SOQE 2017

Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017

Patrick Koopmann · Sebastian Rudolph ·
Renate A. Schmidt · Christoph Wernhard (Eds.)

Editors
Patrick Koopmann
Technische Universität Dresden
Dresden
Germany

Sebastian Rudolph
Technische Universität Dresden
Dresden
Germany

Renate A. Schmidt
The University of Manchester
Manchester
UK

Christoph Wernhard
Technische Universität Dresden
Dresden
Germany

Originally published online by CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073).

Preface

Second-order quantifier elimination (SOQE) is the problem of computing from a given logic formula involving quantifiers upon second-order objects such as predicates an equivalent first-order formula, or, in other words, an equivalent formula in which these quantified second-order objects do no longer occur. The problem can be studied for various logics, including classical propositional and first-order logic as well as modal and description logics. In slight variations SOQE is also known as forgetting, projection, predicate elimination and uniform interpolation.

SOQE bears strong relationships to Craig interpolation, definability and computation of definientia, the notion of conservative theory extension, abduction and notions of weakest sufficient and strongest necessary condition, to correspondence theory relationships, as well as to generalizations of Boolean unification to predicate logic.

Various important research subjects of current interest are based on SOQE and these related notions, as is reflected in the topics addressed in workshop program: In the area of knowledge representation they include forgetting, uniform interpolation and abduction in description logics, modularization, reuse, versioning and summarization of ontologies, forgetting in variants of logic programming that are relevant for ontology reasoning, and query reformulation on the basis of interpolation. The *unified correspondence* research program, which recently emerged from the study of algorithmic correspondence and canonicity, now by itself reaches into further areas such as proof theory. SOQE has applications in the verification of distributed systems. The investigation of SOQE with respect to specific fragments of first-order logic is – much less researched than decidability – an area with open challenges, where, however, ways of progress can be indicated. For Boolean unification on the basis of predicate logic, like SOQE an operation with formulas as output, various relationships to SOQE can be shown. In the *Algebra of Logic* program of the 19th century SOQE was identified as an important operation, which makes the study of historical aspects and the passage of SOQE to modern logic particularly interesting. Special forms of SOQE are essential components of state-of-the-art SAT-solvers. SOQE provides an exemplary framework for studying the dichotomy of expressivity versus complexity.

The first *Workshop on Second-order Quantifier Elimination and Related Topics (SOQE 2017)* was held in the International Center for Computational Logic (ICCL) at Technische Universität Dresden in Dresden, Germany, during 6-8 December 2017. The workshop aimed at bringing together researchers working on SOQE and related topics in a dedicated event. Its program includes nine invited talks, two tutorials and nine research talks, acquired with an open call for submissions of original research, adaptions of relevant research published elsewhere and discussions of research in progress.

We would like to thank all those involved for their enthusiasm and high-quality contributions, in particular, the invited speakers, the authors of tutorials and research presentations, the members of the program committee, and Romy Thieme who assisted with the local organization.

The organizational framework for the workshop was provided by the ICCL at TU Dresden, an interdisciplinary center of competence in research and teaching in the field of Computational Logic, with special emphasis on algebra, logic, and formal methods in computer science, founded in 2003. The workshop was made possible through funding by *Deutsche Forschungsgemeinschaft (DFG)* for the project *The Second-Order Approach and its Application to View-Based Query Processing* (grant WE 5641/1-1).

December 2017

Patrick Koopmann
Sebastian Rudolph
Renate A. Schmidt
Christoph Wernhard

Organization

SOQE 2017 was organized by the International Center for Computational Logic of Technische Universität Dresden, Germany.

Program Chairs

Patrick Koopmann	Technische Universität Dresden, Germany
Sebastian Rudolph	Technische Universität Dresden, Germany
Renate A. Schmidt	The University of Manchester, UK
Christoph Wernhard	Technische Universität Dresden, Germany

Program Commitee

James Delgrande	Simon Fraser University, Canada
Andreas Herzig	IRIT, France
Stefan Hetzl	Technische Universität Wien, Austria
Patrick Koopmann	Technische Universität Dresden, Germany
Andreas Nonnengart	DFKI, Germany
Hans Jürgen Ohlbach	Ludwig-Maximilians-Universität München, Germany
Alessandra Palmigiano	Technische Universiteit Delft, Netherlands
Sebastian Rudolph	Technische Universität Dresden, Germany
Renate A. Schmidt	The University of Manchester, UK
Viorica Sofronie-Stokkermans	Universität Koblenz-Landau, Germany
David Toman	University of Waterloo, Canada
Dirk Walther	Fraunhofer IVI, Germany
Christoph Wernhard	Technische Universität Dresden, Germany
Frank Wolter	University of Liverpool, UK
Richard Zach	University of Calgary, Canada

Organization Chair

Christoph Wernhard	Technische Universität Dresden, Germany

Funding

The workshop was supported by Deutsche Forschungsgemeinschaft (DFG) with grant WE 5641/1-1.

Contents

Abstracts of Invited Talks

Abstracts of Tutorials

Research Presentations

Interpolation for Query Reformulation (Abstract of Invited Talk)

Michael Benedikt

Department of Computer Science, University of Oxford, UK

In this talk I will explain query reformulation problems – given a formula Q and a background theory Σ, the goal is to translate Q, either into another formula or a direct implementation, such that the translation is equivalent to Q according to Σ, and such that the translation satisfies additional interface restrictions – e.g. restrictions on the vocabulary. I will review the approach to solving these problems via interpolation which has been investigated by several groups of researchers over the last few years, presenting the properties of a proof system and interpolation algorithm we might desire for the application to reformulation. I will then give a quick tour of some proof methods and associated interpolation methods proposed in the past, with some tentative remarks about how these stack up against the requirements.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

Algorithmic Correspondence and Canonicity for Non-Classical Logics (Abstract of Invited Talk)

Willem Conradie

University of Johannesburg, South Africa

In this talk I give an overview of the work on algorithmic approaches to correspondence and canonicity for non-classical logics in which I have been involved over the past decade, and which has evolved into the research programme now being called 'unified correspondence'. In the first part I will discuss work that was a collaboration with Valentin Goranko and Dimiter Vakarelov, while the second part details work with Alessandra Palmigiano and a number of other collaborators.

Sahlqvist Theory. As is well known, every modal formula defines a second-order property of Kripke frames. Sahlqvist's famous theorem [31] gives a syntactic definition of a class of modal formulas, the *Sahlqvist formulas*, each of which defines an first-order class of frames and is canonical. Over the years, many extensions, variations and analogues of this result have appeared, including alternative proofs in e.g. [32], generalizations to arbitrary modal signatures [30], variations of the correspondence language [28, 1], Sahlqvist-type results for hybrid logics [4], various substructural logics [26, 18, 21], mu-calculus [2], and enlargements of the Sahlqvist class to e.g. the *inductive* formulas of [24], to mention but a few. Another natural approach to the modal correspondence problem is to apply second-order quantifier elimination algorithms to the frame-translations of modal formulas. It has been shown, for example, that the algorithms SCAN [20] and DLS [17] both succeed in computing first-order equivalents for all Sahqvist formulas [23, 5].

SQEMA. SQEMA is an acronym for Second-Order Quantifier elimination in Modal logic using Ackernann's Lemma. As this would suggest, SQEMA is related to DLS in its use of the Ackermann lemma as the engine for eliminating predicate variables and of equivalence preserving rewrite-rules to prepare formulas for the application of the former. A major difference, however, is that SQEMA is specifically targeted at propositional modal logics which it does not translate into second-order logic, but manipulates directly. However, modal logic itself cannot express the required equivalences, formulated as rewrite rules, so an enriched hybrid language with inverse (temporal) modalities is required. SQEMA is strong enough to compute first-order correspondents for at least all inductive formulas. Perhaps more surprisingly, it is possible to extract a proof of canonicity (in

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

the form of d-persistence) for every formula on which SQEMA succeeds [11]. Schmidt has introduced another algorithm based on Ackermann's Lemma which is optimized for implementation purposes [33].

Extensions of SQEMA. SQEMA extends in an unproblematic way to polyadic (purely) modal languages and hybrid languages [12]. Extensions using a recursive version of the Ackermann lemma enable SQEMA to find correspondents in first-order logic with least fixed points for some non-elementary modal formulas like the Löb formula [10]. Relaxing the syntactic requirement of positivity in the Ackermann rule to monotonicity, yields a more general 'semantic' algorithm [9]. Including various substitution rules results in an extension of SQEMA [13] that can handle all Vakaralov's complex formulas [34].

ALBA. SQEMA and its variations are applicable to (extensions of) modal logics based on classical propositional logic. The distributive modal logic of Gehrke, Nagahashi and Venema [22] is similar to intuitionistic modal logic but lacks the implication, and has four unary modalities, which can be though of as 'possibly', 'necessarily', 'possibly not' and 'necessarily not'. Distributive lattices with operators provide the algebraic semantics for this logic which a discrete duality links to Kripke frames enriched with partial ordering relations. The ALBA algorithm [14] (an acronym for Ackermann Lemma Based Algorithm) is a successor of SQEMA which is adapted to this setting. The loss of classical negation has far reaching consequences requiring major changes and making ALBA a distinctively different algorithm from SQEMA. Simultaneously, the move to this more general environment helps to clarify the essentially order-theoretic and algebraic nature of the properties underlying Salhqvist's theorem and algorithms like SQEMA and ALBA.

Unified Correspondence. These insights are explored and developed in [8] as a framework for unifying disparate correspondence and canonicity results in the literature and as a methodology for formulating and proving new ones in a wide range of logics. One of the most general instances of this is a Sahlqvist-style result for logics with algebraic semantics based on possibly non-distributive lattices with operators exhibiting a wide range of order-theoretic behaviours [15]. Giving up distributivity results in the original ALAB algorithm's strategy becoming unsound in significant aspects. This calls for a new approach where formulas are no longer decomposed connective-by-connective, but where their order-theoretic properties (as term functions on algebras) determine the applicability of rules which extract subformulas directly. This framework covers many well known logics including the Full Lambek and Lambek calculus, (co- and bi-) intuitionistic multi-modal logic, Prior's MIPC and Dunn's Positive Modal Logic. Furthermore, normality of modal operators is by no means a prerequisite for the unified correspondence approach to work, as is shown in [29].

Extensions of ALBA. Although the results for possibly non-distributive logics outlined in the previous paragraph are very general, the particular features of many logics require special treatment and therefore customised versions of ALBA and bespoke realizations of the unified correspondence paradigm. These include mu-calculi, already studied from a Sahlqvist-theoretic perspective in [2], where the presence of fixed-point binders significantly complicates the order theoretic considerations and requires special rules [6, 7]. Hybrid logics pose no problem as far as correspondence is concerned, since nominals and the other typical syntactic machinery do not introduce second-order quantification, but canonicity and completeness results require innovative treatment [16]. Correspondence for many-valued modal logic is easy to obtain once ALBA is seen to be applicable via an appropriate algebraic duality [3].

Other Applications of Unified Correspondence. Although the original purpose of SQEMA and ALBA is to eliminate second-order quantifiers, the fact that they both also guarantee canonicity already indicates that their usefulness goes beyond this. Other such applications include the dual characterizations of classes of finite lattices [19], the identification of the syntactic shape of axioms which can be translated into structural rules of a proper display calculus [25] and of internal Gentzen calculi for the logics of strict implication [27].

References

1. van Benthem, J.: Modal frame correspondences and fixed-points. Studia Logica 83(1-3), 133–155 (2006)
2. van Benthem, J., Bezhanishvili, N., Hodkinson, I.: Sahlqvist correspondence for modal mu-calculus. Studia Logica 100(1-2), 31–60 (2012)
3. Britz, C.: Correspondence theory in many-valued modal logics. Master's thesis, University of Johannesburg, South Africa (2016)
4. ten Cate, B., Marx, M., Viana, J.P.: Hybrid logics with Sahlqvist axioms. Logic Journal of the IGPL (3), 293–300 (2006)
5. Conradie, W.: On the strength and scope of DLS. Journal of Applied Non-Classical Logics 16(3-4), 279–296 (2006)
6. Conradie, W., Craig, A.: Canonicity results for mu-calculi: an algorithmic approach. Journal of Logic and Computation 27(3), 705–748 (2017)
7. Conradie, W., Fomatati, Y., Palmigiano, A., Sourabh, S.: Algorithmic correspondence for intuitionistic modal mu-calculus. Theoretical Computer Science 564, 30–62 (2015), http://www.sciencedirect.com/science/article/pii/S0304397514008196
8. Conradie, W., Ghilardi, S., Palmigiano, A.: Unified correspondence. In: Baltag, A., Smets, S. (eds.) Johan van Benthem on Logic and Information Dynamics, Outstanding Contributions to Logic, vol. 5, pp. 933–975. Springer International Publishing (2014)
9. Conradie, W., Goranko, V.: Algorithmic correspondence and completeness in modal logic IV: Semantic extensions of SQEMA. Journal of Applied Non-Classical Logics 18(2-3), 175–211 (2008)
10. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. v. recursive extensions of sqema. Journal of Applied Logic 8(4), 319–333 (2010)

11. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. I. The core algorithm SQEMA. Logical Methods in Computer Science (2006)
12. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. II. Polyadic and hybrid extensions of the algorithm SQEMA. Journal of Logic and Computation 16(5), 579–612 (2006)
13. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. III. extensions of the algorithm SQEMA with substitutions. Fundamenta Informaticae 92(4), 307–343 (2009)
14. Conradie, W., Palmigiano, A.: Algorithmic correspondence and canonicity for distributive modal logic. Annals of Pure and Applied Logic 163(3), 338 – 376 (2012)
15. Conradie, W., Palmigiano, A.: Algorithmic correspondence and canonicity for non-distributive logics (Submitted ArXiv preprint 160308515)
16. Conradie, W., Robinson, C.: On Sahlqvist theory for hybrid logic. Journal of Logic and Computation 27(3), 867–900 (2017)
17. Doherty, P., Łukaszewicz, W., Szalas, A.: Computing circumscription revisited: A reduction algorithm. Journal of Automated Reasoning 18(3), 297–336 (1997)
18. Dunn, M., Gehrke, M., Palmigiano, A.: Canonical extensions and relational completeness of some substructural logics. Journal of Symbolic Logic 70(3), 713–740 (2005)
19. Frittella, S., Palmigiano, A., Santocanale, L.: Dual characterizations for finite lattices via correspondence theory for monotone modal logic. Journal of Logic and Computation 27(3), 639–678 (2017)
20. Gabbay, D.M., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. South African Computer Journal 7, 35–43 (1992)
21. Gehrke, M.: Generalized Kripke frames. Studia Logica 84(2), 241–275 (2006)
22. Gehrke, M., Nagahashi, H., Venema, Y.: A Sahlqvist theorem for distributive modal logic. Annals of Pure and Applied Logic 131(1-3), 65–102 (2005)
23. Goranko, V., Hustadt, U., Schmidt, R.A., Vakarelov, D.: SCAN is complete for all Sahlqvist formulae. In: Berghammer, R., Möller, B., Struth, G. (eds.) Revised Selected Papers of the 7th International Seminar on Relational Methods in Computer Science and the 2nd International Workshop on Applications of Kleene Algebra, Bad Malente, Germany, May 12-17, 2003. pp. 149–162. Springer (2004)
24. Goranko, V., Vakarelov, D.: Elementary canonical formulae: Extending Sahlqvist's theorem. Annals of Pure and Applied Logic 141(1-2), 180–217 (2006)
25. Greco, G., Ma, M., Palmigiano, A., Tzimoulis, A., Zhao, Z.: Unified correspondence as a proof-theoretic tool. Journal of Logic and Computation (2016, doi: 101093/logcom/exw022 ArXiv preprint 160308204)
26. Kurtonina, N.: Categorical inference and modal logic. Journal of Logic, Language, and Information 7 (1998)
27. Ma, M., Zhao, Z.: Unified correspondence and proof theory for strict implication. Journal of Logic and Computation 27(3), 921–960 (2017)
28. Ohlbach, H.J., Schmidt, R.A.: Functional translation and second-order frame properties of modal logics. Journal of Logic and Computation 7(5), 581–603 (1997)
29. Palmigiano, A., Sourabh, S., Zhao, Z.: Sahlqvist theory for impossible worlds. Journal of Logic and Computation 27(3), 775–816 (2017)
30. de Rijke, M., Venema, Y.: Sahlqvist's theorem for Boolean algebras with operators with an application to cylindric algebras. Studia Logica 54(1), 61–78 (1995)
31. Sahlqvist, H.: Completeness and correspondence in the first and second order semantics for modal logic. In: Kanger, S. (ed.) Studies in Logic and the Foundations of Mathematics, vol. 82, pp. 110–143. North-Holland, Amsterdam (1975)

32. Sambin, G., Vaccaro, V.: A new proof of Sahlqvist's theorem on modal definability and completeness. Journal of Symbolic Logic 54(3), 992–999 (1989)
33. Schmidt, R.A.: The Ackermann approach for modal logic, correspondence theory and second-order reduction. Journal of Applied Logic 10(1), 52–74 (2012)
34. Vakarelov, D.: Modal definability in languages with a finite number of propositional variables, and a new extention of the Sahlqvist class. In: Advances in Modal Logic, vol. 4, pp. 495–518. King's College Publications (2003)

Focusing on a Vocabulary: Ontology Inseparability, Uniform Interpolation and Modularity (Abstract of Invited Talk)

Boris Konev

Department of Computer Science, University of Liverpool, UK
konev@liverpool.ac.uk

Standardisation and wide acceptance of the web ontology language OWL and its profiles have led to the proliferation of description logic ontologies, especially in the medical, bioinformatics and semantic web domains. The sheer size and complexity make it virtually impossible for a human to comprehend the underlying logical structure of an ontology as a whole, so it can be advantageous for ontology engineers to concentrate on specific parts of an ontology. On the other hand, local changes to a logical theory, and interactions between such changes, can have unpredictable non-local effects. Ontology inseparability, closely linked with the notion of conservative extension, is a powerful tool to capture non-local dependencies between ontology terms within a given vocabulary, depending on a specific application scenario. In this talk, we consider different notions of ontology inseparability and their applications to modularity, forgetting and logical difference.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

Conservative Extensions in Description Logics and Beyond (Abstract of Invited Talk)

Carsten Lutz

Department of Computer Science, University of Bremen, Germany

In description logic (DL), deciding whether a logical theory is a conservative extension of another theory is a fundamental reasoning task with applications in ontology modularity and reuse, ontology versioning, and ontology summarization [1]. It is well-known that conservative extensions are decidable in many DLs and that they can often be characterized elegantly in term of model theoretic notions such as bisimulations, simulations, or homomorphisms. In this talk, we discuss two current topics in conservative extensions.

First, we consider versions of conservative extensions that are defined in terms of data and querying [2]. We show that when ontologies are formulated in the description logic $\mathcal{ALC}$ and queries are conjunctive queries (CQs), then the resulting decision problem is undecidable. Remarkably, decidability is regained when CQs are replaced with unions of conjunctive queries (UCQs). We also consider the unexpectedly dramatic effects of admitting inverse roles in ontologies [3], namely that, in model-theoretic characterizations, homomorphisms have to be replaced with bounded homomorphisms, resulting in considerable technical challenges in designing decision procedures.

And second, we study the decidability of conservative extensions in more expressive decidable fragments of first-order logic such as the two-variable fragment and the guarded fragment [4]. We show undecidability for these two fragments and decidability for the two-variable guarded fragment. The latter rests on a model-theoretic characterization that is considerably more complex than for many standard DLs. Again, boundedness of the relevant model-theoretic notion (which in this case is GF_2-bisimulation) plays an important role.

References

1. Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, Michael Zakharyaschev: Inseparability and Conservative Extensions of Description Logic Ontologies: A Survey. Reasoning Web 2016: 27-89
2. Elena Botoeva, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, Michael Zakharyaschev: Query-Based Entailment and Inseparability for ALC Ontologies. IJCAI 2016: 1001-1007
3. Jean Christoph Jung, Carsten Lutz, Mauricio Martel, Thomas Schneider: Query Conservative Extensions in Horn Description Logics with Inverse Roles. IJCAI 2017: 1116-1122
4. Jean Christoph Jung, Carsten Lutz, Mauricio Martel, Thomas Schneider, Frank Wolter: Conservative Extensions in Guarded and Two-Variable Fragments. ICALP 2017: 108:1-108:14

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

Elimination Techniques in Modern Propositional Logic Reasoning (Abstract of Invited Talk)

Norbert Manthey

nmanthey@conp-solutions.com

The satisfiability testing (SAT) problem is one of the most relevant problems of computer science, as SAT is the representative problem for the complexity class $\mathcal{NP}$ [3]. Due to the numerous improvements to SAT solvers, many industrial problems are successfully reduced to SAT [3]. The highly optimized and specialized SAT solvers make these improvements accessible, and with such systems solving problems via SAT became effective.

Many recent improvements in SAT solvers are related to data structures, search heuristics or problem simplifications. However, the major reasoning techniques in propositional logic is resolution on clauses, used in unit propagation, variable elimination as well as clause learning [6, 7, 13]. State-of-the-art SAT solvers primarily use this technique to guide their search [1].

Both from a reasoning strength, as well as from an empirical analysis point of view, these systems still benefit from further problem simplifications, specifically variable elimination, where elimination is not only performed on pure clauses, but also on XOR constraints as well as cardinality constraints [2, 4, 8, 14]. For the two more expressive constraint types, constraints can even be extracted from formulas in CNF.

The relations between formulas F before and after F' a simplification have been described in [11]. For applied SAT solving, not only performance matters, but also the ability of constructing models for the original formula F based on a model for F'. All above mentioned elimination techniques have this property.

Further simplification techniques rely on removing clauses, e.g. blocked clause elimination [9]. From a proof complexity point of view, the counter technique – adding blocked clauses [10] – can lead to a much more powerful reasoning than resolution, namely introducing fresh variables via extended resolution [5]. Attempts on introducing fresh variables automatically exist [12], but are currently more used during encoding a problem into CNF than as a reasoning technique during search. Again, a model for the original formula can always be constructed based on a model of the simplified formula.

References

1. *Proceedings of SAT Competition 2017: Solver and Benchmark Descriptions*, volume B-2017-1 of *Publication series B*. University of Helsinki, Helsinki, Finland, 2017.
2. A. Balint and N. Manthey. Boosting the performance of SLS and CDCL solvers by preprocessor tuning. In *POS-13*, volume 29 of *EPiC Series*, pages 1–14. EasyChair, 2014.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

3. A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
4. A. Biere, D. Le Berre, E. Lonca, and N. Manthey. Detecting cardinality constraints in CNF. In *SAT 2014*, volume 8561 of *LNCS*, pages 285–301, 2014.
5. S. A. Cook. A short proof of the pigeon hole principle using extended resolution. *SIGACT News*, 8(4):28–32, Oct. 1976.
6. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
7. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.
8. N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *SAT 2005*, volume 3569 of *LNCS*, pages 61–75, 2005.
9. M. Järvisalo, A. Biere, and M. J. Heule. Blocked clause elimination. In *TACAS*, volume 6015 of *LNCS*, pages 129–144, 2010.
10. O. Kullmann. On a generalization of extended resolution. *Discrete Applied Mathematics*, 96-97(1):149–176, 1999.
11. N. Manthey. *Towards Next Generation Sequential and Parallel SAT Solvers*. PhD thesis, TU Dresden, 2014.
12. N. Manthey, M. J. Heule, and A. Biere. Automated reencoding of Boolean formulas. In *Hardware and Software: Verification and Testing*, volume 7857 of *LNCS*, pages 102–117, 2013.
13. J. P. Marques-Silva and K. A. Sakallah. GRASP – a new search algorithm for satisfiability. ICCAD '96, pages 220–227. IEEE Computer Society, 1996.
14. M. Soos, K. Nohl, and C. Castelluccia. Extending SAT solvers to cryptographic problems. In *SAT 2009*, volume 5584 of *LNCS*, pages 244–257, 2009.

Automated Forgetting and Uniform Interpolation: Three Tools (Abstract of Invited Talk)

Renate A. Schmidt

School of Computer Science, The University of Manchester, UK

Forgetting eliminates symbols from a knowledge base so that consequences over the remaining symbols in the signature are preserved. In logic the problem has been studied as the uniform interpolation problem. Uniform interpolation is a notion related to the Craig interpolation problem, but is stronger.

In computer science the importance of forgetting can be found in the knowledge representation literature, specification refinement literature and the area of description logic-based ontology engineering. In ontology-based information processing, forgetting allows users to focus on specific parts of ontologies in order to create decompositions and restricted views for in depth analysis or sharing with other users. Forgetting is also useful for information hiding, explanation generation, semantic difference computation and ontology debugging. Forgetting is an inherently difficult problem, much harder than standard reasoning (satisfiability and validity testing), and very few logics are known to be complete for forgetting (or have the uniform interpolation property). These not so encouraging premises should however not prevent us from developing practical methods for computing forgetting solutions and uniform interpolants.

My presentation gives an overview of the methods and success stories of three forgetting tools:

- SCAN, which performs second-order quantifier elimination [1, 2],
- LETHE, which solves the uniform interpolation problem for many expressive description problems extending $\mathcal{ALC}$ [3, 4], and
- FAME, which computes semantic forgetting solutions for description logics of different expressivity [5, 6].

References

1. D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7:35–43, 1992. Also published in B. Nebel, C. Rich, W. R. Swartout, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 425–436. Morgan Kaufmann, 1992.
2. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*, volume 12 of *Studies in Logic: Mathematical Logic and Foundations*. College Publications, 2008.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

3. P. Koopmann and R. A. Schmidt. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In P. Fontaine, C. Ringeissen, and R. A. Schmidt, editors, *Proceedings of the 9th International Symposium on Frontiers of Combining Systems (FroCoS 2013)*, volume 8152 of *Lecture Notes in Artificial Intelligence*, pages 87–102. Springer, 2013.
4. P. Koopmann and R. A. Schmidt. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning (IJCAR 2014)*, volume 8562 of *Lecture Notes in Artificial Intelligence*, pages 434–448. Springer, 2014.
5. Y. Zhao and R. A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-ontologies. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*, pages 1345–1352. AAAI Press/IJCAI, 2016.
6. Y. Zhao and R. A. Schmidt. Role forgetting for $\mathcal{ALCOQH}(\nabla)$-ontologies using an Ackermann approach. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17)*, pages 1354–1361. AAAI Press/IJCAI, 2017.

Swinging between Expressiveness and Complexity in Second-Order Formalisms: A Case Study (Abstract of Invited Talk)

Andrzej Szałas[1,2]

[1] Institute of Informatics, University of Warsaw, Poland
[2] Department of Computer and Information Science, Linköping University, Sweden
andrzej.szalas@{mimuw.edu.pl,liu.se}

Second-order logic proved very useful in formalizing phenomena related to many forms of reasoning both monotonic and nonmonotonic. One of the misconceptions about various forms of second-order formalisms is that, in general, they are not amenable to practical use due to their high complexity. In fact, it is often the case that restricted, but quite general uses of second-order quantifier elimination allow for the constructive reduction of such second-order theories to logically equivalent first-order or fixpoint theories, as shown in many cases, e.g., in correspondence theory for modal logics [1,5,10,11,12,14], computing circumscription [2,8] and many others [6].

When modeling complex phenomena, like those related to commonsense reasoning, it proved useful to first swing up to the general case, using as complex logical tools as needed, and then to swing down by isolating fragments of general (second- or higher-order) theories admitting efficient reasoning techniques. This is evident, e.g., in the case of circumscription where the general case is second-order while large classes of formulas admit second-order quantifier elimination [2,8] This approach is also applied in [3] where a technique for computing weakest sufficient and strongest necessary conditions for first-order theories using second-order quantifier elimination is provided. Given a theory expressing properties of concepts, these conditions, proposed for the propositional case in [9], allow one to define the best approximations of concepts under the theory, assuming that some concepts have to be forgotten.

In [4], a highly expressive framework for qualitative preference modeling has been introduced. The framework uses generalized circumscription [7] which allows for predicates (and thus formulas) to be minimized/maximized relative to arbitrary pre-orders (reflexive and transitive). It has also been shown in [4] how a large variety of preference theories extended with cardinality constraints, can in fact be reduced to logically equivalent first-order theories using second-order quantifier elimination techniques developed for that purpose.

This talk will be devoted to a case study of combining the techniques of [3,4] to swing up to a powerful higher-order formalism for approximating concepts when the underlying theories contain qualitative preferences and cardinality constraints. Then. suitable techniques and restrictions of the general theory will be indicated to swing down to computationally friendly cases. Of course, using techniques extending [13] (or, e.g., [15,16]), one can embed this formalism in description logics using suitable second-order quantifier elimination techniques. This will also be demonstrated during the talk.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

References

1. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic: I. the core algorithm SQEMA. Logical Methods in Computer Science 2(1:5), 1–26 (2006)
2. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Autom. Reasoning 18(3), 297–336 (1997)
3. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing strongest necessary and weakest sufficient conditions of first-order formulas. In: Nebel, B. (ed.) Proc 17th IJCAI: Int. Joint Conf. on AI. pp. 145 – 151 (2001)
4. Doherty, P., Szałas, A.: Reasoning with qualitative preferences and cardinalities using generalized circumscription. In: Brewka, G., Lang, J. (eds.) Proc. 11th Conf. KR: Principles of Knowledge Representation and Reasoning. pp. 560–570. AAAI Press (2008)
5. Gabbay, D.M., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. South African Computer Journal 7, 35–43 (1992)
6. Gabbay, D., Schmidt, R., Szałas, A.: Second-Order Quantifier Elimination. Foundations, Computational Aspects and Applications, Studies in Logic, vol. 12. College Publications (2008)
7. Lifschitz, V.: Some results on circumscription. In: Proc. 1st AAAI Workshop on Nonmonotonic Reasoning. pp. 151–164 (1984)
8. Lifschitz, V.: Circumscription. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A. (eds.) Handbook of Artificial Intelligence and Logic Programming. vol. 3, pp. 297–352. Oxford University Press (1991)
9. Lin, F.: On strongest necessary and weakest sufficient conditions. In: Cohn, A., Giunchiglia, F., Selman, B. (eds.) Proc. 7th Conf. KR: Principles of Knowledge Representation and Reasoning. pp. 167–175. Morgan Kaufmann Pub., Inc. (2000)
10. Nonnengart, A., Szałas, A.: A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In: Orłowska, E. (ed.) Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa. Studies in Fuzziness and Soft Computing, vol. 24, pp. 307–328. Springer (1998)
11. Sahlqvist, H.: Correspondence and completeness in the first- and second-order semantics for modal logic. In: Kanger, S. (ed.) Proc. 3rd Scandinavial Logic Symposium. pp. 110–143. North-Holland, Amsterdam (1975)
12. Szałas, A.: On the correspondence between modal and classical logic: An automated approach. Journal of Logic and Computation 3, 605–620 (1993)
13. Szałas, A.: Second-order reasoning in description logics. Journal of Applied Non-Classical Logics 16(3-4), 517–530 (2006)
14. vanBenthem, J.: Correspondence theory. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic. vol. 2, pp. 167–247. D. Reidel Pub. Co. (1984)
15. Wernhard, C.: Second-order quantifier elimination on relational monadic formulas – A basic method and some less expected applications. In: de Nivelle, H. (ed.) Proc. 24th Int. Conf. TABLEAUX 2015. LNCS (LNAI), vol. 9323, pp. 249–265. Springer (2015)
16. Zhao, Y., Schmidt, R.: Role forgetting for ALCOQH(universal role)-ontologies using an Ackermann-based approach. In: Sierra, C. (ed.) Proc. 26th IJCAI: Int. Joint Conf. on AI. pp. 1354–1361 (2017)

Forgetting for Logic Programs/Existential Rules (Abstract of Invited Talk)

Kewen Wang (collaboration with Zhe Wang)

Griffith University, Australia

The notion of forgetting has been investigated extensively for various types of logic programs. Syntactically, a logic program can be Horn, normal (non- disjunctive), disjunctive, nested and existential. The semantics for most of such logic programs is based on either classical semantics or stable models. In this talk, we will discuss some major approaches to forgetting in logic programming, especially, the class of existential rules, a family of expressive ontology languages, which inherit desired expressive and reasoning properties from both description logics and logic programming. Yet it is challenging to establish a theory of forgetting for existential rules. We will introduce a theory of forgetting for existential rules in terms of query answering based a novel notion of unfolding. A result of forgetting may not be expressible in existential rules, and we then capture the expressibility of forgetting by a variant of boundedness. While the expressibility is undecidable in general, we identify a decidable fragment. Finally, we provide an algorithm for forgetting in this fragment.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

Ackermann-Based Forgetting for Expressive Description Logics (Abstract of Invited Talk)

Yizheng Zhao

The University of Manchester, UK
yizheng.zhao@manchester.ac.uk

Forgetting refers to a non-standard reasoning problem concerned with eliminating terms (i.e., concept and role names) from description logic-based ontologies whilst preserving sufficiently many logical consequences up to the remaining terms. On one hand, it turns out to be a very useful technique in ontology-based information processing, as it allows users to focus on specific parts of ontologies in order to create restricted views or decompositions for in-depth analysis or sharing with other users; forgetting is also useful for information hiding, explanation generation, logical difference computation, and ontology debugging and repair. On the other hand, forgetting is an inherently difficult problem – it is much harder than standard reasoning (satisfiability testing) – and very few logics are known to be complete for forgetting, there has been insufficient research on the topic and very few forgetting tools are available at present.

This work investigates practical methods for semantic forgetting in expressive description logics not considered so far. In particular, we present a practical method for forgetting concept and role names from ontologies expressible in the description logic $\mathcal{ALCOIH}(\triangledown, \sqcap)$, i.e., the basic $\mathcal{ALC}$ extended with nominals, inverse roles, role inclusions, the universal role and role conjunctions, and a practical method for forgetting role names from ontologies expressible in the description logic $\mathcal{ALCOQH}(\triangledown, \sqcap)$, i.e., $\mathcal{ALCOH}(\triangledown, \sqcap)$ additionally extended with qualified number restrictions. Being based on non-trivial generalisations of a monotonicity property called Ackermann's Lemma, these methods are the first and the only approaches so far that allow for concept and role forgetting in description logics with nominals and that allow role forgetting in description logics with qualified number restrictions. The methods are goal-oriented and incremental. They always terminate and are sound in the sense that the forgetting solution is equivalent to the original ontology up to the interpretations of the forgotten names, possibly with the interpretations of the newly-introduced nominals or concept definers during the forgetting process.

