past cases. In addition, *transparency*, in not trying to "hide conflicting evidence" [9, p. 134], is identified as desirable. To this end, AA-CBR utilises a notion of most similar past cases, called *nearest cases*, which, roughly, are $\subseteq$-maximal subsets of the new case N [2]. They are important in that whenever a nearest case is unique, then the AA outcome matches the outcome of that nearest case. Still further, even when there are possibly more than one nearest case, the grounded extension $\mathbb{G}$ contains them all, be they of agreeing or diverging outcomes, thus adhering to the transparency requirement. However, simply presenting the nearest case(s) as explanation does not "help the user to understand how the symptoms connect with the solution" [9, p. 128]. Here, the argumentative nature of AA-CBR naturally lends itself to a method of explanation based not only on the nearest cases, but on a dialectical exchange of past cases too.

In AA-CBR, the notion of AA outcome allows to determine algorithmically whether a new case N should be assigned the default outcome (d) or not ($\bar{d}$), by determining whether or not (respectively) the default case belongs to the grounded extension of the corresponding AA framework. Explanations for AA outcomes, *AA-CBR explanations* henceforth, are then obtained naturally, by exploiting the argumentative re-interpretation afforded by the corresponding AA framework. In particular, AA-CBR explanations are defined in terms of *dispute trees* [2, 6]: an explanation for why the AA outcome of a new case N is the default outcome d, is an *admissible* dispute tree rooted at the default case argument $(\emptyset, d)$; an explanation for why the AA outcome of a new case N is the opposite $\bar{d}$ of the default outcome, is a *maximal* dispute tree rooted at the default case argument $(\emptyset, d)$; see [2, 3] for details. As such, AA-CBR explanations are guaranteed to exist (particularly because AA-CBR always yields an AA outcome for any new case). Moreover, the set of proponent's arguments in an AA-CBR explanation is guaranteed to be contained (respectively, to not be contained) in the grounded extension $\mathbb{G}$ of the corresponding AA framework whenever the AA outcome is d (respectively, $\bar{d}$).

In general, AA-CBR explanations indicate whether the proponent has a desirable default outcome in mind, as illustrated in the following example.

Example 2 (Example 1 ctd.)**.** Let there be an additional case $(\{\mathsf{S},\mathsf{E},\mathsf{O}\},-)$ in Example 1 (say that a Small En-suite room in an Open-plan flat goes for even more than £900). The corresponding AA framework is depicted in Figure 3.

Here, $\mathbb{G} = \{(\{\mathsf{S},\mathsf{E},\mathsf{O},\mathsf{G}\},?), (\{\mathsf{S},\mathsf{E},\mathsf{O}\},-), (\{\mathsf{S}\},-)\}$ is the grounded extension, so the AA outcome of $\{\mathsf{S},\mathsf{E},\mathsf{O},\mathsf{G}\}$ is $-$, for which the AA-CBR explanations are (in linear notation) $\mathcal{T}_0 : [\mathtt{P}:(\{\emptyset\},+)] — [\mathtt{O}:(\{\mathsf{S}\},-)] — [\mathtt{P}:(\{\mathsf{S},\mathsf{E}\},+)] — [\mathtt{O}:(\{\mathsf{S},\mathsf{E},\mathsf{O}\},-)]$ and $\mathcal{T}_0' : [\mathtt{P}:(\{\emptyset\},+])$ — $[\mathtt{O}:(\{\mathsf{S}\},-)] — [\mathtt{P}:(\{\mathsf{S},\mathsf{O}\},+)] — [\mathtt{O}:(\{\mathsf{S},\mathsf{E},\mathsf{O}\},-)]$.

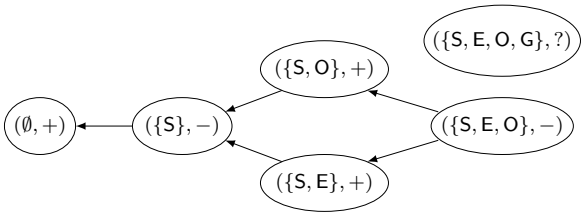


Figure 3: The AA framework from Example 2.

Both explanations suggest to Bob that while with only either En-suite or Open-plan flat Bob is in his assumed price range, with *both* factors he is no more. Presumably, Bob is better off increasing his desirable price for the room.

A further notion of *lean explanations* is defined in AA-CBR [3] to avoid overpopulation of explanations with irrelevant arguments, where the concept of relevance captures, roughly, cases/arguments that share some factors with the new case. Lean AA-CBR explanations keep the desirable properties of AA-CBR explanations, at the same time ensuring that they concern only arguments that are relevant for the AA outcome, as well as structuring past cases. Still further, lean AA-CBR explanations are better suited to indicate whether and how a modification of the new case would result in a different, possibly more desirable, outcome.

References

[1] Latifa Al-Abdulkarim, Katie Atkinson, and Trevor J M Bench-Capon. Abstract Dialectical Frameworks for Legal Reasoning. In *Leg. Knowl. Inf. Syst.*, 61–70. IOS Press, 2014.

[2] Kristijonas Čyras, Ken Satoh, and Francesca Toni. Abstract Argumentation for Case-Based Reasoning. In *KR*, 549–552, 2016.

[3] Kristijonas Čyras, Ken Satoh, and Francesca Toni. Explanation for Case-Based Reasoning via Abstract Argumentation. In *COMMA*, 243–254, 2016.

[4] Ramon López de Mántaras. Case-Based Reasoning. In *Mach. Learn. Its Appl. Adv. Lect.*, 127–145. Springer, 2001.

[5] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games. *Artif. Intell.*, 77:321–357, 1995.

[6] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing Ideal Sceptical Argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

[7] Henry Prakken, Adam Wyner, Trevor J M Bench-Capon, and Katie Atkinson. A Formalization of Argumentation Schemes for Legal Case-Based Reasoning in ASPIC+. *J. Log. Comput.*, 25(5):1141–1166, 2015.

[8] Michael Richter and Rosina Weber. *Case-Based Reasoning*. Springer, 2013.

[9] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in Case-Based Reasoning–Perspectives and Goals. *Artif. Intell. Rev.*, 24(2):109–143, 2005.

Automatically Discovering Human-readable Rules to Predict Effective Compiler Settings for Embedded Software

Craig Blackmore Oliver Ray Kerstin Eder

Department of Computer Science, University of Bristol
{craig.blackmore, oliver.ray, kerstin.eder}@bristol.ac.uk

Abstract: This paper proposes improvements to a recent logic-based approach that optimises the selection of compiler flags on an embedded architecture. First we present and evaluate a new and more effective approach to obtain training data for the model. Then we analyse the human-readable rules that our method produces to gain insights into the learned model and briefly suggest further ways to improve the approach. We provide our training data online to encourage others to apply their techniques to this problem.

1 Introduction

This paper builds on our recent logic-based approach that seeks to optimise the selection of compiler flags on an embedded architecture [1]. Modern compilers offer a range of compiler flags to control optimisation. The optimal configuration of compiler flags is dependent on the target program and architecture. Finding good configurations is a hard task due to the large number of flags available and complex, often unknown interactions between them.

Previous work searched for effective configurations using Iterative Compilation [5], which compiles and runs the target program with several different configurations and selects the one that gives the best execution time. Although this finds good results, it is very time-consuming.

As Iterative Compilation is infeasible in practice, recent studies sought to automatically predict which compiler flags to enable by analysing characteristics of the target program. Most have been propositional machine learning efforts [3] which rely on vectors of statistical aggregates that summarise features of the program code.

Our Inductive Logic Programming (ILP) [4] based approach aims to automatically discover relevant features for the machine learning task rather than relying on predetermined features that may not necessarily be most relevant for the task. This is a first-order logic approach that learns human-readable rules to relate effective compiler flags to specific program features.

As in our original study, we develop and evaluate our improvements by using the GCC compiler toolchain[1] to compile for the STM32VLDISCOVERY embedded system development board[2]. We use 81 programs (21 more than the original study) from the state-of-the-art Bristol/Embecosm Embedded Benchmark Suite (BEEBS)[3] to measure the execution times of a diverse set of programs compiled with different configurations.

After summarising our original method we present a new, more effective approach for generating the training data for the model and we evaluate it using leave-one-out cross validation. Our revised method outperforms the original on two thirds of the programs. Next we discuss insights gained from our method's human-readable rules. To fully exploit the expressive power of ILP, further work on program representation will be required. To enable this, we have published our data online and invite the community to apply their techniques to this problem.[4]

2 Original ILP-based compiler tuning (ILP-Rand)

Inductive Logic Programming (ILP) [4] seeks to discover hypotheses H that generalise relations based on background knowledge B that describes characteristics of the problem, positive examples E^+ for which the relation holds and negative examples E^- for which it does not hold.

In the context of compiler flag selection, our goal is to learn the relation badFlag(P,F). which can be read as "Flag F is a bad flag for program P". A 'bad flag' is a flag which degrades performance for the target program and a 'good flag' is one which improves performance. We chose the predicate badFlag/2 because we found it was more effective to selectively disable flags rather than predict which ones to enable.

The training data consists of B which describes the source code, positive examples E^+ of significantly bad flags and negative examples E^- of significantly good flags.

We used Milepost GCC [3] to extract a Datalog encoding of GCC's internal Intermediate Representation (IR). This forms the basis for B. We also added auxiliary predicates which enable the IR to be generalised.

To determine good and bad flags we generated 1000 random configurations of 133 compiler flags and measured their performance on each program. For each program a set of good configurations was identified by choosing the configurations that performed within 5% of the best. If a flag appeared in at least 75% of these good configurations it was classed as good and if it appeared in less than 25% it was classed as bad. These parameters were chosen based on preliminary experiments.

[1] http://gcc.gnu.org/
[2] http://www.st.com/en/evaluation-tools/stm32vldiscovery.html
[3] http://beebs.eu
[4] http://github.com/craigblackmore/ilp-ce

3 Revised ILP-based compiler tuning (ILP-CE)

We propose three improvements to ILP-Rand. Firstly, instead of using random sampling to search for good configurations, we use Combined Elimination (CE) [5] which finds better results in less time [2]. Intuitively, CE begins with all flags enabled and continually disables the flag with the biggest negative impact on the program's performance, until no further gains are made.

Secondly, we have developed an alternative way to identify good and bad flag examples. Using random data leads to noisy examples as flags could appear frequently or infrequently in the set of good configurations by chance (especially when the set of good configurations is small). These noisy examples confuse the learning process.