The methods are implemented in Java using the OWL API and the prototypical implementation, namely, Fame, was evaluated on several corpora of publicly accessible biomedical ontologies (in order to verify the practicality of the methods). Despite our methods not being complete, the evaluation results showed that Fame was successful in most test cases (i.e., forgot all concept and role names specified in the forgetting signatures) within a very short period of time.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

Resolution Based Uniform Interpolation and Forgetting for Expressive Description Logics (Abstract of Tutorial)

Patrick Koopmann

Institute of Theoretical Comp. Science, Technische Universität Dresden, Germany
patrick.koopmann@tu-dresden.de

This tutorial discusses resolution-based methods for computing uniform interpolants in various expressive description logics (DLs) and their implementations, and their relations to second-order quantifier elimination. DLs [1] are fragments of first-order logic used to define terminological knowledge in form of ontologies, using a set of unary and binary predicates called concept and role names, which form the signature of the ontology. *Forgetting* a set of given concept and role names from an ontology means computing a new ontology that does not use these concept and role names, but preserves all logical entailments of the original ontology over the remaining signature. The resulting ontology is then a uniform interpolant of the original ontology for the remaining signature. There is a range of applications for uniform interpolants, such as for ontology reuse, ontology analysis, or computing logical differences.

Theoretical results on uniform interpolation seem discouraging. In most known DLs, uniform interpolants do not always exist, and already the problem of deciding whether a uniform interpolant exists for a given ontology and signature is ExpTime-complete for the Horn DL $\mathcal{EL}$ [7], and 2ExpTime-complete for the expressive DL $\mathcal{ALC}$ [8]. Moreover, in both DLs, uniform interpolants have in the worst case a size that is triple-exponential in the size of the input ontology [9,8]. The former problem can be solved by computing uniform interpolants in DLs that have fixpoint operators, which often ensures that uniform interpolants do always exist. Regarding the size of the uniform interpolants, experiments indicate that the worst-case rarely occurs with real-life ontologies, and that uniform interpolants are in most cases not larger than the original ontology. However, due to the high worst case complexity, the practical computation of uniform interpolants in expressive DLs requires dedicated and goal-oriented procedures.

Forgetting and uniform interpolation are similar to second-order quantifier elimination in the sense that the goal is to eliminate predicates from a logical formulae, while preserving logical entailments in the remaining signature. However, while the result of second-order quantifier preserves all second-order entailments of the original formula, uniform interpolants only preserve entailments that can be expressed in the DL at hand. Despite these differences, the similarity to second-order quantifier elimination motivates the use of similar techniques for computing uniform interpolants. This tutorial presents such an approach, which is based on the idea of computing relevant entailments using resolution. For this, it uses a resolution-based calculus that was first presented in [2] for the DL $\mathcal{ALC}$, and later extended to more expressive DLs such as $\mathcal{SHQ}$ [3] and $\mathcal{SIF}$ [5], as

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

well as to knowledge bases with ABoxes [6]. In theory, these calculi can be seen as a consequence-based reasoning procedures that could also be used for classical reasoning tasks such as consistency-checking or classification. However, in order to be suited for computing uniform interpolants, they have to satisfy additional completeness conditions tailored towards the computation of uniform interpolants. We will present some of these calculi, and discuss algorithms and optimisations used in the tool LETHE [4] for computing uniform interpolants.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) Frontiers of Combining Systems: 9th International Symposium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings. pp. 87–102. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-40885-4_7
3. Koopmann, P., Schmidt, R.A.: Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Automated Reasoning: 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings. Springer International Publishing, Cham (2014)
4. Koopmann, P., Schmidt, R.A.: LETHE: Saturation-based reasoning for non-standard reasoning tasks. In: Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015) co-located with the 28th International Workshop on Description Logics (DL 2015), Athens, Greece, June 6, 2015. pp. 23–30 (2015), http://ceur-ws.org/Vol-1387/paper_9.pdf
5. Koopmann, P., Schmidt, R.A.: Saturated-based forgetting in the description logic $\mathcal{SIF}$. In: Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015. (2015), http://ceur-ws.org/Vol-1350/paper-53.pdf
6. Koopmann, P., Schmidt, R.A.: Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 175–181 (2015), http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9981
7. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic EL. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012 (2012), http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4511
8. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. pp. 989–995 (2011), https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170
9. Nikitina, N., Rudolph, S.: (Non-)succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. Artif. Intell. 215, 120–140 (2014), https://doi.org/10.1016/j.artint.2014.06.005

Cut Elimination and Second Order Quantifier Elimination (Abstract of Tutorial)

Alessandra Palmigiano
with Giuseppe Greco, Minghui Ma, Apostolos Tzimoulis, Zhiguang Zhao

Delft University of Technology, Netherlands

Display calculi, pioneered by Belnap [1], are a proof-theoretic framework generalizing Gentzen's sequent calculi, which have succeeded in endowing a large class of modal logics with cut-free sequent calculi in a uniform and modular way. The robustness and modularity of display calculi are rooted in a general methodology for proving cut-elimination, which identifies conditions on the design of sequent calculi which guarantee the success of a certain uniform strategy for syntactic cut elimination.

Recently, systematic connections have been established between algorithmic correspondence theory, well known from the area of modal logic, and the theory of display calculi. These connections originate from some seminal observations made by Kracht [5], in the context of his characterization of the modal axioms which can be effectively transformed into 'analytic' structural rules of display calculi. In this context, a rule is 'analytic' if adding it to a display calculus preserves Belnap's cut-elimination theorem.

The present tutorial illustrates these connections. Specifically, after introducing (proper) display calculi and discussing the uniform strategy for their cut elimination, I will discuss how the two main tools of unified correspondence theory [3], [2] (namely, (a) the *ALBA algorithm* for second order quantifier elimination, and (b) the syntactically defined class of *inductive inequalities* in each logical/algebraic signature of normal distributive lattice expansions) can be used to produce analytic calculi for a certain subclass of inductive formulas (the *analytic inductive* inequalities), and to exhaustively characterize this subclass as the class of the 'properly displayable' logics.

Time permitting, I will also discuss how the methodology of *multi-type calculi* [4] can be used to circumvent this exhaustive characterization, and export these techniques also to *non analytic* logics.

References

1. N. Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.
2. W. Conradie, S. Ghilardi, and A. Palmigiano. Unified correspondence. In A. Baltag and S. Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, volume 5 of *Outstanding Contributions to Logic*, pages 933–975. Springer International Publishing, 2014.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

3. W. Conradie and A. Palmigiano. Algorithmic correspondence and canonicity for non-distributive logics. Submitted. ArXiv preprint 1603.08515.
4. S. Frittella, G. Greco, A. Kurz, A. Palmigiano, and V. Sikimić. Multi-type sequent calculi. In M. Z. A. Indrzejczak and J. Kaczmarek, editors, *Proceedings of Trends in Logic XIII*, pages 81–93. Łodz University Press, 2014.
5. M. Kracht. Power and weakness of the modal display calculus. In *Proof theory of modal logic*, volume 2 of *Applied Logic Series*, pages 93–121. Kluwer, 1996.

A Preliminary Comparison of Forgetting Solutions Computed using SCAN, LETHE and FAME

Ruba Alassaf and Renate A. Schmidt

School of Computer Science, The University of Manchester, UK

Abstract. *Forgetting, in description logic, is a non-standard reasoning technique. The aim of this technique is to eliminate concept and role symbols from a knowledge base, whilst preserving all logical consequences up to the remaining symbols. In this research, three forgetting tools were used to understand the relationship between the approaches on which they are based: SCAN, LETHE and FAME. The approach applied LETHE to description logic-based ontologies to investigate the solutions being computed. The solutions were then compared with those being computed using the other forgetting tools, SCAN and FAME.*

Key words: Forgetting, Ontology, Resolution, Ackermann's Lemma.

1 Introduction

In previous research, a number of practical approaches have been developed to perform forgetting. Generally, these tools fall within two categories: those which use resolution and others which use methods that exploit Ackermann's Lemma. The aim of this research was to compare the solutions being computed between the different methods using real world ontologies. The three tools used to represent the different approaches analyzed in this research are SCAN [2, 5], LETHE [3, 4], and FAME [6].

The first tool, SCAN, computes forgetting solutions for knowledge-bases expressed in first-order logic using a resolution-based approach, namely the SCAN algorithm [2]. Since forgetting in first-order logic is not generally solvable, SCAN is not guaranteed to find a solution. SCAN uses Skolemization to eliminate existential quantifiers, and therefore Skolem terms may appear in SCAN's solutions.

Similarly, LETHE, the second tool, computes its forgetting solution for description logic-based ontologies using a resolution-based method. However, it has been proven that the approach used to develop LETHE will find forgetting solutions for the description logics it can handle [3]. Furthermore, a point that will be useful later is that the forgetting solutions computed using LETHE may involve definer symbols, which are not part of the original vocabulary. This is due to applying flattening techniques to the ontology and sometimes failing to eliminate the extra symbols introduced during the computation [3].

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

Finally, the last tool, FAME, computes its forgetting solutions for description logic-based ontologies using a method based on Ackermann's Lemma. Similar to LETHE, the method is guaranteed to terminate. However, the method is not forgetting complete and may fail to eliminate some of the symbols [6]. The symbols it fails to eliminate are present in the result computed by the system.

The difference between the tools lie in the expressivity of the logics being supported, as well as the forgetting methodology being used. The aim of this research was to apply one of the forgetting tools, LETHE, to description logic-based ontologies to investigate the solutions computed, and compare them with those computed using SCAN and FAME. The focus was on concept forgetting in the description logic $\mathcal{ALC}$.

2 Our Tool

For the purposes of the comparison in this research, a tool has been developed to investigate the solutions computed using LETHE, which were then compared with those computed using SCAN and FAME. FAME and LETHE have the advantage of being able to process large ontologies, hence two separate versions of the system were created: one that compares LETHE and SCAN, and another that compares FAME and LETHE using larger ontologies. An effort has been made to maximize the size of the ontologies that SCAN can accept. However, the considered ontologies were still not as large as desired. This is due to boundaries in the static data structures of SCAN.

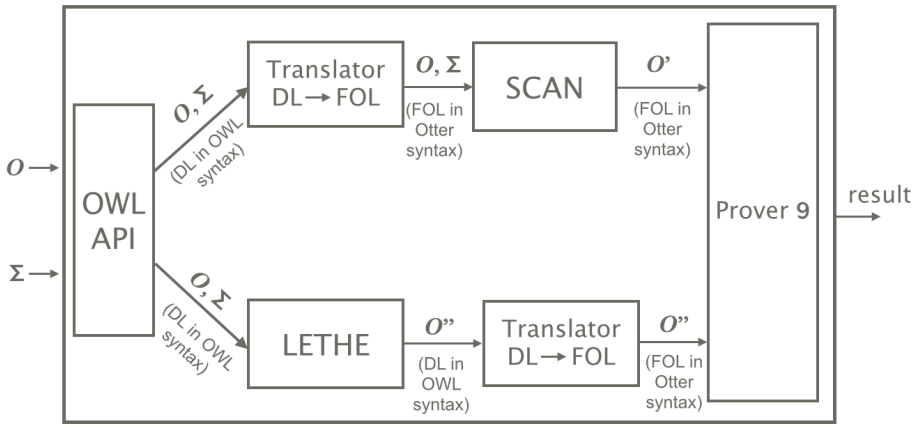

Fig. 1. An abstract design of our tool using SCAN and LETHE.

Figure 1 and Figure 2 give an overview of the components of the system and show how they are used. As can be seen in Figure 1, the tool parses an ontology

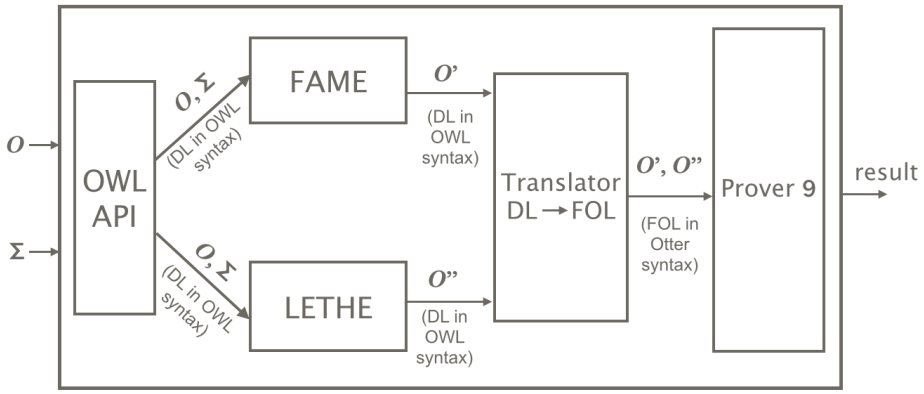

Fig. 2. An abstract design of our tool for comparing LETHE and FAME.

$\mathcal{O}$ expressed in description logic in the OWL syntax, using the OWL API, and a set of concept symbols Σ, that is referred to as the forgetting vocabulary. A description logic to first-order logic translator that uses the established relationship between them [1] has been built. The tool uses the translator to translate an ontology expressed in OWL syntax to an equivalent set of first-order formulae expressed in Otter syntax. It is used to translate the input ontology into an equivalent input for SCAN. After each of the two systems have computed its solution, a first-order logic theorem prover called Prover9[1] was used to check for entailments between the results. The translator was used again to translate LETHE's results to first-order logic as Prover9 uses the Otter syntax, too.

The system shown in Figure 2 is another version of the one used in Figure 1, except that the appropriate changes were made to use FAME instead of SCAN.

3 Results of Comparing SCAN and LETHE

In this section, the results of the comparison made between the solutions produced using SCAN and LETHE are presented. Following this, the results of the comparison made between the solutions produced using LETHE and FAME are presented in the next section.

In both comparisons, a set of randomly selected concepts was forgotten. The cardinality of the forgetting vocabulary varied depending on the number of concepts present in the ontology. However, for each ontology, an effort has been made to test and evaluate the forgetting solutions computed, as a result of forgetting sets of concepts of different sizes, ranging from low to high. In ontologies expressed in description logics that are more expressive than description logic $\mathcal{ALC}$, an $\mathcal{ALC}$ fragment of the ontologies were used. The tests where LETHE

[1] https://www.cs.unm.edu/~mccune/prover9/

produces results that contain definer symbols were excluded as they are not part of the original vocabulary.

After testing and evaluating fragments of different ontologies from the OBO Foundry ontology repository[2], the following results were observed.

1. The solutions computed using SCAN always entail the solutions computed using LETHE.
2. The solutions computed using LETHE always entail the solutions computed using SCAN if no Skolem terms occur in SCAN's solutions. However, if Skolem terms do occur in the solution, the solution computed using LETHE entail the subset of clauses and formulas that do not contain Skolem terms.
3. In some cases, the subset of clauses and formulas in SCAN's solution, that do not contain Skolem terms, is sufficient to entail the forgetting solutions computed using LETHE. The exact reason behind this occurrence raises interesting new questions of research.
4. If SCAN and LETHE both terminate, the approach used to develop SCAN outperforms the approach used to develop LETHE. However, it must be noted that this is the average situation.
5. In some cases the two approaches agree on the difficulty of the forgetting problem. However, in other cases, it was observed that LETHE found some forgetting problems to be difficult, while SCAN found a solution relatively quickly. It was speculated that the reverse occurs as well, but currently there is not sufficient evidence to support it.

4 Results of Comparing LETHE and FAME

Based on the testing and evaluation performed using fragments of ontologies from the Oxford ontology repository[3], the following results were observed:

1. If FAME successfully forgets all the symbols in the forgetting vocabulary and produces a solution that is expressed in description logic $\mathcal{ALC}$, rather than a more expressive description logic such as $\mathcal{ALCI}$, then the solutions produced by FAME and LETHE entail one another.
2. If FAME successfully forgets all the symbols in the forgetting vocabulary, but produces a solution in a more expressive description logic, namely $\mathcal{ALCI}$, then the solutions produced by FAME entail those produced by LETHE.
3. If FAME fails to forget a subset of the forgetting vocabulary and LETHE produces a solution that does not contain any unwanted symbols such as definer symbols, then the solutions produced by FAME entail those produced by LETHE. This case captures the strength of resolution as it shows that there exist cases where a forgetting problem can be solved using resolution but not using an Ackermann-based approach. Consequently, this opens the door to consider a hybrid technique that benefits from the speed of FAME and the power of LETHE.

[2] http://obofoundry.org
[3] http://www.cs.ox.ac.uk/isg/ontologies/

4. One of the differences between FAME and LETHE is that LETHE introduces definer symbols using standard flattening techniques to structurally transform the ontology into what is called normal form, in order to simplify its computations. We found that if LETHE introduces definer symbols and fails to eliminate all of them, FAME fails to forget at least one symbol in the ontology. Consequently, if FAME fails to forget at least one symbol and LETHE fails to eliminate at least one definer symbol, then no entailments can be inferred. This is obvious because each tool produces solutions that contain symbols that are not present in the solutions of the other tool. However, interestingly, one of the flattening techniques used to produce definer symbols in LETHE is inspired by Ackermann's Lemma; instead of using the Lemma to eliminate a symbol, it is used to introduce one. Hence, this raises the question: what is the relationship between the definer symbols that LETHE failed to eliminate and the symbols FAME failed to forget? We expect that these are cases of symbols with cyclic dependencies, which remains to be checked.

5 Concluding Remarks

Parts of the theory were confirmed and verified using a tool that was developed for the purposes of this research. The tool aims to compare the solutions computed using the three forgetting tools: SCAN, LETHE and FAME. It provided insight into the relationship between the three forgetting tools in the absence of a theoretical comparison.

The possibilities of a hybrid system are currently being investigated in an attempt to improve the performance while exploiting the power of resolution. A naive implementation has shown interesting results, particularly, for one of the forgetting problems. This problem took the resolution rules, implemented in LETHE, more than 14 hours to compute a solution. On the other hand, FAME, which exploits Ackermann's Lemma, successfully forgets all the symbols in the forgetting problem, except for one, in two seconds. The hybrid method which first applies FAME and then LETHE to the result computed by FAME successfully forgets all the symbols in the forgetting problem in three seconds. It would be interesting to develop heuristics that select the approach to use for each symbol in the forgetting vocabulary.

Moreover, we are interested in investigating the relationships between the tools and the underlying approaches more deeply. For example, we have not yet found any counterexamples to the following statement: when FAME successfully forgets all the symbols in the forgetting vocabulary, but produces results in $\mathcal{ALCI}$, we have that the solutions produced by LETHE entail a subset of those produced by FAME, namely the subset of solutions that are expressed in description logic $\mathcal{ALC}$. Additionally, no cases where FAME successfully forgets all the symbols and LETHE fails to eliminate a definer symbol were found during this research. We would like to investigate if this can theoretically happen. For concept forgetting in $\mathcal{ALC}$, we expect that this cannot happen. Finally, we would

like to continue investigating the relationship between the approaches used in the various systems but for more expressive description logics.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. University of Cambridge Press, Cambridge, UK, 2007.
2. D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*, volume 12 of *Studies in Logic: Mathematical Logic and Foundations*. College Publications, 2008.
3. P. Koopmann. *Practical Uniform Interpolation for Expressive Description Logics*. PhD thesis, University of Manchester, 2015.
4. P. Koopmann and R. A. Schmidt. Implementation and evaluation of forgetting in $\mathcal{ALC}$-ontologies. In *Proceedings of the 7th International Workshop on Modular Ontologies co-located with the 12th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR 2013), Corunna, Spain, September 15, 2013.*, 2013.
5. H. J. Ohlbach, T. Engel, R. A. Schmidt, and D. M. Gabbay. Scan: Home page. `http://www.mettel-prover.org/scan/index.html`. [Online; accessed 29-October-2017].
6. Y. Zhao and R. A. Schmidt. Concept forgetting in $\mathcal{ALCOI}$-ontologies using an Ackermann approach. In M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, editors, *The Semantic Web, 14th International Semantic Web Conference, ISWC 2015*, volume 9366 of *Lecture Notes in Computer Science*, pages 587–602. Springer, 2015.

Forgetting-Based Abduction in $\mathcal{ALC}$-Ontologies (Extended Abstract)

Warren Del-Pinto, Renate A. Schmidt

University of Manchester, UK
{warren.del-pinto,renate.schmidt}@manchester.ac.uk

Abstract. This paper presents a method for abduction in description logic ontologies. The method is based on forgetting and contrapositive reasoning and can produce semantically minimal hypotheses for TBox and ABox abduction in the description logic $\mathcal{ALC}$. The method is not restricted to Horn clauses or atomic observations and hypotheses. Key considerations when using forgetting for abduction are addressed. A Java prototype has been implemented, making use of the resolution-based forgetting method implemented in the tool LETHE. Experimental results over a corpus of ontologies show the practicality of the method across a number of scenarios.

1 Introduction

This paper presents a resolution-based method for performing both TBox and ABox abduction in $\mathcal{ALC}$ ontologies, which utilises the forgetting method developed in [7–10]. The observations and hypotheses can contain atomic and complex concepts. Currently, the method is restricted to the expressibility of $\mathcal{ALC}$ and cannot compute hypotheses for problems involving role assertions. System properties, essential postprocessing steps and other considerations are discussed with an evaluation of the system on various ontologies.

The contributions of this paper are as follows: (1) Abduction in ontologies is framed in terms of forgetting and contrapositive reasoning. Important considerations, such as extracting hypotheses from uniform interpolants, are discussed with proposed solutions. (2) We show that the uniform interpolation method in [9] can not only be used for TBox abduction [10] but also ABox abduction. (3) A unified method for TBox and ABox abduction based on forgetting is presented. This method can compute complex hypotheses from complex observations, finding a semantically minimal hypothesis for each observation in terms of a set of abducible symbols defined by a forgetting signature. (4) The practicality of the system is evaluated on a corpus of real-world ontologies.

2 Abduction in DL Ontologies

The aim of abduction is to compute a hypothesis to explain a new observation. This task is usually split into the tasks of TBox and ABox abduction, for which

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

the observation ψ and the hypothesis $\mathcal{H}$ take the form of sets of TBox axioms and ABox axioms respectively [2]. Two of the most common requirements are expressed in conditions (i) and (ii) of Definition 1 [13, 5], which is the form of the abduction problem considered in this paper.

Definition 1. ***Abduction in Ontologies.*** *Let $\mathcal{O}$ be an ontology, $\mathcal{S}_A$ be a set of abducible symbols, and ψ an ABox or TBox axiom such that $\mathcal{O}, \psi \not\models \bot$ and $\mathcal{O} \not\models \psi$. The* abduction problem *is to find a hypothesis $\mathcal{H}$ in the form of a set of axioms, consisting only of symbols in $\mathcal{S}_A$, such that:* (i) $\mathcal{O}, \mathcal{H} \not\models \bot$, (ii) $\mathcal{O}, \mathcal{H} \models \psi$ *and* (iii) *there is no other hypothesis $\mathcal{H}'$ such that $sig(\mathcal{H}') \subseteq \mathcal{S}_A$, $\mathcal{O}, \mathcal{H}' \models \psi$ and $\mathcal{H} \models \mathcal{H}'$, unless $\mathcal{O}, \mathcal{H}' \equiv \mathcal{O}, \mathcal{H}$.*

The condition $\mathcal{O}, \psi \not\models \bot$ ensures that neither the ontology itself nor the ontology with the observation entail false, otherwise everything follows. The condition $\mathcal{O} \not\models \psi$ imposes the constraint that the observation should not already follow from the original ontology, otherwise the problem is deductive and the abduction solution is simply $\mathcal{H} = \top$. For $\mathcal{ALC}$, it is also worth noting that for ABox abduction both ψ and $\mathcal{H}$ must take the form of ABox axioms, while in the case of TBox abduction both must take the form of TBox axioms.

Even with these conditions, the number of possible hypotheses is often intractably large. There is also the problem of finding the preferred hypotheses among these possible solutions. Thus, a variety of additional constraints are often considered to further restrict the space of abductive hypotheses [2].

The proposed method computes consistent, explanatory and semantically minimal hypotheses in accordance with Definition 1. The semantic minimality constraint in condition (iii) restricts the hypotheses to those that make the fewest assumptions, and is the "strong" semantic minimality constraint in [4]. This is a desirable characteristic for comparing hypotheses in many applications [14].

A set of *abducibles* is usually defined to further restrict the abductive hypotheses. The set of abducibles defines a subset of symbols in $\mathcal{O}$ that may appear in the hypothesis $\mathcal{H}$. Here, the abducibles are defined by a forgetting signature, as the proposed method utilises forgetting to compute hypotheses that satisfy the conditions outlined for the abduction task in Definition 1.

3 Forgetting and Uniform Interpolation

Forgetting, also known as uniform interpolation, is a process of finding a compact representation of an ontology by hiding or removing subsets of symbols within it. Here, the term *symbols* refers to concept and role names in the signature of the ontology. The symbols to be hidden are defined by a forgetting signature $\mathcal{F}$, which consists of a subset of symbols in the ontology $\mathcal{O}$. The symbols in $\mathcal{F}$ should be removed from $\mathcal{O}$, while preserving all entailments of $\mathcal{O}$ that can be represented using the restricted signature $sig(\mathcal{O}) \setminus \mathcal{F}$, resulting in a new ontology.

Definition 2. ***Uniform Interpolation in $\mathcal{ALC}$.*** *Let $\mathcal{O}$ be an $\mathcal{ALC}$-ontology and $\mathcal{F}$ a signature of symbols to be forgotten from $\mathcal{O}$. Let $\mathcal{S}_A = sig(\mathcal{O}) \setminus \mathcal{F}$ be the*

complement of $\mathcal{F}$. *The* uniform interpolation problem [12] *is the task of finding an ontology* $\mathcal{V}$ *such that the following conditions hold:* (i) $sig(\mathcal{V}) \subseteq \mathcal{S}_A$, (ii) *for any set of axioms* β: $\mathcal{O} \models \beta$ *iff* $\mathcal{V} \models \beta$ *provided that* $sig(\beta) \subseteq \mathcal{S}_A$. *The ontology* $\mathcal{V}$ *is a* uniform interpolant *of* $\mathcal{O}$ *for the signature* $\mathcal{S}_A$.

Theorem 1. $\mathcal{V}$ *is the uniform interpolant of ontology* $\mathcal{O}$ *for signature* $\mathcal{S}_A$ *iff* $\mathcal{V}$ *is the strongest necessary entailment of* $\mathcal{O}$ *in the signature* $\mathcal{S}_A$.

It is not necessarily the case that the source and target languages are the same. In [9] the uniform interpolant (or forgetting solution) of an ontology in $\mathcal{ALC}$ may need to be expressed in an extension of $\mathcal{ALC}$, which includes the following additions: (i) fixpoint operators or definer symbols for representing cycles and (ii) disjunctions of ABox axioms. Here, "definer symbols" refer to new symbols that are not present in the original ontology. "Cycles" refer to uniform interpolants that are not finitely representable in $\mathcal{ALC}$. However, evaluations on real-world ontologies have shown that the majority of uniform interpolants can be represented in pure $\mathcal{ALC}$. In the case where the result does contain cycles, we represent this result using fixpoints. Thus, definer symbols do not appear in any of the uniform interpolants computed.

The proposed abduction method utilises the resolution-based forgetting method developed in [7–10], which can compute uniform interpolants of $\mathcal{ALC}$ ontologies by forgetting both concept and role symbols in the original ontology. Here, this method is referred to as $Int_{\mathcal{ALC}}$. Two key characteristics for computing uniform interpolants that are essential to the proposed abduction method are as follows.

Theorem 2. *The uniform interpolation method has the following properties:* (1) ***Soundness:*** *any ontology* $\mathcal{O}'$ *returned by applying* $Int_{\mathcal{ALC}}$ *to an ontology* $\mathcal{O}$ *is a uniform interpolant and hence satisfies criteria* (i) *and* (ii) *in Definition 2.* (2) ***Interpolation Completeness:*** *if there exists a uniform interpolant* $\mathcal{O}'$ *of ontology* $\mathcal{O}$, *then* $Int_{\mathcal{ALC}}$ *returns an ontology* $\mathcal{V}$ *such that* $\mathcal{V} \equiv \mathcal{O}'$.

For any combination of an $\mathcal{ALC}$ ontology $\mathcal{O}$ and forgetting signature $\mathcal{F}$, $Int_{\mathcal{ALC}}$ returns a finite uniform interpolant [9, 6].

The method $Int_{\mathcal{ALC}}$ relies on the transformation of the ontology to a normal form given by a set of clauses of concept literals. The rules of the forgetting calculus utilised in $Int_{\mathcal{ALC}}$ can be found in [9]. Definer symbols are introduced to represent concepts that occur under the scope of a quantifier. Resolution inferences are then made on literals including the symbols present in $\mathcal{F}$ and the definer symbols. Once all possible inferences have been made, any clauses containing symbols in $\mathcal{F}$ are removed and the definer symbols are eliminated resulting in an ontology $\mathcal{O}'$ that is free of all symbols in $\mathcal{F}$.

4 A Forgetting-Based Abduction Method

The resolution-based nature of the $Int_{\mathcal{ALC}}$ makes it well suited to abduction via contrapositive reasoning. The calculus is applicable not only to TBox abduction [10], but also to ABox abduction in a single unified framework. Algorithm 1

below outlines our forgetting-based method which utilises $Int_{\mathcal{ALC}}$ to compute semantically minimal abductive hypotheses by exploiting contrapositive reasoning as shown in the following theorem.

Theorem 3. *Let $\mathcal{O}$ be an ontology, ψ an observation and $\mathcal{H}$ a hypothesis. Then the following holds: $\mathcal{O}, \mathcal{H} \models \psi$ iff $\mathcal{O}, \neg\psi \models \neg\mathcal{H}$.*

Algorithm 1 Forgetting-Based Abduction. *The algorithm computes hypothesis $\mathcal{H}$ for an observation ψ relative to ontology $\mathcal{O}$. It is assumed that ψ is a single axiom, which can also be a conjunction of assertions over a single individual "a" which does not occur in the ontology $\mathcal{O}$. Two cases are considered:* (i) *ψ is an ABox axiom $C(a)$ or* (ii) *ψ is a TBox axiom $C \sqsubseteq D$ where C and D can be any atomic or complex $\mathcal{ALC}$ concepts. The steps for both cases* (i) *and* (ii) *are as follows:*

1. *Negate the observation to obtain $\neg\psi$. In case* (i) *$\neg\psi = \neg C(a)$, while in case* (ii) *$\neg\psi = (C \sqcap \neg D)(a)$.*
2. *Choose a forgetting signature set $\mathcal{F}$ such that $\mathcal{F} \cap sig(\neg\psi) \neq \emptyset$, where at least one of the symbols in both $\mathcal{F}$ and $\neg\psi$ occurs with opposite polarity in the ontology $\mathcal{O}$. Let $\mathcal{S}_A = sig(\mathcal{O}) \setminus \mathcal{F}$.*
3. *Use $Int_{\mathcal{ALC}}$ to compute a uniform interpolant of $(O, \neg\psi)$ for $\mathcal{S}_A$ by forgetting the symbols in $\mathcal{F}$.*
4. *Let $\mathcal{V}$ be the uniform interpolant computed. Apply filtering to $\mathcal{V}$ to obtain the set of axioms $\mathcal{V}^* \subseteq \mathcal{V}$ that are dependent on $\neg\psi$. This means that the axioms in $\mathcal{V}^*$ are conclusions of inferences in $Int_{\mathcal{ALC}}$ with clauses from $\neg\psi$.*
5. *Assuming $\mathcal{V}^* = \{\overline{\alpha}_1(a), ..., \overline{\alpha}_k(a)\}$, let* (i) *$\mathcal{H}_I = (\alpha_1 \sqcup ... \sqcup \alpha_k)(a)$ when ψ is an ABox axiom, and* (ii) *$\mathcal{H}_I = \top \sqsubseteq (\alpha_1 \sqcup ... \sqcup \alpha_k)$ when ψ is a TBox axiom, where $\alpha_i \equiv \neg\overline{\alpha}_i$ for any $1 \leq i \leq k$.*
6. *In the case* (i) *of ABox abduction, let $\mathcal{H}_f = \mathcal{H}_I$. In the case* (ii) *of TBox abduction, an additional check is needed to ensure consistency of the hypothesis: $\mathcal{O}, \mathcal{H}_I \not\models \bot$. If this succeeds, then $\mathcal{H}_f = \mathcal{H}_I$.*

This procedure can be performed iteratively over a set of observations. In the event that cycles involving definer symbols occur in $\mathcal{V}$, these will be represented using fixpoint operators. It is important to note that $\mathcal{F}$ must contain at least one symbol in the observation ψ, as described in step 2. This enables the computation of inferences between the ontology $\mathcal{O}$ and the negated observation $\neg\psi$ using $Int_{\mathcal{ALC}}$, ensuring that the set of axioms $\mathcal{V}^*$ is computed. Otherwise, the trivial hypothesis $\mathcal{H}_f = \psi$ will be obtained. It is also worth noting the choice of representation for the negated observation $\neg\psi$. For ABox abduction, the negation of an ABox axiom $\psi = C(a)$ is simply $\neg\psi = \neg C(a)$. For TBox abduction, the negation of a TBox axiom $\psi = C \sqsubseteq D$ can be represented in several ways [10]: we choose to include a fresh individual name a and represent as an ABox axiom $\neg\psi = (C \sqcap \neg D)(a)$. Thus, for both TBox and ABox abduction $\neg\psi$ takes the form of an ABox axiom. This choice is exploited in the extraction of $\mathcal{V}^*$ from the uniform interpolant $\mathcal{V}$, as described in the next section.

The exclusion of role assertions is due to the fact that the method $Int_{\mathcal{ALC}}$ does not cater directly for negated role assertions. As a result, observations containing role assertions cannot currently be handled by our abduction method.

5 Extraction of $\mathcal{V}^*$ from $\mathcal{V}$

Many of the entailments in $\mathcal{V}$ do not involve inferences with the negated observation $\neg\psi$. These entailments do not contribute towards an explanation for ψ, and must be removed to reduce redundancy and guarantee consistency of the hypothesis returned. This leaves the set of axioms $\mathcal{V}^*$, which consists of axioms obtained by inferences in $Int_{\mathcal{ALC}}$ with either clauses in $\neg\psi$ or with clauses previously derived by inferences with $\neg\psi$.

There are several ways to remove the unnecessary axioms in $\overline{\mathcal{V}^*}$. These include checking the consistency of each disjunct in $\mathcal{H}_I$ with the ontology $\mathcal{O}$ with an external reasoner or performing subsumption deletion between axioms in $\mathcal{V}$ and those in $\mathcal{O}$. Both of these methods are computationally expensive, particularly as there are often a large number of axioms in $\mathcal{V}$. A third possibility is to trace the dependency on $\neg\psi$ as the inferences are performed in $Int_{\mathcal{ALC}}$. However, an alternate method was devised to eliminate these guaranteed redundancies without relying on an external reasoner, subsumption deletion or dependency tracing. This method utilises a property of the forgetting calculus $Int_{\mathcal{ALC}}$.

Theorem 4. Efficient Filtering of $\mathcal{V}$. *Let $\mathcal{O}$, ψ and $\mathcal{V}$ be defined as in Algorithm 1, where $\neg\psi$ is an ABox axiom $\neg C(a)$. For any α in the uniform interpolant $\mathcal{V}$, $\alpha \in \mathcal{V}^*$ iff the signature $sig_I(\alpha)$ of individuals contains a.*

After computing the set of axioms $\mathcal{V}^*$, this set is negated to obtain a hypothesis $\mathcal{H}_I$, exploiting contrapositive reasoning as in Theorem 3. This is outlined in step 5 of Algorithm 1. We have that $\mathcal{O}, \neg\psi \models \mathcal{V}^*$ iff $\mathcal{O}, \mathcal{H}_I \models \psi$ where $\neg\mathcal{V}^* \equiv \mathcal{H}_I$.