We propose a 'sensitivity analysis' which finds flags that exhibit a significant impact on the target program. For each program, we first select the best configuration found by CE. Then each enabled flag is disabled one-by-one – if disabling the flag makes performance significantly worse, then it is classed as a good flag. Similarly, each disabled flag is enabled one-by-one and, if enabling the flag degrades performance significantly, it is classed as a bad flag.

This sensitivity analysis gives more reliable examples that cannot appear by chance and it also allows all programs to be included in training, even if only one good configuration is found for them. With the previous approach, it was impossible to determine whether the flags had appeared in a single configuration by chance.

Finally, we fixed an oversight in the design of BEEBS that had enabled the compiler to over-optimise based on knowledge of the test input data used by some benchmarks [2]. This increases confidence that the gains are representative of realistic applications.

Our improved approach outperforms ILP-Rand on 54 out of 81 benchmarks (Fig. 1). In many cases the performance is close to the best known configuration.

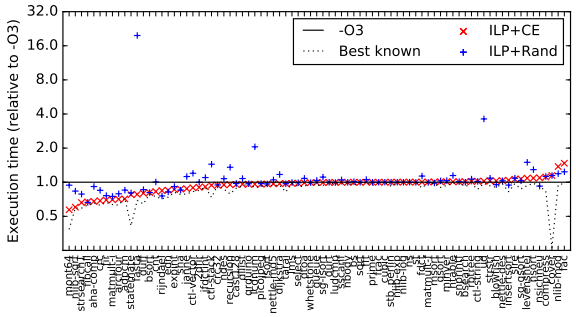


Figure 1: Leave-one-out cross-validation of ILP+CE and ILP+Rand (logarithmic scale)

4 Insights from Human-readable Rules

ILP+CE found 25 rules to control flags. These can easily be translated into understandable sentences. For example the following rule states that flag '-fschedule-insns' should be disabled if the program has a function with an expression that contains a 64-bit integer variable:

```
badFlag(P,'-fschedule-insns') :-
  expr_type(P,F,Expr,Type),
  expr_int_size(P,F,Type,64),
  expr_var(P,F,Expr,Var).
```

Another example is a little less clear but may provide some useful information. The following rule states that '-fguess-branch-probability' should be disabled if the program has a function with at least one edge, at least two floating point expressions and an exceptional expression. This suggests that the compiler's branch prediction may perform poorly for floating point programs on the target platform.

```
badFlag(P,'-fguess-branch-probability')
  :- edge_src(P,F,Src,BB),
     expr_class(P,F,Expr,exceptional),
     expr_code2(P,F,real_type).
```

Seven of the rules are actually facts that determine certain flags should always be disabled on our target platform. Four of these flags were also found in [2].

5 Conclusion and Future Work

Our revised method generates more robust training examples that will enable future work to focus on improving background knowledge. In this respect, we plan to explore alternative program representations and devise additional auxiliary rules to fully exploit the expressivity of ILP.

Acknowledgements

This work was supported by EPSRC award ref. 1628029.

References

[1] C. Blackmore, O. Ray, and K. Eder. A logic programming approach to predict effective compiler settings for embedded software. *Theory and Practice of Logic Programming*, 15(4-5):481–494, 2015.

[2] C. Blackmore, O. Ray, and K. Eder. Automatically tuning the GCC compiler to optimize the performance of applications running on the ARM Cortex-M3. *arXiv:1703.08228 [cs.DC]*, 2017.

[3] G. Fursin et al. Milepost GCC: Machine learning enabled self-tuning compiler. *Int. J. of Parallel Programming*, 39(3):296–327, 2011.

[4] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The J. of Logic Programming*, 19:629–679, 1994.

[5] Z. Pan and R. Eigenmann. Fast and effective orchestration of compiler optimizations for automatic performance tuning. In *Proc. of the Int. Symp. on Code Generation and Optimization*. IEEE, 2006.

The role of textualisation and argumentation in understanding the machine learning process: a position paper

Kacper Sokol Peter Flach

Department of Computer Science, University of Bristol
{k.sokol, peter.flach}@bristol.ac.uk

Abstract: Understanding data, models and predictions is important for any machine learning application. Due to the limitations of our spatial perception and intuition, analysing high-dimensional data is inherently difficult. Furthermore, black-box models achieving high predictive accuracy are widely used, yet the logic behind their predictions is often opaque. Use of textualisation – a natural language narrative of selected phenomenon – can tackle these shortcomings. When extended with argumentation theory we could envisage machine learning models and predictions arguing persuasively for their choices.

1 Introduction

In recent years machine learning has witnessed a big technological leap and proliferation in everyday life. Predictive models initially flourished in the Internet fuelling shopping recommendations and internet search. Nowadays they are becoming a vital part of decision support systems used in judicial system, politics, finance, credit scoring and job appointments. This widespread adaptation of machine learning algorithms and their influence on our every-day life is often criticised for unfairness[1]. In the wake of algorithms taking supposedly "optimal" decisions for human matters the European Union has introduced the "right to explanation"; it entitles involved parties to receive an explanation of the algorithm's decision[2]. Moreover, protected features such as gender and race cannot be used in predictive models to prevent discrimination.

In the digital age data are easy to collect; machine learning models are simple to use, learn and deploy with packages such as `scikit-learn` and `weka`. Nevertheless, they are rarely understood and inspected in detail before deployment as the main objective is to maximise the predictive accuracy, which rarely includes social costs. Given large amounts and high dimensionality of data, learnt models and the nature of their predictions can easily become incomprehensible. Their better understanding could result in selecting the correct features and model for the task, hence guarantee fair predictions in deployment.

2 Research problem

To address these issues machine learning experts have developed techniques to inspect data, models and predictions. Understanding data is the most difficult part of the process; researchers use correlation maps to discover dependencies and interactions between features, yet these are limited to pairwise correlation coefficient. Approaches such as t-SNE and PCA allow to project high-dimensional data into 2 or 3 dimensions that can be visually inspected [2]. Understanding models and predictions is a difficult task as well. Most commonly, researchers use white-box models when transparency is crucial. Decision trees and rule learners, for example, can be read as conjunctions of logical conditions. Predictions of linear regression can be described with corresponding feature weights (but this makes assumptions about the commeasurability of features). Black-box models like deep neural networks, on the other hand, are almost impossible to interpret. Their predictions can be understood by applying post-hoc methods. These usually build a simple representation of the decision criterion (e.g. linear model) in the neighbourhood of the instance of interest.

In general, these techniques can be divided into two groups.

Model-dependent are developed with a specific machine learning task in mind e.g. linear model and feature weights, conjunction of logical conditions in a rule. They suffer from scalability issues as each model family requires its own approach.

Model-agnostic (usually post-hoc) can be applied to any task e.g. local linear model. They are versatile but they use white-box models as backbone, hence they inherit their limitations.

Finally, all of these approaches to understand the data, models and predictions share one commonality: they use *visualisation* in the core of their descriptive power. This is powerful but also has limitations as our intuition in high-dimensional spaces is flawed – we often call this the curse of dimensionality. Moreover, they *describe* but do not *explain*: they provide statistics and characteristics that quantify models' behaviour but not their reasoning. While in some cases model and features are simple (small and shallow decision tree) and description is more or less equivalent to explanation, such models are rarely the norm in today's data-rich world, where multidimensional data renders white-box models incomprehensible.

Therefore, we need explanations supported by *arguments* that lead to understanding. Explaining a model means providing a high-level insight into its decision system, e.g. self-

[1] "Weapons of Math Destruction" by Cathy O'Neil
[2] General Data Protection Regulation (EU 2016/679)

driving cars stop before zebra crossing if a moving object is detected nearby. Explaining a prediction means presenting a thought process supported with arguments, e.g. a self-driving car crossed the junction without stopping because the light was green and the pedestrian crossing was empty. Last but not least, explaining data means to understand their patterns as used by machine learning algorithms to make inferences, e.g. this picture shows a pedestrian crossing because there are vertical white stripes on the road.

3 Approach

The simplest and most common approach to characterise a machine learning component, in particular data, is *(statistical) summarisation*. These are usually numerical tables and lists which can be difficult to digest for non-experts. Their role is to describe properties of the data by providing information from the system to the user. A more advanced analytic tool is *visualisation*; graphical representation of data in a form of plots and figures is more insightful. It is also descriptive but sometimes the communication can be bidirectional e.g. interactive plots. Visualisations are often supported by a small narrative in a form of caption, which increases their informativeness. To overcome the curse of dimensionality we can use *textualisation* – narratives accompanied by statistics and figures. Natural language can express concepts of arbitrary complexity and dimensionality. Moreover, it is proven more insightful and effective than presenting raw, numerical and visual data [4]. Finally, we suggest to make use of *argumentation* – structural, logical narratives accounting for every disputable statement. It provides explanation leading to understanding rather than informative description; a long overdue approach.

Using natural language to describe machine learning data, models and predictions is uncommon. Narrative was used to present data analysis in the Automatic Statistician project[3]. [1] generated reposts of control systems and used narrative to present anomalies in operating system logs. [4] developed a system that synthesises medical reports from neonatal intensive care unit data to support medical decisions. Application of argumentation theory in machine learning is even less common despite its capability of argumenting classification choices, hence providing invaluable insight into model reasoning [3].

4 Contributions and directions

Expressiveness and versatility of *textualisation* and logical reasoning behind *argumentation* point toward benefits of its wider adaptation in machine learning. We aim to improve the currently available data-to-text frameworks. Such systems mostly use (conditional) templating, hence they require manual engineering and are limited to a particular application domain. We aim to develop a flexible narrative generation platform that accepts a variety of data types. We will showcase capabilities of our platform by composing narratives of experimental results so often used in the *results* section of our papers. Such a system, for example, would provide a textual narrative comparing performance of a novel algorithm against state-of-the-art solutions only based on accuracy table and experiment meta-data.

Generating a summary of arbitrary data requires a unified feature and meta-data representation. We will introduce a feature annotation approach useful in explaining the data, models and predictions. For example, knowing that two features are length measurements expressed in the same unit would vouch for a model using their mean, while averaging time-stamp and temperature is counter-intuitive.

Allowing the data-to-text framework to use arbitrary approach to analyse feature interactions and dependencies could help avoid using combinations of features that are equivalent to a protected feature. If these interactions are complex, describing them with natural language is preferable to numerical coefficients, graphs and figures.

Finally, full potential of argumentation theory has not yet been applied to explain machine learning approaches. We will integrate it with other parts of our data-to-text system to produce model-agnostic explanations that yield better understanding of our field. For example, a model recognising activities based on environmental sensor data could argue in favour of its choice. By integrating a human into the learning loop fallacies could be identified and the model refined.