In the ABox abduction case, the unnecessary axioms in $\mathcal{V} \setminus \mathcal{V}^*$ account for all possible inconsistencies in $\mathcal{H}_I$, and no further processing is required. This was confirmed empirically by the lack of any difference between $\mathcal{H}_I$ and $\mathcal{H}_f$ in the experimental evaluations in Tables 2 and 3. $\mathcal{H}_I$ represents the hypothesis prior to the following additional check of $\mathcal{O}, \mathcal{H}_I \not\models \bot$. For TBox abduction, this test is needed to ensure that the hypothesis returned by the system is not inconsistent with the original ontology. This is due to the transformation from an ABox assertion to a TBox axiom $\top \sqsubseteq (\alpha_1 \sqcup ... \sqcup \alpha_k)$ described in step 5 of Algorithm 1. This transformation is necessary as in $\mathcal{ALC}$, if the observation ψ is a TBox axiom then the hypothesis must also be a TBox axiom to ensure the condition in Definition 1(ii) is satisfied.

6 Properties of Method

Key properties of the abduction method are presented here. These properties hold for consistent $\mathcal{ALC}$ ontologies and the characteristics of the computed hy-

potheses are relative to the signature defined by eliminating the symbols contained within the chosen forgetting signature $\mathcal{F}$. If cycles occur in uniform interpolants, these are represented using fixpoint operators.

The proposed abduction method computes semantically minimal hypotheses. This can be seen in terms of strongest necessary and weakest sufficient conditions [11, 1]. The uniform interpolant computed by $Int_{\mathcal{ALC}}$ is the strongest necessary set of entailments of the original ontology, as in Theorem 1. Thus, the axioms $\mathcal{V}^*$ can be seen as a set of strongest necessary entailments of $(\mathcal{O}, \neg\psi)$ that depend upon the observation ψ. As discussed in [11], strongest necessary and weakest sufficient conditions are dual conditions. Thus, by negating the set of axioms $\mathcal{V}^*$ under contrapositive reasoning, a weakest sufficient hypothesis $\mathcal{H}_f$ is obtained.

Theorem 5. *Let $\mathcal{O}$, ψ, $\mathcal{H}_I$ and $\mathcal{H}_f$ be defined as in Algorithm 1. The following conditions hold for the hypothesis $\mathcal{H}_f$:* (i) $\mathcal{O}, \mathcal{H}_f \not\models \bot$, (ii) $\mathcal{O}, \mathcal{H}_f \models \psi$ *and* (iii) $\mathcal{H}_f$ *is a weakest sufficient explanation, i.e, if there is a $\mathcal{H}$ such that* (i) *and* (ii) *hold and $\mathcal{H}_f \models \mathcal{H}$, then $\mathcal{O}, \mathcal{H}_f \equiv \mathcal{O}, \mathcal{H}$.*

Theorem 6. Completeness with respect to $\mathcal{S}_A$. *For an ontology $\mathcal{O}$ and observation ψ, if there exists a consistent, semantically minimal hypothesis $\mathcal{H}'$ within the signature $\mathcal{S}_A = sig(\mathcal{O}) \setminus \mathcal{F}$ such that $(O, \mathcal{H}') \models \psi$, then the proposed method returns a hypothesis $\mathcal{H}_f$ such that $\mathcal{H}_f \equiv \mathcal{H}'$.*

Several other properties of the method are worth noting. Firstly, in the case that ψ is an ABox axiom, each disjunct $\alpha_i(a)$ in the final hypothesis $\mathcal{H}_f$ is also a hypothesis since $\mathcal{O}, \alpha_i(a) \models \mathcal{O}, (\alpha_1 \sqcup ... \sqcup \alpha_k)(a)$ for $1 \leq i \leq k$. Secondly, it is possible to iteratively compute semantically stronger hypotheses due to the fact that symbols are iteratively eliminated in the forgetting loop of $Int_{\mathcal{ALC}}$.

Below is an example of an ABox abduction problem by [14, 15], used to illustrate the abduction procedure given by Algorithm 1 and the semantic minimality of the hypothesis returned.

Example 1. *Consider the following ontology $\mathcal{O}$, consisting of two TBox axioms β_1 and β_2, and the observation ψ:*

β_1: Professor $\sqcup$ Scientist $\sqsubseteq$ Academic

β_2: AssocProfessor $\sqsubseteq$ Professor

ψ : Academic(Jack)

The steps in applying the proposed method are as follows: (1) *Negate ψ to obtain $\neg$Academic(Jack).* (2) *Choose a forgetting signature $\mathcal{F}$ such that $\mathcal{F} \cap sig(\neg\psi) \neq \emptyset$, in this case: $\mathcal{F} = \{Academic\}$.* (3) *Obtain the uniform interpolant $\mathcal{V}$ by applying $Int_{\mathcal{ALC}}$ to $(\mathcal{O}, \neg\psi)$ with the forgetting signature $\mathcal{F}$. Using $\mathcal{F} = \{Academic\}$, the following uniform interpolant $\mathcal{V}$ is obtained with $Int_{\mathcal{ALC}}$:*

$\overline{\alpha}_1$: AssocProfessor $\sqsubseteq$ Professor

$\overline{\alpha}_2$: $\neg$ Professor(Jack)

$\overline{\alpha}_3$: $\neg$ Scientist(Jack)

(4) *Obtain the set of axioms $\mathcal{V}^*$ that are dependent on $\neg\psi$ by applying the filtering described in Theorem 4 to the uniform interpolant $\mathcal{V}$. The first entailment $\overline{\alpha}_1$ is filtered out as it follows directly from the ontology $\mathcal{O}$ and does not contain the*

individual observed: Jack. The entailments $\overline{\alpha}_2$ *and* $\overline{\alpha}_3$ *are dependent on* $\neg\psi$ *and are retained.* (5) *Negate* $\mathcal{V}^*$ *to obtain a hypothesis* $\mathcal{H}_I$*:*

$\mathcal{H}_I$ = (Professor ⊔ Scientist)(Jack).

(6) *In this case,* ψ *is an ABox axiom as in case* (i) *of Algorithm 1. Thus, no further checks are required and* $\mathcal{H}_I = \mathcal{H}_f$*.*

The disjuncts of the hypothesis in this example are also (stronger) hypotheses: Professor(Jack) and Scientist(Jack), as is the conjunction of these two.

7 Experimental Evaluation

A Java prototype was implemented using the OWL-API[1] and the library of the utilised forgetting method: LETHE[2]. A set of experiments was then carried out over ontologies from the NCBO BioPortal[3] and OBO [4] repositories, plus the LUBM benchmark [3] and Semintec [5] financial ontologies. The ontologies were converted to their $\mathcal{ALC}$ fragments: axioms not representable in $\mathcal{ALC}$ were deleted while others, such as domain and range restrictions, were replaced by equivalent $\mathcal{ALC}$ axioms. The characteristics of the resulting corpus are shown in Table 1. The experiments were performed on a machine using a 4.00GHz Intel Core i7-6700K CPU with 16GB of RAM.

Ontology Name	TBox Size	ABox Size	Number of Concepts	Number of Roles
BFO	52	0	35	0
HOM	83	0	66	0
LUBM	87	0	44	24
Semintec	199	65189	61	16
DOID	7892	0	11663	15
ICF	1910	6597	1597	41
OBI	28888	196	3691	67
NATPRO	68565	42763	9464	12

Table 1. Characteristics of the experimental corpus.

For each ontology, 30 observations were generated. For TBox abduction, each set of observations contained random TBox axioms from the associated ontology, consisting of atomic or complex concepts. For each individual test, the TBox axiom was first removed from the ontology then used as an observation. For ABox abduction, observations were randomly generated using assertions or arbitrary concepts in the ontology. This was done to emulate information that may be observed in practice. $\mathcal{F}$ was limited to the smallest possible signature: a randomly

[1] http://owlapi.sourceforge.net/
[2] http://www.cs.man.ac.uk/∼koopmanp/lethe/index.html
[3] https://bioportal.bioontology.org/
[4] http://www.obofoundry.org/
[5] http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm

selected concept symbol from ψ. The assumption was that a user may begin by finding the most general hypotheses, which has the benefit that stronger hypotheses can be found using the iterative abduction process described earlier.

The results are shown in Table 2. In almost all cases a semantically minimal hypothesis was computed within the time limit on LETHE. For the OBI ontology, for two TBox and five ABox cases, LETHE timed out before a uniform interpolant was computed. Given more time a uniform interpolant would be found. The time taken to compute $\mathcal{H}_f$ varied considerably with the ontology size, as did the size of $\mathcal{H}_f$. The difference between the number of axioms filtered and the size of $\mathcal{H}_f$, particularly for the larger ontologies, supports the need for efficient filtering such as the proposal in Theorem 4. The sizes of $\mathcal{H}_I$ and $\mathcal{H}_f$ were equal for the ABox abduction results, indicating that all unnecessary entailments in the uniform interpolants were removed by the proposed filtering. For TBox abduction the two values were different in all cases, leaving room for improvement in the filtering used to avoid the need for additional checks on $\mathcal{H}_I$.

ABox abduction

Ont. Name	Mean Time /s	Max. Time /s	Axioms Filtered from $\mathcal{V}$	Mean size /disjuncts $\mathcal{H}_\mathcal{I}$	Mean size /disjuncts $\mathcal{H}_f$
BFO	0.03	0.25	51	1.0	1.0
HOM	0.03	0.25	82	1.8	1.8
LUBM	0.04	0.24	92	2.0	2.0
Semin.	2.13	5.73	66757	1.3	1.3
DOID	0.57	1.68	8508	4.8	4.8
ICF	1.16	4.11	8490	3.3	3.3
OBI*	5.34	22.44	29360	5.6	5.6
NATP	46.44	399.31	111329	9.4	9.4

TBox abduction

Ont. Name	Mean Time /s	Max. Time /s	Axioms Filtered from $\mathcal{V}$	Mean size /disjuncts $\mathcal{H}_\mathcal{I}$	Mean size /disjuncts $\mathcal{H}_f$
BFO	0.03	0.35	51	2.1	1.5
HOM	0.03	0.32	81	3.3	3.0
LUBM	0.04	0.31	89	2.3	1.8
Semin.	4.06	9.28	69900	5.2	0.5
DOID	0.51	1.46	7890	3.2	2.6
ICF	0.50	1.38	8504	4.3	3.9
OBI*	53.16	94.74	29059	92.4	92.3
NATP	412.34	685.60	111196	130.0	125.1

Table 2. Results obtained for (i) ABox and (ii) TBox abduction over 30 observations with a forgetting signature of size 1. * indicates ontologies for which LETHE did not terminate within 120 seconds in at least one case.

8 Conclusion and Future Work

In this paper, a method for performing both TBox and ABox abduction in $\mathcal{ALC}$ ontologies was presented. The method uses forgetting and can compute complex hypotheses to explain both atomic and complex observations. The computed hypotheses were shown to be semantically minimal within a specified set of symbols. The practicality of the method was illustrated empirically across a corpus of real-world ontologies.

The method will be extended to perform abduction in more expressive description logics and to handle statements involving role assertions. These aims will likely be achieved by extending the $Int_{\mathcal{ALC}}$ calculus [9] to handle negated role assertions. Another option would be to investigate other methods for forgetting [16]. Another aim is to investigate the use of the iterative abduction procedure described earlier to compute increasingly stronger hypotheses.

References

1. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing strongest necessary and weakest sufficient conditions of first-order formulas. In: Proc. IJCAI'01. pp. 145–151. AAAI Press (2001)
2. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. In: Proc. OWL: Experiences and Directions (2006)
3. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 3, 158–182 (2005)
4. Halland, K., Britz, A., Klarman, S.: Tbox abduction in $\mathcal{ALC}$ using a DL tableau. In: Proc. DL'14. pp. 556–566. CEUR-WS.org (2014)
5. Klarman, S., Endriss, U., Schlobach, S.: Abox abduction in the description logic $\mathcal{ALC}$. Journal of Automated Reasoning 46, 43–80 (2011)
6. Koopmann, P.: Practical Uniform Interpolation for Expressive Description Logics. Ph.D. thesis, The University of Manchester, UK (2015)
7. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In: LPAR'13. LNCS, vol. 8312, pp. 552–567. Springer (2013)
8. Koopmann, P., Schmidt, R.A.: Uniform interpolation of $\mathcal{ALC}$ ontologies using fixpoints. In: Proc. FroCoS13. LNCS, vol. 8152, pp. 87–102. Springer (2013)
9. Koopmann, P., Schmidt, R.A.: Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In: Proc. AAAI'15. pp. 175–181. AAAI Press (2015)
10. Koopmann, P., Schmidt, R.A.: LETHE: Saturation based reasoning for non-standard reasoning tasks. In: Proc. ORE'15. pp. 23–30. CEUR-WS.org (2015)
11. Lin, F.: On strongest necessary and weakest sufficient conditions. Artificial Intelligence 128 pp. 143–159 (2001)
12. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proc. IJCAI'11. pp. 989–995. AAAI Press (2011)
13. Mooney, R.J.: Integrating abduction and induction in machine learning. In: Abduction and Induction. pp. 181–191. P. A. Flach and A. C. Kakas, Eds. Applied Logic Series. Kluwer (2000)
14. Pukancová, J., Homola, M.: Tableau-based ABox abduction for description logics: preliminary report. In: Proc. DL'16. CEUR-WS.org (2016)
15. Pukancová, J., Homola, M.: Tableau-based ABox abduction for the $\mathcal{ALCHO}$ description logic. In: Proc. DL'17,. CEUR-WS.org (2017)
16. Zhao, Y., Schmidt, R.: Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu^+(\nabla, \sqcap)$-ontologies. In: Proc. IJCAI'16. AAAI Press (2016)

Second Order Quantifier Elimination: Towards Verification Applications

Silvio Ghilardi[0000−0001−6449−6883] (✉) and Elena Pagani[0000−0001−7162−5997]

Università degli Studi di Milano, Milano, Italy
{silvio.ghilardi, elena.pagani}@unimi.it

Abstract. We develop *quantifier elimination procedures* for a fragment of higher order logic arising from the formalization of distributed systems (especially of fault-tolerant ones). Such procedures can be used in symbolic manipulations like the computation of Pre/Post images and of projections. We show in particular that our procedures are quite effective in producing *counter abstractions* that can be model-checked using standard SMT technology.

1 Introduction

Building accurate *declarative* models of distributed systems requires some complex logic, because integer and boolean variables are not sufficient: since such systems are *parameterized* (i.e. they are composed by a finite but unspecified number of processes), one needs to use arrays [1, 17] and, in the fault-tolerant case, also cardinality constraints for arrays [3, 4, 6, 12]. Since arrays are modeled by function symbols, when symbolic manipulations require to eliminate them, some form of higher-order quantifier elimination or of higher order abstraction is needed. Although in many situations existentially quantified array variables can be eliminated via explicit definitions (see the implementations in [9, 18]), this is no longer the case for concurrent and reactive systems exhibiting a large degree of non determinism.

Quantifier elimination is a rare phenomenon in second order logic, but not completely unexpected, witness the large literature on correspondence theory in modal logic. For our intended applications, some quantifier elimination results were already mentioned in [4] (Section 7.1, Thm 4) and a preliminary implementation is already available [15]. In this paper, we extend the results from our previous paper [4] by obtaining quantifier elimination for formulae containing matrices (i.e. binary arrays) and, more important, by covering formulae having *an extra universal quantifier* (Theorems 2 and 3 below). Such expansions allow us to produce arithmetic projections of more systems and to analyze benchmarks already covered in [15] in a more fine-grained way, so as to better match the pseudo-code specifications of the original papers from the distributed algorithms literature.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

The paper is organized as follows: in Section 2 we introduce preliminary notation for higher order logic, in Section 3 we describe our quantifier elimination results and in Section 4 we show how to apply the results to verification problems. This paper is focused on algorithmic procedures; the reader can find the detailed analysis of a benchmark in an Appendix of the extended (online available) version [16] (more examples, analyzed with the same methods but requiring much weaker quantifier elimination results,[1] can be found in [15], where also some experiments are reported).

2 Higher Order Logic and Flat Constraints

In order to have enough expressive power, we use higher order logic, more specifically *Church's type theory* (see e.g. [5] for an introduction to the subject).[2] It should be noticed, however, that our primary aim is *to supply a framework for model-checking and not to build a deductive system.* Thus we shall introduce below only suitable languages (via higher order signatures) and a semantics for such languages - such semantics can be specified e.g. inside any classical foundational system for set theory. In addition, as typical for model-checking, we want to constrain our semantics so that certain sorts have a fixed meaning: the primitive sort $\mathbb{Z}$ has to be interpreted as the (standard) set of integers, the sort Ω has to be interpreted as the set of truth values $\{\mathtt{tt}, \mathtt{ff}\}$; moreover, some primitive sorted operations like $+, 0, S$ (addition, zero, successor for natural numbers) and $\wedge, \vee, \rightarrow, \neg$ (Boolean operations for truth values) must have their natural interpretation. Some sorts might be *enumerated*, i.e. they must be interpreted as a specific finite 'set of values' $\{\mathtt{a}_0, \ldots, \mathtt{a}_k\}$, where the $\mathtt{a}_i$'s are mentioned among the constants of the language and are assumed to be distinct. Finally, we may ask for a primitive sort to be interpreted as a *finite set* (by abuse, we shall call such sorts *finite*): for instance, we shall constrain in this way the sort `Proc` modeling the set of processes in a distributed system. In addition, if a sort is interpreted into a finite set, we may constrain some numerical parameter (usually, the parameter we choose for this is named `N`) to indicate the cardinality of such finite set. The notion of constrained signature below incorporates all the above requirements in a general framework.

A *constrained signature* Σ consists of a set of (primitive) sorts and of a set of (primitive) sorted function symbols,[3] together with a class $\mathcal{C}_\Sigma$ of Σ-structures, called the *models* of Σ.[4] Using primitive sorts, *types* can be built up using ex-

[1] Quantifier elimination required in the benchmarls analyzed in [15] is in fact essentially confined to the BAPA-fragment known since [26].

[2] Some notation we use might look slightly non-standard; it is similar to the notation of [27].

[3] These include 0-ary function symbols, called constants; constants of sort $\mathbb{Z}$ will be called (arithmetic) *parameters*.

[4] In the standard model-checking literature, $\mathcal{C}_\Sigma$ is a singleton; here we must allow *many* structures in $\mathcal{C}_\Sigma$, because our model-checking problems are *parametric*: the sort modeling the set of processes of our system specifications must be interpreted onto

ponentiation (= functions type); *terms* can be built up using variables, function symbols, as well as λ-abstraction and functional application.

Our constrained signatures always include the sort Ω of truth-values; terms of type Ω are called *formulae* (we use greek letters $\alpha, \beta, \ldots, \phi, \psi, \ldots$ for them). For a type S, the type $S \to \Omega$ is indicated as $\wp(S)$ and called the *power set* of S; if S is constrained to be interpreted as a finite set, Σ might contain a cardinality operator $\sharp : \wp(S) \longrightarrow \mathbb{Z}$, whose interpretation is assumed to be the intended one ($\sharp s$ is the number of the elements of s - as such it is always a nonnegative number). If ϕ is a formula and S a type, we use $\{x^S \mid \phi\}$ or just $\{x \mid \phi\}$ for $\lambda x^S \phi$. We assume to have binary equality predicates for each type; universal and existential quantifiers for formulæ can be introduced by standard abbreviations (see e.g. [27]). We shall use the roman letters $x, y, \ldots, i, j, \ldots, v, w, \ldots$ for variables (of course, each variable is suitably typed, but types are left implicit if confusion does not arise). Bold letters like $\mathbf{v}$ (or underlined letters like $\underline{x}$) are used for tuples of free variables; below, we indicate with $t(\mathbf{v})$ the fact that the term t has free variables included in the list $\mathbf{v}$ (whenever this happens, we say that t is a $\mathbf{v}$-*term*, or a $\mathbf{v}$-*formula* if it has type Ω). The result of a simultaneous substitution of the tuple of variables $\mathbf{v}$ by the tuple of (type matching) terms $\underline{u}$ in t is denoted by $t(\underline{u}/\mathbf{v})$ or directly as $t(\underline{u})$.

Given a tuple of variables $\mathbf{v}$, a Σ-*interpretation* of $\mathbf{v}$ in a model $\mathcal{M} \in \mathcal{C}_\Sigma$ is a function $\mathcal{I}$ mapping each variable onto an element of the correponding type (as interpreted in $\mathcal{M}$). The evaluation of a term $t(\mathbf{v})$ according to $\mathcal{I}$ is recursively defined in the standard way and is written as $t_{\mathcal{M},\mathcal{I}}$. A Σ-formula $\phi(\mathbf{v})$ is *true* under $\mathcal{M}, \mathcal{I}$ iff it evaluates to $\mathtt{tt}$ (in this case, we may also say that $\mathbf{v}_{\mathcal{M},\mathcal{I}}$ *satisfies* ϕ); ϕ is *valid* iff it is true for all models $\mathcal{M} \in \mathcal{C}_\Sigma$ and all interpretations $\mathcal{I}$ of $\mathbf{v}$ over $\mathcal{M}$. We write $\models_\Sigma \phi$ (or just $\models \phi$) to mean that ϕ is valid and $\phi \models_\Sigma \psi$ (or just $\phi \models \psi$) to mean that $\phi \to \psi$ is valid; we say that ϕ and ψ are Σ-*equivalent* (or just equivalent) iff $\phi \leftrightarrow \psi$ is valid.

2.1 Flat Cardinality Constraints

Let us fix a constrained signature Σ for the remaining part of the paper. Such Σ should be adequate for modeling parameterized systems, hence we assume that Σ consists of:

(i) the integer sort $\mathbb{Z}$, together with some parameters (i.e. free individual constants) as well as all operations and predicates of linear arithmetic (namely, $0, 1, +, -, =, <, \equiv_n$);
(ii) the enumerated truth value sort Ω, with the constants $\mathtt{tt}, \mathtt{ff}$ and the Boolean operations on them;

---

a finite set whose cardinality is not a priori fixed. Our definition of a 'constrained signature' is analogous to the definition of a 'theory' in SMT literature; in fact, in SMT literature, a 'theory' is just a pair given by a signature and a class of structures. When transferred to a higher order context, such definition coincides with that of a 'constrained signature' above (thus our formal preliminary definitions are very similar to e.g. that of [29]).

(iii) a finite sort `Proc`, whose cardinality is constrained to be equal to the arithmetic parameter `N` (in the applications, this sort is used to represent the processes acting in our distributed systems);

(iv) a further sort `Data`, with appropriate operations, modeling local data; we assume that (a) *first-order quantifier elimination* holds for `Data`, meaning that all first-order formulæ built up from `Data`-atoms (i.e. from variables of type `Data` using operations and predicates relative to the sort `Data`) are equivalent to quantifier-free ones; (b) *ground* (i.e. variable-free) `Data`-atoms are equivalent to $\bot$ or to $\top$.

In principle, we could consider having finitely many signatures for data instead of just one, but this generalization is only apparent because one can use product sorts and recover component sorts via suitable pairing and projection operations.

If `Data` is an enumerated sort, we call Σ *finitary*; the subsignature Σ_0 of Σ obtained by restricting to sorts and operations in (i)-(ii) is called the *arithmetic* subsignature of Σ.

In the syntactic definitions below, we freely take inspiration from [3], however the present framework is greatly simplified because we do not view `Proc` as a subsort of $\mathbb{Z}$, like in [3]; in addition, notice that Σ does not contain operations or relation symbols specific to the sort `Proc` (apart from equality) - this restriction reduces terms of sort `Proc` to just variables.

Below, besides *integer* variables (namely variables of sort $\mathbb{Z}$), *data* variables (namely variables of sort `Data`) and *index* variables (namely variables of sort `Proc`), we use two other kinds of variables, that we call *array-ids* and *matrix-ids*. An array-id is a variable of type $\texttt{Proc} \to \texttt{Data}$ or of type $\texttt{Proc} \to \mathbb{Z}$ and a matrix-id is a variable of type $\texttt{Proc} \to (\texttt{Proc} \to \texttt{Data})$ or of type $\texttt{Proc} \to (\texttt{Proc} \to \mathbb{Z})$. Array-ids and matrix-ids of codomain sort $\mathbb{Z}$ are called *arithmetical* array-ids or matrix-ids; if `Data` is enumerated, array-ids and matrix-ids of codomain sort `Data` are called *finitary*. If M is a matrix-id and i, y are index variables, we may write $M_i(y)$ or $M(i, y)$ instead of $M(i)(y)$.

Let us now introduce some useful classes of formulæ.

- *Open formulæ*: these are built up from atomic formulæ containing arithmetic parameters and the above mentioned variables, using Boolean connectives only (no binders, i.e. no λ-abstractors and no quantifiers).
- *1-Flat formulæ*: these are formulæ of the kind $\phi(\sharp\{x \mid \psi_1\}/z_1, \ldots, \sharp\{x \mid \psi_n\}/z_n)$, where $\phi(z_1, \ldots, z_n), \psi_1, \ldots, \psi_n$ are open and x is a variable of type `Proc`.
- Given an index variable i, a formula ϕ is said to be *i-uniform* with respect to a matrix-id M (resp. an array-id a) iff i is not used as a bounded variable in ϕ and the only terms occurring in ϕ containing an occurrence of M (resp. of a) are of the kind $M_i(y)$ (resp. $a(i)$) for a variable y.

Notice that, some quantified formulæ can be rewritten as 1-flat formulæ: for instance $\forall i\ (a(i) = c \rightarrow b(i) = d)$ is the same as $\sharp\{i \mid a(i) = c \rightarrow b(i) = d\} = \mathtt{N}$,[5] and similarly $\exists i\ (a(i) = c)$ can be re-written as $\sharp\{i \mid a(i) = c\} > 0$.

Remark 1. 1-Flat formulæ of this paper are slightly different from the flat formulæ of [3, 4] (they roughly correspond to the flat formulæ of degree 1 of [4]); the definition here is not recursive and is simplified by the fact that we do not have nonvariable terms of type `Proc`; on the other hand, we allow matrix-ids to occur in our formulæ, whereas the syntax of [3, 4] is restricted to array-ids.

3 Quantifier Elimination

In this technical section we state and prove the quantifier elimination results we need. Let us fix a constrained signature Σ like in Subsection 2.1. We first investigate in a closer way our open formulæ. Notice first that if an open formula is *pure* (i.e. it does not contain array-ids or matrix-ids), then it is a Boolean combination of arithmetic, index or data atoms, where:

- arithmetic atoms are built up from variables of sort $\mathbb{Z}$, parameters (i.e free constants of sort $\mathbb{Z}$), by using $=, <, \equiv_n$ as predicates and $+, -, 0, 1$ as function symbols;
- index atoms are of the kind $i = j$, where i, j are variables of sort `Proc` (we do not consider further operations and predicates for this sort - apart from equality - in this paper);
- data atoms are built up from variables of sort `Data` by applying some specific set of predicates and operations (predicates include equality, all arguments of such predicates and operations are of type `Data`).

By assumption (see Subsection 2.1), quantifier elimination holds for first-order `Data`-formulæ, but this result extends very easily to all pure first-order formulæ. We state this formally as a Lemma:

Lemma 1. *Any pure first-order formula is equivalent to an open pure first-order formula.*

Proof. Using prenex formula transformations, it is sufficient to show how to eliminate a quantifier $\exists x\, \alpha$, where α is open and pure. Actually, using disjunctive normal forms, we can assume that α is a conjunction of literals. Pushing the existential quantifier inside, we can assume that such literals are all arithmetic, all index or all data literals, depending on the sort of x. The case of arithmetic literals is covered by Presburger quantifier elimination [28], whereas the case of data literals is covered by our assumption. It remains to consider the case of index literals; excluding trivial cases where the existential quantifier is redundant or eliminable by substitution, we are left with the case where α is $x \neq y_1 \wedge \cdots \wedge x \neq$

[5] Strictly speaking, this formula is 1-flat only after a bound variable renaming (we need to rename i to x). We always feel free to apply such α-conversions in the paper.

y_n. By introducing a disjunction of cases (and by distributing the existential quantifier over such disjunction and removing redundant variables), we reduce to a disjunction of formulæ of the kind

$$\exists x\ (x \neq y_1 \wedge \cdots \wedge x \neq y_{n'} \wedge \bigwedge_{i \neq j} y_i \neq y_j)$$

The latter is equivalent to $\mathtt{N} > \bar{n}'$, where $\bar{n}'$ is $1 + \cdots + 1$ (n'-times). ⊣

In case array-ids and matrix-ids do not occur, 1-flat formulæ can also be trivialized:[6]

Lemma 2. *A 1-flat formula without array-ids and matrix-ids is equivalent to a pure formula.*

Proof. Let us eliminate subterms t of the kind $\sharp\{x \mid \alpha\}$ (with pure α) inside a pure formula ϕ. We can first remove from α arithmetic and data atoms, as well as index atoms not containing x, by the following equivalence (let A be the atom to be removed):

$$\phi \leftrightarrow ([A \wedge \phi(\top/A)] \vee [\neg A \wedge \phi(\bot/A)]) \quad .$$

By Venn regions decomposition, we can assume that α is a conjunction of literals. In addition, if t is of the kind $\sharp\{x \mid x = i \wedge \alpha\}$, we can remove it using the equivalence:

$$\phi \leftrightarrow ([\alpha(i/x) \wedge \phi(1/t)] \vee [\neg\alpha(i/x) \wedge \phi(0/t)]) \quad .$$

Thus we are left only with the case in which t is $\sharp\{x \mid \bigwedge_{s=1}^{n} x \neq i_s\}$; we can also assume that ϕ entails $\bigwedge_{s \neq s'} i_s \neq i_{s'}$ (otherwise we can force this by making ϕ a disjunction of case distinctions). Then we can remove t using

$$\phi \leftrightarrow (\mathtt{N} \geq \bar{n} \wedge \phi(\mathtt{N} - \bar{n}/t)) \vee (\mathtt{N} < \bar{n} \wedge \phi(0/t)) \quad .$$

Once all t are removed (one by one), the statement is proved. ⊣

It is now convenient to introduce a notation for open (not necessarily pure) formulæ (from now on we shall reserve the letters $\alpha, \beta, \ldots$ to first-order *pure* formulæ, to evidentiate them). Considering that there are no operation symbols of sort `Proc`, the only new terms that might arise in open non pure formulæ (wrt pure formulæ) are of the kind $a(i)$ or $M_i(j)$, where a is an array-id, M is a matrix-id and i, j are variables of sort `Proc`. Thus we may write an open formula ϕ as the formula obtained by replacing in a pure formula some arithmetic variables with terms of the kind $a(i)$ or $M_i(j)$. If our open ϕ does not contain matrix-ids, we can write it as

$$\alpha(\underline{z}, \underline{k}, \mathbf{a}(\underline{k})/\underline{e}, \underline{d}) \quad \text{or simply as } \alpha(\underline{z}, \underline{k}, \mathbf{a}(\underline{k}), \underline{d}) \tag{1}$$

[6] If the sort `Proc` is identified with a definable finite subset of $\mathbb{Z}$, the result still holds but is much less trivial: to get it, one must apply results from Presburger arithmetic with counting quantifiers [30].

where $\alpha(\underline{z}, \underline{k}, \underline{e}, \underline{d})$ is pure, $\underline{z}$ is a tuple of arithmetic variables, $\underline{k}$ is a tuple of index variables, $\underline{d}$ is a tuple of data variables, $\mathbf{a}$ is a tuple of array-ids (the $\underline{e}$ might be arithmetic or `Data`-variables depending on the types of the $\mathbf{a}$); if $\mathbf{a} = a_1, \ldots, a_n$ and $\underline{k} = k_1, \ldots, k_m$, then $\mathbf{a}(\underline{k})$ is the tuple

$$a_1(k_1), \ldots, a_1(k_m), \ldots, a_n(k_1), \ldots, a_n(k_m)$$

so that the matching tuple of data variables $\underline{e}$ must be indexed as $e_{11}, \ldots, e_{nm}$.

A 1-flat formula without matrix-ids is then written as

$$\alpha(\underline{z}, \underline{k}, \mathbf{a}(\underline{k}), \underline{d}, \sharp\{x \mid \beta_1(\underline{z}, x, \underline{k}, \mathbf{a}(x), \mathbf{a}(\underline{k}), \underline{d})\}, \ldots, \sharp\{x \mid \beta_s(\underline{z}, x, \underline{k}, \mathbf{a}(x), \mathbf{a}(\underline{k}), \underline{d})\})$$

or (with some abuse of notation) shortly as

$$\alpha(\underline{z}, \underline{k}, \mathbf{a}(\underline{k}), \underline{d}, \sharp\{x \mid \underline{\beta}(\underline{z}, x, \underline{k}, \mathbf{a}(x), \mathbf{a}(\underline{k}), \underline{d})\}) \tag{2}$$

where $\underline{\beta}$ is a tuple of formulæ (we use the convention that $\sharp\{x \mid \underline{\beta}\}$ stands for the tuple of terms $\sharp\{x \mid \beta_1\}, \ldots, \sharp\{x \mid \beta_s\}$). Displaying 1-flat formulæ with matrix-ids requires an even more complex notation, that we won't use though. These notations are apparently cumbersome but have the merit of displaying the essential information on how our formulæ are built up from pure formulæ.

We now state a first quantifier elimination result (this is Theorem 4 from [4], we nevertheless report the proof in an appendix of the extended version [16] of the present paper):

Theorem 1. *Suppose that ϕ is a 1-flat formula containing the array ids $\mathbf{a}, \mathbf{a}'$ (and not containing matrix-ids); then the formula $\exists \mathbf{a}'\, \phi$ is equivalent to a formula $\exists \underline{e}\, \psi$, where the $\underline{e}$ are arithmetic and data variables, ψ is 1-flat and contains only the array-ids $\mathbf{a}$.*

The following Corollary follows from Theorem 1 and Lemmas 2,1:

Corollary 1. *Suppose that ϕ is a 1-flat formula containing the array ids $\mathbf{a}$ (and not containing matrix-ids); then the formula $\exists \mathbf{a}\, \phi$ is equivalent to an open pure formula.*

Notice that the above result (as it happens with all our quantifier elimination results) immediately implies that 1-flat formulæ not containing matrix-ids are decidable for satisfiability. If the sort `Data` is enumerated and all array-ids are finitary, we can improve Corollary 1 above by including an extra quantified variable (this is useful for benchmarks, see the Appendix of [16] for an example):

Theorem 2. *Let the sort* `Data` *be enumerated and let the 1-flat formula ϕ contain only the finitary array ids $\mathbf{a}$ (and no matrix-ids); then the formula*

$$\exists \mathbf{a}\, \forall i\, \exists \underline{y}\, \phi \tag{3}$$

(where i is an index variable and the $\underline{y}$ are arithmetic and data variables) is equivalent to an open pure formula.

Proof. Let $\mathtt{Data}$ be enumerated as $\{\mathtt{a}_0, \ldots, \mathtt{a}_k\}$; let $\underline{z}$ be the arithmetic variables occurring freely in (3) and let $\underline{k} = k_1, \ldots, k_n$ be the index variables occurring freely in (3) (thus i is not among the $\underline{k}$ and the $\underline{y}$ are not among the $\underline{z}$). We can assume that the $\underline{y}$ are arithmetic variables because, since $\mathtt{Data}$ is enumerated, existential data variables can be elimitated via disjunctions. For simplicity, we assume that (3) contains only one array-id, let it be a.[7]

Before working on the formula (3), it is better to make some preprocessing steps. Our final outcome will be that of producing a disjunction of existentially quantified pure formulæ logically equivalent to (3): in fact, we introduce extra existentially quantified variables to be eliminated in the very end using Lemma 1. We need also to introduce extra information to complete (3): this extra information is achieved by rewriting (3) as a disjunction (each disjunct formalizes a suitable guess) and by operating on each disjunct separately.

Concretely, we shall freely assume that $\forall i\, \exists \underline{y}\, \phi$ in (3) is of the kind

$$\begin{array}{l} \mathtt{Diff}(\underline{k}) \ \wedge \ \bigwedge_i (a(k_i) = \mathtt{a}_{l_i}) \ \wedge \bigwedge_j (u_j = \sharp\{x \mid a(x) = \mathtt{a}_j\}) \ \wedge \\ \wedge \ \forall i\, \exists \underline{y}\, \phi'(\underline{z}, \underline{y}, i, \underline{k}, a(i), \sharp\{x \mid \underline{\beta}(\underline{z}, \underline{y}, x, i, \underline{k}, a(x), a(i))\}) \end{array} \tag{4}$$

where

- the formula $\mathtt{Diff}(\underline{k})$ says that the $\underline{k}$ are pairwise distinct (i.e. it is $\bigwedge_{i \neq j} k_i \neq k_j$): this can be assumed without loss of generality, because one can guess a partition (introducing a disjunction over all partitions) and make the appropriate replacements so as to keep only one representative for each equivalence class of variables;
- since $\mathtt{Data}$ is enumerated we can guess (via a disjunction) for each k_i the $\mathtt{a}_{l_i}$ which is the value of $a(k_i)$ (then, all occurrences of the term in the remaining part of the formula can be replaced by this $\mathtt{a}_{l_i}$);
- the u_j are fresh arithmetic variables indicating the cardinality of the set of indices whose a-value is $\mathtt{a}_j$ (these u_j are the extra existentially quantified variables to be eliminated in the very end by Lemma 1);
- $\underline{\beta}$ are open formulæ as displayed and ϕ' is a 1-flat formula as displayed (notice that the terms $a(k_i)$ do not occur anymore here, because we can assume that they have been replaced by the corresponding $\mathtt{a}_{l_i}$).

We now operate further transformations on the subformula $\forall i\, \exists \underline{y}\, \phi'$: we want to show that this formula is equivalent to a 1-flat formula (hence without the quantifier $\forall i$), so that the claim of the Theorem follows from an application of Corollary 1 and Lemma 1 - by these results in fact all quantified variables in (3) can be eliminated in favor of a pure open formula in which only the $\underline{k}, \underline{z}$ occur. When manipulating $\forall i\, \exists \underline{y}\, \phi'$ below, we assume all the information we have from (4), namely that the $\underline{k}$ are all distinct and that the values of the $a(k_i)$ are known.

[7] This is without loss of generality: since $\mathtt{Data}$ is enumerated and the $\mathbf{a}$ are finitary, one may take a product of $\mathtt{Data}$ and replace the tuple $\mathbf{a}$ with a single array with values in such a product.

As a first step, we can distinguish the case in which i is equal to some of the $\underline{k}$ from the case in which it is different from all of them; in the latter case, we can also guess the value of $a(i)$. This observation shows that $\forall i\, \phi'$ is equal to the conjunction of an open formula (expressing what happens if i is equal to any of the $\underline{k}$) with the conjunctions (varying $\mathtt{a_j}$ in our enumerated data)

$$\forall i.\ \ \mathtt{Diff}(i,\underline{k}) \wedge a(i) = \mathtt{a_j} \rightarrow \exists \underline{y}\, \phi''(\underline{z},\underline{y},i,\underline{k},\sharp\{x \mid \underline{\beta}'(\underline{z},\underline{y},x,i,\underline{k},a(x))\}) \qquad (5)$$

where the $\phi'', \underline{\beta}'$ are obtained from the $\phi', \underline{\beta}$ by replacing $a(i)$ with $\mathtt{a_j}$. Again, it will be sufficient to show that (5) is equivalent to an open formula.

First observe that ϕ'' is obtained from a pure formula by replacing arithmetic variables with the terms $\sharp\{x \mid \underline{\beta}'(\underline{z},\underline{y},x,i,\underline{k},a(x))\}$; since equality is the only predicate of sort `Proc` (and there are no function symbols of sort `Proc`), the only atoms of sort `Proc` that might occur in a pure formula are of the kind $i = k_s, k_s = k_{s'}$ for some $s \neq s'$, but these can all be replaced by $\bot$ because we have $\mathtt{Diff}(i,\underline{k})$ in the antecedent of the implication of (5). As a consequence ϕ'' *can be displayed as* $\phi''(\underline{z},\underline{y},\sharp\{x \mid \underline{\beta}'(\underline{z},x,i,\underline{k},a(x))\})$.

A similar observation applies also to the $\underline{\beta}'$, however here we must take into consideration also atoms of the kind $x = i, x = k_s$. Thus, the $\underline{\beta}'$ are built up using Boolean conectives from atoms of the kind $x = i, x = k_s$, from arithmetic atoms $A(\underline{z},\underline{y})$ and from `Data`-atoms that might contain the term $a(x)$. We can disregard arithmetic atoms, because for each such atom $A(\underline{z},\underline{y})$ we may rewrite ϕ'' as

$$[A(\underline{z},\underline{y}) \wedge \phi''(\underline{z},\underline{y},\sharp\{x \mid \underline{\beta}'(\top/A)\})] \ \vee\ [\neg A(\underline{z},\underline{y}) \wedge \phi''(\underline{z},\underline{y},\sharp\{x \mid \underline{\beta}'(\bot/A)\})]\ . \qquad (6)$$

Thus the $\underline{\beta}'$ can be displayed as $\underline{\beta}'(x,i,\underline{k},a(x))$.

When $x = i$ or $x = k_s$ (for some s) the $\underline{\beta}'$ can be simplified to $\top$ or $\bot$ because we know the values of $a(i), a(k_s)$ (and as a consequence the numbers $\sharp\{x \mid x = i \wedge \underline{\beta}'\}, \sharp\{x \mid x = k_s \wedge \underline{\beta}'\}$ are 0/1-tuples). In conclusion we have that, for some tuple of numbers $\underline{m}$[8] that can be computed, we have that (5) is equivalent to

$$\forall i.\ \ \mathtt{Diff}(i,\underline{k}) \wedge a(i) = \mathtt{a_j} \rightarrow \exists \underline{y}\, \phi''(\underline{z},\underline{y},\underline{\bar{m}} + \sharp\{x \mid \mathtt{Diff}(x,i,\underline{k}) \wedge \underline{\beta}''(a(x))\}) \quad (7)$$

where $\underline{\beta}''$ is obtained from $\underline{\beta}'$ by replacing the atoms $x = i, x = k_s$ with $\bot$. Fix now some β''_s from the tuple $\underline{\beta}''$; for every enumerated data $\mathtt{a}_k$, each of the formulæ $\beta''_s(\mathtt{a}_k)$ simplify to either $\top$ or $\bot$ and, since we know that $u_k = \sharp\{x \mid a(x) = \mathtt{a}_k\}$ from (4), we can deduce that $\sharp\{x \mid \mathtt{Diff}(x,i,\underline{k}) \wedge a(x) = \mathtt{a}_k \wedge \beta''_s(\mathtt{a}_k)\}$ is equal to either 0 (in case $\beta''_s(\mathtt{a}_k)$ simplifies to $\bot$) or to $u_k - n_k$, where n_k is the number of the $\underline{k}, i$ for which we know that $a(\underline{k}), a(i)$ is equal to $\mathtt{a}_k$. As a consequence $\sharp\{x \mid \mathtt{Diff}(x,i,\underline{k}) \wedge \underline{\beta}''(a(x))\}$ is equal to $\sum_k (u_k - n_k)$ (where the sum extends to all k such that $\beta''_s(\mathtt{a}_k)$ simplifies to $\top$).

[8] This tuple depends on j, i.e. on the $\mathtt{a}_j$ used in the antecedent of (5) (we do not indicate this dependency for simplicity).

All this can be summarized by saying that we can rewrite (7) as

$$\forall i.\ \ \mathtt{Diff}(i,\underline{k}) \wedge a(i) = \mathtt{a_j} \rightarrow \exists \underline{y}\, \theta_j(\underline{y},\underline{z},\underline{u}) \tag{8}$$

where the formulæ θ_j are pure (the tuple $\underline{u}$ is the tuple of the u_j from (4)). By Presburger quantifier elimination, we can drop the $\exists \underline{y}$, thus getting

$$\forall i.\ \ \mathtt{Diff}(i,\underline{k}) \wedge a(i) = \mathtt{a_j} \rightarrow \theta'_j(\underline{z},\underline{u}) \tag{9}$$

Since now θ'_j does not contain occurrences of i, we can rewrite this as

$$\exists i\ (\mathtt{Diff}(i,\underline{k}) \wedge a(i) = \mathtt{a_j}) \rightarrow \theta'_j(\underline{z},\underline{u}) \tag{10}$$

and finally as

$$\sharp\{x \mid \mathtt{Diff}(x,\underline{k}) \wedge a(x) = \mathtt{a_j}\} > 0 \rightarrow \theta'_j(\underline{z},\underline{u}) \tag{11}$$

This is a 1-flat formula. To sum up, our original formula (3) is equivalent to a formula of the kind $\exists a\, \exists \underline{u}\, \vartheta$, where ϑ is 1-flat. Then (after swapping the quantifiers $\exists a\, \exists \underline{u}$) we can first use Theorem 1 to remove $\exists a$ and then Lemma 1 to produce an equivalent pure open formula (involving just the arithmetic variables $\underline{z}$ and the index variables $\underline{k}$). ⊣

In case we have uniformity, we can further extend the above result to cover formulæ in which arithmetic array-ids and matrix-ids occur (see again the Appendix of [16] for an example of the use of this result):

Theorem 3. *Let the sort* `Data` *be enumerated and let i be an index variable; suppose that all matrix-ids* $\mathbf{M}$ *occurring in the 1-flat formula ϕ are i-uniform and that all array-ids* $\mathbf{a}$ *occurring in ϕ are either finitary or i-uniform. Then the formula*

$$\exists \mathbf{a}\, \exists \mathbf{M}\, \forall i\, \exists \underline{y}\, \phi \tag{12}$$

(where the $\underline{y}$ are arithmetic and data variables) is equivalent to an open pure formula.

Proof. The first step is to remove $\exists M$ for each $M \in \mathbf{M}$, using uniformity. In fact, by uniformity, M occurs in ϕ only inside terms of the kind $M_i(y)$ (for some index variable y); thus, using *choice axiom* (in the form of an anti-skolemization), we can rewrite (12) as

$$\exists \mathbf{a}\, \forall i\, \exists \mathbf{b}\ \exists \underline{y}\, \phi(\cdots \mathbf{b}/\mathbf{M}_i \cdots) \tag{13}$$

and then we can swap the existential quantifiers $\exists \mathbf{b} \exists \underline{y}$ and apply Theorem 1, thus obtaining a formula of the kind $\exists \mathbf{a}\, \forall i\, \exists \underline{y}\, \exists \underline{e}\, \psi$ where the $\underline{e}$ are further arithmetic or data variables, ψ is 1-flat and contains only the array-id $\mathbf{a}$. Let us now split the $\mathbf{a}$ as $\mathbf{a}', \mathbf{a}''$, where the $\mathbf{a}''$ are i-uniform and the $\mathbf{a}'$ are finitary. We can apply the same anti-skolemization argument to the $\mathbf{a}''$ and rewrite $\exists \mathbf{a}'\, \mathbf{a}''\, \forall i\, \exists \underline{y}\, \exists \underline{e}\, \psi$ as $\exists \mathbf{a}'\, \forall i\, \exists \underline{z}\, \exists \underline{y}\, \exists \underline{e}\, \psi(\underline{z}/\mathbf{a}''(i))$, where the $\underline{z}$ are fresh arithmetic variables replacing the terms $\mathbf{a}''(i)$ in ψ. Now Theorem 2 can be used to eliminate the $\mathbf{a}'$. ⊣

4 System Specifications and Arithmetic Projections

We now go to verification applications. We summarize the essential machinery for making quantifier elimination to apply (for more information, see [15]).

Definition 1. *A* system specification $\mathcal{S}$ *is a tuple*

$$\mathcal{S} \;=\; (\Sigma, \mathbf{v}, \Phi, \iota, \tau) \tag{14}$$

where (i) Σ *is a constrained signature, (ii)* $\mathbf{v}$ *is a tuple of variables, (iii)* Φ, ι *are* $\mathbf{v}$*-formulæ, (iv)* τ *is a* $(\mathbf{v}, \mathbf{v}')$*-formula (here the* $\mathbf{v}'$ *are renamed copies of the* $\mathbf{v}$*) such that*

$$\iota(\mathbf{v}) \models_\Sigma \Phi(\mathbf{v}), \qquad \Phi(\mathbf{v}) \wedge \tau(\mathbf{v}, \mathbf{v}') \models_\Sigma \Phi(\mathbf{v}') \quad . \tag{15}$$

In the above definition, the $\mathbf{v}$ are meant to be the variables specifying the system status, ι is meant to describe initial states and τ is meant to describe the transition relation. The $\mathbf{v}$-formula Φ, as it is evident from (15), describes an invariant of the system (known to the user). Invariants are quite useful - and often essential - in concrete verification tasks, that's why we included them in Definition 1.

A *safety problem* for a system specification $\mathcal{S}$ like above is a $\mathbf{v}$-formula $\upsilon(\mathbf{v})$; the system is *safe with respect to* υ iff there is no $n \geq 0$ such that the formula

$$\iota(\mathbf{v}_0) \wedge \tau(\mathbf{v}_0, \mathbf{v}_1) \wedge \cdots \wedge \tau(\mathbf{v}_{n-1}, \mathbf{v}_n) \wedge \upsilon(\mathbf{v}_n)$$

is satisfiable.

Directly attacking safety problems for a system like (14) might be a too difficult task, that's why it is useful to replace it with a simpler system: in our applications, we shall try to replace $\mathcal{S}$ by some $\mathcal{S}'$ whose variables are all integer variables. To this aim, we 'project' $\mathcal{S}$ onto a subsystem $\mathcal{S}'$, i.e. onto a system comprising only some of the variables of $\mathcal{S}$. In order to give a precise definition of what we have in mind, we must first consider subsignatures: here a *subsignature* Σ_0 of Σ is a signature obtained from Σ by dropping some symbols of Σ and taking as Σ_0-models the class $\mathcal{C}_{\Sigma_0}$ of the restrictions $\mathcal{M}_{|\Sigma_0}$ to the Σ_0-symbols of the structures $\mathcal{M} \in \mathcal{C}_\Sigma$. The following proposition is immediate:

Proposition 1. *Let* $\mathcal{S}_0 = (\Sigma_0, \mathbf{v}_0, \Phi_0, \iota_0, \tau_0)$ *and* $\mathcal{S} = (\Sigma, \mathbf{v}, \Phi, \iota, \tau)$ *be system specifications, with respective safety problems* $\upsilon(\mathbf{v}_0)$ *and* $\upsilon(\mathbf{v})$*. Suppose that* Σ_0 *is a subsignature of* Σ *and let* $\mathbf{v} = \mathbf{v}_0, \mathbf{v}_1$*; suppose also that the following hold:*

(i) $\models_\Sigma \Phi_0(\mathbf{v}_0) \leftrightarrow \exists \mathbf{v}_1 \Phi(\mathbf{v}_0, \mathbf{v}_1)$*;*
(ii) $\models_\Sigma \iota_0(\mathbf{v}_0) \leftrightarrow \exists \mathbf{v}_1 \iota(\mathbf{v}_0, \mathbf{v}_1)$*;*
(iii) $\models_\Sigma \tau_0(\mathbf{v}_0, \mathbf{v}_0') \leftrightarrow \exists \mathbf{v}_1 \exists \mathbf{v}_1' (\Phi(\mathbf{v}_0, \mathbf{v}_1) \wedge \tau(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_0', \mathbf{v}_1'))$*;*
(iv) $\models_\Sigma \upsilon_0(\mathbf{v}_0) \leftrightarrow \exists \mathbf{v}_1 \upsilon(\mathbf{v}_0, \mathbf{v}_1)$*.*

Then if $\mathcal{S}_0$ *is safe with respect to* υ_0*, so it is* $\mathcal{S}$ *with respect to* υ*.*[9]

[9] Notice that only the right-to-left implications of (i)-(iv) are needed for the proposition to hold; however, if we have also the left-to-right implications, the system specification $\mathcal{S}_0$ is better, in a sense that can be specified formally [15] (intuitively, the system specification $\mathcal{S}_0$ would be the best approximation of $\mathcal{S}$ that we can make using only the variables $\mathbf{v}_0$).

The system specification $\mathcal{S}_0$ satisfying the above condition (i)-(iii) with respect to $\mathcal{S}$ is called the $(\Sigma_0, \mathbf{v}_0)$-projection of $\mathcal{S}$; if Σ_0 is the arithmetical subsignature of Σ and $\mathbf{v}_0$ are all the arithmetic variables of $\mathcal{S}$, $\mathcal{S}_0$ is called the *arithmetic projection* of $\mathcal{S}$.

Theorem 4. *If Φ, ι, τ do not contain matrix-ids and are of the kind $\exists k_1 \cdots \exists k_n \phi$ for a 1-flat formula ϕ and for index variables $k_1, \ldots, k_n$, then $\mathcal{S} = (\Sigma, \mathbf{v}, \Phi, \iota, \tau)$ has an (effectively computable) arithmetic projection.*

Proof. Let $\mathbf{v}$ be $\underline{z}, \mathbf{a}$, where the $\underline{z}$ are arithmetic variables and the $\mathbf{a}$ are array variables; we also abbreviate $k_1, \ldots, k_n$ as $\underline{k}$. We need to show that a formula of the kind

$$\exists \mathbf{a} \, \exists \underline{k} \; \alpha(\underline{z}, \underline{k}, \mathbf{a}(\underline{k}), \sharp\{x \mid \underline{\beta}(\underline{z}, x, \underline{k}, \mathbf{a}(x), \mathbf{a}(\underline{k}))\}) \tag{16}$$

is equivalent to a pure arithmetic formula.[10] But this is indeed the case: just swap the existential quantifiers and apply Corollary 1 and Lemma 1. The result follows because there are no ground index atoms and all ground data atoms are equivalent to $\top$ or to $\bot$, according to our assumptions from Subsection 2.1.

Next result concerns specifications using matrix-ids in a finitary signature.

Theorem 5. *Let the sort* `Data` *be enumerated and let Φ, ι, τ be disjunctions of formulæ of the kind*

$$\exists \underline{k} \; \forall i \; \exists \underline{y} \; \phi \tag{17}$$

where ϕ is 1-flat, $\underline{k}$ are index variables, $\underline{y}$ are arithmetic and data variables and i is an index variable such that all matrix variables and all non-finitary array variables from $\mathbf{v}$ are i-uniform in ϕ; then $\mathcal{S} = (\Sigma, \mathbf{v}, \Phi, \iota, \tau)$ has an (effectively computable) arithmetic projection.

Proof. Similar to the proof of Theorem 4, using Theorem 3 instead of Corollary 1.

To sum up, given a system specification $\mathcal{S} = (\Sigma, \mathbf{v}, \Phi, \iota, \tau)$ and a safety problem $\upsilon(\mathbf{v})$, if the formulae $\Phi, \iota, \tau, \upsilon$ satisfy suitable syntactic restrictions so that our quantifier elimination results apply, *we can compute the arithmetic projection $\mathcal{S}_0$ of $\mathcal{S}$ and try to show that $\mathcal{S}_0$ is safe with respect to $\upsilon_0(\mathbf{v}_0)$* (the latter is the formula obtained in its turn by eliminating the higher order variables from $\upsilon(\mathbf{v})$).[11] Thus a model ckecking problem formulated in higher order logic can be solved via a model checking problem for counter systems (i.e. for system specifications in a purely arithmetic signature). The literature on distributed systems confirms that this is a viable approach: since long time it has been observed that counter systems [10,11,13] can be sufficient to specify problems like cache coherence or broadcast protocols. Recently, counter abstractions have been

[10] In view of condition (iii) of Proposition 1, we need also the observation that formulæ like (16) are closed under conjunctions.

[11] A more sophisticated strategy would preprocess the system specification $\mathcal{S}$ by artificially adding to it some extra integer variables counting certain definable sets (see the Appendix of [16] for an example on how this works).

effectively used also in the verification of fault-tolerant distributed algorithms [2, 22–24].

It should be noticed that safety problems for counter systems are themselves undecidable, however the sophisticated machinery (predicate abstraction [14], IC3 [8,20], etc.) developed inside the SMT community lead to impressively performing tools like μZ [21], nuXmv [7], SeaHorn [19], ... which are nowadays being successfully used to solve many verification problems regarding counter systems.

Arithmetic projections obtained by our methods are far from trivial: the reader may realize this by looking at the detailed analysis of a classical benchmark in the Appendix of [16] (more examples are described in [15]). In all such cases, the resulting safety model-checking problems for the arithmetic projections are solved instantaneously by μZ (the SMT-HORN module of z3).

5 Conclusions

We have investigated quantifier elimination results for fragments of higher order logic suggested by verification applications in the distributed algorithms area. We have shown how to apply such results in order to automatically produce arithmetic projections that can be effectively handled by state-of-the-art SMT-based model checkers. Similar applications can be devised for forward/backward model checking, along the lines sketched in [4]. We won't discuss and compare our approach here with the different approaches from the literature, the reader is referred to the final section of [15] for some information in this sense. We only point out that the main merit of the approach we propose is that of being purely *declarative*: our starting point is the informal description of the algorithms (e.g. in some pseudo-code) and our first step is a direct translation into a standard logical formalism (typically, classical Church type theory), without relying for instance on ad hoc automata devices or on ad hoc specification formalisms. We believe that this choice can ensure flexibility and portability of our methods.

A potentially weak point to be taken care is the *complexity* of the algorithms we employ: in fact, the procedure for quantifier elimination used in the proof of Theorems 1, 2, 3 produces super-exponential blow-ups of the size of the formulæ it is applied to. Notice however that, when building a counters simulation of a concrete algorithm, such a heavy procedure is applied to each instruction (or to each block of instructions) separately, i.e. not to the whole code. Moreover, it is not difficult to realize (going through the details for our benchmarks) that it is hardly the case that the quantifier elimination procedure is applied in its full generality: in fact, it is always applied to easier fragments, where complexity reduces (recall for instance the content of footnote 1). The same observation applies also to the instances of the Presburger quantifier elimination procedure that are invoked in our manipulations: usually, they are confined to difference logic formulæ or to formulæ where quantifiers can be eliminated by simple instantiations. The identification of such shortcuts and the study of the related

complexities is important for future work and preliminary to any substantial implementation effort.

Another delicate point is related to the *syntactic limitations* we require on the formulæ describing system specifications (see the statements of Theorems 4 and 5): such syntactic limitations are needed to ensure higher order quantifier elimination. Although it seems that a significant amount of benchmarks are captured despite such limitations, it is essential to develop techniques applicable in more general cases. To this aim, we observe that just overapproximations are needed to build simulations and that, even if the best simulation may not exist, still practically useful simulations might be produced. In fact, quantifier elimination is just an extreme solution to symbol elimination problems. Symbol elimination and interpolation are a well-known technique to build invariants, abstractions and overapproximations, and for this reason their investigation has deserved considerable attention in the automated reasoning literature [25]; extensions to higher-order fragments might be useful in our context too.

References

1. Alberti, F., Bruttomesso, R., Ghilardi, S., Ranise, S., Sharygina, N.: Lazy Abstraction with Interpolants for Arrays. In: LPAR. pp. 46–61 (2012)
2. Alberti, F., Ghilardi, S., Orsini, A., Pagani, E.: Counter Abstractions in Model Checking of Distributed Broadcast Algorithms: Some Case Studies. In: Proc. CILC. pp. 102–117. CEUR Proceedings (2016)
3. Alberti, F., Ghilardi, S., Pagani, E.: Counting Constraints in Flat Array Fragments. In: Proc. IJCAR. Lecture Notes in Computer Science, vol. 9706, pp. 65–81 (2016)
4. Alberti, F., Ghilardi, S., Pagani, E.: Cardinality Constraints for Arrays (decidability results and applications). Formal Methods in System Design (2017)
5. Andrews, P.B.: An introduction to mathematical logic and type theory: to truth through proof, Applied Logic Series, vol. 27. Kluwer Academic Publishers, Dordrecht, second edn. (2002)
6. Bjørner, N., von Gleissenthall, K., Rybalchenko, A.: Cardinalities and Universal Quantifiers for Verifying Parameterized Systems. In: Proc. of the 37th ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI) (2016)
7. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv Symbolic Model Checker. In: CAV. pp. 334–342 (2014)
8. Cimatti, A., Griggio, A.: Software model checking via IC3. In: CAV. pp. 277–293 (2012)
9. Conchon, S., Goel, A., Krstic, S., Mebsout, A., Zaïdi, F.: Cubicle: A parallel smt-based model checker for parameterized systems - tool paper. In: CAV. pp. 718–724 (2012)
10. Delzanno, G.: Constraint-Based Verification of Parameterized Cache Coherence Protocols. Formal Methods in System Design 23(3), 257–301 (2003)
11. Delzanno, G., Esparza, J., Podelski, A.: Constraint-Based Analysis of Broadcast Protocols. In: Proc. of CSL. LNCS, vol. 1683, pp. 50–66 (1999)
12. Dragoj, C., Henzinger, T., Veith, H., Widder, J., Zufferey, D.: A Logic-based Framework for Verifying Consensus Algorithms. In: Proc. of VMCAI (2014)

13. Esparza, J., Finkel, A., Mayr, R.: On the Verification of Broadcast Protocols. In: Proc. of LICS. pp. 352–359. IEEE Computer Society (1999)
14. Flanagan, C., Qadeer, S.: Predicate abstraction for software verification. In: POPL. pp. 191–202 (2002)
15. Ghilardi, S., Pagani, E.: Counter Simulations via Higher Order Quantifier Elimination: a preliminary report. In: Proc. of PxTP. EPTCS (2017), (preliminary workshop version available from authors' webpages)
16. Ghilardi, S., Pagani, E.: Second Order Quantifier Elimination: Towards Verification Applications. (2017), (extended version available from authors' webpages)
17. Ghilardi, S., Ranise, S.: Backward Reachability of Array-based Systems by SMT solving: Termination and Invariant Synthesis. Logical Methods in Computer Science 6(4) (2010)
18. Ghilardi, S., Ranise, S.: MCMT: A Model Checker Modulo Theories. In: IJCAR. pp. 22–29 (2010)
19. Gurfinkel, A., Kahsai, T., Komuravelli, A., Navas, J.A.: The SeaHorn Verification Framework. In: CAV. pp. 343–361 (2015)
20. Hoder, K., Bjørner, N.: Generalized Property Directed Reachability. In: SAT. pp. 157–171 (2012)
21. Hoder, K., Bjørner, N., deMoura, L.: μZ– An Efficient Engine for Fixed Points with Constraints. In: CAV. pp. 457–462 (2011)
22. John, A., Konnov, I., Schmid, U., Veith, H., Widder, J.: Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In: Proc. Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD). pp. 201–209 (Aug 2013)
23. John, A., Konnov, I., Schmid, U., Veith, H., Widder, J.: Towards Modeling and Model Checking Fault-Tolerant Distributed Algorithms. In: Proc. Int'l SPIN Symposium on Model Checking of Software. Lecture Notes in Computer Science, vol. 7976, pp. 209–226. Springer (Jul 2013)
24. Konnov, I.V., Veith, H., Widder, J.: What You Always Wanted to Know About Model Checking of Fault-Tolerant Distributed Algorithms. In: Perspectives of System Informatics - 10th International Andrei Ershov Informatics Conference, PSI 2015, in Memory of Helmut Veith, Kazan and Innopolis, Russia, August 24-27, 2015, Revised Selected Papers. pp. 6–21 (2015)
25. Kovács, L., Voronkov, A.: Interpolation and Symbol Elimination. In: Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings. pp. 199–213 (2009)
26. Kuncak, V., Nguyen, H.H., Rinard, M.: Deciding Boolean Algebra with Presburger Arithmetic. Journal of Automated Reasoning 36(3) (2006)
27. Lambek, J., Scott, P.J.: Introduction to higher order categorical logic, Cambridge Studies in Advanced Mathematics, vol. 7. Cambridge University Press, Cambridge (1988), reprint of the 1986 original
28. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. Warszawa (1929)
29. Reynolds, A., Deters, M., Kuncak, V., Tinelli, C., Barrett, C.W.: Counterexample-guided quantifier instantiation for synthesis in SMT. In: Proc. CAV. pp. 198–216 (2015)
30. Schweikardt, N.: Arithmetic, First-Order Logic, and Counting Quantifiers. ACM TOCL pp. 1–35 (2004)

Computing $\mathcal{ALCH}$-Subsumption Modules Using Uniform Interpolation

Patrick Koopmann[1] and Jieying Chen[2]

[1] Institute of Theoretical Comp. Science, Technische Universität Dresden, Germany
patrick.koopmann@tu-dresden.de
[2] LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France
jieying.chen@lri.fr

Abstract. We investigate how minimal subsumption modules can be extracted using methods for uniform interpolation and forgetting. Given an ontology and a signature of concept and role names, a subsumption module is a subset of the ontology that preserves all logical entailments that can be expressed in the description logic of the ontology using only terms in the specified signature. As such, they are useful for ontology reuse and ontology analysis. While there exists a range of methods for computing or approximating minimal modules for a range of module types, we are not aware of a practical, implemented method for computing minimal subsumption modules in description logics beyond $\mathcal{ELH}$. In this paper, we present a method that uses uniform interpolation/forgetting to compute subsumption modules in $\mathcal{ALCH}$, and which under certain conditions guarantees minimality of the extracted modules. As a side product, our method computes a so-called LK subsumption module, which over-approximates the union of all minimal subsumption modules, and as such may already have applications of its own. We further present an initial evaluation of this method on a varied corpus of ontologies.

1 Introduction

Description Logics [1] (DLs) are a well-investigated family of logics that are commonly used to describe terminological knowledge in form of ontologies. Applications in areas such as medicine, biology and the semantic web have lead to the development of very large ontologies that, with growing size, become harder to understand and maintain. Due to the complexity of existing ontologies, there are areas where it is useful to extract a subset of the ontology, a so-called *module*, based on a set of terms of interest. For example, when developing a new ontology for a specialised application, one may want to reuse knowledge from an existing ontology. If this ontology covers a large domain of concepts, not all information in it will be relevant for the application at hand, so that it makes sense to first extract a module of the ontology that is sufficient for the application. Secondly, for maintaining an existing ontology it may be important for an ontology engineer to understand which axioms in the ontology are responsible for which of its logical entailments. Modules give the engineer an overview on the axioms of

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

the ontology that contribute to any entailment over a selected set of terms. This allows him to browse the ontology in a more directed manner guided by the terms he is interested in. In the mentioned applications, it is usually desirable to extract a module that is optimal in some sense, for example minimal w.r.t. set inclusion.

There are a range of different notions and properties for modules that have been defined in the literature, and correspondingly a range of methods for module extraction have been developed [2,8]. For some of those notions, such as semantic modules, deciding whether a subset of the ontology is a module that is minimal w.r.t. set inclusion is undecidable already for ontologies formulated in the lightweight DL $\mathcal{EL}$ [13]. In this paper, however, we consider a notion for which computing a minimal module is decidable. More precisely, we are interested in computing *minimal subsumption modules*, which are modules that preserve all logical entailments in the form of concept inclusions over the specified signature of terms.While a method for computing minimal subsumption modules in acyclic $\mathcal{EL}$ ontologies has been presented in [4], in this paper we focus on the more expressive DL $\mathcal{ALCH}$. Already for $\mathcal{ALC}$ ontologies, deciding whether a subset of the ontology is a subsumption module, is known to be 2ExpTime-complete [6], but we are not aware of a practical implementation for extracting minimal subsumption modules in DLs that are more expressive than $\mathcal{ELH}$ [3].

The core idea of our method is to use uniform interpolation [22] to compute a finite representation of the entailments the module has to preserve, together with techniques from axiom-pinpointing [28]. The method can compute small subsumption modules of $\mathcal{ALCH}$-ontologies for signatures that contain all role symbols, which under certain conditions guarantees minimality of the computed modules. As a side-product, the method computes a *lean kernel (LK) subsumption module*, an over-approximation of all minimal subsumption modules, which, as our evaluation indicates, is computationally cheaper to compute and usually not much larger than the minimal subsumption module. Moreover, we believe LK subsumption modules may have applications on its own: if subsumption modules are used by ontology engineers to investigate information with respect to certain signatures, it might be useful to have an overview of all the axioms that contribute to this information: this overview is provided, if over-approximated, by the LK subsumption module.

Our method only supports signatures that contain all role names, while arbitrary signatures are left for future work. Modules for this type of signatures have a property that makes them especially useful for ontology reuse. Specifically, as we show, modules for signatures that include all role names provide for a weak form of *robustness under replacement* [12].

The paper is structured as follows. We first recall related work on module extraction and uniform interpolation in Section 2, and give the preliminaries on $\mathcal{ALCH}$, subsumption modules and uniform interpolation in Section 3. We then describe our core algorithm for computing subsumption modules based on axiom pin-pointing in Section 4. A central idea for reducing the number of entailment checks is to use a technique to quickly compute uniform interpolants for differ-

ent subsets of the input ontology, for which we compute an *annotated uniform interpolant* defined in Section 5. As a by-product, the annotated uniform interpolant encodes an upper approximation of the minimal subsumption module, which indeed over-approximates all minimal subsumption modules. We call this module lean kernel subsumption module, as they are similar to lean kernels in axiom pinpointing, which we discuss in more detail in Section 6. Finally, we give results from an initial evaluation in Section 7 and conclude with a discussion in Section 8.