5 Summary

Narrative is a promising direction for better understanding machine learning data, models and prediction. It can describe concepts of arbitrary complexity and when accompanied with argumentation theory it can explain them. When combined with state-of-the-art statistical and machine learning models it can vastly improve our understanding of the algorithms that we use, as well as the predictions they produce and the data on which they are based.

[3] https://www.automaticstatistician.com

References

[1] Rachel Farrell, Gordon Pace, and Michael Rosner. A framework for the generation of computer system diagnostics in natural language using finite state methods. In *Proceedings of the 15th European Workshop on Natural Language Generation, 52–56*, 2015.

[2] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[3] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10-15):922–937, 2007.

[4] François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7):789–816, 2009.

An Inductive Logic Programming approach for analysing Cyber-Attacks

Oliver Ray[1] Samuel Hicks[1] Steve Moyle[2]

[1] Department of Computer Science, University of Bristol
oliver.ray@bristol.ac.uk, sam.hicks.2014@my.bristol.ac.uk
[2] Acute Intelligence
katarapkokid@hotmail.com

Abstract: This paper describes a proof-of-principle study aimed at developing an Inductive Logic Programming (ILP) tool to assist experts in analysing cyber-attacks using logs acquired by an eavesdropping device placed in the host computer's network to monitor various aspects of its communications traffic. Using the CryptoWall ransomeware attack as an example, we show how ILP can be used to interactively learn rules describing malware behaviour that are comparable to those hand-crafted by a human.

1 Introduction

This paper builds on a recent case-study [1] which showed the Inductive Logic Programming (ILP) system Aleph [2] can relearn meaningful rules about malware behaviour using a small amount of carefully curated log data (extracted by a professional cyber-security collaborator from a sandbox computer that was deliberately infected with the CryptoWall ransomeware [3]) with respect to a carefully chosen set of positive and negative examples (selected by us to allow Aleph to automatically learn the intended rules).

We note that our work differs from previous applications of ILP to cyber-security (such as [4]) in that it aims to help humans understand novel types of attack rather than to help them detect known types of attack. This paper extends our previous work in two key ways, as described in the following two sections. For more information on the CryptoWall-4 attack and our Prolog representation of network traffic data, the reader is referred back to [1].

2 NIDS: Network Intrusion Detection System

Our first contribution was to develop a Network Intrusion Detection System (NIDS) and collect ten week's worth of network traffic data generated by a genuine small business network. We also developed a tool to transform this data into the Prolog format we previously used to represent the CryptoWall sandbox data provided by our collaborator. The hardware for our NIDS was based on a Raspberry Pi and the software was based on the open source Bro Network Security Monitor [5]. This resulted in the experimental setup shown in Figure 1. Due to the huge amount of network data as compared to the tiny amount of sandbox data, we stored these logs in two separate files that we call the 'CryptoLog' and the 'NetLog' respectively. They were kept separate as it was not possible to apply Aleph to the NetLog, since even running a simple Prolog query on that huge file takes a very significant amount of time.

3 ACUITY: Abducing Constraints for User-guided Induction of a TheorY

Our second contribution was to develop an extension of Aleph that facilitates the interactive learning of rules by helping the user find counter-examples to overly general hypotheses and avoiding the generation of logically redundant specialisations of refuted rules. The first part was done by showing the extent of hypotheses over the CryptoLog and NetLog. This allowed the user to easily spot potential counterexamples. The second part was done by a Skolemisation method that overcame a key problem of Aleph's existing mechanisms for dealing with incorrect hypotheses: namely its tendency to become stuck in descending chains of logically equivalent versions of a rejected hypotheses. Figure 2 outlines the key features of our system compared to Aleph; and Figure 3 shows the intended rules from [1] that the user was easily able to find using our new system.

Acknowledgements

We thank Paul Byrne for the CryptoLog data, and EPSRC for supporting this work through a summer bursary.

References

[1] O. Ray, S. Hicks and S. Moyle. *Using ILP to Analyse Ransomware Attacks*. Proc. ILP'16 (to appear). http://ilp16.doc.ic.ac.uk/program/papers/46,

[2] A. Srinivasan. *The Aleph Manual.* http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/, 2007.

[3] BleepingComputer.com. *CryptoWall 4.0: Help_Your_Files Ransomware Support Topic*. http://www.bleepingcomputer.com/forums/t/595215/cryptowall-40-help-your-files-ransomware-support-topic/, November 2015.

[4] S. Moyle and J. Heasman. *Machine Learning to Detect Intrusion Strategies*. KES 2003, LNCS 2773:371-378, 2003

[5] The Bro Project. *The Bro Network Security Monitor*. https://www.bro.org/, 2014.

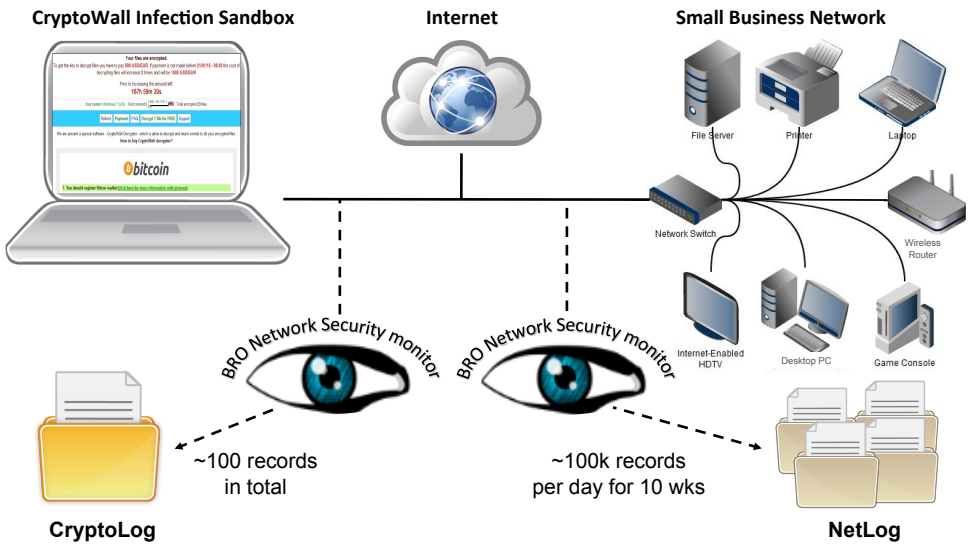


Figure 1: Data Collection: NIDS

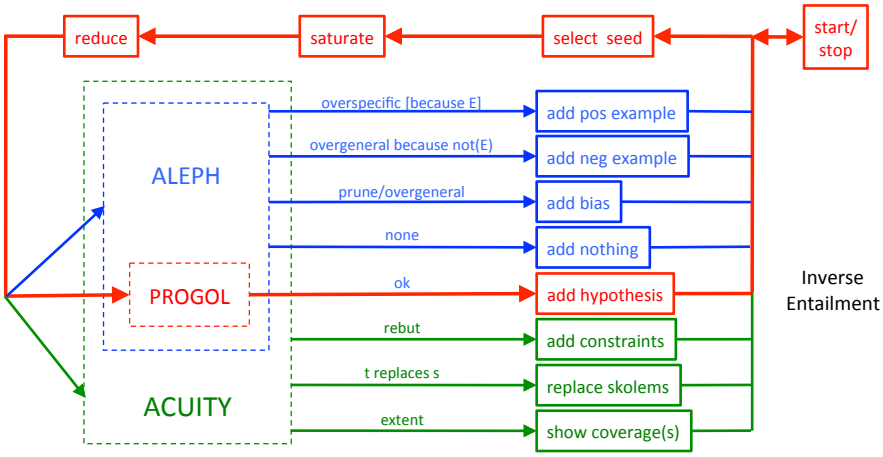


Figure 2: Interactive Learning: ACUITY

```
malware_domain(Domain):-
  http_domain_name_parameter(Machine,Domain,Name1,Param1),
  http_domain_name_parameter(Machine,Domain,Name2,Param1),
  http_domain_name_parameter(Machine,Domain,Name1,Param2),
  Name1 \= Name2, Param1 \= Param2.

malware_fetch(A,B,C):-
   malware_domain(C), http(A,D,B,E,'POST',C,F,G,H,I,J,K,L,M,
   vector('text/plain'),N,vector('text/plain')), gt1000(K).
```

Figure 3: Example of learnt rules

Forgetting Role Symbols in $\mathcal{ALCOQH}(\triangledown)$-Ontologies

Yizheng Zhao Renate A. Schmidt

School of Computer Science, The University of Manchester, UK
{yizheng.zhao,renate.schmidt}@manchester.ac.uk

Abstract: Forgetting (in description logic contexts) refers to a non-standard reasoning problem concerned with eliminating concept and role symbols from description logic ontologies while preserving all logical consequences up to the remaining symbols in their initial signatures. While early work focused mostly on forgetting concept symbols, we in this work turn the attention to role symbol forgetting. In particular, we describe a practical method of role forgetting for ontologies expressible in $\mathcal{ALCOQH}(\triangledown)$, i.e., the basic description logic $\mathcal{ALC}$ extended with nominals, qualified number restrictions, role inclusions and the universal role.

1 Introduction

Early work in the area of description logics focused mostly on forgetting concept symbols, as role forgetting was realised to be significantly harder. An important known reason is that the solution of role forgetting often requires more expressivity than is available in the source language. In this work we describe a practical method for role forgetting in expressive description logics not considered so far. In particular, the method accommodates ontologies expressible in the description logic $\mathcal{ALCOQH}$ and the extension with the universal role $\triangledown$. The extended expressivity enriches the target language, making it expressive enough to represent the forgetting solution which otherwise would have been lost. For instance, the solution of forgetting the role symbol $\{r\}$ from the ontology $\{A_1 \sqsubseteq \geq 2r.B_1, A_2 \sqsubseteq \leq 1r.B_2\}$ is $\{A_1 \sqsubseteq \geq 2\triangledown.B_1, A_1 \sqcap A_2 \sqsubseteq \geq 1\triangledown.(B_1 \sqcap \neg B_2)\}$, whereas in a description logic without the universal role $\triangledown$, the uniform interpolant is $\{\top\}$, which is weaker. The method is goal-oriented and incremental. It always terminates and is sound in the sense that the forgetting solution is equivalent to the original ontology up to the symbols that have been forgotten. Our method is nearly role forgetting complete for $\mathcal{ALCOQH}(\triangledown)$-ontologies, and we characterise cases where the method is complete. Only problematic are cases where forgetting a role symbol would require the combinations of certain cardinality constraints and role inclusions.