2 Related Work

There is a range of types and properties of modules that have been investigated in the literature, surveys of which can be found in [12] and [2]. Usually, modules are computed on the basis of an ontology and a signature Σ, i.e. a set of concept and role names, and preserve certain properties of the ontology with respect to that signature Σ. Examples include *semantic modules*, which preserve all models of the ontology when restricted to Σ [13] and *subsumption modules*, which preserve all logical entailments in the form of concept inclusions over Σ that can be expressed in the description logic under consideration [4,3]. Apart from minimality under set inclusion, additional properties have been considered such as *self-containedness* (the module is also a module with respect to its own signature) and depletedness (the remaining ontology only entails tautologies in the specified signature, i.e. all relevant information is in the module). Deciding whether a subset of the ontology is a semantic module for a signature is undecidable already for $\mathcal{EL}$-ontologies [13], and consequently minimal (depleting, self-contained) semantic modules can only be approximated in practice. For $\mathcal{EL}$ and $\mathcal{ALCI}$, an exception are modules of acyclic ontologies and for signatures that contain only concept names, for which methods to extract depleting modules have been implemented in the tool MEX [13]. A well-known approximation of semantic modules are locality-based modules, of which syntactical variants, such as $\top\bot*$-modules, can be computed very cheaply [8,14]. However, locality-based modules may still contain a large portion of the original ontology [27]. A more refined technique for extracting semantic modules is presented in [5], which computes lower and upper approximations of minimal depleting modules in $\mathcal{ALCQI}$ using QBF-reasoning. Depending on the application, modules that only preserve entailments in a certain query-language may be sufficient. A method that approximates minimal modules taylored towards specific query languages uses datalog reasoning and has been presented in [26].

For computing minimal subsumption modules of $\mathcal{ELH}$-terminologies, a method has been presented in [3]. An alternative approach is presented in [4], which uses a black-box search algorithm that detects axioms that can be safely removed without causing a *logical difference*, i.e. a difference in the set of entailed concept inclusions over the selected signature. For computing logical differences, the authors use the tool CEX [11], whose latest version supports the computation of logical differences between $\mathcal{ELH}^r$-ontologies [21]. However, in principle, the

same algorithm could be used for ontologies formulated in any logic for which a tool for computing logical differences exists. We tried this for $\mathcal{ALCH}$ ontologies, deploying the tool LETHE [17] that allows for computing logical differences in $\mathcal{ALCH}$ for arbitrary signatures, provided a uniform interpolant exists for them. However, we found that this approach is too computationally expensive in practice.

A notion strongly related to that of subsumption modules is that of uniform interpolants, which are also computed as part of our method. Both subsumption modules and uniform interpolants preserve all logical entailments in the specified signature that can be expressed in the respective description logic. However, while subsumption modules are subsets of the input ontology, uniform interpolants are themselves completely formulated in the specified signature, and may therefore contain axioms that do not occur in the original ontology. In fact, in the worst case, already for the description logics $\mathcal{EL}$ and $\mathcal{ALC}$, the uniform interpolant may have a size that is triple exponential in the size of the input ontology [23,22]. Despite this discouraging theoretical result, various practical methods for computing uniform interpolants in expressive description logics have been developed [20,16,18,31]. In fact, it turns out that in practice, uniform interpolants are often of moderate size.

3 Preliminaries

We recall the description logic $\mathcal{ALCH}$ [1], as well as the notions of subsumption modules and uniform interpolants.

Let N_c and N_r be two disjoint, countably infinite, sets of respectively *concept names* and *role names*. A signature $\Sigma \subseteq N_c \cup N_r$ is a finite set of concept names and role names.

The set of *concepts* C, D, *TBox axioms* α and *RBox axioms* β the set of $\mathcal{ALCH}$*-inclusions* α are built according to the following grammar rules:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$
$$\alpha ::= C \sqsubseteq C \mid C \equiv C$$
$$\beta ::= r \sqsubseteq s \mid r \equiv s$$

where $A \in N_c$ and $r \in N_r$. A TBox is a finite set of TBox axioms, an RBox a finite set of RBox axioms, and an ontology is the union of a TBox and RBox.

The semantics of $\mathcal{ALCH}$ is defined using interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the domain $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function assigning each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and every role name r to a binary relation $r^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$. Then $\cdot^{\mathcal{I}}$ is inductively extended to complex concepts by: $(\top)^{\mathcal{I}} := \Delta^{\mathcal{I}}$, $(\bot)^{\mathcal{I}} := \emptyset$, $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y \in C^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}}\}$, and $(\forall r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \forall (x, y) \in r^{\mathcal{I}} : y \in C^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ satisfies a TBox axiom $C \sqsubseteq D$ ($C \equiv D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$). It satisfies an RBox axiom $r \sqsubseteq s$ ($r \sqsubseteq s$) iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ ($r^{\mathcal{I}} = s^{\mathcal{I}}$).

We write $\mathcal{I} \models \alpha$ if $\mathcal{I}$ satisfies the axiom α. An interpretation $\mathcal{I}$ is a *model* of an ontology $\mathcal{O}$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{O}$. An axiom α *is entailed by* $\mathcal{O}$, written $\mathcal{O} \models \alpha$, if for all models $\mathcal{I}$ of $\mathcal{O}$, we have that $\mathcal{I} \models \alpha$.

A *signature* is a (countable but possibly infinite) set $\Sigma \subseteq N_r \cup N_c$ of concept and role names. Given a concept/axiom/ontology α, we denote by $\mathsf{sig}(\alpha)$ the set of concept and role names occuring in α.

Definition 1 (Σ-Inseparability, Subsumption Module, Uniform Interpolant.). *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two $\mathcal{ALCH}$-ontologies, and let Σ be a signature. Then $\mathcal{O}_1$ and $\mathcal{O}_2$ are* Σ-inseparable*, denoted as $\mathcal{O}_1 \equiv_\Sigma \mathcal{O}_2$, iff for every axiom α s.t. $\mathsf{sig}(\alpha) \subseteq \Sigma$, we have $\mathcal{O}_1 \models \alpha$ iff $\mathcal{O}_2 \models \alpha$.*

A Σ-subsumption module of $\mathcal{O}$ *is an ontology $\mathcal{M}$ s.t. $\mathcal{O} \equiv_\Sigma \mathcal{M}$ and $\mathcal{M} \subseteq \mathcal{O}$. $\mathcal{M}$ is* minimal *iff there exists no Σ-subsumption module $\mathcal{M}'$ of $\mathcal{O}$ s.t. $\mathcal{M}' \subsetneq \mathcal{M}$.*

A uniform interpolant of $\mathcal{O}$ for Σ *is an ontology $\mathcal{O}_\Sigma$ s.t. $\mathcal{O} \equiv_\Sigma \mathcal{O}_\Sigma$ and $\mathsf{sig}(\mathcal{O}_\Sigma) \subseteq \Sigma$.*

4 Minimal Subsumption Modules as Justifications

A related problem to minimal subsumption module extraction is that of computing justifications [28]. Given an ontology $\mathcal{O}$ and an axiom α that is entailed by $\mathcal{O}$, a *justification for α in $\mathcal{O}$* is a subset $\mathcal{J}$ of $\mathcal{O}$ s.t. $\mathcal{J}$ is minimal w.r.t. $\subsetneq$ and $\mathcal{J} \models \alpha$. We can generalise this notion to *justifications of ontologies* $\mathcal{O}'$ by asking for minimal subsets $\mathcal{J} \subseteq \mathcal{O}$ s.t. $\mathcal{J} \models \mathcal{O}'$. One easily sees that every minimal subsumption module is a justification of a uniform interpolant: for a given ontology $\mathcal{O}$ and signature Σ, a uniform interpolant $\mathcal{O}^\Sigma$ captures all entailments of $\mathcal{O}$ that are in Σ, and therefore, any subset of $\mathcal{O}$ that entails $\mathcal{O}^\Sigma$ entails all axioms that are in Σ. Therefore, one possible approach for computing minimal subsumption modules is to first compute a uniform interpolant, and then compute a justification for it using any DL reasoner that supports this. As there are implemented systems for both for uniform interpolation (e.g. [20,17,31]), and for computing justifications (reasoners such as HermiT [7], JFact [30] and Pellet [29] support this directly via the OWL API [9]), it seems that such a method could be implemented without much effort.

However, there are two short-comings of this approach. First, it is well-known that uniform interpolants do not always exists for any pair of ontology and signature. For this reason, existing methods for uniform interpolation either only compute approximations of the uniform interpolant, or they compute a uniform interpolant in an extended language that uses greatest fixpoint operators. Unfortunately, we are not aware of any reasoner that supports fixpoint operators, so that the method can only be applied for ontology-signature pairs for which there exist a uniform interpolant without fixpoint operators. Secondly, in first experiments of this idea we quickly found out that reasoners such as HermiT, Pellet and JFact struggle with the computation of justifications for large entailments such as uniform interpolants. While in this paper, we offer no general solution for the case in which there is no uniform interpolant without fixpoints,

to overcome the more practical problem of computing justifications for uniform interpolants without fixpoint operators, we developed a more refined approach based on ideas for computing justifications.

Given an ontology $\mathcal{O}_1$ and a set of entailed axioms $\mathcal{O}_2$, such as a uniform interpolant, we can compute a justification for $\mathcal{O}_2$ in $\mathcal{O}_1$ using the following algorithm **A1**.

1. Input: ontology $\mathcal{O}_1$, entailed set of axioms $\mathcal{O}_2$
2. For each $\alpha \in \mathcal{O}_1$:
   (a) Set $\mathcal{O}_1' = \mathcal{O}_1 \setminus \{\alpha\}$
   (b) If $\mathcal{O}_1' \models \mathcal{O}_2$, set $\mathcal{O}_1 = \mathcal{O}_1'$
3. Return $\mathcal{O}_1$

If in Step 2b), we test entailment of $\mathcal{O}_2$ by checking one axiom after the other, this algorithm has to perform a quadratic number of entailment tests, namely one for each pair (α, β) of axioms $\alpha \in \mathcal{O}_1$ and $\beta \in \mathcal{O}_2$. However, if $\mathcal{O}_1$ and $\mathcal{O}_2$ overlap syntactically, this number of tests can be reduced, as we do not need to call a reasoner for axioms that are already in $\mathcal{O}_1$. Unfortunately, if $\mathcal{O}_2$ is a uniform interpolant of $\mathcal{O}_1$ for Σ, it only contains axioms in Σ, and is therefore unlikely to syntactically overlap with $\mathcal{O}_1$, especially if Σ is significantly smaller than the signature of $\mathcal{O}_1$. To overcome this problem, we propose the following algorithm **A2**, which checks for entailment of the uniform interpolant by another uniform interpolant.

1. Input: Ontology $\mathcal{O}$, signature Σ
2. Initialise $\mathcal{O}_m$ to the $\top\bot*$-module of $\mathcal{O}$ for Σ
3. Compute the uniform interpolant $\mathcal{O}^\Sigma$ of $\mathcal{O}_m$ for Σ
4. For each $\beta \in \mathcal{O}_m$:
   (a) Compute the uniform interpolant $\mathcal{O}_2^\Sigma$ of $\mathcal{O}_m \setminus \{\beta\}$ for Σ
   (b) Set $\mathcal{O}_d = \mathcal{O}^\Sigma \setminus \mathcal{O}_2^\Sigma$
   (c) If $\mathcal{O}_2^\Sigma \models \mathcal{O}_d$, set $\mathcal{O}_m = \mathcal{O}_m \setminus \{\beta\}$
5. Return $\mathcal{O}_m$

Of course, the improvement of this method relies on the shape of the uniform interpolants we compute: in general, $\mathcal{O}^\Sigma$ and $\mathcal{O}_2^\Sigma$ may not overlap at all, so that $\mathcal{O}_d$ may have the same size as $\mathcal{O}^\Sigma$. However, as our experiments confirmed, if the uniform interpolants are computed wisely, in the algorithm above $\mathcal{O}_d$ is usually significantly smaller than $\mathcal{O}_2^\Sigma$, and in fact often contains only a single axiom. Therefore, the algorithm is expected to require a much smaller number of entailment checks than **A1**. However, we now need to compute a uniform interpolant in every iteration, which usually is a much more expensive operation than testing for entailment of axioms. Luckily, as it turns out, for signatures Σ s.t. $N_r \subseteq \Sigma$, we can compute the required uniform interpolants very efficiently if we compute an *annotated uniform interpolant* first, from which all required uniform interpolants can then be obtained by simple replacement operations.

5 Annotated Uniform Interpolants

In the following, for simplicity, let $\mathcal{O}$ be the input ontology of our method. Let $N_a \subseteq N_c$ be a special set of concept names called *annotation concepts*, which we assume to be disjoint from the signature of $\mathcal{O}$, and let $\mathsf{A} : \mathcal{O} \to N_c$ be a bijective function that maps each axiom in $\mathcal{O}$ to an annotation concept. To improve readability, we write $\mathsf{A}(\alpha)$ as A_α.

Note that in $\mathcal{ALCH}$, every TBox axiom is equivalent to a set of TBox axioms of the form $C \sqsubseteq D$. Given a TBox axiom α, we denote by $\mathsf{gci}(\alpha)$ the set of GCIs that is equivalent to α.

Definition 2. *Given an axiom α, the* annotation α_a of α *is defined as*

$$\{C \sqsubseteq D \sqcup \mathsf{A}_\alpha \mid C \sqsubseteq D \in \mathsf{gci}(\alpha)\}.$$

Given an ontology $\mathcal{O}$, the annotation $\mathcal{O}_a$ of $\mathcal{O}$ is the union of all annotations of axioms in $\mathcal{O}$. Given a signature Σ, a annotated uniform interpolant of $\mathcal{O}$ for Σ *is a uniform interpolant $\mathcal{O}_a^\Sigma$ of the annotation of $\mathcal{O}$ for the signature $\Sigma \cup \{\mathsf{A}_\alpha \mid \alpha \in \mathcal{O}\}$.*

Note that the annotation $\mathcal{O}_a$ of an ontology $\mathcal{O}$ is usually not a conservative extension. Specifically, $\mathcal{O}_a \not\models C \sqsubseteq D$ may not hold even for $C \sqsubseteq D \in \mathcal{O}$, due to the added disjuncts. Instead, all non-tautological entailments now involve annotation concepts that refer to the axioms that have been used to infer the axiom.

The idea of the annotation concepts is to track which axioms contributed to computing a uniform interpolant. This way, we can easily obtain a uniform interpolant of any subset of the original ontology. Specifically, given an annotated uniform interpolant of $\mathcal{O}$ for Σ, where $N_r \subseteq \Sigma$, we can obtain uniform interpolants for Σ of any subset $\mathcal{O}'$ of $\mathcal{O}$ as follows: we replace every annotation concept A_α s.t. $\alpha \in \mathcal{O}'$ by $\bot$, and every remaining annotation concept by $\top$.

Example 1. Consider the following ontology $\mathcal{O}$.

$$\exists r.\top \sqsubseteq A \sqcup B \qquad A \equiv \exists r.B$$

The following ontology is a uniform interpolant of $\mathcal{O}$ for $\Sigma = \{A, r\}$.

$$\begin{aligned} A &\sqsubseteq \exists r.\top \\ \exists r.(\exists r.\top \sqcap \neg A) &\sqsubseteq A \end{aligned}$$

To track which axioms contributed to the uniform interpolant, we compute the annotated uniform interpolant. For this, we first compute the annotation of $\mathcal{O}$, which is the following.

$$\begin{aligned} \exists r.\top &\sqsubseteq A \sqcup B \sqcup \mathsf{A}_{\exists r.\top \sqsubseteq A \sqcup B} \\ A &\sqsubseteq \exists r.B \sqcup \mathsf{A}_{A \equiv \exists r.B} \\ \exists r.B &\sqsubseteq A \sqcup \mathsf{A}_{A \equiv \exists r.B} \end{aligned}$$

By interpolating the annotation of $\mathcal{O}$, we obtain the following annotated uniform interpolant of $\mathcal{O}$ for Σ.

$$A \sqsubseteq \exists r.\top \sqcup \mathsf{A}_{A \equiv \exists r.B}$$
$$\exists r.(\exists r.\top \sqcap \neg A \sqcap \neg \mathsf{A}_{\exists r.\top \sqsubseteq A \sqcup B}) \sqsubseteq A \sqcup \mathsf{A}_{A \equiv \exists r.B}$$

The annotation concepts mark which parts of the uniform interpolant where influenced by which axiom. If we replace every annotation concept by $\bot$, we obtain a uniform interpolant of $\mathcal{O}$ again. If instead, we replace $\mathsf{A}_{A \equiv \exists r.B}$ by $\bot$ and $\mathsf{A}_{\exists r.\top \sqsubseteq A \sqcup B}$ by $\top$, we obtain the following uniform interpolant of $A \equiv \exists r.B$:

$$A \sqsubseteq \exists r.\top \sqcup \bot$$
$$\exists r.(\exists r.\top \sqcap \neg A \sqcap \neg \top) \sqsubseteq A \sqcup \bot,$$

which can be simplified to $\{A \sqsubseteq \exists r.\top\}$, as the second axiom is tautological. In a same way, we obtain that a uniform interpolant of $\exists r.\top \sqsubseteq A \sqcup B$ is $\{\bot \sqsubseteq \top\}$.

This technique however only works for signatures that contain all role symbols of the original ontology. The following lemma, which can be shown by inspection of the uniform interpolation method presented in [15], provides the central property of uniform interpolants which make our technique possible.

Lemma 1. *Let $\mathcal{O}$ be an ontology and Σ a signature s.t. $N_r \subseteq \Sigma$. Let $\mathcal{O}_1 \subseteq \mathcal{O}$ be such that $\mathsf{sig}(\mathcal{O}_1) = \Sigma$ and $\mathcal{O}_1$ contains no RBox axioms, and let $\mathcal{O}_2 = \mathcal{O} \setminus \mathcal{O}_1$. Further, let $\mathcal{O}_2^\Sigma$ be a uniform interpolant of $\mathcal{O}_2$ for Σ. Then, $\mathcal{O}_1 \cup \mathcal{O}_2^\Sigma$ is a uniform interpolant of $\mathcal{O}$ for Σ.*

As a corollary of this lemma, we obtain the robustness property of subsumption modules for signatures Σ s.t. $N_r \subseteq \Sigma$ which was claimed in the introduction, and which can equivalently be proved based on a corresponding result for $\mathcal{ALC}$ from [12].

Corollary 1 (Weak robustness under replacement). *Let $\mathcal{O}$ be an ontology, Σ a signature s.t. $N_r \subseteq \Sigma$, and $\mathcal{M}$ be a Σ-subsumption module for $\mathcal{O}$. Let $\mathcal{O}'$ be an ontology s.t. $\mathsf{sig}(\mathcal{O}') \cap \mathsf{sig}(\mathcal{O}) \subseteq \Sigma$ and $\mathcal{O}'$ contains no RBox axioms containing role names from $\mathcal{O}$. Then, for any axiom α s.t. $\mathsf{sig}(\alpha) \subseteq \mathsf{sig}(\mathcal{O}') \cup \Sigma$, we have $\mathcal{O} \cup \mathcal{O}' \models \alpha$ iff $\mathcal{M} \cup \mathcal{O}' \models \alpha$.*

We can now show that uniform interpolants of subsets of the original ontology can indeed be computed using the replacement operations described earlier. (Recall that annotated uniform interpolants are defined as uniform interpolants of the annotated ontology, for the given signature extended by annotation concepts.)

Theorem 1. *Let $\mathcal{O}$ be an ontology, Σ a signature s.t. $N_r \subseteq \Sigma$, and $\mathcal{O}_a^\Sigma$ an annotated uniform interpolant of $\mathcal{O}$ for Σ. Let $\mathcal{O}_1 \subseteq \mathcal{O}$. Then, the ontology*

$$\mathcal{O}_1^\Sigma = \mathcal{O}_a^\Sigma[\mathsf{A}_\alpha \mapsto \bot \mid \alpha \in \mathcal{O}_1][\mathsf{A}_\alpha \mapsto \top \mid \alpha \in \mathcal{O} \setminus \mathcal{O}_1]$$

is a uniform interpolant of $\mathcal{O}_1$ for Σ.

Proof. Let $\mathcal{O}$, Σ, $\mathcal{O}_a^\Sigma$ and $\mathcal{O}_1$ be as in the lemma. Let $\mathcal{O}_a$ be the annotation of $\mathcal{O}$, and extend $\mathcal{O}_a$ to following ontology $\mathcal{O}_2$.

$$\mathcal{O}_a \cup \{\mathsf{A}_\alpha \equiv \bot \mid \alpha \in \mathcal{O}_1\} \cup \{\mathsf{A}_\alpha \equiv \top \mid \mathsf{A}_\alpha \in \mathcal{O} \setminus \mathcal{O}_1\}$$

By looking at the way axioms are annotated, one easily establishes that the uniform interpolant of $\mathcal{O}_2$ for $\mathsf{sig}(\mathcal{O})$ is equivalent to $\mathcal{O}_1$. Also, one easily sees that this uniform interpolant is simply obtained by replacing every annotation concept A_α s.t. $\alpha \in \mathcal{O}$ by $\bot$, and every annotation concept A_α s.t. $\mathsf{A}_\alpha \in \mathcal{O} \setminus \mathcal{O}_1$ by $\top$. Since uniform interpolation is commutative, we can obtain a uniform interpolant of $\mathcal{O}_1$ for Σ, starting from $\mathcal{O}_2$, in two ways. Either we first compute $\mathcal{O}_1$ as uniform interpolant of $\mathcal{O}_2$, and compute then the uniform interpolant of $\mathcal{O}_1$ for Σ. Or we first compute the uniform interpolant of $\mathcal{O}_2$ for $\Sigma \cup N_a$, of which we then compute the uniform interpolant for Σ. As observed earlier, this last step is simply performed by replacing annotation concepts by $\bot$ respectively $\top$. By Lemma 1, the uniform interpolant of $\mathcal{O}_2$ is equivalent to the uniform interpolant of $\mathcal{O}_a$—the annotated uniform interpolant—together with the additional axioms in $\mathcal{O}_2$, which means, these axioms are only involved in computing the second uniform interpolant. Therefore, we obtain the same ontology if we first compute the annotated uniform interpolant of $\mathcal{O}$, and then perform the substitution on the annotated uniform interpolant. □

6 LK Subsumption Modules

Theorem 1 allows us to apply Algorithm **A2** without having to use an expensive uniform interpolation method in each step. Another consequence of Theorem 1 is that, if we take the axioms associated with the set of annotation concepts occurring in the annotated uniform interpolant, we obtain a set of axioms that contains all minimal subsumption modules. This module is not necessarily equal to the union of all minimal subsumption modules, since the annotated uniform interpolant may contain tautological axioms, so that it is an over-approximation. We call this module *lean kernel subsumption module* (LK subsumption module for short), since it contains all axioms that were involved in computing the uniform interpolant, similar to lean kernels in SAT-solving [19]. LK subsumption modules can be computed more cheaply than minimal subsumption modules, as they do not require any further subsumption tests after the annotated uniform interpolant is computed. We believe that they also have a special use in ontology engineering: given a set of concept names, they allow the ontology engineer to quickly examine all the axioms that are involved in inferring any entailments involving no other concept names.

Definition 3. *Let $\mathcal{O}$ be an ontology and Σ a signature. An ontology $\mathcal{M}_{lk}^\Sigma$ is a Σ LK subsumption module iff there exists an annotated uniform interpolant $\mathcal{O}_a^\Sigma$ of $\mathcal{O}$ for Σ that is obtained by collecting all axioms that belong to some annotation concepts occurring in $\mathcal{O}_a^\Sigma$:*

$$\mathcal{M}_{lk}^\Sigma = \{\alpha \mid \mathsf{A}_\alpha \in \mathsf{sig}(\mathcal{O}_a^\Sigma)\}.$$

Corollary 2. *Given an ontology $\mathcal{O}$, a signature Σ s.t. $N_r \subseteq \Sigma$ and a Σ LK subsumption module $\mathcal{M}_{lk}^{\Sigma}$ for $\mathcal{O}$, we have $\mathcal{M} \subseteq \mathcal{M}_{lk}^{\Sigma}$ for every minimal Σ-subsumption module $\mathcal{M}$ of $\mathcal{O}$.*

Since LK subsumption modules can be obtained from the annotated uniform interpolant without additional subsumption tests, they can always be computed even in the case where the annotated uniform interpolant contains fixpoint operators. However, different to minimal subsumption modules they do not offer any minimality guarantees: which axioms are contained in the LK subsumption module depends on the uniform interpolation procedure used, and as uniform interpolants may contain redundant and tautological information, it is in general possible that an LK subsumption module contains axioms that are not included in any minimal subsumption module.

The requirement that the signature contains all role names is indeed crucial for the correctness of our method, as is exemplified by the following example.

Example 2. Take following ontology $\mathcal{O}$:

$$A \sqsubseteq \exists r.B \qquad A \sqsubseteq C \qquad B \sqsubseteq \bot,$$

and consider the signature $\Sigma_1 = \{A, r\}$. The annotation $\mathcal{O}_a$ of $\mathcal{O}$ looks as follows.

$$A \sqsubseteq \exists r.B \sqcup \mathsf{A}_{A \sqsubseteq \exists r.B}$$
$$A \sqsubseteq C \sqcup \mathsf{A}_{A \sqsubseteq C}$$
$$B \sqsubseteq \bot \sqcup \mathsf{A}_{B \sqsubseteq \bot}$$

A uniform interpolant $\mathcal{O}_a^{\Sigma}$ for $\mathcal{O}_a$ for $\{A, r, \mathsf{A}_{A \sqsubseteq \exists r.B}, \mathsf{A}_{A \sqsubseteq C}, \mathsf{A}_{B \sqsubseteq \bot}\}$ is

$$A \sqsubseteq \exists r.\mathsf{A}_{B \sqsubseteq \bot} \sqcup \mathsf{A}_{A \sqsubseteq \exists r.B}.$$

Consequently, the Σ_1 LK subsumption module contains the first and the last axiom of $\mathcal{O}$. One easily verifies that these two axioms also form the only minimal Σ_1-subsumption module of $\mathcal{O}$.

Now consider the signature $\Sigma_2 = \{A, C\}$. A close look at the original ontology shows that in fact, the concept A is unsatisfiable, which can be inferred just from the first and the last axiom. This makes the second axiom redundant, and therefore the minimal subsumption module for Σ_2 is the same as for Σ_1. However, the uniform interpolant $\mathcal{O}_a^{\Sigma_2}$ for $\mathcal{O}_a$ for $\{A, C, \mathsf{A}_{A \sqsubseteq \exists r.B}, \mathsf{A}_{A \sqsubseteq C}, \mathsf{A}_{B \sqsubseteq \bot}\}$ just contains the following axiom.

$$A \sqsubseteq C \sqcup \mathsf{A}_{A \sqsubseteq C}$$

That is, the LK subsumption module contains just one axiom, which is exactly the only axiom that does not occur in the minimal subsumption module. The reason that the annotated uniform interpolant does not contain any more axioms is that the axiom $A \sqsubseteq \exists r.\mathsf{A}_{B \sqsubseteq \bot} \sqcup \mathsf{A}_{A \sqsubseteq \exists r.B}$ has no non-tautological $\mathcal{ALCH}$-entailment that does not also make use of the role name r.

Finally, for the complete signature $\Sigma_3 = \{A, r, C\}$, the Σ_3 LK subsumption module contains all axioms. However, as we already observed, the second axiom is redundant, and thus, this subsumption module is not minimal.

7 Evaluation

To evaluate how our method performs in practice, we implemented a Java prototype of our method and did an initial evaluation on a set of 96 ontologies that were taken from the classification track for OWL DL ontologies at the ORE competition 2014 [24]. The prototype was implemented in Java 1.7, using the OWL-API [9] for ontology access, Lethe [17] for computing the annotated uniform interpolants, and MORe [25] as a reasoner in the minimisation step. MORe is a hybrid reasoner that utilises the OWL DL reasoner HermiT [7] together with the $\mathcal{EL}$ reasoner ELK [10]. As it was to be expected that the module as well as the uniform interpolants could often be expressed purely in $\mathcal{EL}$, this reasoner was selected to improve reasoner performance for these cases. In fact, we noticed that the minimalisation step was usually significantly faster when using MORe than when using HermiT.

We were particularly interested in the performance for computing small subsumption modules. In particular, we were interested in the following questions: 1) how well does the method perform in practice for computing LK subsumption modules and minimal subsumption modules, and 2) can minimal subsumption modules in $\mathcal{ALCH}$ be expected to be smaller than modules extracted by alternative methods, such as locality-based modules. Since we expected both the computation of the annotated uniform interpolant, as well as the minimisation step afterwards, to be costly in practice, we computed subsumption modules in a relaxed setting. For this, we used a timeout for the uniform interpolation step. Lethe computes uniform interpolants by eliminating names outside of the specified signature one after the other. If it did not succeed in computing the uniform interpolant within the specified timeout, we continued the computation with the uniform interpolant it computed so far, thus obtaining LK subsumption modules and minimal subsumption modules for an extended signature. This way, we were able to get an upper bound on the size of the minimal subsumption module even if the annotated uniform interpolant was too difficult to compute. Furthermore, note that the annotated uniform interpolant computed in the first step may contain fixpoint expressions, so that the LK subsumption module cannot be minimised in the second step, as we cannot test for entailment of axioms with fixpoint expressions using MORe. To still get an idea about the minimisation potential of our approach, we simply treated entailment of axioms with fixpoint expressions as failure, unless the axiom was syntactically contained in the current module. Note that this may result in a module that is not minimal, similar to when the uniform interpolation procedure created a timeout.

The ontologies for our experiments were selected as follows. We selected our ontologies from the track *"Classification of OWL DL ontologies"* of the OWL Reasoner Evaluation competition 2015, because they provide a small, yet well balanced mix of ontologies with very different properties [24]. As our method only supports $\mathcal{ALCH}$ ontologies, we further removed from each ontology the axioms that were not in $\mathcal{ALCH}$, where we kept n-ary equivalence and disjointness axioms, as well as concept inclusions. From this set of 315 ontologies, we selected

$\lvert\Sigma \cap N_c\rvert$	10	25	50
Success Rate	92.2%	90.3%	90.6%
Minimal	71.2%	70.3%	68.7%
Fixpoints	20.8%	19.8%	23.6%
UI Timeout	2.0%	2.2%	2.0%
Duration (s)	0.4 / 594.0 / 3.7 / 20.6	0.4 / 591.9 / 4.1 / 27.1	0.5 / 592.0 / 6.1 / 32.5
Size $\top\bot*$-Module	0 / 1021 / 131.0 / 204.4	0 / 1374 / 164.0 / 256.7	0 / 1047 / 204.0 / 260.3
Size LK Module	0 / 723 / 103.5 / 170.1	0 / 1054 / 144.0 / 216.7	0 / 835 / 170.0 / 218.5
Size Min. Module	0 / 723 / 103.0 / 169.8	0 / 1051 / 142.0 / 216.1	0 / 814 / 169.0 / 218.0
Size Original Ontology	186 / 8926 / 1792.0 / 2547.5		

Table 1. Results of our evaluation for signatures containing 10/25/50 concept names (minimal/maximal/median/average).

$\lvert\Sigma \cap N_c\rvert$	100	150
Success Rate	87.7%	85.7%
Minimal	66.5%	66.0%
Fixpoints	21.7%	20.6%
UI Timeout	2.6%	1.7%
Duration (s)	0.9 / 584.5 / 9.1 / 39.1	1.0 / 594.9 / 11.6 / 39.7
Size $\top\bot*$-Mod	1 / 1374 / 305.0 / 352.2	6 / 1242 / 385.0 / 435.8
Size LK Mod	1 / 1054 / 267.0 / 301.2	6 / 1242 / 342.0 / 382.1
Size Minimised Mod	1 / 1051 / 264.0 / 300.0	6 / 1234 / 336.0 / 379.9
Size Original Ontology	186 / 8926 / 1792.0 / 2547.5	

Table 2. Results of our evaluation for signatures containing 100/150 concept names (minimal/maximal/median/average).

those that contained less than 10,000 axioms and more than 150 concept names, resulting in a set of 96 ontologies in total.

The experiment was performed on a server running Ubuntu 15.10 with Intel Xeon 2.50GHz cluster Core 4 Duo CPU with 64GiB RAM. We used our prototype to compute subsumption modules for randomly created signatures containing 10, 25, 50, 100 and 150 concept names, were we used 30 samples per signature size. The timeout for the interpolation procedure was set to 5 minutes, while the overall timeout was set to 10 minutes. The results of our experiment are shown in Table 2. The success rate shows the number of runs in which the method succeeded within the timeout. The next rows show the percentage of runs that produced modules which were guaranteed to be minimal (Row 3), that did not guarantee minimality due to fixpoints in the uniform interpolant (Row 4), and that did not guarantee minimality due to a timeout in the interpolation procedure (Row 5). Note that the latter two cases may overlap. We then list the minimal, maximal, median and average duration of computing the module in the successful runs. We obtained median durations between 4.1 and 11.6 seconds, which might be reasonable for daily applications. However, as to be expected, in general our method is more computationally expensive than other methods for module extraction.

The following rows compare the sizes of the TBoxes of the $\top\bot*$-modules, the LK subsumption modules and the minimised subsumption modules. Note

that, because the signatures always contained all role names, the $\top\bot*$-modules as well as the subsumption modules always had the same RBox as the original ontology. It is for this reason that we focus on the *TBoxes* of the modules to provide for a more meaningful comparison. The LK subsumption modules were on average 12.3%–16.8% smaller than the $\top\bot*$-modules, while the minimised variants differed only little in size to the LK subsumption modules. This shows that in practice, LK subsumption modules provide for good approximations of minimal subsumption modules. The last row shows for comparison the minimal, maximal, median and average size of the input ontologies.

8 Conclusion and Future Work

We presented a method for extracting subsumption modules in $\mathcal{ALCH}$ for signatures Σ s.t. $N_r \subseteq \Sigma$. The method ensures minimality of the extracted modules provided that a uniform interpolant without fixpoint operators can be computed for the given ontology and signature. In the first step, the method computes an annotated uniform interpolant, from which an approximation of the minimal subsumption module, the LK subsumption module can be obtained. This module is then minimised in a subsequent step by comparing entailments of corresponding uniform interpolants. Our evaluation indicates that in most cases, the LK subsumption module is already minimal or close-to minimal, so that the second step can be omitted. As for LK subsumption modules, we do not have a restriction for cyclic ontologies, this means that our method provides for good approximations of minimal subsumption modules for $\mathcal{ALCH}$ ontologies in general. Our evaluation indicates that our method is often able to compute subsumption modules that are smaller than locality-based modules. However, in some cases, the computation of these modules was quite time-consuming. Our implementation uses a timeout for the uniform interpolation step, and computes a subsumption module for the signature for which a uniform interpolant could be computed within that timeout. It would be interesting to see the exact effect of this timeout on the computed subsumption modules. For small timeouts, our implementation would compute an LK subsumption module for an extended signature that in addition contains those concept names that are especially hard for Lethe to eliminate, which usually is only a small fraction of complete signature. It is possible that those LK subsumption modules provide for close approximations of the LK subsumption modules for the given signature, though computable in much shorter time.

There are obvious short-comings of our evaluation that should be addressed in future work. First, the sample size used for the signatures was very small, which is why we are currently running the experiments with a higher number of signatures per ontology. Second, we only compared our method with the syntactical method for computing $\top\bot*$-modules. An alternative obvious choice for comparison would be AMEX [5], which can approximate depleting semantic modules of $\mathcal{ALCQI}$ ontologies. Such a comparison could give insights on whether subsumption modules are in practice smaller than semantic modules, or whether

they are mostly similar. Third, it would be interesting to evaluate our method on larger ontologies and not only ontologies with at most 10,000 axioms. Furthermore, it would be interesting to test our method with other implementations for uniform interpolation than LETHE, for example the Ackermann-based method presented in [31].

Open problems with our approach are 1) how to obtain an optimal, practical method in the case uniform interpolants contain fixpoints, and 2) how to compute minimal subsumption modules for signatures that do not contain all role names. In order to tackle these, it might be necessary to use the uniform interpolation method not as a black box, but to modify its implementation to track inferences directly. Again, it is possible that techniques from the area of justification and axiom-pinpointing could be used here.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA (2007)
2. Botoeva, E., Konev, B., Lutz, C., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Inseparability and conservative extensions of description logic ontologies: A survey. In: Pan, J.Z., Calvanese, D., Eiter, T., Horrocks, I., Kifer, M., Lin, F., Zhao, Y. (eds.) Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering: 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures. pp. 27–89. Springer International Publishing, Cham (2017), https://doi.org/10.1007/978-3-319-49493-7_2
3. Chen, J., Ludwig, M., Ma, Y., Walther, D.: Zooming in on ontologies: Minimal modules and best excerpts. In: The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I. pp. 173–189 (2017)
4. Chen, J., Ludwig, M., Walther, D.: On computing minimal $\mathcal{EL}$-subsumption modules. In: Proceedings of the Joint Ontology Workshops 2016 Episode 2: The French Summer of Ontology co-located with the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016), Annecy, France, July 6-9, 2016. (2016), http://ceur-ws.org/Vol-1660/womocoe-paper6.pdf
5. Gatens, W., Konev, B., Wolter, F.: Lower and upper approximations for depleting modules of description logic ontologies. In: Proceedings of the Twenty-first European Conference on Artificial Intelligence. pp. 345–350. IOS Press (2014)
6. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006. pp. 187–197 (2006), http://www.aaai.org/Library/KR/2006/kr06-021.php
7. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
8. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research (JAIR) 31(1), 273–318 (2008)

9. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
10. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK. Journal of Automated Reasoning 53(1), 1–61 (2014)
11. Konev, B., Ludwig, M., Walther, D., Wolter, F.: The logical difference for the lightweight description logic $\mathcal{EL}$. Journal of Artificial Intelligence Research (JAIR) 44, 633–708 (2012)
12. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal properties of modularisation. In: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, pp. 25–66. Springer (2009), https://doi.org/10.1007/978-3-642-01907-4_3
13. Konev, B., Lutz, C., Walther, D., Wolter, F.: Model-theoretic inseparability and modularity of description logic ontologies. Artificial Intelligence 203, 66–103 (2013)
14. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to DL-Lite. Artificial Intelligence 174(15), 1093–1141 (2010)
15. Koopmann, P., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In: Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings. pp. 552–567 (2013), https://doi.org/10.1007/978-3-642-45221-5_37
16. Koopmann, P., Schmidt, R.A.: Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Automated Reasoning: 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings. pp. 434–448. Springer International Publishing, Cham (2014), https://doi.org/10.1007/978-3-319-08587-6_34
17. Koopmann, P., Schmidt, R.A.: Lethe: Saturation-based reasoning for non-standard reasoning tasks. In: Proc. ORE'15. pp. 23–30 (2015)
18. Koopmann, P., Schmidt, R.A.: Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with ABoxes. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 175–181 (2015), http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9981
19. Kullmann, O., Lynce, I., Marques-Silva, J.: Categorisation of clauses in conjunctive normal forms: Minimally unsatisfiable sub-clause-sets and the lean kernel. In: Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings. pp. 22–35 (2006), https://doi.org/10.1007/11814948_4
20. Ludwig, M., Konev, B.: Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014 (2014), http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7985
21. Ludwig, M., Walther, D.: The logical difference for $\mathcal{ELH}^r$-terminologies using hypergraphs. In: ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014). pp. 555–560 (2014), https://doi.org/10.3233/978-1-61499-419-0-555
22. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: IJCAI 2011, Proceedings of the 22nd International

Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. pp. 989–995 (2011), https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170
23. Nikitina, N., Rudolph, S.: (Non-)succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. Artif. Intell. 215, 120–140 (2014), https://doi.org/10.1016/j.artint.2014.06.005
24. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL reasoner evaluation (ORE) 2015 competition report. Journal of Automated Reasoning pp. 1–28 (2015)
25. Romero, A.A., Grau, B.C., Horrocks, I.: MORe: Modular combination of owl reasoners for ontology classification. In: International Semantic Web Conference. pp. 1–16. Springer (2012)
26. Romero, A.A., Kaminski, M., Grau, B.C., Horrocks, I.: Module extraction in expressive ontology languages via datalog reasoning. J. Artif. Intell. Res. 55, 499–564 (2016), https://doi.org/10.1613/jair.4898
27. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proceedings of DL'09. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
28. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003. pp. 355–362 (2003), http://ijcai.org/Proceedings/03/Papers/053.pdf
29. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: science, services and agents on the World Wide Web 5(2), 51–53 (2007)
30. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. Automated reasoning pp. 292–297 (2006)
31. Zhao, Y., Schmidt, R.A.: Forgetting concept and role symbols in $\mathcal{ALCOIH}\mu+(\nabla,\cap)$-ontologies. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 1345–1352. AAAI Press (2016)

Towards Elimination of Second-Order Quantifiers in the Separated Fragment

Marco Voigt ✉

Max Planck Institute for Informatics and Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Saarbrücken, Germany
mvoigt@mpi-inf.mpg.de

Abstract. It is a classical result that the monadic fragment of second-order logic admits elimination of second-order quantifiers. Recently, the separated fragment (SF) of first-order logic has been introduced. SF generalizes the monadic first-order fragment without equality, while preserving decidability of the satisfiability problem. Therefore, it is a natural question to ask whether SF also admits elimination of second-order quantifiers. Interestingly, already Ackermann answered this question in the negative as far as full SF with unrestricted occurrences of second-order quantifiers is concerned. However, with appropriate restrictions on the syntax of a second-order version of SF, one could hope to define a substantial extension of the monadic fragment that admits second-order quantifier elimination. The present note is about preliminary results of ongoing research in this direction. As a first positive result a restricted second-order version of SF is defined that admits the elimination of at least one existential second-order quantifier. The elimination of existential second-order quantifiers from a monadic sentence without equality constitutes a special case of the methods presented here.

Keywords: Second-order quantifier elimination · separated fragment · monadic fragment

1 Introduction

It is a classical result that the monadic fragment of second-order logic admits elimination of second-order quantifiers. This was discovered by Löwenheim [6], Skolem [7], and Behmann [2].

Recently, the *separated fragment (SF)* of first-order logic has been introduced [8]. It constitutes a syntactic generalization of well-known first-order fragments: the Bernays–Schönfinkel–Ramsey fragment—the class of relational $\exists^*\forall^*$ sentences—and the monadic first-order fragment without equality—the class of relational sentences over predicate symbols of arity at most one. The satisfiability problem for SF sentences (*SF-Sat*) is decidable, but computationally very hard: SF-Sat is k-NExpTime-hard for every positive integer k [10]. In other words, SF-Sat is non-elementary. The definition of SF is based on restricting the syntax of first-order sentences in prenex normal form. However, neither the

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at http://ceur-ws.org.

arity of predicate symbols nor the shape of quantifier prefixes is restricted. The defining principle for SF sentences is that universally and existentially quantified variables do not occur together in atoms. Leading existential quantifiers are exempt from this rule. The sentence $\forall x_1 \exists y_1 \forall x_2 \exists y_2.\, R(x_1, x_2) \leftrightarrow Q(y_1, y_2)$ is an exemplary SF sentence.

As SF generalizes the monadic first-order fragment without equality while retaining a decidable satisfiability problem, it is natural to ask whether a second-order version of SF admits elimination of second-order quantifiers. Interestingly, already Ackermann gave a negative answer to this question. In an article from 1935 [1], Ackermann argued that the quantifier $\exists P$ in the following formula cannot be eliminated: $\exists P.\ P(x) \wedge \neg P(y) \wedge \forall uv.\ \neg P(u) \vee P(v) \vee \neg N(u, v)$. The only atom in this formula that could potentially break the separateness condition is $N(u, v)$. But since both variables u and v are universally quantified, universal variables are separated from existential variables and the sentence is in SF.

Although Ackermann's observation seems to be discouraging, it only means that there is, apparently, no straight-forward way of extending the quantifier-elimination techniques that work for second-order monadic logic to the separated fragment. The purpose of this note is to present certain syntactic restrictions that allow the elimination of existentially quantified unary predicate symbols in separated formulas. The presented results are of a preliminary character and are not yet fully developed. They provide only a first hint at some directions that might be worth following in future work.

In Section 2 we present the used notation and some basic results. A definition of the separated fragment is given in Section 3. The main result is developed in Section 4 and concisely formulated in Theorem 8. Finally, Section 5 concludes with a discussion of the results and future directions.

2 Notation and Preliminaries

We consider second-order logic formulas with equality. We call a formula *relational* if it contains neither function nor constant symbols. In all formulas, if not explicitly stated otherwise, we tacitly assume that no variable occurs freely and bound at the same time and that no variable is bound by two different occurrences of quantifiers. For convenience, we sometimes identify tuples $\bar{\mathbf{x}}$ of variables with the set containing all the variables that occur in $\bar{\mathbf{x}}$. By $\mathrm{vars}(\varphi)$ we denote the set of all variables occurring in φ.

The symbol $\models$ denotes the *is-a-model-of* relation as well as semantic entailment of formulas, i.e. $\varphi \models \psi$ holds whenever for every structure $\mathcal{A}$ and every variable assignment β, $\mathcal{A}, \beta \models \varphi$ entails $\mathcal{A}, \beta \models \psi$. The symbol $\models\!\!\!\models$ denotes *semantic equivalence* of formulas, i.e. $\varphi \models\!\!\!\models \psi$ holds whenever $\varphi \models \psi$ and $\psi \models \varphi$.

The following are standard lemmas that we simply add for completeness.

Lemma 1 (Miniscoping). *Let φ, ψ, χ be formulas, and assume that x and y do not occur freely in χ. We have the following equivalences, where $\circ \in \{\wedge, \vee\}$:*

(i) $\exists y.(\varphi \vee \psi) \models\!\!\!\models (\exists y.\varphi) \vee (\exists y.\psi)$ *(ii)* $\forall x.(\varphi \wedge \psi) \models\!\!\!\models (\forall x.\varphi) \wedge (\forall x.\psi)$

(iii) $\exists y.(\varphi \circ \chi) \models\!\!\!\models (\exists y.\varphi) \circ \chi$ *(iv)* $\forall x.(\varphi \circ \chi) \models\!\!\!\models (\forall x.\varphi) \circ \chi$

Lemma 2. *Let $\psi[t]$ be some second-order formula in which the term t occurs. Let x be some first-order variable that does not occur in $\psi[t]$. Then, $\psi[t]$ is semantically equivalent to $\forall x.\, x = t \to \psi[x]$, where $\psi[x]$ is derived from $\psi[t]$ by replacing every occurrence of t with the variable x.*

3 The Separated Fragment

Consider a second-order formula φ. We say that two disjoint sets of first-order variables X and Y are *separated in* φ if and only if for every atom A in φ we have $\mathrm{vars}(A) \cap X = \emptyset$ or $\mathrm{vars}(A) \cap Y = \emptyset$.

The following definition of the separated fragment is a slightly simplified version of the fragment investigated in [8] and in [10]. In contrast to the original, we do not consider constant symbols here.

Definition 3 (Separated fragment (SF)). *The* separated fragment (SF) *of first-order logic consists of all relational first-order sentences with equality that are of the form $\exists \bar{\mathbf{z}} \forall \bar{\mathbf{x}}_1 \exists \bar{\mathbf{y}}_1 \ldots \forall \bar{\mathbf{x}}_n \exists \bar{\mathbf{y}}_n.\, \psi$, in which ψ is quantifier free, and in which the two sets $\bar{\mathbf{x}}_1 \cup \ldots \cup \bar{\mathbf{x}}_n$ and $\bar{\mathbf{y}}_1 \cup \ldots \cup \bar{\mathbf{y}}_n$ are separated. The tuples $\bar{\mathbf{z}}$ and $\bar{\mathbf{y}}_n$ may be empty, i.e. the quantifier prefix does not have to start with an existential quantifier and it does not have to end with an existential quantifier either.*

Notice that the variables in $\bar{\mathbf{z}}$ are not subject to any restriction concerning their occurrences.

It is not hard to see that SF generalizes the Bernays–Schönfinkel–Ramsey fragment (relational $\exists^*\forall^*$ prenex formulas with equality) and the monadic first-order fragment without equality (see [8], Theorem 9). The reason why certain monadic sentences with equality do not belong to SF is that, although SF sentences may contain equality, non-separated equations are not allowed in SF. For example, the sentence $\exists y \forall x.\, x = y$ belongs to SF whereas $\forall x \exists y.\, x = y$ does not.

As already mentioned in the introduction, it is known that the satisfiability problem for SF sentences (*SF-Sat*) is decidable and non-elementary [8, 10]. These results rely on an equivalence-preserving transformation from SF into the Bernays–Schönfinkel–Ramsey fragment (BSR): for every SF sentence there is an equivalent sentence in the BSR fragment. This transformation will be the starting point for showing that quantifier elimination is possible for a certain extension of the separated fragment with second-order quantifiers.

Lemma 4. *Let $\varphi := \forall \bar{\mathbf{x}}_1 \exists \bar{\mathbf{y}}_1 \ldots \forall \bar{\mathbf{x}}_n \exists \bar{\mathbf{y}}_n.\, \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$ be a relational first-order formula in which ψ is quantifier free and the sets $\bar{\mathbf{x}} := \bar{\mathbf{x}}_1 \cup \ldots \cup \bar{\mathbf{x}}_n$ and $\bar{\mathbf{y}} := \bar{\mathbf{y}}_1 \cup \ldots \cup \bar{\mathbf{y}}_n$ are separated. Moreover, we assume that every variable occurring in the quantifier prefix and in $\bar{\mathbf{z}}$ also occurs in the matrix ψ.*

Let $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{m_1} \subseteq \bar{\mathbf{x}}$ and $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{m_2} \subseteq \bar{\mathbf{y}}$ be partitions of the sets $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$, respectively, such that the $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{m_1}, \widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{m_2}$ are nonempty, pairwise disjoint, and pairwise separated in φ. Then, φ is equivalent to a finite disjunction of formulas of the form

$$\Big(\bigwedge_k \forall\widetilde{\mathbf{x}}''_k.\ \bigvee_\ell K_{k\ell}(\widetilde{\mathbf{x}}''_k, \bar{\mathbf{z}})\Big)\ \wedge\ \Big(\bigwedge_i \exists\widetilde{\mathbf{y}}''_i.\ \bigwedge_j L_{ij}(\widetilde{\mathbf{y}}''_i, \bar{\mathbf{z}})\Big)\ ,$$

where the $K_{k\ell}$ and the L_{ij} are literals whose atoms are renamed variants of atoms that occur in φ. Moreover, any two sets $\widetilde{\mathbf{x}}''_{k_1}, \widetilde{\mathbf{x}}''_{k_2}$ with $k_1 \neq k_2$, $\widetilde{\mathbf{y}}''_{i_1}, \widetilde{\mathbf{y}}''_{i_2}$ with $i_1 \neq i_2$, and $\widetilde{\mathbf{x}}''_k, \widetilde{\mathbf{y}}''_i$ are separated in the resulting formula.

Proof. The proof is an adaptation of the proof of Lemma 12 in [10]. For convenience, we pretend that $\bar{\mathbf{z}}$ is empty. The argument works for nonempty $\bar{\mathbf{z}}$ as well. We will make use of the following auxiliary lemma:

Claim I (Lemma 11 in [10]):
Let I and J_i, $i \in I$, be sets that are finite, nonempty, and pairwise disjoint. The elements of these sets serve as indices. Let

$$\exists\bar{\mathbf{v}}.\bigwedge_{i\in I}\Big(\chi_i(\bar{\mathbf{u}}) \vee \bigvee_{k\in J_i} \eta_k(\bar{\mathbf{v}}, \bar{\mathbf{u}})\Big)$$

be some first-order formula where the χ_i and the η_k denote arbitrary subformulas that we treat as indivisible units in what follows. We say that $f : I \to \bigcup_{i\in I} J_i$ is a *selector* if for every $i \in I$ we have $f(i) \in J_i$. We denote the set of all selectors of this form by $\mathcal{F}$.
Then, the above formula is equivalent to

$$\bigwedge_{\substack{S\subseteq I\\ S\neq\emptyset}} \Big(\bigvee_{i\in S}\chi_i(\bar{\mathbf{u}})\Big) \vee \bigvee_{f\in\mathcal{F}}\Big(\exists\bar{\mathbf{v}}.\bigwedge_{i\in S}\eta_{f(i)}(\bar{\mathbf{v}}, \bar{\mathbf{u}})\Big)\ .$$

◊

We transform φ into an equivalent CNF formula of the form

$$\forall\bar{\mathbf{x}}_1\exists\bar{\mathbf{y}}_1\ldots\forall\bar{\mathbf{x}}_n\exists\bar{\mathbf{y}}_n.\ \bigwedge_{i\in I}\Big(\chi_i(\bar{\mathbf{x}}) \vee \bigvee_{k\in J_i} L_k(\bar{\mathbf{y}})\Big)$$

where I and the J_i are finite, pairwise disjoint sets of indices, the subformulas χ_i are disjunctions of literals, and the L_k are literals. By Claim I, we can construct an equivalent formula of the form

$$\varphi' := \forall\bar{\mathbf{x}}_1\exists\bar{\mathbf{y}}_1\ldots\forall\bar{\mathbf{x}}_n.\ \bigwedge_{\substack{S\subseteq I\\ S\neq\emptyset}} \Big(\bigvee_{i\in S}\chi_i(\bar{\mathbf{x}})\Big) \vee \bigvee_{f\in\mathcal{F}}\Big(\exists\bar{\mathbf{y}}_n.\bigwedge_{i\in S}\eta_{f(i)}(\bar{\mathbf{y}})\Big)$$

where $\mathcal{F}$ is the set of all selectors over the index sets J_i, $i \in I$. Applying miniscoping (Lemma 1), we move inward the universal quantifier block $\forall\bar{\mathbf{x}}_n$ and thus obtain

$$\varphi'' := \forall\bar{\mathbf{x}}_1\exists\bar{\mathbf{y}}_1\ldots\exists\bar{\mathbf{y}}_{n-1}.\ \bigwedge_{\substack{S\subseteq I\\ S\neq\emptyset}} \Big(\forall\bar{\mathbf{x}}_n.\bigvee_{i\in S}\chi_i(\bar{\mathbf{x}})\Big) \vee \bigvee_{f\in\mathcal{F}}\Big(\exists\bar{\mathbf{y}}_n.\bigwedge_{i\in S}\eta_{f(i)}(\bar{\mathbf{y}})\Big)\ .$$

We now iterate these two steps in an alternating fashion until all quantifier blocks have been moved inwards in the described way. The constituents of the result $\varphi^{(3)} := \bigwedge_q \Big(\chi_q^{(3)} \vee \bigvee_p \eta_{qp}^{(3)}\Big)$ of this process have the form

$$\chi_q^{(3)} = \forall \bar{\mathbf{x}}_1. \bigvee_{\ell_1} \forall \bar{\mathbf{x}}_2. \bigvee_{\ell_2} \Big(\dots \Big(\bigvee_{\ell_{n-1}} \forall \bar{\mathbf{x}}_n. \bigvee_{i \in S_{\ell_1,\dots,\ell_{n-1}}} \chi_i(\bar{\mathbf{x}}) \Big) \dots \Big)$$

where the $S_{\ell_1,\dots,\ell_n - 1}$ are certain subsets of I and the χ_i are still disjunctions of literals, and

$$\eta_{qp}^{(3)} = \exists \bar{\mathbf{y}}_1. \bigwedge_{\ell_1} \exists \bar{\mathbf{y}}_2. \bigwedge_{\ell_2} \Big(\dots \Big(\bigwedge_{\ell_{n-1}} \exists \bar{\mathbf{y}}_n. \bigwedge_{k \in J_{\ell_1,\dots,\ell_{n-1}}} L_k(\bar{\mathbf{y}}) \Big) \dots \Big)$$

where the $J_{\ell_1,\dots,\ell_{n-1}}$ are certain subsets of $\bigcup_{i \in I} J_i$.

By definition of the sets $\widetilde{\mathbf{x}}_1, \dots, \widetilde{\mathbf{x}}_{m_1}$, which are pairwise separated in the $\chi_q^{(3)}$, we can rewrite every $\chi_q^{(3)}$ into the following form by regrouping the inner disjuncts:

$$\chi_q^{(4)} = \forall \bar{\mathbf{x}}_1. \bigvee_{\ell_1} \forall \bar{\mathbf{x}}_2. \bigvee_{\ell_2} \Big(\dots \Big(\bigvee_{\ell_{n-1}} \forall \bar{\mathbf{x}}_n. \bigvee_{i' = 1,\dots,m_1} \chi'_{\bar{\ell} i'}(\widetilde{\mathbf{x}}_{i'}) \Big) \dots \Big)$$

where the $\chi'_{\bar{\ell} i'}$ are (possibly empty) disjunctions of literals. Analogously, we rewrite every $\eta_{qp}^{(3)}$ into the form

$$\eta_{qp}^{(4)} = \exists \bar{\mathbf{y}}_1. \bigwedge_{\ell_1} \exists \bar{\mathbf{y}}_2. \bigwedge_{\ell_2} \Big(\dots \Big(\bigwedge_{\ell_{n-1}} \exists \bar{\mathbf{y}}_n. \bigwedge_{j' = 1,\dots,m_2} \eta'_{\bar{\ell} j'}(\widetilde{\mathbf{y}}_j) \Big) \dots \Big)$$

where the $\eta'_{\bar{\ell} j'}$ are (possibly empty) conjunctions of literals.

We then observe the following equivalences, starting from $\chi_q^{(4)}$:

$$\begin{aligned}
&\forall \bar{\mathbf{x}}_1. \bigvee_{\ell_1} \forall \bar{\mathbf{x}}_2. \bigvee_{\ell_2} \Big(\dots \Big(\bigvee_{\ell_{n-1}} \forall \bar{\mathbf{x}}_n. \bigvee_{i' = 1,\dots,m_1} \chi'_{\bar{\ell} i'}(\widetilde{\mathbf{x}}_{i'}) \Big) \dots \Big) \\
&\eqsim \forall \bar{\mathbf{x}}_1. \bigvee_{\ell_1} \forall \bar{\mathbf{x}}_2. \bigvee_{\ell_2} \Big(\dots \Big(\bigvee_{\ell_{n-1}} \bigvee_{i' = 1,\dots,m_1} \forall (\bar{\mathbf{x}}_n \cap \widetilde{\mathbf{x}}_{i'}).\ \chi'_{\bar{\ell} i'}(\widetilde{\mathbf{x}}_{i'}) \Big) \dots \Big) \\
&\eqsim \forall \bar{\mathbf{x}}_1. \bigvee_{\ell_1} \forall \bar{\mathbf{x}}_2. \bigvee_{\ell_2} \Big(\dots \Big(\bigvee_{i' = 1,\dots,m_1} \bigvee_{\ell'_{n-1}} \forall (\bar{\mathbf{x}}_n \cap \widetilde{\mathbf{x}}_{i'}).\ \chi'_{\bar{\ell}' i'}(\widetilde{\mathbf{x}}_{i'}) \Big) \dots \Big) \\
&\qquad \vdots \\
&\eqsim \bigvee_{i' = 1,\dots,m_1} \forall (\bar{\mathbf{x}}_1 \cap \widetilde{\mathbf{x}}_{i'}). \bigvee_{\ell'_1} \forall (\bar{\mathbf{x}}_2 \cap \widetilde{\mathbf{x}}_{i'}). \bigvee_{\ell'_2} \Big(\dots \Big(\bigvee_{\ell'_{n-1}} \forall (\bar{\mathbf{x}}_n \cap \widetilde{\mathbf{x}}_{i'}).\ \chi'_{\bar{\ell}' i'}(\widetilde{\mathbf{x}}_{i'}) \Big) \dots \Big) \\
&\eqsim \bigvee_{i' = 1,\dots,m_1} \forall \widetilde{\mathbf{x}}'_{i'}.\ \chi''_{i'}(\widetilde{\mathbf{x}}'_{i'}) \ ,
\end{aligned}$$

where the $\chi''_{i'}$ are disjunctions of literals. Before moving universal quantifiers outwards in the last step of the above transformation, bound variables are renamed such that all quantifiers bind pairwise distinct variables. Analogously, we have

$$\eta^{(4)}_{qp} \models \bigwedge_{j'=1,\ldots,m_2} \exists \widetilde{\mathbf{y}}'_{j'}.\ \eta''_{j'}(\widetilde{\mathbf{y}}'_{i'}) \ ,$$

where the $\eta''_{j'}$ are conjunctions of literals.

Consequently, we have rewritten $\varphi^{(3)} = \bigwedge_q \Big(\chi^{(3)}_q \vee \bigvee_p \eta^{(3)}_{qp}\Big)$ into an equivalent formula $\varphi^{(4)}$ of the form

$$\varphi^{(4)} = \bigwedge_q \Big(\Big(\bigvee_{i'=1,\ldots,m_1} \forall \widetilde{\mathbf{x}}'_{i'}.\ \chi''_{qi'}(\widetilde{\mathbf{x}}'_{i'}) \Big) \vee \Big(\bigvee_p \bigwedge_{j'=1,\ldots,m_2} \exists \widetilde{\mathbf{y}}'_{j'}.\ \eta''_{qpj'}(\widetilde{\mathbf{y}}'_{i'}) \Big) \Big) \ .$$

After renaming bound variables again such that all quantifiers bind pairwise distinct variables, we transform $\varphi^{(4)}$ into an equivalent formula that is a disjunction of formulas of the form

$$\bigwedge_k \Big(\forall \widetilde{\mathbf{x}}''_k.\ \bigvee_\ell K_{k\ell}(\widetilde{\mathbf{x}}''_k) \Big) \ \wedge \ \bigwedge_i \Big(\exists \widetilde{\mathbf{y}}''_i.\ \bigwedge_j L_{ij}(\widetilde{\mathbf{y}}''_i) \Big) \ .$$

□

The just proven lemma will provide the syntactic transformations necessary to eliminate second-order quantifiers that occur in a separated formula under certain conditions.

Example 5. We have already mentioned the following SF sentence in the introduction: $\varphi := \forall x_1 \exists y_1 \forall x_2 \exists y_2.\, R(x_1, x_2) \leftrightarrow Q(y_1, y_2)$. As indicated by Lemma 4, nested alternating quantifiers can be transformed away. An intermediate result of this process is

$$\begin{aligned} \forall x_1 \exists y_1. \Big(\big(\forall x_2.\, R(x_1, x_2)\big) \vee \big(\exists y_2.\, \neg Q(y_1, y_2)\big) \Big) \\ \wedge \Big(\big(\forall x_2.\, \neg R(x_1, x_2)\big) \vee \big(\exists y_2.\, Q(y_1, y_2)\big) \Big) \ . \end{aligned}$$

Continuing the transformation process, we eventually obtain

$$\begin{aligned} &\Big(\exists y_1 y_2 y_3.\, Q(y_1, y_2) \wedge \neg Q(y_1, y_3) \Big) \\ &\vee \Big(\big(\forall x_1 x_2.\, R(x_1, x_2)\big) \wedge \big(\exists y_1 y_2.\, Q(y_1, y_2)\big) \Big) \\ &\vee \Big(\big(\forall x_1 x_2.\, \neg R(x_1, x_2)\big) \wedge \big(\exists y_1 y_2.\, \neg Q(y_1, y_2)\big) \Big) \\ &\vee \Big(\big(\forall x_1 x_2 x_3.\, R(x_1, x_2) \vee \neg R(x_1, x_3)\big) \wedge \big(\exists y_1 y_2.\, Q(y_1, y_2)\big) \wedge \big(\exists y_3 y_4.\, \neg Q(y_3, y_4)\big) \Big) \ , \end{aligned}$$

which is equivalent to φ but does not contain any quantifier alternation.

4 Elimination of Second-Order Quantifiers

In this section we formulate syntactic restrictions that enable the elimination of second-order quantifiers over unary predicates from sentences that belong to the separated fragment. The notion of separation of sets of variables in a formula plays a central role in our criterion. However, this time it is not only of interest that universal variables are separated from existential variables. It is rather of importance that within each set of non-separated variables there is at most one that occurs as the argument of the predicate symbol that is bound by the quantifier we intend to eliminate.

Lemma 6. *Let $\varphi := \forall\bar{\mathbf{x}}_1\exists\bar{\mathbf{y}}_1 \ldots \forall\bar{\mathbf{x}}_n\exists\bar{\mathbf{y}}_n.\, \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$ be a relational first-order formula in which ψ is quantifier free and the sets $\bar{\mathbf{x}} := \bar{\mathbf{x}}_1 \cup \ldots \cup \bar{\mathbf{x}}_n$ and $\bar{\mathbf{y}} := \bar{\mathbf{y}}_1 \cup \ldots \cup \bar{\mathbf{y}}_n$ are separated. We assume that every variable occurring in the quantifier prefix and in $\bar{\mathbf{z}}$ also occurs in the matrix ψ.*

Let $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{m_1} \subseteq \bar{\mathbf{x}}$ and $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{m_2} \subseteq \bar{\mathbf{y}}$ be partitions of the sets $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$, respectively, such that the $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{m_1}, \widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{m_2}$ are nonempty, pairwise disjoint, and pairwise separated in φ. Let P be a unary predicate symbol satisfying the following conditions:

(1) For every set $\widetilde{\mathbf{x}}_i$, $1 \leq i \leq m_1$, there is at most one variable $x_i^* \in \widetilde{\mathbf{x}}_i$ for which φ contains atoms $P(x_i^*)$.
(2) For every set $\widetilde{\mathbf{y}}_i$, $1 \leq i \leq m_2$, there is at most one variable $y_i^* \in \widetilde{\mathbf{y}}_i$ for which φ contains atoms $P(y_i^*)$.

Then $\exists P.\, \varphi$ is equivalent to a finite disjunction of formulas of the form

$$\begin{aligned}\theta(\bar{\mathbf{z}}) \;\wedge\; \exists P.\; & \bigwedge_{k_1}\Big(\forall\widetilde{\mathbf{x}}'_{k_1}.\; \chi_{k_1}(\widetilde{\mathbf{x}}'_{k_1}, \bar{\mathbf{z}}) \vee P(x^*_{k_1})\Big) \wedge \bigwedge_{k_2}\Big(\forall\widetilde{\mathbf{x}}'_{k_2}.\; \chi'_{k_2}(\widetilde{\mathbf{x}}'_{k_2}, \bar{\mathbf{z}}) \vee \neg P(x^*_{k_2})\Big) \\ & \wedge \bigwedge_{i_1}\Big(\exists\widetilde{\mathbf{y}}'_{i_1}.\; \eta_{i_1}(\widetilde{\mathbf{y}}'_{i_1}, \bar{\mathbf{z}}) \wedge P(y^*_{i_1})\Big) \wedge \bigwedge_{i_2}\Big(\exists\widetilde{\mathbf{y}}'_{i_2}.\; \eta'_{i_2}(\widetilde{\mathbf{y}}'_{i_2}, \bar{\mathbf{z}}) \wedge \neg P(y^*_{i_2})\Big) \\ & \wedge \bigwedge_{\ell_1} P(z^*_{\ell_1}) \wedge \bigwedge_{\ell_2} \neg P(z^*_{\ell_2})\;,\end{aligned}$$

where (a) the χ_{k_1} and the χ'_{k_2} are disjunctions of literals and the η_{i_1} and the η'_{i_2} are conjunctions of literals, (b) all the atoms in θ and in the χ_{k_1}, χ'_{k_2}, η_{i_1}, and η_{i_2} are renamed variants of atoms that occur in φ and do not contain the predicate symbol P, and (c) the variables $z^*_{\ell_1}$, $z^*_{\ell_2}$ are pairwise distinct and stem from $\bar{\mathbf{z}}$.

Proof. By Lemma 4, we know that φ can be rewritten to an equivalent formula that is a finite disjunction of formulas in which no universal quantifier lies within the scope of an existential quantifier and vice versa. We apply this transformation to φ and obtain a formula as described in Lemma 4. In the next step, we isolate atoms that exclusively contain variables from $\bar{\mathbf{z}}$, narrow the scopes of first-order quantifiers so that these atoms are not within their scopes anymore, and transform

the resulting formulas into a formula φ' that is a disjunction of formulas of the form

$$\Big(\bigwedge_k \forall\widetilde{\mathbf{x}}'_k.\ \bigvee_\ell K_{k\ell}(\widetilde{\mathbf{x}}'_k,\bar{\mathbf{z}})\Big)\ \wedge\ \Big(\bigwedge_i \exists\widetilde{\mathbf{y}}'_i.\ \bigwedge_j L_{ij}(\widetilde{\mathbf{y}}'_i,\bar{\mathbf{z}})\Big)\ \wedge\ \bigwedge_r M_r(\bar{\mathbf{z}})\ ,$$

where the $K_{k\ell}$ and the L_{ij} are literals whose atoms are renamed variants of atoms that occur in φ and contain at least one variable from some $\widetilde{\mathbf{x}}'_k$ or $\widetilde{\mathbf{y}}'_i$. The M_r are literals whose atoms occur in φ and contain exclusively variables from $\bar{\mathbf{z}}$. Moreover, any two sets $\widetilde{\mathbf{x}}'_{k_1},\widetilde{\mathbf{x}}'_{k_2}$ with $k_1 \neq k_2$, $\widetilde{\mathbf{y}}'_{i_1},\widetilde{\mathbf{y}}'_{i_2}$ with $i_1 \neq i_2$, and $\widetilde{\mathbf{x}}'_k,\widetilde{\mathbf{y}}'_i$ are separated in φ'. By inspection of the transformations performed in the proof of Lemma 4, we observe that Conditions (1) and (2) are preserved such that they also apply to the sets $\widetilde{\mathbf{x}}'_k$ and $\widetilde{\mathbf{y}}'_i$ with respect to variables x^*_k and y^*_i, respectively.