2 Definition of Forgetting

Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_O}$ be three pairwise disjoint sets of *concept symbols (atomic concepts)*, *role symbols (atomic roles)* and *individuals (nominals)*, respectively. Let $\mathrm{sig}_{\mathsf{R}}(X)$ denote the role symbols occurring in X, where X ranges over concepts, roles, axioms, clauses, ontologies of axioms, or sets of clauses. Let $r \in \mathsf{N_R}$ be any role symbol, and let $\mathcal{I}$ and $\mathcal{I}'$ be any interpretations. We say $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to* r, or r*-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ coincide but differ possibly in the interpretations of r. More generally, $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent up to a set* Σ *of role symbols*, or Σ*-equivalent*, if $\mathcal{I}$ and $\mathcal{I}'$ are identical but differ possibly in the interpretations of the symbols in Σ.

Definition 1 (Forgetting in $\mathcal{ALCOQH}(\triangledown)$) *Let $\mathcal{O}$ and $\mathcal{O}'$ be two $\mathcal{ALCOQH}(\triangledown)$-ontologies, and let Σ be any subset of* $\mathrm{sig}_{\mathsf{R}}(\mathcal{O})$. *$\mathcal{O}'$ is a* solution of forgetting Σ *from $\mathcal{O}$, if the following conditions hold: (i)* $\mathrm{sig}_{\mathsf{R}}(\mathcal{O}') \subseteq \mathrm{sig}_{\mathsf{R}}(\mathcal{O}) \backslash \Sigma$, *and (ii) for any interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{O}'$ iff $\mathcal{I}' \models \mathcal{O}$, for some interpretation $\mathcal{I}'$ Σ-equivalent to $\mathcal{I}$.*

In this work, Σ is assumed to be a set of role symbols to be forgotten. The symbol in Σ being forgotten is referred to as the *pivot* in our method. An axiom (clause) that contains the pivot is called a *pivot-axiom* (*pivot-clause*).

3 Approach to Eliminating Single Role Symbols

Since in a description logic with nominals, ABox assertions can equivalently be expressed as TBox axioms (via nominals), we assume w.l.o.g. that an ontology contains only TBox and RBox axioms in this work. Given an ontology $\mathcal{O}$ and a set $\Sigma \subseteq \mathrm{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten, computing the solution of forgetting Σ from $\mathcal{O}$ can be reduced to the problem of eliminating single symbols in Σ.

We then describe our approach to eliminating single role symbols from a set of TBox and RBox clauses expressible in $\mathcal{ALCOQH}(\triangledown)$. In particular, the approach has two key ingredients: (i) the conversion of pivot-clauses into *reduced form*, and (ii) a set of elimination rules, namely, *Ackermann rules*. The Ackermann rules are non-trivial generalisations of Ackermann's Lemma [1], and allow a role symbol to be eliminated from a set of clauses in reduced form.

Definition 2 *For $r \in \mathsf{N_R}$ the pivot, a TBox pivot-clause is in* reduced form *if it has the form $E \sqcup \geq m r.F$ or $E \sqcup \leq n r.F$, where E and F are concepts that do not contain r, and $m \geq 1$ and $n \geq 0$ are integers. An RBox pivot-clause is in* reduced form *if it has the form $\neg s \sqcup r$ or $s \sqcup \neg r$, where $s \neq r$ is a role symbol. A set $\mathcal{N}$ of clauses is in* reduced form *if every pivot-clause in $\mathcal{N}$ is in reduced form.*

These reduced forms include all basic forms of TBox and RBox clauses in which a role symbol could occur. While an RBox pivot-clause is always in reduced form, this is not true for a TBox pivot-clause. A TBox pivot-clause not in reduced form has the form $E \sqcup \geq m\mathcal{S}.F$ or $E \sqcup \leq n\mathcal{S}.F$, where $\mathcal{S}$ can be any role (including the pivot symbol), and

Ackermann Rule

$$\frac{\mathcal{N}, \overbrace{\neg s_1 \sqcup r, \ldots, \neg s_v \sqcup r}^{\mathcal{P}^+_{\mathcal{R}}(r)}, \overbrace{E_1 \sqcup \leq y_1 r.F_1, \ldots, E_n \sqcup \leq y_n r.F_n}^{\mathcal{P}^-_{\mathcal{T}}(r)}, \overbrace{t_1 \sqcup \neg r, \ldots, t_w \sqcup \neg r}^{\mathcal{P}^-_{\mathcal{R}}(r)}}{\mathcal{N}, \textbf{BLOCK}(\mathcal{P}^+_{\mathcal{R}}(r), E_1 \sqcup \leq y_1 r.F_1), ..., \textbf{BLOCK}(\mathcal{P}^+_{\mathcal{R}}(r), E_n \sqcup \leq y_n r.F_n), \textbf{BLOCK}(\mathcal{P}^+_{\mathcal{R}}(r), t_1 \sqcup \neg r), ..., \textbf{BLOCK}(\mathcal{P}^+_{\mathcal{R}}(r), t_w \sqcup \neg r)}$$

$\textbf{BLOCK}(\mathcal{P}^+_{\mathcal{R}}(r), E_j \sqcup \leq y_j r.F_j)$ denotes the set: $\{E_j \sqcup \leq y_j s_1.F_j, \ldots, E_j \sqcup \leq y_j s_v.F_j\}$.
$\textbf{BLOCK}(\mathcal{P}^+_{\mathcal{R}}(r), t_k \sqcup \neg r)$ denotes the set: $\{\neg s_1 \sqcup t_k, \ldots, \neg s_v \sqcup t_k\}$.

Figure 1: A sample Ackermann rule for eliminating $r \in \text{sig}_{\mathsf{R}}(\mathcal{N})$ from a set $\mathcal{N}$ of clauses in reduced form

E and F are concepts with at least one of them containing the pivot; $(\leq 1 r.A) \sqcup (\geq 2 s. \geq 1 r.B)$ is such an example. Finding the reduced form of a TBox pivot-clause is not always possible unless definer symbols are introduced. *Definer symbols* are auxiliary concept symbols that do not occur in the present ontology and are introduced in the way described in [4]. In this way, any $\mathcal{ALCOQH}(\triangledown)$-ontology can be transformed into a set of clauses in reduced form.

For space reasons we do not present all Ackermann rules in this work; instead we provide a sample Ackermann rule in Figure 1, giving readers a flavour of the rules.

4 Key Aspects of the Method

The forgetting process in our method comprises three main phases: the conversion of a given ontology $\mathcal{O}$ into a set $\mathcal{N}$ of clauses (**the first phase**), the Σ-symbol elimination phase (**the central phase**), and the definer symbol elimination phase (**the final phase**). It is always assumed that as soon as a forgetting solution is computed, the remaining phases are skipped.

Input: The input to the method are (i) an $\mathcal{ALCOQH}(\triangledown)$-ontology $\mathcal{O}$ of TBox and RBox axioms, and (ii) a set $\Sigma \subseteq \text{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten. Σ-symbols can be flexibly specified in our method.

The first phase: The first phase of the forgetting process transforms $\mathcal{O}$ into a set $\mathcal{N}$ of clauses using standard clausal normal form transformations.

The central phase: Central to the forgetting process is the Σ-symbol elimination phase, which is an iteration of several rounds in which single Σ-symbols are eliminated. Specifically, the method attempts to eliminate Σ-symbols one by one using the approach described in the previous section. In each elimination round, the method performs two steps. The first step transforms every TBox pivot-clause into reduced form, so that one of the Ackermann rules can be applied. The second step eliminates the pivot by applying an appropriate Ackermann rule. Upon the intermediate result returned at the end of each round, the method repeats the same steps in the next round (if necessary) for the elimination of another symbol in Σ. If, using our Ackermann approach, a Σ-symbol has been found ineliminable from the present ontology (i.e., none of the Ackermann rules is applicable to the current reduced form), the method skips the current round and attempts to eliminate another symbol in Σ (if necessary).

The final phase: To help the conversion of TBox clauses into reduced form, definer symbols may be introduced during the elimination rounds. The final phase of the forgetting process eliminates the definer symbols using the technique for definer elimination described in [2].

Output: What the method returns at the end of the forgetting process, if the forgetting is successful, is a finite set $\mathcal{O}'$ of clauses that do not contain the symbols in Σ.

Theorem 1 (Termination and Soundness) *For any $\mathcal{ALCOQH}(\triangledown)$-ontology and any set $\Sigma \subseteq \text{sig}_{\mathsf{R}}(\mathcal{O})$ of role symbols to be forgotten, the method always terminates and returns a finite set $\mathcal{O}'$ of clauses. If $\mathcal{O}'$ does not contain any symbols in Σ, the method is* successful. *$\mathcal{O}'$ is then a solution of forgetting Σ from $\mathcal{O}$. If neither $\mathcal{O}$ nor $\mathcal{O}'$ use the universal role, $\mathcal{O}'$ is an $\mathcal{ALCOQH}$-ontology. Otherwise it is an $\mathcal{ALCOQH}(\triangledown)$-ontology.*

Theorem 2 (Completeness) *Let $\mathcal{O}$ be any $\mathcal{ALCOQH}(\triangledown)$ ontology, and let Σ be any subset of $\text{sig}_{\mathsf{R}}(\mathcal{O})$. The method is guaranteed to compute a solution of forgetting Σ from $\mathcal{O}$, iff one of the following conditions holds for each $r \in \Sigma$: (i) $\mathcal{O}$ does not include any RBox axioms of the form $\neg \mathcal{S} \sqcup r$ for $\mathcal{S}$ any role symbol; (ii) $\mathcal{O}$ does not include any TBox axioms containing a number restriction of the form $\geq n r.D$; (iii) $\mathcal{O}$ does not include any TBox axioms containing a number restriction of the form $\leq n r.D$ for $n \geq 1$.*

5 Remarks

This work follows an Ackermann approach to forgetting for a number of expressive description logic ontologies [3, 4].

References

[1] W. Ackermann. Untersuchungen $\ddot{u}$ber das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110(1):390–413, 1935.

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

[3] Y. Zhao and R. A. Schmidt. Concept forgetting in $\mathcal{ALCOI}$-ontologies using an Ackermann approach. In *Proc. ISWC'15*, volume 9366 of *LNCS*, pages 587–602. Springer, 2015.

[4] Y. Zhao and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^+(\triangledown, \sqcap)$-ontologies. In *Proc. IJCAI'16*, pages 1345–1352. IJCAI/AAAI Press, 2016.

Forgetting for Abduction in $\mathcal{ALC}$-Ontologies

Warren Del-Pinto Renate Schmidt

University of Manchester, UK

{warren.del-pinto, renate.schmidt}@manchester.ac.uk

Abstract: Abductive reasoning seeks to find explanatory hypotheses for new observations. We outline the use of forgetting to perform abduction in ontologies expressed in the description logic $\mathcal{ALC}$. We then suggest a method for large-scale evaluation of abductive reasoning in this context, and discuss how this will be used to find ways of guiding hypothesis generation by selection of an appropriate forgetting signature.

1 Introduction

Abduction is one of the three main forms of commonsense reasoning alongside deduction and induction, as originally classified by Charles Sanders Peirce [7]. The aim of abductive reasoning is to generate hypotheses to explain a given observation, making use of available background information. For example, given the information "All birds have feathers" and the observation "Gary has feathers", we may obtain the hypothesis "Gary is a bird" via abduction. Abductive reasoning is a key component of tasks in a variety of domains including medical diagnostics, scientific discovery and natural language interpretation.

Here we specifically consider abduction in description logic (DL) ontologies. An ontology is a knowledge base consisting of two main components: a TBox and an ABox. The TBox contains information regarding general concepts and the ABox concerns instances of concepts known as "individuals". In this context, the aim of abduction is to generate and add explanatory hypotheses to a given ontology so that the resulting ontology entails a given observation. These hypotheses should not contradict existing information. This aim can be expressed as follows: given an ontology O and an observation ϕ, find a hypothesis H that can be added to O such that (i) $O \cup H \models \phi$ and (ii) $O \cup H \not\models \bot$ [6].

The need for abductive reasoning within ontologies has been argued by Elsenbroich et al [2], with applications across many domains that utilise DL ontologies such as medical informatics, AI and computational linguistics [3]. This broad applicability has led to a variety of work on performing abduction in DL ontologies. Examples include investigations of the computational complexity of abduction in the less expressive description logic $\mathcal{EL}$ [1] through to methods for abduction in the more expressive logic $\mathcal{ALC}$ [3].

Here, we focus on the use of forgetting [5, 9] for abductive reasoning in $\mathcal{ALC}$.

2 Forgetting

Forgetting is a process by which subsets of symbols in a given ontology can be hidden or removed. This process is defined by a forgetting signature $\mathcal{F}$, where the aim is that all symbols $S_i \in \mathcal{F}$ are removed from the ontology. These symbols should be removed in such a way that all information entailed by the original ontology that is expressible in the remaining symbols is preserved [5]. Forgetting in $\mathcal{ALC}$ is more difficult than standard reasoning based tasks. Potential issues with the resulting ontology include the possibility that it cannot be represented by a finite collection of $\mathcal{ALC}$ axioms [4] and the fact that the size of a forgetting solution can be triple exponential with respect to the original ontology [5].

In this work, we utilise a resolution-based method for logical forgetting presented by Koopmann and Schmidt [5, 4]. Our choice of method was motivated by the following factors: 1) The method can represent non-finite forgetting results using fixpoints [4], which may be required to successfully perform certain abductions. 2) Support is included for both TBoxes and ABoxes [5]. Thus, abductions are not limited to cases involving only general concepts. 3) The size of the forgetting result is constrained to a double exponential bound with respect to the input and the method is guaranteed to terminate [5]. This reduces the potential complexity of the resulting abduction procedure.

These factors lead us to believe that this method for forgetting will provide an efficient and reliable basis for abduction in $\mathcal{ALC}$-ontologies.

3 Forgetting for Abduction

We apply forgetting to the task of abductive reasoning by exploiting the fact that, given an ontology O, an observation ϕ and a hypothesis H: $O \cup H \models \phi$ iff $O \cup \neg\phi \models \neg H$. Thus, we first negate a given observation then add it to the original ontology. We then perform forgetting on this combination with a signature $\mathcal{F}$, which results in a negated hypothesis consisting of symbols not contained within $\mathcal{F}$. By negating this result once more, we obtain an explanatory hypothesis for the observation in terms of a specific set of symbols.

Due to the reliance on a defined signature of symbols $\mathcal{F}$, the abduction process can be directed according to a specific goal. The choice of symbols in $\mathcal{F}$ allows domain experts to utilise their own intuition to seek specific types of hypothesis for a given observation. Alternatively, the selection of forgetting signatures provides a way of guiding the abduction procedure towards hypotheses that are preferable with respect to a general abduction scheme or a domain of

interest. This potential for guiding abduction via forgetting signatures forms part of our research focus.

4 Guiding Hypothesis Generation

An issue faced by logical abduction is that often an impractically large number of possible hypotheses are generated for a given explanation. However, narrowing the space of generated hypothesis is difficult. It is first necessary to ensure that the remaining hypotheses are the most relevant or preferred ones, but this depends upon the nature of the domain, task and specific ontology under consideration.

It is clear that it is necessary to develop guiding principles for abduction to ensure that it is practical for use in real-world ontologies. Using forgetting as described in Section 3 provides a way to achieve this goal: by selecting appropriate forgetting signatures.

In order to understand how these signatures can be used to guide abduction, it is necessary to perform large scale testing of abduction using real ontologies. Doing so will enable the characterization of the process of selecting signatures in terms of different abduction schema, such as those suggested by Stickel [8].

One way to perform many test abductions would be to manually create a testing dataset of observations. This is not feasible, as it would require hand-crafting sufficiently sized sets of non-trivial observations for each specific ontology used. An alternative would be to randomly generate observations based on symbols in the ontology. This is likely to be unreliable, and may result in a large number of trivial observations.

The method we will use in this research is to utilise existing axioms in real ontologies as observations. Given an ontology, we start by selecting and removing a random axiom. This axiom is then negated and added to the original ontology, exploiting the fact that $O \cup \neg\phi \models \neg H$ as outlined in Section 3. Performing abduction with a set of forgetting signatures then results in a set of hypotheses, each of which "explain" the original axiom that was removed. The sets of hypotheses generated using each set of forgetting signatures can then be compared to determine how the choice of signature impacts the nature of the resulting hypotheses.

This approach circumvents the need for manual crafting of non-trivial observations or reliance on random generation, guaranteeing that the observations used for evaluation take the form of actual information that may occur in the domain of interest.

5 Conclusion and Ongoing Work

We outlined a method for abductive reasoning in description logic ontologies which uses forgetting to compute hypotheses, where the hypotheses are expressed in the symbols present in a provided forgetting signature [5]. We then suggested a method for evaluating abduction in DL ontologies using information present in existing axioms.

The next step in this research is to identify key statistics of the resulting hypotheses that can be used to guide our choice of forgetting signature in different scenarios. Preliminary suggestions for these include the average length of the hypotheses (potentially relevant to the "most-specific" and "least-specific" characterisations of abduction outlined by Stickel [8]), the overall number of hypotheses generated and perhaps the percentage of hypotheses that are of the form $H \sqsubseteq C$ where C is the observation provided. Another possible direction would be to utilise these characteristics in conjunction with inductive learning methods, in order to learn how to select effective forgetting signatures for a given objective based on previous abductions.

References

[1] M. Bienvenu. Complexity of abduction in the $\mathcal{EL}$ family of lightweight description logics. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, pages 220–230, 2008.

[2] C. Elsenbroich, O. Kutz, and U. Sattler. A case for abductive reasoning over ontologies. In *Proceedings of OWL: Experiences and Directions*, 2006.

[3] S. Klarman, U. Endriss, and S. Schlobach. Abox abduction in the description logic $\mathcal{ALC}$. In *Journal of Automated Reasoning 46*, pages 43–80, 2011.

[4] P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$ ontologies using fixpoints. In *Proceedings of the 9th International Symposium on Frontiers of Combining Systems*, pages 87–102. volume 8152 of LNCS, Springer, 2013.

[5] P. Koopmann and R. A. Schmidt. Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with aboxes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[6] R. J. Mooney. Integrating abduction and induction in machine learning. In *Abduction and Induction*, pages 181–191, 2000.

[7] C. S. Peirce. Deduction, induction and hypothesis. In *Popular Science Monthly 13*, pages 470–482, 1878.

[8] M. E. Stickel. A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. In *Annals of Mathematics and Artificial Intelligence 4*, pages 89–106, 1991.

[9] Y. Zhao and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-ontologies. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016.

A Framework for Axiom Selection in 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 an approach for axiom selection in large theories, which is based on the abstraction-refinement framework. The proposed approach consists of two approximations: over-approximation and under-approximation and their combination.

1 Introduction

Selection of relevant axioms from large theories is one of the challenging problems in automated reasoning. Several methods have been proposed to approach this problem. Some of them are based on the syntactic or semantic structure of the axioms and conjecture formulas [2, 1]. Other methods use machine learning to take advantage of previous knowledge about proving conjectures [3]. What those methods have in common are two phases of a whole process for proving a conjecture: one is the axiom selection phase, and the other one is reasoning phase. Those phases are performed in a sequential way. First, the axiom selection takes place, then using the selected axioms the reasoning process starts. Our proposed approach, which is based on the abstraction-refinement framework [4] has the purpose of interleaving the axioms selection and reasoning phases. It allows a more dynamic interaction between both phases.

2 Preliminaries

A *concrete axiom* is a fact that is stored in the current used large theory A for reasoning. This A is a set of concrete axioms. A *strengthening abstraction function* α_s for axiom selection is a mapping $\alpha_s : A \to \hat{A}^s$, which assigns a concrete axiom in A to a stronger abstract axiom in $\hat{A}^s$ where $\hat{A}^s$ is a set of abstract axioms and $\hat{A}^s \models A$.

A *stronger abstract axiom* $\hat{a}^s$ is an element of $\hat{A}^s$ such that $\hat{a}^s = \alpha_s(a)$ and $\hat{a}^s \models a$, where a is a concrete axiom; i.e., α_s strengthens the axioms in the sense that they are a more general representation of the concrete axioms. A *concretisation function* γ_s is a mapping $\gamma_s : \hat{A}^s \to A$.

A *weakening abstraction function* α_w is a mapping $\alpha_w : A \to \hat{A}^w$, which assigns a concrete axioms in A to a weaker abstract axiom in $\hat{A}^w$ where $\hat{A}^w$ is a set of abstract axioms such that $A \models \hat{A}^w$. A *weaker abstract axiom* $\hat{a}^w$ is an element of $\hat{A}^w$ such that $\hat{a}^w = \alpha_w(a)$ and $a \models \hat{a}^w$ where a is a concrete axiom.

Abstraction refinement is a process to approximate an abstract representation of axioms $\hat{A}$ to their concrete representation A. *Weakening abstraction refinement* is a process to construct $\hat{A}^{s\prime}$, which is a closer representation of A from $\hat{A}^s$ such that $\hat{A}^s \models \hat{A}^{s\prime}$ and $\hat{A}^{s\prime} \models A$; i.e., this refinement weakens the abstract axioms in $\hat{A}^s$. However those weak abstract axioms still stronger than the concrete axioms. *Strengthening abstraction refinement* constructs $\hat{A}^{w\prime}$ which is an approximation to the set of concrete axioms A, such that $A \models \hat{A}^{w\prime}$ and $\hat{A}^{w\prime} \models \hat{A}^w$.