This enables us to regroup the disjunctions and conjunctions in the constituents of φ' such that each of these disjuncts has the form

$$\begin{aligned}
&\bigwedge_{k'}\Big(\forall\widetilde{\mathbf{x}}'_{k'}.\Big(\bigvee_{\ell'} K_{k'\ell'}(\widetilde{\mathbf{x}}'_{k'},\bar{\mathbf{z}})\Big) \vee [\neg]P(x^*_{k'})\Big)\\
&\wedge\ \bigwedge_{i'}\Big(\exists\widetilde{\mathbf{y}}'_{i'}.\Big(\bigwedge_{j'} L_{i'j'}(\widetilde{\mathbf{y}}'_{i'},\bar{\mathbf{z}})\Big) \wedge [\neg]P(y^*_{i'})\Big)\\
&\wedge\ \Big(\bigwedge_{r'} M_{r'}(\bar{\mathbf{z}})\Big) \wedge \bigwedge_q [\neg]P(z^*_q)\ ,
\end{aligned}$$

where the literals $K_{k'\ell'}$, $L_{i'j'}$, and $M_{r'}$ do not contain the predicate symbol P. The variables z^*_q stem from $\bar{\mathbf{z}}$. Moreover, we replace disjuncts (conjuncts) which contain two literals $P(v)$ and $\neg P(v)$ with the logical constant `true` (`false`). Having this, it only remains to regroup conjuncts and distribute the existential quantifier $\exists P$ over the topmost disjunction, in order to obtain the formula advertised in the lemma. □

The formula resulting from the lemma gives us the right starting point for the elimination of the second-order quantifier $\exists P$ from a formula. Before we elaborate on this, we present the lemma that we shall employ for elimination.

Lemma 7 (Basic elimination lemma, see [11] and [2]). *Let P be a unary predicate symbol and let χ,η be first-order formulas in which P does not occur. Then, $\exists P.\ \big(\forall x.\,\chi \vee P(x)\big) \wedge \big(\forall x.\,\eta \vee \neg P(x)\big)$ is semantically equivalent to $\forall x.\,\chi \vee \eta$.*

Consider a formula $\varphi := \forall\bar{\mathbf{x}}_1\exists\bar{\mathbf{y}}_1 \ldots \forall\bar{\mathbf{x}}_n\exists\bar{\mathbf{y}}_n.\psi(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\mathbf{z}})$ as described in Lemma 6. Moreover, let there be sets $\widetilde{\mathbf{x}}_1,\ldots,\widetilde{\mathbf{x}}_{m_1}$ and $\widetilde{\mathbf{y}}_1,\ldots,\widetilde{\mathbf{y}}_{m_2}$ and a unary predicate symbol P as described in the lemma. Then, Lemma 6 stipulates the existence of a formula equivalent to φ that is a disjunction of formulas of the form

$$\begin{aligned}
\theta(\bar{\mathbf{z}})\ \wedge\ \exists P.\ &\bigwedge_{k_1}\Big(\forall\widetilde{\mathbf{x}}'_{k_1}.\ \chi_{k_1}(\widetilde{\mathbf{x}}'_{k_1},\bar{\mathbf{z}}) \vee P(x^*_{k_1})\Big) \wedge \bigwedge_{k_2}\Big(\forall\widetilde{\mathbf{x}}'_{k_2}.\ \chi'_{k_2}(\widetilde{\mathbf{x}}'_{k_2},\bar{\mathbf{z}}) \vee \neg P(x^*_{k_2})\Big)\\
&\wedge \bigwedge_{i_1}\Big(\exists\widetilde{\mathbf{y}}'_{i_1}.\ \eta_{i_1}(\widetilde{\mathbf{y}}'_{i_1},\bar{\mathbf{z}}) \wedge P(y^*_{i_1})\Big) \wedge \bigwedge_{i_2}\Big(\exists\widetilde{\mathbf{y}}'_{i_2}.\ \eta'_{i_2}(\widetilde{\mathbf{y}}'_{i_2},\bar{\mathbf{z}}) \wedge \neg P(y^*_{i_2})\Big)\\
&\wedge \bigwedge_{\ell_1} P(z^*_{\ell_1}) \wedge \bigwedge_{\ell_2} \neg P(z^*_{\ell_2})\ ,
\end{aligned}$$

in which we can eliminate the quantifier $\exists P$ as follows. The shape of the above formula is very similar to what Behmann called "Eliminationshauptform" in [2] (see [11] for a modern exposition of Behmann's results related to quantifier elimination). With the next two transformation steps we come closer to the syntactic shape of the "Eliminationshauptform". First, we narrow the scope of the first-order quantifiers that do not bind variables x_k^* or y_i^*.

$$\begin{aligned}
\theta(\bar{\mathbf{z}}) \;\wedge\; \exists P.\; & \bigwedge_{k_1} \Big(\forall x_{k_1}^*.\; \underbrace{\big(\forall(\widetilde{\mathbf{x}}'_{k_1} \setminus \{x_{k_1}^*\}).\; \chi_{k_1}(\widetilde{\mathbf{x}}'_{k_1}, \bar{\mathbf{z}})\big)}_{=:\, \chi_{k_1}^*} \vee P(x_{k_1}^*) \Big) \\
& \wedge \bigwedge_{k_2} \Big(\forall x_{k_2}^*.\; \underbrace{\big(\forall(\widetilde{\mathbf{x}}'_{k_2} \setminus \{x_{k_2}^*\}).\; \chi_{k_2}(\widetilde{\mathbf{x}}'_{k_2}, \bar{\mathbf{z}})\big)}_{=:\, \chi_{k_2}^*} \vee \neg P(x_{k_2}^*) \Big) \\
& \wedge \bigwedge_{i_1} \Big(\exists y_{i_1}^*.\; \underbrace{\big(\exists(\widetilde{\mathbf{y}}'_{i_1} \setminus \{y_{i_1}^*\}).\; \eta'_{i_1}(\widetilde{\mathbf{y}}'_{i_1}, \bar{\mathbf{z}})\big)}_{=:\, \eta_{i_1}^*} \wedge P(y_{i_1}^*) \Big) \\
& \wedge \bigwedge_{i_2} \Big(\exists y_{i_2}^*.\; \underbrace{\big(\exists(\widetilde{\mathbf{y}}'_{i_2} \setminus \{y_{i_2}^*\}).\; \eta'_{i_2}(\widetilde{\mathbf{y}}'_{i_2}, \bar{\mathbf{z}})\big)}_{=:\, \eta_{i_2}^*} \wedge \neg P(y_{i_2}^*) \Big) \\
& \wedge \bigwedge_{\ell_1} P(z_{\ell_1}^*) \wedge \bigwedge_{\ell_2} \neg P(z_{\ell_2}^*)
\end{aligned}$$

Next, we treat the subformulas χ_k^* and η_i^* as indivisible units, move universal quantifiers outwards that occur in different conjuncts (and merge them while doing so), pull first-order existential quantifiers outwards (without merging them), and rename the variables that are bound by the moved quantifiers. Moreover, we reorder the conjunctions in the scope of the quantifier blocks $\exists \bar{\mathbf{u}}$ and $\exists \bar{\mathbf{v}}$.

$$\begin{aligned}
\theta(\bar{\mathbf{z}}) \;\wedge\; \exists P.\; & \Big(\forall x.\; \underbrace{\Big(\bigwedge_{k_1} \chi_{k_1}^*\big[x_{k_1}^*/x\big] \Big)}_{=:\, \chi_1^*(x, \bar{\mathbf{z}})} \vee P(x) \Big) \\
& \wedge \Big(\forall x.\; \underbrace{\Big(\bigwedge_{k_2} \chi_{k_2}^*\big[x_{k_2}^*/x\big] \Big)}_{=:\, \chi_2^*(x, \bar{\mathbf{z}})} \vee \neg P(x) \Big) \\
& \wedge \Big(\exists \bar{\mathbf{u}}.\; \underbrace{\Big(\bigwedge_{i_1} \eta_{i_1}^*\big[y_{i_1}^*/u_{i_1}\big] \Big)}_{=:\, \eta_1^*(\bar{\mathbf{u}}, \bar{\mathbf{z}})} \wedge \bigwedge_{i_1} P(u_{i_1}) \Big) \\
& \wedge \Big(\exists \bar{\mathbf{v}}.\; \underbrace{\Big(\bigwedge_{i_2} \eta_{i_2}^*\big[y_{i_2}^*/v_{i_2}\big] \Big)}_{=:\, \eta_2^*(\bar{\mathbf{v}}, \bar{\mathbf{z}})} \wedge \bigwedge_{i_2} \neg P(v_{i_2}) \Big) \\
& \wedge \Big(\bigwedge_{\ell_1} P(z_{\ell_1}^*) \wedge \bigwedge_{\ell_2} \neg P(z_{\ell_2}^*) \Big)
\end{aligned}$$

In what follows we treat the χ_1^*, χ_2^* and η_1^*, η_2^* as indivisible units. One more step remains to establish a kind of "Eliminationshauptform". We move the quantifier blocks $\exists \bar{\mathbf{u}}$ and $\exists \bar{\mathbf{v}}$ outwards over the $\exists P$, reorder the conjuncts within the scope of $\exists P$, and narrow the scope of $\exists P$ such that it does not contain the η_1^*, η_2^* anymore. Moreover, we make use of Lemma 2 and turn the literals $P(u_{i_1})$ into subformulas $\forall x.\, x = u_{i_1} \to P(x)$. We proceed analogously with the literals $\neg P(v_{i_2})$, $P(z^*_{\ell_1})$, and $\neg P(z^*_{\ell_2})$.

$$\begin{aligned}
&\theta(\bar{\mathbf{z}}) \;\wedge\; \exists \bar{\mathbf{u}} \bar{\mathbf{v}}.\, \eta_1^*(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta_2^*(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
&\qquad \wedge \exists P.\, \Big(\forall x.\, \chi_1^*(x, \bar{\mathbf{z}}) \vee\; P(x)\Big) \;\wedge\; \Big(\forall x.\, \chi_2^*(x, \bar{\mathbf{z}}) \vee\; \neg P(x)\Big) \\
&\qquad\qquad \wedge\; \Big(\forall x.\, \bigwedge_{i_1} \big(x = u_{i_1} \to P(x)\big)\Big) \;\wedge\; \Big(\forall x.\, \bigwedge_{i_2} \big(x = v_{i_2} \to \neg P(x)\big)\Big) \\
&\qquad\qquad \wedge\; \Big(\forall x.\, \bigwedge_{\ell_1} \big(x = z^*_{\ell_1} \to P(x)\big)\Big) \;\wedge\; \Big(\forall x.\, \bigwedge_{\ell_2} \big(x = z^*_{\ell_2} \to \neg P(x)\big)\Big)
\end{aligned}$$

At this point, the subformula staring with $\exists P$ is in "Eliminationshauptform". After converting the implications into disjunctions and factoring out the $[\neg]P(x)$, we arrive at a formula from which the second-order quantifier $\exists P$ can be eliminated immediately via the basic elimination lemma.

$$\begin{aligned}
&\theta(\bar{\mathbf{z}}) \;\wedge\; \exists \bar{\mathbf{u}} \bar{\mathbf{v}}.\, \eta_1^*(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta_2^*(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
&\qquad \wedge \exists P.\, \Big(\forall x.\, \big(\chi_1^*(x, \bar{\mathbf{z}}) \wedge \bigwedge_{i_1} x \neq u_{i_1} \wedge \bigwedge_{\ell_1} x \neq z^*_{\ell_1}\big) \vee P(x)\Big) \\
&\qquad\qquad \wedge\; \Big(\forall x.\, \big(\chi_2^*(x, \bar{\mathbf{z}}) \wedge \bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2}\big) \vee \neg P(x)\Big)
\end{aligned}$$

Using Lemma 7, we eliminate the quantifier $\exists P$ and obtain the following result.

$$\begin{aligned}
&\theta(\bar{\mathbf{z}}) \;\wedge\; \exists \bar{\mathbf{u}} \bar{\mathbf{v}}.\, \eta_1^*(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta_2^*(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
&\qquad \wedge \forall x.\, \Big(\big(\chi_1^*(x, \bar{\mathbf{z}}) \wedge \bigwedge_{i_1} x \neq u_{i_1} \wedge \bigwedge_{\ell_1} x \neq z^*_{\ell_1}\big) \\
&\qquad\qquad \vee \big(\chi_2^*(x, \bar{\mathbf{z}}) \wedge \bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2}\big)\Big)
\end{aligned}$$

In order to convert this result into a somewhat nicer form, we proceed as described in the proof of Lemma 19 in [11]. In particular, we remove the disequations $x \neq y$, where x is a universally quantified variable. To this end, we first distribute disjunction over conjunction within the scope of the quantifier $\forall x$.

$$\begin{aligned}
&\theta(\bar{\mathbf{z}}) \;\wedge\; \exists \bar{\mathbf{u}} \bar{\mathbf{v}}.\, \eta_1^*(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta_2^*(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
&\qquad \wedge\; \forall x.\, \Big(\chi_1^*(x, \bar{\mathbf{z}}) \vee \chi_2^*(x, \bar{\mathbf{z}})\Big) \\
&\qquad\qquad \wedge\; \Big(\big(\bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2}\big) \vee \chi_1^*(x, \bar{\mathbf{z}})\Big)
\end{aligned}$$

$$\wedge\ \Big((\bigwedge_{i_1} x \neq u_{i_1} \wedge \bigwedge_{\ell_1} x \neq z^*_{\ell_1}) \vee \chi^*_2(x, \bar{\mathbf{z}})\Big)$$
$$\wedge\ \Big((\bigwedge_{i_1} x \neq u_{i_1} \wedge \bigwedge_{\ell_1} x \neq z^*_{\ell_1}) \vee (\bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2})\Big)$$

Next, we factor the subformulas χ^*_1, χ^*_2, and $\bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2}$ into the conjunctions with which they are disjunctively connected. Moreover, we turn the resulting disjunctions into implications.

$$\begin{aligned}
\theta(\bar{\mathbf{z}})\ \wedge\ & \exists \bar{\mathbf{u}}\bar{\mathbf{v}}.\ \eta^*_1(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta^*_2(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
& \wedge\ \forall x.\ \Big(\chi^*_1(x, \bar{\mathbf{z}}) \vee \chi^*_2(x, \bar{\mathbf{z}})\Big) \\
& \qquad \wedge\ \Big(\bigwedge_{i_2} \big(x = v_{i_2} \to \chi^*_1(x, \bar{\mathbf{z}})\big) \wedge \bigwedge_{\ell_2} \big(x = z^*_{\ell_2} \to \chi^*_1(x, \bar{\mathbf{z}})\big)\Big) \\
& \qquad \wedge\ \Big(\bigwedge_{i_1} \big(x = u_{i_1} \to \chi^*_2(x, \bar{\mathbf{z}})\big) \wedge \bigwedge_{\ell_1} \big(x = z^*_{\ell_1} \to \chi^*_2(x, \bar{\mathbf{z}})\big)\Big) \\
& \qquad \wedge\ \Big(\bigwedge_{i_1} \big(x \neq u_{i_1} \to (\bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2})\big)\Big) \\
& \qquad \wedge\ \Big(\bigwedge_{\ell_1} \big(x \neq z^*_{\ell_1} \to (\bigwedge_{i_2} x \neq v_{i_2} \wedge \bigwedge_{\ell_2} x \neq z^*_{\ell_2})\big)\Big)
\end{aligned}$$

Finally, we apply Lemma 2 in a reverse fashion to remove the universal variable x from some of the subformulas.

$$\begin{aligned}
& \theta(\bar{\mathbf{z}}) \wedge \big(\forall x.\ \chi^*_1(x, \bar{\mathbf{z}}) \vee \chi^*_2(x, \bar{\mathbf{z}})\big) \\
& \wedge \exists \bar{\mathbf{u}}\bar{\mathbf{v}}.\ \eta^*_1(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta^*_2(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
& \qquad \wedge \bigwedge_{i_2} \chi^*_1(v_{i_2}, \bar{\mathbf{z}})\ \wedge \bigwedge_{\ell_2} \chi^*_1(z^*_{\ell_2}, \bar{\mathbf{z}})\ \wedge \bigwedge_{i_1} \chi^*_2(u_{i_1}, \bar{\mathbf{z}})\ \wedge \bigwedge_{\ell_1} \chi^*_2(z^*_{\ell_1}, \bar{\mathbf{z}}) \\
& \qquad \wedge \bigwedge_{i_1}\bigwedge_{i_2} u_{i_1} \neq v_{i_2}\ \wedge \bigwedge_{i_1}\bigwedge_{\ell_2} u_{i_1} \neq z^*_{\ell_2}\ \wedge \bigwedge_{\ell_1}\bigwedge_{i_2} z^*_{\ell_1} \neq v_{i_2}\ \wedge \bigwedge_{\ell_1}\bigwedge_{\ell_2} z^*_{\ell_1} \neq z^*_{\ell_2}
\end{aligned}$$

Consequently, we get the following result.

Theorem 8. *Let $\varphi := \forall \bar{\mathbf{x}}_1 \exists \bar{\mathbf{y}}_1 \ldots \forall \bar{\mathbf{x}}_n \exists \bar{\mathbf{y}}_n.\, \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$ be a relational first-order formula in which ψ is quantifier free and the sets $\bar{\mathbf{x}} := \bar{\mathbf{x}}_1 \cup \ldots \cup \bar{\mathbf{x}}_n$ and $\bar{\mathbf{y}} := \bar{\mathbf{y}}_1 \cup \ldots \cup \bar{\mathbf{y}}_n$ are separated. We assume that every variable occurring in the quantifier prefix and in $\bar{\mathbf{z}}$ also occurs in the matrix ψ.*

Let $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{m_1} \subseteq \bar{\mathbf{x}}$ and $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{m_2} \subseteq \bar{\mathbf{y}}$ be partitions of the sets $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$, respectively, such that the $\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{m_1}, \widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{m_2}$ are nonempty, pairwise disjoint, and pairwise separated in φ. Let P be a unary predicate symbol satisfying the following conditions:

(1) For every set $\widetilde{\mathbf{x}}_i$, $1 \leq i \leq m_1$, there is at most one variable $x^*_i \in \widetilde{\mathbf{x}}_i$ for which φ contains atoms $P(x^*_i)$.

(2) For every set $\widetilde{\mathbf{y}}_i$, $1 \leq i \leq m_2$, there is at most one variable $y_i^* \in \widetilde{\mathbf{y}}_i$ for which φ contains atoms $P(y_i^*)$.

Then $\exists P.\, \varphi$ is equivalent to some first-order formula φ' that is a finite disjunction of formulas of the form

$$\begin{aligned}
&\theta(\bar{\mathbf{z}}) \wedge \big(\forall x.\ \chi_1^*(x, \bar{\mathbf{z}}) \vee \chi_2^*(x, \bar{\mathbf{z}})\big) \\
&\wedge\ \exists \bar{\mathbf{u}} \bar{\mathbf{v}}.\ \eta_1^*(\bar{\mathbf{u}}, \bar{\mathbf{z}}) \wedge \eta_2^*(\bar{\mathbf{v}}, \bar{\mathbf{z}}) \\
&\qquad \wedge \bigwedge_{i_2} \chi_1^*(v_{i_2}, \bar{\mathbf{z}}) \ \wedge \bigwedge_{\ell_2} \chi_1^*(z_{\ell_2}^*, \bar{\mathbf{z}}) \ \wedge \bigwedge_{i_1} \chi_2^*(u_{i_1}, \bar{\mathbf{z}}) \ \wedge \bigwedge_{\ell_1} \chi_2^*(z_{\ell_1}^*, \bar{\mathbf{z}}) \\
&\qquad \wedge \bigwedge_{i_1} \bigwedge_{i_2} u_{i_1} \neq v_{i_2} \ \wedge \bigwedge_{i_1} \bigwedge_{\ell_2} u_{i_1} \neq z_{\ell_2}^* \ \wedge \bigwedge_{\ell_1} \bigwedge_{i_2} z_{\ell_1}^* \neq v_{i_2} \ \wedge \bigwedge_{\ell_1} \bigwedge_{\ell_2} z_{\ell_1}^* \neq z_{\ell_2}^*
\end{aligned}$$

where all free predicate symbols and all free first-order variables also occur freely in $\exists P.\, \varphi$. Moreover, all the u_{i_1} are variables from $\bar{\mathbf{u}}$, the v_{i_2} are from $\bar{\mathbf{v}}$, and the $z_{\ell_1}^*$ and $z_{\ell_2}^*$ are certain free variables from $\bar{\mathbf{z}}$.

Example 9. Consider the sentence $\varphi := \exists P. \forall x_1 \exists y \forall x_2.\, R(x_1, x_2) \leftrightarrow P(y)$. We transform it into the equivalent sentence

$$\begin{aligned}
\exists P.\ &\Big(\forall x_1 x_2 x_3.\, R(x_1, x_2) \vee \neg R(x_1, x_3)\Big) \\
&\wedge \Big(\big(\forall x_1 x_2.\, R(x_1, x_2)\big) \vee \big(\exists y.\, \neg P(y)\big)\Big) \\
&\wedge \Big(\big(\forall x_1 x_2.\, \neg R(x_1, x_2)\big) \vee \big(\exists y.\, P(y)\big)\Big)\ .
\end{aligned}$$

For the sake of simplicity, we narrow the scope of $\exists P$ so that it only stretches over the last two conjuncts, which we thereafter transform into a disjunction of conjunctions. This yields

$$\begin{aligned}
&\Big(\forall x_1 x_2 x_3.\, R(x_1, x_2) \vee \neg R(x_1, x_3)\Big) \\
&\wedge\ \Big(\exists P.\ \Big(\big(\forall x_1 x_2.\, R(x_1, x_2)\big) \wedge \big(\exists y.\, P(y)\big)\Big) \\
&\qquad\quad \vee \Big(\big(\forall x_1 x_2.\, \neg R(x_1, x_2)\big) \wedge \big(\exists y.\, \neg P(y)\big)\Big) \\
&\qquad\quad \vee \Big(\big(\exists y.\, P(y)\big) \wedge \big(\exists y.\, \neg P(y)\big)\Big)\Big)\ .
\end{aligned}$$

Since we can distribute the quantifier $\exists P$ over disjunction, it is enough to eliminate $\exists P$ in the following three formulas:

$$\begin{aligned}
&(1)\ \exists P.\, \exists y.\, P(y) \\
&\quad \models\!\models\ \exists y.\, \exists P.\, \forall x.\, \big(x \neq y \vee P(x)\big) \wedge \big(\texttt{true} \vee \neg P(x)\big) \\
&\quad \models\!\models\ \exists y \forall x.\, x \neq y \vee \texttt{true} \\
&\quad \models\!\models\ \texttt{true} \\
&(2)\ \exists P.\, \exists y.\, \neg P(y) \\
&\quad \models\!\models\ \exists y.\, \exists P.\, \forall x.\, \big(x \neq y \vee \neg P(x)\big) \wedge \big(\texttt{true} \vee P(x)\big) \\
&\quad \models\!\models\ \exists y \forall x.\, x \neq y \vee \texttt{true} \\
&\quad \models\!\models\ \texttt{true}
\end{aligned}$$

(3) $\exists P.\,\big(\exists y.\,P(x)\big) \wedge \big(\exists y.\,\neg P(x)\big)$
$\quad\models\quad \exists y_1 y_2.\,\exists P.\,\big(\forall x.\,x \neq y_1 \vee P(x)\big) \wedge \big(\forall x.\,x \neq y_2 \vee \neg P(x)\big)$
$\quad\models\quad \exists y_1 y_2 \forall x.\,x \neq y_1 \vee x \neq y_2$
$\quad\models\quad \exists y_1 y_2.\,y_1 \neq y_2$

Hence, φ is semantically equivalent to

$$\begin{aligned}&\big(\forall x_1 x_2 x_3.\,R(x_1,x_2) \vee \neg R(x_1,x_3)\big)\\ &\wedge \Big(\big(\forall x_1 x_2.\,R(x_1,x_2)\big) \vee \big(\forall x_1 x_2.\,\neg R(x_1,x_2)\big) \vee \big(\exists y_1 y_2.\,y_1 \neq y_2\big)\Big)\,.\end{aligned}$$

Several remarks regrading the shape of the resulting formulas in Theorem 8 are in order. (a) Although the elimination of $\exists P$ potentially introduces new (dis)equations, these only involve existentially quantified and free variables. This means, the separation conditions are not violated by these newly introduced equations. Hence, the introduction of these atoms in one elimination step does not pose an obstacle to the iterated elimination of multiple existential second-order quantifiers. (b) As the subformulas $\chi_1^*(v_{i_2}, \bar{\mathbf{z}})$ may contain universal quantifiers $\forall w$ and atoms $R(\ldots w \ldots v_{i_2} \ldots)$, the separateness condition regarding universally and existentially quantified variables might be violated when introducing the subformulas $\chi_1^*(v_{i_2}, \bar{\mathbf{z}})$ and, similarly, the subformulas $\chi_2^*(u_{i_1}, \bar{\mathbf{z}})$. (c) Perhaps more severely, the introduction of atoms $R(\ldots w \ldots v_{i_2} \ldots)$ may create a connection between sets $\widetilde{\mathbf{x}}_k$ and $\widetilde{\mathbf{y}}_i$, if $w \in \widetilde{\mathbf{x}}_k$ and $v_{i_2} \in \widetilde{\mathbf{y}}_i$. Then, the sets $\widetilde{\mathbf{x}}_k$ and $\widetilde{\mathbf{y}}_i$ are not separated anymore in formulas that contain the new atom. Similar effects might affect pairs $\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{x}}_{k'}$ and $\widetilde{\mathbf{y}}_i, \widetilde{\mathbf{y}}_{i'}$. Hence, if we were to predict whether elimination of both second-order quantifiers in a formula $\exists Q \exists P.\,\varphi$ is possible using the methods outlined above, we would need to predict which sets of variables will be separated in the formula that results from eliminating $\exists P$.

The above observations seem to make it hard to formulate a version of Theorem 8 that clearly facilitates iterative elimination of multiple quantifiers. On the other hand, it might be worthwhile to base the theorem on a generalization of the separated fragment that still has a decidable satisfiability problem. The *generalized Bernays–Schönfinkel–Ramsey fragment (GBSR)* is described in [9]. In GBSR sentences universally and existentially quantified variables may occur together in atoms under certain restrictions. Roughly speaking, if the existential variable is quantified outside the scope of the quantifier binding the universal variable, the two may occur jointly in atoms. Every GBSR sentence can be transformed into an equivalent sentence that is a finite disjunction of formulas of the form $\exists \bar{\mathbf{y}}.\,\bigwedge_i \forall \bar{\mathbf{x}}_i.\,\bigvee_j L_{ij}(\bar{\mathbf{y}}, \bar{\mathbf{x}}_i)$ where the L_{ij} are literals. Hence, Observation (b) might cause fewer troubles in the GBSR setting.

Another interesting aspect is that the symmetry regarding the two Conditions (1) and (2) in Theorem 8 is perhaps more restrictive than necessary. It seems that Condition (2) is obsolete, as the resulting formula in Lemma 6 could be generalized in such a way that the restriction imposed by (2) is not satisfied but second-order quantifiers can still be eliminated.

Altogether, it is subject to future investigations whether Theorem 8 can be enhanced to facilitate iterative elimination of multiple quantifiers.

5 Discussion

We have developed a preliminary result regarding the elimination of second-order quantifiers in a logic fragment that extends the monadic first-order fragment without equality and the Bernays–Schönfinkel–Ramsey fragment.

Notice that the elimination of $\exists P$ from a formula $\exists P.\, \varphi$, where φ is a monadic first-order formula *without equality*, constitutes a special case of the method shown in the present note. The reason is that, if every atom contains at most one variable, then variables cannot occur jointly in atoms. Hence, given a monadic first-order formula φ without equality, any two singleton sets $\{x\}, \{y\}$ of variables are separated in φ. Consequently, any formula $\exists P.\, \varphi$ with monadic φ satisfies the prerequisites of Theorem 8, if we choose the sets $\widetilde{\mathbf{x}}_k$ and $\widetilde{\mathbf{y}}_i$ to be singleton sets covering all the variables that occur bound in φ.

The presented result can only be a first step towards the formulation of a novel fragment of second-order logic that (a) extends the monadic second-order fragment, (b) is based on the concept of separateness of certain variables at the atomic level, (c) admits elimination of second-order quantifiers, also in an iterated fashion. The discussion following Theorem 8 already makes clear that a lot remains to be done, in order to achieve this goal. Furthermore, there seems to be no good reason to confine ourselves to the elimination of quantifiers over unary predicates, but aim for higher arities as well. Moreover, (b) can be weakened by taking boolean structure into account instead of only concentrating on the atoms in a given formula. For example, the formula $\exists P. \forall xy.\, P(x) \wedge \big(P(y) \vee R(x,y)\big)$ does not satisfy the prerequisites of Theorem 8, as $\{x\}$ and $\{y\}$ are not separated and the set $\{x, y\}$ contains two variables that occur as arguments of P. However, the theorem can be applied to the equivalent formula $\exists P. \forall x_1 x_2 y.\, P(x_1) \wedge \big(P(y) \vee R(x_2, y)\big)$, as the sets $\{x_1\}$ and $\{x_2, y\}$ are separated and x_2 does not occur as argument of P. As a third possible improvement, equations between universal and existential variables should be allowed in a less restrictive way than they are in the present note. To this end, some of the methods that are used to handle equations during quantifier elimination in the monadic second-order fragment might be applicable in the more general setting as well.

In the present note we concentrate on transforming the input formulas syntactically until the basic elimination lemma (Lemma 7) is applicable. In future work, it is of course advisable to also try other known approaches, such as the ones described in [4], e.g. the SCAN algorithm, the DLS* algorithm, hierarchical theorem proving, or variations thereof. The unmodifed DLS algorithm, as presented in [4], fails on the logic fragment described in Theorem 8 in the present note. In particular, the preprocessing phase is not always able to transform the input into the required form, although this is possible in principle. This is already true for monadic sentences such as $\varphi := \exists P. \forall x \exists y.\, \big(\neg P(x) \vee P(y)\big) \wedge \big(P(x) \vee \neg P(y)\big)$, which is equivalent to $\exists P. \forall x \exists y.\, P(x) \leftrightarrow P(y)$. Conradie gave a necessary and sufficient condition on the syntax of formulas in which DLS can successfully eliminate an existential second-order quantifier [3]. It turns out that the occurrences of P in φ violate Conradie's condition in many ways. (Every occurrence of P is in *malignant* conjunctions and disjunctions and inside a $\forall\exists$-*scope*.) Nonetheless,

it is not hard to see that there is a first-order formula that is equivalent to φ, namely `true`. A slight modification of the DLS preprocessing step in the spirit of Claim I, used in the proof of Lemma 4, might already solve this particular issue.

Acknowledgement The present author is indebted to the anonymous reviewers for their constructive criticism and valuable suggestions.

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Mathematische Annalen 110, 390–413 (1935)
2. Behmann, H.: Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. Mathematische Annalen 86(3–4), 163–229 (1922)
3. Conradie, W.: On the strength and scope of DLS. Journal of Applied Non-Classical Logics 16(3-4), 279–296 (2006)
4. Gabbay, D., Schmidt, R., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications (2008)
5. van Heijenoort, J.: From Frege to Gödel – A Source Book in Mathematical Logic, 1879–1931. Harvard University Press (2002)
6. Löwenheim, L.: Über Möglichkeiten im Relativkalkül. Mathematische Annalen 76, 447–470 (1915), an English translation can be found in [5].
7. Skolem, T.: Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme welche gewisse Klassen von Aussagen betreffen. Videnskapsselskapets Skrifter I. Mat.-Nat. Klasse (3) (1919)
8. Sturm, T., Voigt, M., Weidenbach, C.: Deciding First-Order Satisfiability when Universal and Existential Variables are Separated. In: Logic in Computer Science (LICS'16). pp. 86–95 (2016)
9. Voigt, M.: On Generalizing Decidable Standard Prefix Classes of First-Order Logic Submitted, a preprint version is available under arXiv:1706.03949 [cs.LO].
10. Voigt, M.: A Fine-Grained Hierarchy of Hard Problems in the Separated Fragment. In: Logic in Computer Science (LICS'17). pp. 1–12 (2017)
11. Wernhard, C.: Heinrich Behmann's Contributions to Second-Order Quantifier Elimination from the View of Computational Logic. Tech. rep., TU Dresden (2015)

Approximating Resultants of Existential Second-Order Quantifier Elimination upon Universal Relational First-Order Formulas

Christoph Wernhard
TU Dresden, Germany

Abstract. We investigate second-order quantifier elimination for a class of formulas characterized by a restriction on the quantifier prefix: existential predicate quantifiers followed by universal individual quantifiers and a relational matrix. For a given second-order formula of this class a possibly infinite sequence of universal first-order formulas that have increasing strength and are all entailed by the second-order formula can be constructed. Any first-order consequence of the second-order formula is a consequence of some member of the sequence. The sequence provides a recursive base for the first-order theory of the second-order formula, in the sense investigated by Craig. The restricted formula class allows to derive further properties, for example that the set of those members of the sequence that are equivalent to the second-order formula, or, more generally, have the same first-order consequences, is co-recursively enumerable. Also the set of first-order formulas that entails the second-order formula is co-recursively enumerable. These properties are proven with formula-based tools used in automated deduction, such as domain closure axioms, eliminating individual quantifiers by ground expansion, predicate quantifier elimination with Ackermann's Lemma, Craig interpolation and decidability of the Bernays-Schönfinkel-Ramsey class.

1 Introduction

The objective of second-order quantifier elimination is to compute from a given second-order formula a so-called *resultant* (from German *Resultante*, used by Schröder [17]), that is, an equivalent first-order formula. Finding such a resultant is not in general possible for arbitrary second-order formulas. The early technical investigations of second-order quantifier elimination on the basis of first-order logic were done jointly with the early investigations of the decision problem: Löwenheim [14], Skolem [18] and Behmann [3] gave decision methods for relational[1] monadic[2] formulas with equality. Their methods are based on existentially quantifying upon all predicates in the formula to decide and computing a resultant by elimination. The obtained resultant of a relational monadic formula is then a truth value constant or a formula that just constrains domain cardinality (see, e.g., [23]).

Whereas much is known today about decidable fragments of first-order logic, see, e.g., [4], knowledge about fragments on which second-order quantifier elim-

[1] No function symbols with exception of individual constants.
[2] No predicate symbols with arity larger than one.

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

ination succeeds seems still scarce [12]. Ackermann [1] was the first to publish[3] a proof that second-order quantifier elimination on the basis of first-order logic does not succeed in general, by means of a second-order formalization of the induction axiom. Conradie [5] formulated quite specific syntactic conditions under which the modern DLS algorithm [7] for second-order quantifier elimination succeeds. However, it appears that for the case of simultaneous elimination of multiple existential predicate quantifiers only a sufficient condition has been found. In addition, DLS as considered there does not cover the relational monadic case. Van Benthem and Doets [21] consider three classes of relational formulas with existential second-order quantifier prefix, distinguished by the subsequent first-order quantifier prefix: existential first-order quantifiers, universal first-order quantifiers, and first-order quantifier prefixes of the form $\forall x_1 \ldots \forall x_m \exists y_1 \ldots \exists y_n$.[4] As shown in [21], formulas with arbitrary further first-order prefixes can be converted to the latter class by Skolemization. For the second class, it is sketched in [21] with model theoretic arguments that a resultant exists which is an (infinite) disjunction of an (infinite) conjunction of first-order formulas.