ATP_S is used to denote an instance of automated theorem prover which is sound and could be complete or not. On the other hand, ATP_C is used to make reference to an automated theorem prover, which is complete but not necessary sound [6]. The purpose of this ATP is to disprove conjectures more efficiently than a sound and complete ATP.

3 Over-Approximation

This procedure starts by applying the strengthening abstract function α_s to A, to obtain an abstract representation of axioms $\hat{A}^s$, $\hat{A}^s = \alpha_s(A)$. Utilising the set $\hat{A}^s$, the procedure tries to prove the conjecture C using an ATP_C. If the ATP_C disproves the conjecture, the process finishes and responds that the conjecture has been disproved. If ATP_C proves the conjecture or the time limit is reached, the procedure uses the abstract axioms involved in the proof $\hat{A}^s_p$ to retrieve their concrete axioms in A by applying the function γ_s over $\hat{A}^s_p$. The retrieved concrete axioms form a new subset A_p, where $A_p = \gamma_s(\hat{A}^s_p)$. With the new set A_p, the procedure tries again to prove the conjecture using this time an ATP_S. If the ATP_S proves the conjecture, the process stops and provides the proof. Otherwise, if the time limit is reached or the conjecture is disproved, the set of axioms $\hat{A}^s$ is refined using the weakening abstraction refinement. The procedure is repeated utilising the refined set of abstract axioms. This loop finishes when the conjecture is proved or disproved or the time limit of the whole procedure is reached. This approach is expressed in algorithm 1.

3.1 Strengthening Abstraction Function

For the over-approximation, one possible candidate as abstraction function is changing ground terms in the concrete axioms' formulas to variables. This transformation must have the property that the obtained abstract axioms entail the concrete axioms, $\hat{a}^s \models a$.

3.2 Weakening Abstraction Refinement

The abstraction refinement weakens the abstracted axioms in $\hat{A}^s$. In the case of using a strengthening abstraction function that changes ground terms for variables. Those vari-

Input: A set of axioms A and a conjecture C
$\hat{A}^s = \alpha_s(A)$
repeat
  if $ATP_C(\hat{A}^s, C)$ *returns SAT* **then** the conjecture is disproved
  else
    $\hat{A}^s_p = get\_axioms\_from\_proof()$
    $A_p = \gamma_s(\hat{A}^s_p)$
    if $ATP(A_p, C)$ *returns UNSAT* **then** the conjecture is proved
    else $\hat{A}^s := refine\_abstraction(\hat{A}^s)$
  end
until *the conjecture has been proved or disproved or the time limit has been reached*

Algorithm 1: Over-approximation

Input: A set of axioms A and a conjecture C
$\hat{A}^w = \alpha_w(A)$
repeat
  if $ATP_S(\hat{A}^w, C)$ *returns UNSAT* **then** the conjecture is proved
  else `/* refinement of` $\hat{A}^w$ `*/`
    $I \models \hat{A}^w \wedge \neg C$
    find a set $\breve{A}$ such that $I \not\models \breve{A}$
    if $\breve{A} = \emptyset$ **then** the conjecture is disproved
    else $\hat{A}^w := \hat{A}^w \cup \breve{A}$
  end
until *the conjecture has been proved or disproved or the time limit has been reached*

Algorithm 2: Under-approximation

ables can be returned to the ground term that they substitute in the concrete axiom. Another manner to refine the set of abstract axioms is by adding concrete axioms to $\hat{A}^s$ and remove their abstract entities.

4 Under-Approximation

The process starts by applying the weakening 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. 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 a proof is reached. The procedure is shown in algorithm 2.

4.1 Weakening Abstraction Function

In the case of under-approximation, an adaptation of Inst-Gen framework [5] can be used as abstraction function. This abstraction function generates ground instances of the concrete axioms. Another abstraction function can be to remove concrete axioms from the theory.

4.2 Strengthening Abstraction Refinement

One way to refine the set of abstract axioms is by adding to it concrete axioms $\breve{A}$ that turn the model I, which is obtained form ATP_S, into a countermodel, $\breve{A} = \{\breve{a} \mid \breve{a} \in A, I \not\models \breve{a}\}$. Another way to refine the abstract set of axioms is by generating a set of ground instances of axioms $(A\sigma)$ such that $I \not\models A\sigma$, $\breve{A} := A\sigma$.

5 Combined-Approximation

As a future work, we are working in a way to combine the two approximations mentioned before. This combination has the purpose of converging to a proof more rapidly by over and under approximating. One possible approach to combine them is using the over-approximation as a black box. This black box can be used as the ATP_S in the under-approximation process.

References

[1] Geoff Sutcliffe and Yury Puzis. SRASS - A Semantic Relevance Axiom Selection System. *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, 2007*, 4603:295–310, 2007.

[2] Kryštof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. *LNCS*, 6803 LNAI:299–314, 2011.

[3] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskočil. MaLARea SG1 - Machine learner for automated reasoning with semantic guidance. *LNCS*, 5195 LNAI:441–456, 2008.

[4] Edmund M Clarke, Orna Grumberg, and David E Long. Model checking and abstraction. *ACM transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1512–1542, 1994.

[5] Konstantin Korovin. Inst-Gen – A Modular Approach to Instantiation-Based Automated Reasoning. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics: Essays in Memory of Harald Ganzinger*, pages 239–270. Springer, 2013.

[6] Christopher Lynch. Unsound theorem proving. In *International Workshop on Computer Science Logic*, pages 473–487. Springer, 2004.

Mind the Gap: Metric Temporal Logic Translations

Ullrich Hustadt[1] Clare Dixon[1] Ana Ozaki[2]

[1] Dept. of Computer Science, Univ. of Liverpool, {CLDixon, U.Hustadt}@liverpool.ac.uk
[2] Faculty of Computer Science, TU Dresden, Ana.Ozaki@tu-dresden.de

Abstract: We consider the satisfiability problem in Metric Temporal Logic (MTL) with the 'standard semantics' in which a weakly monotonic function allows multiple states to be mapped to the same time time. We present two translations from MTL to Linear Temporal Logic (LTL) for this semantics.

1 Introduction

Metric Temporal Logic (MTL) is an extension of linear time temporal logic that allows us to impose constraints on the interval in which a formula is true or becomes true. For example, the formula $\Box_{[3,\infty)}p$ states that p holds in all states that occur at least 3 moments from now, while $\Diamond_{[2,5]}\neg p$ expresses that $\neg p$ will become true in a state that occurs in the time interval $[2, 5]$. MTL has been used to formalise vehicle routing problems, monitoring of algorithms and cyber-physical systems, among others. In [3] we have considered MTL with point-wise discrete semantics where each state is mapped to a time point on a time line isomorphic to the natural numbers by a strictly monotonic function. For this semantics we defined two different satisfiability preserving translations from MTL to LTL. In the current paper we modify these two translations for a semantic variant of MTL in which states are mapped to time points by a weakly monotonic function, that is, several states can be mapped to the same time point. This *non-strict semantics* is the 'standard semantics' for MTL over discrete time [1]. The modified translations are again polynomial in the size of the MTL formula and the largest constant occurring in an interval (but exponential in the size of the MTL formula due to the binary encoding of the constants). As the satisfiability problem for LTL is PSPACE [5], our translations again preserve the EXPSPACE complexity of the MTL satisfiability problem [1].

2 Metric Temporal Logic Translations

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

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

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

We denote by $\bigcirc^c$ a sequence of c next operators. Further connectives can be defined as usual: $p \vee \neg p \equiv \mathbf{true}$, $\mathbf{true} \equiv \neg(\mathbf{false})$, $\mathbf{true}\mathcal{U}\varphi \equiv \Diamond\varphi$ and $\Diamond\varphi \equiv \neg\Box\neg\varphi$. MTL formulae are constructed in a way similar to LTL, but temporal operators are now bounded by an interval I with natural numbers as end-points or ∞ on the right side. Well formed formulae in MTL are formed according to the rule: $\varphi, \psi := p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \bigcirc_I\varphi \mid (\varphi \mathcal{U}_I \psi)$ where $p \in \mathcal{P}$.

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

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

We omit propositional cases, which are as in LTL. Further connectives can be defined as usual: $\mathbf{true}\mathcal{U}_I\varphi \equiv \Diamond_I\varphi$ and $\Diamond_I\varphi \equiv \neg\Box_I\neg\varphi$. To transform an MTL formula into Negation Normal Form, one uses the constrained dual until $\tilde{\mathcal{U}}_I$ operator [4], defined as $(\varphi\tilde{\mathcal{U}}_I\psi) \equiv \neg(\neg\varphi\mathcal{U}_I\neg\psi)$. An MTL formula φ is in *Negation Normal Form (NNF)* iff the negation operator ($\neg$) occurs only in front of propositional variables. An MTL formula φ is in *Flat Normal Form (FNF)* iff it is of the form $p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \to \psi_i)$ where p_0, p_i are propositional variables or $\mathbf{true}$ and ψ_i is either a formula of propositional logic or it is of the form $\bigcirc_I\psi_1$, $\psi_1\mathcal{U}_I\psi_2$ or $\psi_1\tilde{\mathcal{U}}_I\psi_2$ where ψ_1, ψ_2 are formulae of propositional logic. The transformations into NNF and FNF are satisfiability preserving and can be performed in polynomial time. From

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

Table 1: Gap Translation from MTL to LTL, where α, β are propositional logic formulae, $c_1, c_2 > 0$.

MTL	LTL Time Difference Translation
$(\bigcirc_{[k_1,\infty)}\alpha)^\sharp$	$\bigcirc((\bigvee_{k_1\leq i\leq C}\delta_i^-)\wedge\alpha)$
$(\bigcirc_{[k_1,k_2]}\alpha)^\sharp$	$\bigcirc((\bigvee_{k_1\leq i\leq k_2}\delta_i^-)\wedge\alpha)$
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$\alpha\wedge\bigcirc\bigvee_{1\leq i\leq c_1}((\alpha\wedge\delta_0^-)\mathcal{U}^i(\neg\delta_0^-\wedge\alpha),(\neg\delta_0^-\wedge(\bigvee_{c_1\leq j\leq c_1+C}s_j^i)\wedge\alpha\mathcal{U}\beta))$
$(\alpha\mathcal{U}_{[0,\infty)}\beta)^\sharp$	$\alpha\mathcal{U}\beta$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\alpha\wedge\bigcirc\bigvee_{1\leq i\leq c_2}((\alpha\wedge\delta_0^-)\mathcal{U}^i(\neg\delta_0^-\wedge\alpha),(\neg\delta_0^-\wedge(\bigvee_{c_1\leq j\leq c_2}s_j^i)\wedge(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp))$
$(\alpha\mathcal{U}_{[0,c_2]}\beta)^\sharp$	$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp\vee(\alpha\mathcal{U}_{[1,c_2]}\beta)^\sharp$
$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp$	$\beta\vee(\alpha\wedge\bigcirc((\alpha\wedge\delta_0^-)\mathcal{U}(\beta\wedge\delta_0^-)))$

Table 2: LTL Time Difference Translation from MTL to LTL where α,β are propositional logic formulae, $k_1,k_2\geq 0$, $c_1,c_2>0$, and $\phi\mathcal{U}^n\gamma,\chi$, $n>1$, is a shorthand for $\phi\mathcal{U}(\gamma\wedge\bigcirc(\phi\mathcal{U}^{n-1}\gamma,\chi))$ and $\phi\mathcal{U}^1\gamma,\chi=\phi\mathcal{U}\chi$.

now on assume that our MTL formulae are in NNF and FNF.

Gap Translation We translate MTL formulae for discrete time models into LTL using two new propositional symbols gap and $same$. $\neg gap$ is true in those states σ'_j of σ' such that there is $i\in\mathbb{N}$ with $\tau(i)=j$ and gap is true in all other states of σ'. $same$ is true exactly in those states σ'_j of σ' such that there is $i\in\mathbb{N}$ with $\tau(i)=j$ and, for $i>0$, $\tau(i)=\tau(i-1)$.

The main distinction between the translation in Table 1 and the one in [3] is that here we use nested LTL until operators to make 'progress' in our encoding of the time line whenever we encounter a state with $\neg same$.

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

Time Difference Translation As in [2, 3], it is sufficient to consider timed state sequences where the time difference from a state to its previous state is bounded by C, where $C-1$ is the greatest number occurring in an interval in an MTL formula φ or 1, if none occur. Then, we can encode time differences with a set $\Pi_\delta=\{\delta_i^-\mid 0\leq i\leq C\}$ of propositional variables where each δ_i^- represents a time difference of i w.r.t. the previous state. We also encode variables of the form s_m^n with the meaning that 'the sum of the time differences to the current state from the last n states with a non-zero time difference to the previous one is m'. For our translation, we only need to define these variables up to sums bounded by $2\cdot C$.

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

Let S'_C be the conjunction of the following:

1. $\bigcirc\Box\oplus_{=1}\Pi_\delta$, for $\Pi_\delta=\{\delta_k^-\mid 0\leq k\leq C\}$;
2. $\Box(\delta_k^-\leftrightarrow s_k^1)$, for $1\leq k\leq C$;
3. $\Box\oplus_{\leq 1}\Pi^i$, for $1\leq i\leq 2\cdot C$ and $\Pi^i=\{s_j^i\mid i\leq j\leq 2\cdot C\}$;
4. $\Box((\bigcirc s_k^1\wedge s_l^j)\rightarrow\bigcirc s^{j+1}_{\mathsf{min}(l+k,2\cdot C)})$, for $1<j+1\leq l+k\leq 2\cdot C$.
5. $\Box((\bigcirc\delta_0^-\wedge s_l^j)\rightarrow\bigcirc s_l^j)$, for $1\leq j\leq l\leq 2\cdot C$.

Note that the main difference to S_C in [3] is in Point 5 where we now propagate the variables of the form s_m^n to the next state if the time difference is zero. In the translation itself, shown in Table 2, the main distinction occurs in the translation of the 'until' formulae where we have to nest LTL until operators so that we can count n states with time difference greater than zero and then check whether a variable of the form s_m^n holds.

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

3 Conclusion

We presented two translations from MTL with non-strict semantics to LTL. For their implementation and evaluation see `http://cgi.csc.liv.ac.uk/~ullrich/MTL/`.

References

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

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

[3] Clare Dixon, Ullrich Hustadt, and Ana Ozaki. Metric temporal logic translations over naturals. In *Proc. ARW 2016*, pages 13–14. Dept. of Computer Science, Univ. of Liverpool, 2016.

[4] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Proc. FORMATS 2008*, volume 5215 of *LNCS*, pages 1–13. Springer, 2008.

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

Verifying a Security Protocol for Secure Service Migration in Commercial Cloud Environments

Gayathri Karthick[1] Florian Kammueller[2] Glenford Mapp[3] Mahdi Aiash[4]

School of Science and Technology, Middlesex University,London

Gk419@live.mdx.ac.uk,F.Kammueller,G.Mapp,M.Aiash@mdx.ac.uk

Abstract: Mobile users are making more demands of networks. They want to run applications such as network steaming of audio and video as well as immersive gaming that demand high Qualities-of-Service. One way to address this problem is by using mobile services that move around as users move around. This ensures that low latencies are maintained between the client and the server resulting in a better Quality-of-Experience. In addition, the advent of virtual machine technology for example, VMware, and container technology, such as Docker, have made the migration of services between different Cloud Systems possible. Furthermore, a Service Oriented Architecture that supports service migration in Cloud environments has been proposed. Though there are many things in place to support 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. In particular, it is important that servers do not end up being hosted on unsafe Cloud infrastructure and also Clouds do not end up hosting malicious servers. This paper proposes a new security protocol to address this issue. The protocol is specified and tested using AVISPA. The initial results indicate that the protocol is safe and therefore can be used in practical systems.

1 Introduction

Cloud computing is, therefore, facilitating the migration of data and services. These services are called mobile services and will be used to support mobile users as they move around.

One aspect of the research on mobile services that has been inadequate is support for security [1]. 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. This paper attempts to address these issue by providing a new security protocol for secure service migration [2].

2 Specify the Security Protocol for Mobile Services

In this section, we look at protocols for secure service migration [3].

A Registry or Certificate Authority (CA) is a trusted party that issues signed electronic documents to verify whether the party is a valid entity on the Internet. Electronic documents used as digital certificates are important in public key encryption(PKE), usually, these certificates include the owner's name, public key, the expiration date of the digital certificate, location of the owner and other data of the owner.In addition, the registry can talk securely to both services and Clouds using public key encryption. [3].

- `\The Server{}`: The server is identified by a unique Service_Id. Server_ID, Type of service (TOS), Public Key (PKS).
- `\Cloud Facilties{}`: Cloud A (CA) = Cloud_ID_A, TOS = Cloud, PKA, Resources. Cloud B (CB) = Cloud_ID_B, TOS = Cloud, PKB, Resources.
- `\The Registry{}`: The Registry is the last key component and is used to verify the identities of all servers on the network.
- `\Nonces denoted by N{}`: Nonces are randomly generated numbers which are unforgeable and are used as session tokens, ensuring that requests cannot be repaid by unauthorised personnel.
- `\Timestamps{}`: Timestamps are used explicitly with migration requests and responses..

The full explanation of this second migration is given below:

- **Step 1** The service on Cloud A receives advertisements from Cloud B advertising Cloud's B resources.
- **Step 2** The Server checks the validity of Cloud B.
- **Step 3** Registry authenticates Cloud B
- **Step 4** The server on Cloud A sends a request to migrate to Cloud B
- **Step 5** Cloud B sends a request to make sure that the server, S, is a valid service:
- **Step 6** The Registry validates Cloud B
- **Step 7** Cloud B signals to The Service on Cloud A that it is OK to migrate the service to Cloud B:
- **Step 8** The Service signals to Cloud B to migrate the service:

Algorithm 1 Security Protocol for Migration between Cloud A to Cloud B

1: CB → SA: Advertisement ($CloudID\_B$, TOS, Resources, PKB)
2: SA → R: Verify Identity ($CloudID\_B$, TOS, PKB, Resource, Server_ID, PKS) PKR
3: R → SA: Message: YES ($CloudID\_B$, TOS, PKB, Valid Resources) PKS
4: SA → CB: Migration Request + (Server_ID, TOS, TA, Req.Resource, PKS, NA) PKB
5: CB → R: Verify Identity (Server_ID, PKS, TOS, Cloud ID_B, PKB) PKR
6: R → CB: Message: Yes (Server_ID, TOS, Valid Service) PKB
7: CB → SA: Migration Response + (Cloud ID_B, TOS= Cloud), TB, Resources Granted, NA, NB) PKS
8: SA → CB: Transfer (Migration) + (Server_ID, $CloudID\_B$, Services, NB) PKB
9: CB → SA: $Transfer\_Ack$ (Server_ID, $CloudID\_B$, NA, Tcomp) PKS
10: SB → R: Transfer-Complete + (Server_ID, $CloudID\_B$, TOS, TA, TB, Tcomp) PKR

- **Step 9** Cloud B signals to the server on Cloud A that the migration is complete: Hence, **New Location: (SA→SB)** The service is now started on Cloud B.
- **Step 10** The service on Cloud B signals to the Registry that the migration has been completed:

Figure 1 shows the steps for Cloud to Cloud migration of services.

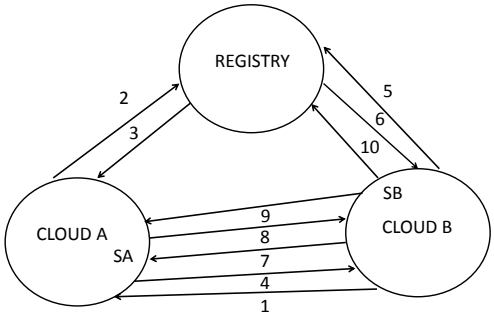


Figure 1: Migration from Cloud A to Cloud B

2.1 Using AVISPA

In this section, AVISPA is used to analyse the protocol specified in the previous section. AVISPA provides automated validation of Internet security protocols and applications.In simple, we modelled in HLPSL and analysed with Avispa tool.It supports for multiple back-end tools for the Analysis of Security Protocols (TA4SP) and it verifies the security properties for a bounded number of sessions.The first back end tool is called OFMC, and the second back end tool is called ATSE, also indicates that the protocol is SAFE. Now, both of the protocol results SAFE, so that the probability of the expected result is accomplished.

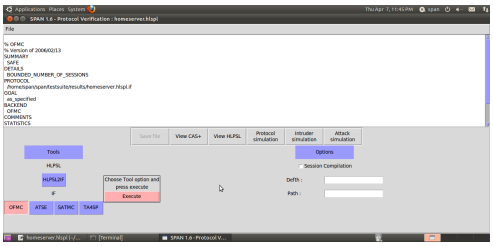


Figure 2: OFMC: Cloud A to Cloud B

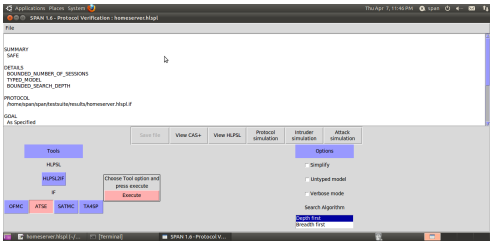


Figure 3: ATSE: Cloud A to Cloud B

3 Conclusion

The protocol is safe under normal operation; the present protocol critically prevents impersonation attacks either by rogue cloud infrastructure hoping to sneer valid services or by malicious servers wanting inflict damage on Cloud infrastructure. We are exploring how this protocol and be enhanced to prevent intruder and man-in-the-middle attacks.

Acknowledgements

References

[1] Mapp G.E Aiash M and Gemikonakli O. Secure Live Migration: Issues and Solutions. In *Proceedings of the Conference on Advanced Information Networking and Applications, AINA 2014, Victoria, Canada*, May 2014.

[2] Phil Honer. Cloud Computing Security Requirements and Solutions: a Systemat6ic Literature Review, 2013.

[3] G Mapp, M Aiash, A Lasebae, and R Phan. Security Models for Heterogeneous Networking. In *International Conference on Secuity and Cryptography (SECRYPT 2010) Athens, Greece*, July 2010.

Diagrammatic Reasoning for Ontology Debugging

Zohreh Shams[1] Mateja Jamnik[1] Gem Stapleton[2] Yuri Sato[2]

[1] Computer Laboratory, University of Cambridge, Cambridge, UK
{zohreh.shams,mateja.jamnik}@cl.cam.ac.uk
[2] Visual Modelling Group, University of Brighton, Brighton, UK
{g.e.stapleton, y.sato}@brighton.ac.uk

Abstract: Ontologies are notoriously hard to define, express and reason about. Many tools have been developed to ease the ontology debugging and reasoning, however they often lack accessibility and formalisation. A visual representation language, *concept diagrams*, was developed for expressing ontologies, which has been empirically proven to be cognitively more accessible to ontology users. Here, we answer the question of "How can concept diagrams be used to reason about inconsistencies and incoherence of ontologies?"'.

1 Introduction

Ontologies are sets of statements that represent properties of individuals, classes and properties, typically expressed using symbolic notations such as description logics (DL) [2] and OWL [1]. Although ontologies are widely used for knowledge representation in domains involving diverse stakeholders, the languages they are expressed in are often inaccessible to those unfamiliar with mathematical notations. This shortcoming has given rise to several visualisation facilities and notation that aid ontology comprehension. However, these notations are either informal or do not fully exploit the potential of formal diagrammatic notations. The design of concept diagrams [5] for expressing ontologies is based on cognitive theories of what makes a diagrammatic notation accessible [3]. Concept diagrams are extensions of Euler diagrams and, in addition to closed curves for set representation, they use dots (spiders) and arrows for individuals and properties, respectively.

Similar to traditional logical systems, concept diagrams are equipped with inference rules which are used for specifying, reasoning and evaluating ontologies. Evaluating ontologies involves debugging them of inconsistencies and incoherence before they can be published. These, so-called *antipatterns*, capture the unintended model-instances of an ontology. An inconsistent ontology is one that cannot have any model and, so, entails anything [4], whereas an incoherent ontology is one that entails an unsatisfiable (i.e., empty) class or property.

Empirical evidence proves that for incoherence checking, novice users perform significantly better with concept diagrams than with OWL [1] or DL [2]. Based on these results, we formalise the use of concept diagrams for reasoning about inconsistencies and incoherence in ontologies by defining inference rules for them.

2 Concept Diagrams

In this section, we informally define what concept diagrams are through an example. Please see [6] for formal syntax and semantics of concept diagrams. The concept diagram in Fig. 1 has the following syntax and semantic interpretation:

- One dot – called a spider – which represents a named individual, s;
- Two *boundary rectangles* (represented by $\square$) each of which represents the universal set.
- Seven curves, representing seven sets, five of which have labels A to E. The two curves without labels represent *anonymous sets*. The spatial relationships between curves and spiders convey semantics. For examples, the syntax within the LHS rectangle says that the individual s is in the set B; B is a subset of A; the sets A and C are disjoint.
- Shading (e.g., inside curve B) which is used to place upper bounds on set cardinality: in a shaded region, all elements are represented by spiders. Thus, the only element in B is s.
- Two arrows, one of which is *solid* and the other one is *dashed* and annotated with ≥ 1. Arrows are used to convey semantics about binary relations, using their sources and targets. The solid arrow asserts that things in A are only related to things in C under op_1. The unlabelled curve, say c_1, targeted by op_1 represents *the* set of things to which elements of A are related under op_1. The dashed arrow is sourced on c_1 and again targets an unlabelled curve. This unlabelled curve, say c_2, represents a subset of D to which elements of c_1 are related under op_2. The inclusion of c_2 inside the curve labelled D expresses that c_2 is subset of D. The dashed arrow's annotation, ≥ 1, places a constraint on the set c_1: all elements of c_1 must be related to at least one element of c_2 under op_2.

3 Concept Diagrams for Debugging Ontologies

Debugging ontologies of incoherence and inconsistencies requires detecting these characteristics first. We first define what it means for an ontology to be incoherent and then give example of inference rules that allow detecting incoherence. We have similar inference rules devised for inconsistency that are not presented here due to space limitation.

To prove that ontology o is incoherent, we have to show that a class or an object property is unsatisfiable. When using a set of concept diagrams, $\mathcal{D}$, to define o, the task

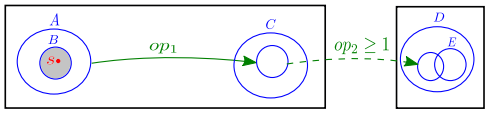
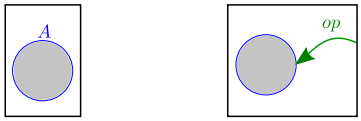


Figure 1: A concept diagram.



(a) A is empty. (b) op is empty.

Figure 2: Incoherence in concept diagrams.

is thus to prove a lemma of the form: (i) a curve labelled A necessarily represents an empty class, or (ii) an arrow labelled op necessarily represents an empty object property.

A lemma of type (i) is proved if, carrying out the proof visually, we derive a diagram in Fig. 2a: an entirely shaded region with no spiders represents the empty set in any model. Type (ii) lemmas are proved if the proof derives a diagram in Fig. 2b, in which the target of the arrow is entirely shaded with no spiders: this target represents the empty set, implying the image of op is empty, thus op is an empty relation.

The space does not allow showcasing proofs of lemmas that establish incoherence and inconsistency using concept diagrams. However, our approach to designing inference rules is driven by the requirements of the proof, rather than in isolation from the proof. We believe that proof driven inference rules give rise to more natural proofs. In contrast, the established common approach to designing inference rules in logic is primarily driven by the requirements of the theoretical properties (e.g., soundness and completeness) of the rules. Fig. 3 shows two examples on inference rules. The top inference rule in Fig. 3 spots an incoherence by showing that A is unsatisfiable. We have that the universal image of op is restricted to B, while there is set A such that the partial image of A under op includes C. However, C and B are disjoint. Since the universal image of op is restricted to B, the image of A under p cannot be outside B, which is clearly not the case here. So A is empty. The bottom inference rule shows that object property op is empty, because the first premise displays the image of op as a subset of intersection of B and C, while the second premise defines B and C as disjoint.

4 Results and Future Work

The set of inference rules we have designed so far are proven sound and are mainly derived from proofs that aim at establishing that a set of ontology axioms are inconsistent or incoherent. Compleness is a desiarble property that we leave for future work (non-trivial). We conjecture that concept diagrams, as defined here, correspond to a fragment of second-order logic with one and two place predicates. One-place and two-place predicates arise due to the use of labelled curves and arrows respectively. Second-order (existential) quantification occurs through the use of unlabelled curves. Although concept diagrams do not contain quantifiers in their syntax, an equivalent fragment of SOL would need to do so. For instance, two non-overlapping labelled curves, A and B say, give rise to (first-order) universal quantification and express $\forall x\, \neg(A(x) \wedge B(x))$. Due to the restricted way in which second-order quantification arises, finding a complete set of inference rules should be possible.

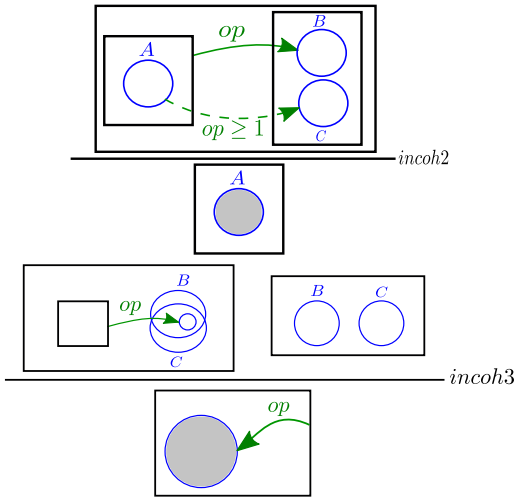


Figure 3: Incoherence examples.

Moreover, in the future we will use concept diagrams and inference rules we have devised for them in building the first mechanised reasoning system for concept diagrams and reasoning about antipatterns in ontology engineering.

Acknowledgements

This research was funded by a Leverhulme Trust Research Project Grant (RPG-2016-082) for the project entitled Accessible Reasoning with Diagrams.

References

[1] The OWL2 web ontology language. https://www.w3.org/TR/owl2-direct-semantics/, Dec. 2016.

[2] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on Ontologies*, pages 21–43. 2009.

[3] Peter Chapman, Gem Stapleton, John Howse, and Ian Oliver. Deriving sound inference rules for concept diagrams. In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2011*, pages 87–94. IEEE, 2011.

[4] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining inconsistencies in OWL ontologies. In *Scalable Uncertainty Management, Third International Conference, SUM 2009*, volume 5785 of *Lecture Notes in Computer Science*, pages 124–137. Springer, 2009.

[5] J. Howse, G. Stapleton, K. Taylor, and P. Chapman. Visualizing ontologies: A case study. In *International Semantic Web Conference*, pages 257–272. Springer, 2011.

[6] Gem Stapleton, John Howse, Peter Chapman, Aidan Delaney, Jim Burton, and Ian Oliver. Formalizing Concept Diagrams. In *Visual Languages and Computing*, pages 182–187. Knowledge Systems Institute, 2013.