Here we will take a closer look on that second class, applying formula-based tools that are used in automated deduction. In particular, our toolkit comprises domain closure axioms as familiar from logical modeling of database semantics, formula normalization and elimination of first-order quantifiers by ground expansion, second-order quantifier elimination by rewriting with an equivalence of a second-order formula of a certain form to a first-order formula (Ackermann's Lemma [1]), a strengthening of Craig interpolation that can be proven with a tableau technique, and decidability of the Bernays-Schönfinkel-Ramsey class.

The main original motivation was to explore possible foundations for applying instance-based [2] techniques as known for theorem proving also to solve elimination tasks. Their underlying principle is Herbrand's theorem, which justifies reducing unsatisfiability of a first-order formula to unsatisfiability of a propositional formula obtained by eliminating first-order quantifiers through ground expansion with terms constructed from the input vocabulary. The general idea of instance-based methods for theorem proving is to successively generate conjunctions of instances of the universally quantified input formulas and test these for propositional unsatisfiability. In presence of various techniques to avoid naive explicit expansion and the ability of recent SAT solvers to handle quite large propositional formulas this a practically feasible approach to first-order theorem proving. Thus, the question came up, whether there are quantifier-free formula expansions that are large enough to ensure that elimination can be performed on these instead of the original formula, in analogy to detecting unsatisfiability.

With the expectation of considering an important but somehow easier special case we focus on relational formulas, which underlie most formalizations of databases. Of course, functions can be represented by predicates, such that

[3] [15, p. 336] and a letter by Ackermann dated 1 November 1928 [22] suggest that Löwenheim earlier obtained similar results.

[4] That these classes are restricted to *relational* formulas can be guessed from the symbolic notation in [21] (actually only the case with a single unquantifed predicate seems considered there) and the observation that if function symbols would be permitted, then the second class would be as expressive as the third one.

without restriction on quantification, the *relational* property is not a limitation of expressive power. We focus on the restriction to *universal* relational formulas. In Skolemized form, formulas of the Bernays-Schönfinkel-Ramsey class are such formulas. Typically, in contrast to many resolution-based methods [10], instance-based methods decide universal relational formulas. An intuitive argument is that these formulas trivially have a largest expansion that needs to be considered to determine satisfiability. Do they also have in some sense a largest expansion that needs to be considered for elimination?

The obtained results are not as positive as desired, but at least shed some light on the properties of elimination problems with certain quantification restrictions. The considered formulas have an existential second-order quantifier prefix, followed by a universal first-order prefix and a quantifier-free formula. It is shown that for such a second-order formula F a sequence $\{G_0, G_1, G_2, \ldots\}$ of universal relational first-order formulas that have (not necessarily strictly) increasing strength and are all entailed by F can be constructed. Any first-order consequence of F is a consequence of some G_i. Formula F has a resultant if and only if it is equivalent to some G_i. The set of the formulas G_i that constitute a resultant of F or, more generally, have the same first-order consequences as F is co-recursively enumerable. Also the set of first-order formulas that entail F is co-recursively enumerable.

Craig [6] considered the question of constructing a so-called *base* for a given formula with existential second-order prefix, that is, a recursive set of first-order formulas whose set of consequences is identical to the set of first-order consequences of the given second-order formula. He notes that this corresponds to a weakened version of Ackermann's [1] generalized notion of *resultant*. Our construction of $\{G_0, G_1, G_2, \ldots\}$ can be viewed as construction of a monotonic base, actually with similar techniques as in [6], but where the restriction to universal relational formulas allows to derive additional properties.

The rest of the paper is structured as follows: Notation and definitions of the considered formula classes are introduced in Sect. 2 and the toolkit of the used techniques is specified in Sect. 3. In Sect. 4 then the main results are stated, proven and informally described, followed by a discussion of related work, potential applications and open issues in Sect. 5. Section 6 concludes the paper.

2 Notation and Preliminaries

We consider second-order formulas where second-order quantification is just upon predicates (in contrast to functions), or, in other words, first-order formulas extended by second-order quantification upon predicates. They are constructed from atoms (including equality atoms), constant operators $\top$, $\bot$, the unary operator $\neg$, binary operators $\wedge, \vee$ and quantifiers $\forall, \exists$ with their usual meaning. Further binary operators $\rightarrow$, $\leftarrow$, $\leftrightarrow$, as well as n-ary versions of $\wedge$ and $\vee$ can be understood as meta-level notation. The operators $\wedge$ and $\vee$ bind stronger than $\rightarrow$, $\leftarrow$ and $\leftrightarrow$. The scope of $\neg$, the quantifiers, and the n-ary connectives is the immediate subformula to the right.

A subformula occurrence has in a given formula *positive (negative) polarity* if it is in the scope of an even (odd) number of negations. A *vocabulary* is a set of *symbols*, that is, predicate symbols (briefly *predicates*), function symbols (briefly *functions*) and *individual symbols*. (Function symbols are assumed to have an arity ≥ 1. Individual symbols are not partitioned into variables and constants. Thus, an individual symbol is – like a predicate in second-order logic – considered as variable if and only if it is bound by a quantifier.) The set of symbols that occur *free* in a formula F is denoted by $\mathcal{V}(F)$, the set of predicate symbols that occur free in F by $\mathcal{V}_P(F)$, and the set of individual symbols that occur free in F (commonly termed the set of *constants* occurring in F) by $\mathcal{V}_C(F)$. The arity of a predicate or function symbol s is denoted by $\mathsf{arity}(s)$.

We use straightforward shorthands based on sequences of terms for expressing application of predicates and functions to arguments, simultaneous comparison of multiple terms and quantifying upon multiple variables: If $\boldsymbol{t} = t_1, \ldots, t_n$ is an n-ary sequence of terms, and s is an n-ary predicate or function symbol, we write $s\boldsymbol{t}$ for $s(t_1, \ldots, t_n)$, or, in case $n = 0$, for s, respectively. If $\boldsymbol{t} = t_1, \ldots, t_n$ and $\boldsymbol{u} = u_1, \ldots, u_n$ are both n-ary sequences of terms, we write $\boldsymbol{t} = \boldsymbol{u}$ for $t_1 = u_1 \wedge \ldots \wedge t_n = u_n$ and $\boldsymbol{t} \neq \boldsymbol{u}$ for $\neg(\boldsymbol{t} = \boldsymbol{u})$. If $\boldsymbol{x} = x_1, \ldots, x_n$ is a sequence of individual symbols or of predicates, we write $\exists \boldsymbol{s}$ ($\forall \boldsymbol{s}$, resp.) for $\exists s_1 \ldots \exists s_n$ ($\forall s_1 \ldots \forall s_n$, resp.).

We consider three specific formula classes whose members are constructed as a second-order quantifier prefix followed by a first-order quantifier prefix and a quantifier-free relational formula of first-order logic with equality. These classes are characterized by restrictions on the second-order and first-order prefix as shown in the following table, where the R in the class names points out the restriction to relational formulas:

Formula class	Second-order prefix	First-order prefix
$\forall$R-formulas	empty	$\forall x_1 \ldots \forall x_n$
$\exists\forall$R-formulas	empty	$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n$
$\exists\forall$R-formulas	$\exists p_1 \ldots \exists p_m$	$\forall x_1 \ldots \forall x_n$

The class $\forall$R-*formulas* is the class of universal relational first-order formulas. The class $\exists\forall$R-*formulas* is also known as Bernays-Schönfinkel-Ramsey class.

If F, G are formulas, we write $F \models G$ for F *entails* G; $\models F$ for F *is valid*; and $F \equiv G$ for F *is equivalent to* G, that is, $F \models G$ *and* $G \models F$. If $\mathfrak{I}$ is an interpretation and F is a formula, we write $\mathfrak{I} \models F$ for $\mathfrak{I}$ *is a model of* F.

3 Underlying Toolkit

We now specify the technical background underlying the proofs of the main results. It is essentially a small toolkit of formula-based concepts and construction techniques used in automated deduction.

3.1 Second-Order Quantifier Elimination

The objective of *second-order quantifier elimination* is to compute for a given second-order formula a first-order *resultant*, which is characterized as follows:

Definition 1 (Resultant). A *resultant* of a second-order formula F is a formula F' such that

1. F' is first-order,
2. $F' \equiv F$,
3. $\mathcal{V}(F') \subseteq \mathcal{V}(F)$.

It follows immediately from condition 2. of Def. 1 that all resultants of a given formula are equivalent. Hence, we also speak of *the* resultant of a second-order formula. From conditions 1. and 3. it follows that none of the quantified predicates possibly occurring in F do occur in F'. They are, so-to-speak, "forgotten" in F'.

The following proposition shows an equivalence of a second-order formula with a certain shape and a first-order formula, due to Ackermann [1]. It can be applied to compute the resultant of a second-order formula that matches its left side by rewriting with the right side. The DLS algorithm [7,5] for second-order quantifier elimination surrounds such rewriting steps with pre- and postprocessing operations.

Proposition 2 (Ackermann's Lemma [1]). *Let p be an n-ary predicate, let G and H be first-order formulas such that p does not occur in G, let $\boldsymbol{x} = x_1, \ldots, x_n$ be a sequence of distinct individual symbols that do not occur bound in G and do not occur in H. Let $H[p \mapsto G]$ stand for H with each occurrence of atoms $p(t_1, \ldots, t_n)$ whose predicate is p replaced by $G\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, that is, G under the substitution that maps x_i to t_i, for $i \in \{1, \ldots, n\}$. If p does not occur with positive polarity in H, then*

$$\exists p\,(\forall \boldsymbol{x}(p\boldsymbol{x} \vee G) \wedge H) \;\equiv\; H[p \mapsto G].$$

Ackermann's lemma also holds in a dual variant: If p does not occur with negative polarity in H, then $\exists p\,(\forall \boldsymbol{x}(\neg p\boldsymbol{x} \vee G) \wedge H) \;\equiv\; H[p \mapsto G]$. For quantifier-free formulas (also with function symbols) with an existential second-order prefix, it is always possible to compute a resultant on the basis of Ackermann's lemma, as demonstrated with the following algorithm:

Algorithm 3 (Second-Order Quantifier Elimination upon Quantifier-Free Formulas).
INPUT: An formula that consists of a prefix of existential predicate quantifiers followed by a quantifier-free formula of first-order logic with equality.
METHOD: Starting with the input formula, repeatedly eliminate the innermost existential second-order quantifier with the following method: Given is a formula $\exists p\, F$, where F is quantifier-free. Convert F to disjunctive normal form $K_1 \vee \ldots \vee K_n$, where each disjunct is a conjunction of literals. Propagate the second-order quantifier inwards to obtain the formula $\exists p\, K_1 \vee \ldots \vee \exists p\, K_n$, which is equivalent to $\exists p\, F$. Eliminate $\exists p$ in each disjunct individually: Arrange the disjunct in the form

$$\exists p\,(p\boldsymbol{t}_1 \wedge \ldots \wedge p\boldsymbol{t}_k \wedge \neg p\boldsymbol{u}_1 \wedge \ldots \wedge \neg p\boldsymbol{u}_l) \wedge K',$$

where p does not occur in K' and $k, l \geq 0$. Rewrite this formula to the equivalent formula

$$\exists p\,(\forall \boldsymbol{x}\,(p\boldsymbol{x} \vee \boldsymbol{x} \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{x} \neq \boldsymbol{t}_k) \wedge \neg p\boldsymbol{u}_1 \wedge \ldots \wedge \neg p\boldsymbol{u}_l) \wedge K',$$

where $\boldsymbol{x}$ is a sequence of fresh individual symbols whose length is the arity of p. Apply Ackermann's Lemma (Prop. 2) to rewrite this formula to its resultant

$$(\boldsymbol{u}_1 \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{u}_1 \neq \boldsymbol{t}_k) \wedge \ldots \wedge (\boldsymbol{u}_l \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{u}_l \neq \boldsymbol{t}_k) \wedge K'.$$

Combine these individual resultants disjunctively to obtain a resultant of $\exists p\, F$. OUTPUT: A resultant of the input formula.

Algorithm 3 also justifies the computation of resultants of formulas of the form

$$\exists p_1 \ldots \exists p_m \exists x_1 \ldots \exists x_n\, F, \tag{i}$$

where F is quantifier-free. These are the formulas of the first of the three classes with existential second-order quantification explicitly considered in [21] as mentioned in the introduction, now generalized by allowing function symbols. Formula (i) is equivalent to $\exists x_1 \ldots \exists x_n \exists p_1 \ldots \exists p_n\, F$, obtained by switching the first- and second-order quantifier prefix. If F' is a resultant of $\exists p_1 \ldots \exists p_n\, F$, obtained for example with Algorithm 3, then $\exists x_1 \ldots \exists x_n F'$ is a resultant of (i).

Algorithm 3 can be considered as a specialized variant of DLS [7], with a simpler pre- and postprocessing that just suffices to handle predicate quantification applied to quantifier-free first-order formulas. In [6] a similar technique is used and some refinements are shown. In [9] a further variant of DLS is introduced that computes resultants of quantifier-free formulas under existential second-order quantification, however constrained such that all occurrences of a quantified predicate are replaced by the same "witness" formula, whereas in Algorithm 3 it is allowed that in the processing of each disjunct K_i a different formula $(\boldsymbol{x} \neq \boldsymbol{t}_1 \wedge \ldots \wedge \boldsymbol{x} \neq \boldsymbol{t}_k)$ is used to replace p. In fact, the algorithm from [9] can be considered as based on Algorithm 3, but followed by a second step in which the different replacement formulas in each disjunct are combined to a single witness that is suitable as replacement in all disjuncts. The necessity of only a single replacement formula is imposed by the restricted form of second-order quantifier elimination through finding a witnesses that is considered in [9], which can be also viewed as finding a Boolean unifier and as finding a particular solution of the Boolean solution problem [24].

3.2 Domain Closure Axioms and Restricted Prefixes

The *domain closure axiom*, which restricts the universe of discourse to just those individuals denoted by constants in the given formula, emerged as a tool for the logical modeling of relational databases [16]. Following [8], we consider a generalized variant that in addition permits a fixed number of further objects.

Definition 4 (Domain Closure Axiom). Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a nonempty set of individual symbols, let $k \in \mathbb{N}_0$, and assume that $x, y_1, \ldots, y_k$ are individual symbols that are not in $\mathcal{C}$. Define

$$\mathsf{DCA}^k_{\mathcal{C}} \stackrel{\text{def}}{=} \exists y_1 \ldots \exists y_k \forall x\, (x = y_1 \vee \ldots \vee x = y_k \vee x = c_1 \vee \ldots \vee x = c_n).$$

It is possible, to specify *domain closure axiom* with a formula as parameter whose free individual symbols are taken as considered set $\mathcal{C}$. However, considered as a function of formulas, *domain closure axiom* is then not "semantic",

that is, it might have semantically different values for semantically equivalent argument formulas, which, despite being equivalent, might have different sets of free individual symbols. To avoid "non-semantic" functions of formulas, we use here a version of *domain closure axiom* that is explicitly parameterized with a set $\mathcal{C}$ of individual symbols.

Domain closure axioms play a role in *general domain circumscription* [8]. As shown with [8, Thm. 6.1], the domain circumscription of an $\forall$R-formula F such that $\mathcal{V}_C(F) \neq \emptyset$ is equivalent to $\mathsf{DCA}^0_{\mathcal{C}} \wedge F$, where $\mathcal{C} = \mathcal{V}_C(F)$.

The remaining propositions in this section show properties of first-order formulas with restricted quantifier prefixes conjoined with domain closure axioms. Roughly, the intuition there is that for these formulas the addition of domain closure axioms does not essentially alter the semantics, but allows to eliminate first-order quantification by ground expansion with respect to the finite set of symbols presupposed in the domain closure axioms. The underlying core property is that from a model of an $\exists\forall$R-formula a further model can be derived that in addition also satisfies the domain closure axiom with respect to length of the existential quantifier prefix and the free individual symbols:

Proposition 5 (Domain Closure Extension for $\exists\forall$R-Formulas). *Let F be an $\exists\forall$R-formula and let $\mathfrak{I}$ be an interpretation such that $\mathfrak{I} \models F$. Then for all non-empty sets $\mathcal{C} \supseteq \mathcal{V}_C(F)$ of individual symbols and for all natural numbers k larger than or equal to the length of the existential quantifier prefix of F there exists an interpretation $\mathfrak{I}'$ such that*

1. *$\mathfrak{I}' \models \mathsf{DCA}^k_{\mathcal{C}} \wedge F$.*
2. *For all ground atoms A constructed from predicates in F and individual symbols in $\mathcal{C}$ it holds that $\mathfrak{I} \models A$ iff $\mathfrak{I}' \models A$.*

This proposition is not hard to prove by considering the submodel of $\mathfrak{I}$ obtained by restricting the domain to the union of the $\leq k$ individuals whose existence is presupposed by the existential quantifier prefix of F and the set of the values of the members of $\mathcal{V}_C(F)$ in $\mathfrak{I}$. From this proposition it follows that if an $\exists\forall$R-formula conjoined with a suitable domain closure axiom entails an $\forall\exists$R-formula, then also the $\exists\forall$R-formula alone entails that formula:

Proposition 6 (Redundancy of Domain Closure for $\exists\forall$R-$\forall\exists$R Entailments). *Let F be an $\exists\forall$R-formula and let G be an $\forall\exists$R-formula, let $\mathcal{C} \supseteq \mathcal{V}_C(F) \cup \mathcal{V}_C(G)$ be a non-empty set of individual symbols, and let k be a natural number that is larger than or equal to the sum of the length of the existential quantifier prefix of F and the length of universal quantifier prefix of G. It then holds that*

$$\text{if } \mathsf{DCA}^k_{\mathcal{C}} \wedge F \models G, \quad \text{then } F \models G.$$

Proof. Assume the left side of the proposition, that is, $\mathsf{DCA}^k_{\mathcal{C}} \wedge F \models G$. Assume further that the right side does not hold. Then there exists an interpretation $\mathfrak{I}$ such that $\mathfrak{I} \models F \wedge \neg G$. Observe that $F \wedge \neg G$ is equivalent to an $\exists\forall$R-formula with k existential quantifiers. Thus, by Prop. 5 there exists an interpretation $\mathfrak{I}'$ such that $\mathfrak{I}' \models \mathsf{DCA}^k_{\mathcal{C}} \wedge F \wedge \neg G$. With the assumption $\mathsf{DCA}^k_{\mathcal{C}} \wedge F \models G$ it then follows that $\mathfrak{I}' \models G \wedge \neg G$, which contradicts with $\mathfrak{I}'$ being an interpretation. □

From this proposition it follows that the semantics of $\forall$R-formulas F is preserved under conjunction with a suitable domain closure axiom:

Proposition 7 (Semanticity of Domain Closure for $\forall$R-formulas). *Let F, G be $\forall$R-formulas, let $\mathcal{C} \supseteq \mathcal{V}_C(F) \cup \mathcal{V}_C(G)$ $\mathcal{C}$ be a non-empty set of individual symbols, and let k be a natural number that is larger than or equal to the maximum of the quantifier prefix lengths of F and G. It then holds that*

$$\textit{if } \mathsf{DCA}^k_{\mathcal{C}} \wedge F \equiv \mathsf{DCA}^k_{\mathcal{C}} \wedge G, \quad \textit{then } F \equiv G.$$

Proof. Follows from Prop. 6. □

3.3 Resultants of $\exists\forall$R-formulas are Universal

Based on a strengthening of Craig's interpolation theorem it can be shown that whenever an $\exists\forall$R-formula has a resultant, then the resultant is equivalent to an $\forall$R-formula, and, moreover, that there is an effective method to compute for any given resultant of an $\exists\forall$R-formula such an equivalent $\forall$R-formula. Before we state this as a proposition, we show the underlying interpolation property:

Proposition 8 (Interpolants with $\exists\forall$R-Formula on the Left). *Let F be an $\exists\forall$R-formula and let G be a first-order formula such that $F \models G$. Then there is an effective method to compute from given F and G a formula H such that*

1. *H is an $\forall$R-formula.*
2. *$F \models H \models G$.*
3. *$\mathcal{V}_P(H) \subseteq \mathcal{V}_P(F) \cap \mathcal{V}_P(G)$.*
4. *$\mathcal{V}_C(H) \subseteq \mathcal{V}_C(F)$.*

Proof. Let G' be G conjoined with tautologies such that $\mathcal{V}_C(G') \supseteq \mathcal{V}_C(F)$, whereas $\mathcal{V}_P(G') = \mathcal{V}_P(G)$ and $G' \equiv G$. Let F' be the $\forall$R-formula obtained from F by renaming the quantified predicates with fresh symbols and dropping the second-order prefix. Compute H as Craig interpolant of F' and G' with the tableau-based interpolant construction method described in [19] and [11]. Conditions 2.–4. of the proposition follow from the properties of Craig interpolants: $F' \models H \models G'$ implies $F \models H \models G$; from $\mathcal{V}(H) \subseteq \mathcal{V}(F') \cap \mathcal{V}(G')$ it follows, since $\mathcal{V}_P(F') = \mathcal{V}_P(F)$ and $\mathcal{V}_P(G') = \mathcal{V}_P(G)$, that $\mathcal{V}_P(H) \subseteq \mathcal{V}_P(F) \cap \mathcal{V}_P(G)$, and, since $\mathcal{V}_C(F') = \mathcal{V}_C(F) \subseteq \mathcal{V}_C(G')$, that $\mathcal{V}_C(H) \subseteq \mathcal{V}_C(F)$. Condition 1. follows from particular features of the interpolant construction method of [19,11], where an existential quantifier in the interpolant would not be introduced if the left formula of the interpolation is universal and all individual symbols that occur free in the left formula also occur free in the right formula. □

Proposition 9 (Resultants of $\exists\forall$R-Formulas are Universal). *A resultant of an $\exists\forall$R formula F is equivalent to an $\forall$R-formula. Moreover, there is an effective method to compute from F and any given resultant a resultant that is an $\forall$R-formula.*

Proof. Assume that G is an arbitrary resultant of F. Thus $F \equiv G$. Let H be computed from F and G according to Prop. 8. It is easy to verify that H an $\forall$R-formula and is a resultant of F. □

4 Approximating Resultants of $\exists\forall_{\mathrm{R}}$-formulas

The main results of the paper are now shown: For a given $\exists\forall_{\mathrm{R}}$-formula F a sequence $\{G_0, G_1, G_2, \ldots\}$ of universal first-order formulas that have (not necessarily strictly) increasing strength and are all entailed by F can be constructed. Any first-order consequence of F is a consequence of some G_i. Formula F has a resultant if and only if it is equivalent to some G_i. However, only methods are provided to detect that the consequences of a given G_i do *not* include all first-order consequences of F, or, respectively, G_i is *not* equivalent to F. This leads to co-recursive enumerability of the first-order formulas that are equivalent to F or entail F, directly in case F has a resultant, or with respect the first-order consequences of F in case it has no resultant. Theorem 11 below makes this precise and shows further properties of the formulas G_i, some of which are underlying the proofs of the mentioned results. First the theorem is stated and proven formally, then the individual claimed properties are described informally.

The following definition is used in the statement of Theorem 11. It specifies notions of entailment and equivalence of second-order formulas modulo the sets of entailed first-order formulas.

Definition 10 (Entailment and Equivalence Modulo First-Order Consequences). For second-order formulas F, G define
(i) $F \models_{\mathsf{FO}} G$ if and only if for all first-order formulas H it holds that if $G \models H$, then $F \models H$.
(ii) $F \equiv_{\mathsf{FO}} G$ if and only if $F \models_{\mathsf{FO}} G$ and $G \models_{\mathsf{FO}} F$.

These notions are helpful to express properties of second-order formulas that possibly have no resultant. Their meaning is identical to the standard notions of entailment and equivalence, respectively, if no such second-order formulas are involved: If G is a first-order formula or a second-order formula that has a resultant, then, also in the case where F is a second-order formula, it holds that $F \models_{\mathsf{FO}} G$ if and only if $F \models G$. If each of F and G is first-order or has a resultant, then $F \equiv_{\mathsf{FO}} G$ if and only if $F \equiv G$.

Theorem 11 (Approximating Resultants of $\exists\forall_{\mathrm{R}}$-formulas). *Let F be an $\exists\forall_{\mathrm{R}}$-formula and let $\mathcal{C} \supseteq \mathcal{V}_C(F)$ be a nonempty set of individual constants. Then there exist $\forall_{\mathrm{R}}$-formulas $G_0, G_1, G_2, \ldots$ such that*

(a) The set $\{G_0, G_1, G_2, \ldots\}$ is recursive.
(b) $G_0 \mathrel{=\!|} G_1 \mathrel{=\!|} G_2 \mathrel{=\!|} \ldots \mathrel{=\!|} F$.
(c) For all $i \in \mathbb{N}_0$ it holds that $\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \equiv \mathsf{DCA}^i_{\mathcal{C}} \wedge F$.
(d) For all $i, j \in \mathbb{N}_0$ such that $i \leq j$ it holds that $\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \models \mathsf{DCA}^j_{\mathcal{C}} \wedge G_j$.
(e) For all $i \in \mathbb{N}_0$ it holds that F has a resultant that is an $\forall_{\mathrm{R}}$-formula with quantifier prefix length i if and only if $G_i \equiv F$.
(f) F has a resultant if and only if there exists a $k \in \mathbb{N}_0$ such that $G_k \equiv F$.
(g) For all $i \in \mathbb{N}_0$ and $\forall_{\mathrm{R}}$-formulas H with quantifier prefix length i it holds that $F \models H$ if and only if $G_i \models H$.
(h) There is an effective method to compute from F and a given first-order formula H such that $F \models H$ a number $k \in \mathbb{N}_0$ such that $G_k \models H$.

(*i*) *The set* $\{i \mid i \in \mathbb{N}_0 \text{ and } G_i \equiv_{\mathsf{FO}} F\}$ *is co-recursively enumerable.*
(*j*) *The set* $\{i \mid i \in \mathbb{N} \text{ and } H_i \models_{\mathsf{FO}} F\}$*, where* $\{H_1, H_2, H_3, \ldots\}$ *is the set of all* $\exists\forall_{\mathrm{R}}$*-formulas, is co-recursively enumerable (under assumption of a countable vocabulary).*

Proof. Let

$$F \;=\; \exists p_1 \ldots \exists p_n \, \forall x_1 \ldots \forall x_m \, F',$$

where F' is quantifier-free. Let $F'[a_1, \ldots, a_m]$ denote F' with x_i replaced by some individual symbol a_i, for all $i \in \{1, \ldots, m\}$. Let $\mathcal{U}$ be a set $\{u_1, u_2, u_3, \ldots\}$ of individual symbols that are not in $\mathcal{C}$ and let $\mathcal{U}_i$ denote $\{u_1, \ldots, u_i\}$. For $i \in \mathbb{N}_0$ construct G_i as

$$G_i \;\stackrel{\text{def}}{=}\; \forall u_1 \ldots \forall u_i \, G_i',$$

where G_i' is the resultant, computed by Algorithm 3, of the following formula:

$$\exists p_1 \ldots \exists p_n \bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m].$$

The following semantic property of G_i, for all $i \in \mathbb{N}_0$, then immediately follows from the construction of G_i:

$$(*) \qquad G_i \;\equiv\; \forall u_1 \ldots \forall u_i \, \exists p_1 \ldots \exists p_n \, \textstyle\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m].$$

(a) Follows from the construction of the formulas G_i: For any given first-order formula whose universal first-order quantifier prefix has length $i \geq 0$ it can be decided whether it is a member of $\{G_0, G_1, G_1, \ldots\}$ by comparing it syntactically with G_i.

(b) Let $i, j \in \mathbb{N}_0$ that $i \leq j$. Since then $\mathcal{U}_j \supseteq \mathcal{U}_i$ it follows that

$$\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_j} F'[a_1, \ldots, a_m] \;\models\; \bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m].$$

With $(*)$ this implies $G_j \models G_i$. To conclude the proof of (b) we show that for an arbitrary number $i \in \mathbb{N}_0$ it holds that $F \models G_i$. This can be proven in the following steps, where the last step, the contraction into G_i, is justified by $(*)$:

$$\begin{aligned}
& F \\
\equiv\; & \exists p_1 \ldots \exists p_n \, \forall x_1 \ldots \forall x_m \, F' \\
\models\; & \exists p_1 \ldots \exists p_n \, \forall u_1 \ldots \forall u_i \, \textstyle\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\models\; & \forall u_1 \ldots \forall u_i \, \exists p_1 \ldots \exists p_n \, \textstyle\bigwedge_{a_1, \ldots, a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\equiv\; & G_i.
\end{aligned}$$

(c) The right-to-left direction follows immediately from (b). The left-to-right direction can be shown in the following steps, where the expansion of G_i at the first step is justified by $(*)$:

$$
\begin{aligned}
& \mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \\
\equiv\ & \mathsf{DCA}^i_{\mathcal{C}} \wedge \forall u_1 \ldots \forall u_i\, \exists p_1 \ldots \exists p_n \bigwedge_{a_1,\ldots,a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\equiv\ & \exists u_1 \ldots \exists u_i\, \mathsf{DCA}^0_{\mathcal{C} \cup \mathcal{U}_i} \wedge \\
& \quad \forall u_1 \ldots \forall u_i\, \exists p_1 \ldots \exists p_n \bigwedge_{a_1,\ldots,a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m] \\
\models\ & \exists u_1 \ldots \exists u_i\, (\mathsf{DCA}^0_{\mathcal{C} \cup \mathcal{U}_i} \wedge \exists p_1 \ldots \exists p_n \bigwedge_{a_1,\ldots,a_m \in \mathcal{C} \cup \mathcal{U}_i} F'[a_1, \ldots, a_m]) \\
\equiv\ & \exists u_1 \ldots \exists u_i\, \mathsf{DCA}^0_{\mathcal{C} \cup \mathcal{U}_i} \wedge \exists p_1 \ldots \exists p_n\, \forall x_1 \ldots \forall x_m\, F' \\
\equiv\ & \mathsf{DCA}^i_{\mathcal{C}} \wedge \exists p_1 \ldots \exists p_n\, \forall x_1 \ldots \forall x_m\, F' \\
\equiv\ & \mathsf{DCA}^i_{\mathcal{C}} \wedge F.
\end{aligned}
$$

(d) From $i \leq j$ it follows that $\mathsf{DCA}^i_{\mathcal{C}} \models \mathsf{DCA}^j_{\mathcal{C}}$. By ($c$) we can conclude $\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \equiv \mathsf{DCA}^i_{\mathcal{C}} \wedge F \models \mathsf{DCA}^j_{\mathcal{C}} \wedge F \equiv \mathsf{DCA}^j_{\mathcal{C}} \wedge G_j$.

(e) The right-to-left direction follows immediately from the construction of G_i: If $G_i \equiv F$, then G_i clearly is a resultant of F that is an $\forall$R-formula with quantifier prefix length i. The left-to-right direction can be show as follows: Assume that there exists an $\forall$R-formula H with quantifier prefix length i that is a resultant of F. Since $H \equiv F$ it follows from (c) that

$$\mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \ \equiv\ \mathsf{DCA}^i_{\mathcal{C}} \wedge H.$$

This equivalence matches the precondition of Prop. 7 that lets us deduce $G_i \equiv H$, implying $G_i \equiv F$.

(f) Follows from (e) and Prop. 9.

(g) The right-to-left direction follows immediately from (b). The left-to-right direction can be shown in the following steps, where the last two equivalences follow from (c) and Prop. 6, respectively:

$$F \models H \ \text{ implies } \ \mathsf{DCA}^i_{\mathcal{C}} \wedge F \models H \ \text{ iff } \ \mathsf{DCA}^i_{\mathcal{C}} \wedge G_i \models H \ \text{ iff } \ G_i \models H.$$

(h) By Prop. 8, we can compute from F and H an $\forall$R-formula K such that $F \models K \models H$. Let k be the length of the quantifier prefix of K. From (g) it follows that $G_k \models K$, hence $G_k \models H$.

(i) The following algorithm halts for a given $i \in \mathbb{N}_0$ if and only if $G_i \not\equiv_{\mathsf{FO}} F$: Let j be an integer variable that satisfies the invariant $j > i$ and is initialized to $i + 1$. Proceed in a loop: Test whether $G_i \models G_j$, that is, whether $G_i \wedge \neg G_j$ is satisfiable. Since G_i and G_j are $\forall$R-formulas, $G_i \wedge \neg G_j$ is an $\exists\forall$R-formula, and thus decidable. If the test succeeds, then halt, else increment j by 1 and re-enter the loop.

That the algorithm indeed halts if and only if $G_i \not\equiv_{\mathsf{FO}} F$ can be shown as follows: By (b) it holds that $F \models G_i$. Hence $G_i \not\equiv_{\mathsf{FO}} F$ if and only $G_i \not\models_{\mathsf{FO}} F$. Consider the case $G_i \not\models_{\mathsf{FO}} F$ and first the subcase where there exists a natural number $j > i$ such that $G_i \not\models G_j$. Then the satisfiability test in the algorithm eventually succeeds and the algorithm halts. Now consider the alternate subcase where no such number j exists. From (b) it then follows that for all $l \in \mathbb{N}_0$ it holds that $G_i \models G_l$. With (h) we can conclude that it holds for all first-order formula H that if $F \models H$, then $G_i \models H$. Hence $G_i \models_{\mathsf{FO}} F$, contradicting the assumption $G_i \not\models_{\mathsf{FO}} F$ made for that case, and thus yielding the alternate subcase impossible.

Now consider the case $G_i \models_{\mathsf{FO}} F$. From the definition of $\models_{\mathsf{FO}}$ it follows that for all $j \geq 0$ it holds that if $F \models G_j$, then $G_i \models G_j$. With (b) it follows that for all $j \geq 0$ it holds that $G_i \models G_j$, which implies that the satisfiability test in the algorithm never succeeds and the algorithm thus loops forever.

(j) The following algorithm halts for a given $\exists\forall_{\mathrm{R}}$-formula H if and only if $H \not\models_{\mathsf{FO}} F$: Let j be an integer variable that is initialized to 0. Proceed in a loop: Test whether $H \not\models G_j$, that is, whether $H \wedge \neg G_j$ is satisfiable. Since H is an $\exists\forall_{\mathrm{R}}$-formula and G_i is an $\forall_{\mathrm{R}}$-formula, $H \wedge \neg G_j$ is an $\exists\forall_{\mathrm{R}}$-formula, and thus decidable. If the test succeeds, then halt, else increment j by 1 and re-enter the loop.

That the algorithm indeed halts if and only if $H \not\models_{\mathsf{FO}} F$ follows since $H \not\models_{\mathsf{FO}} F$ holds if and only if there exists a $j \in \mathbb{N}_0$ such that $H \not\models G_j$, or, equivalently, $H \models_{\mathsf{FO}} F$ if and only if for all $j \in \mathbb{N}_0$ it holds that $H \models G_j$. The left-to-right direction of this equivalence can be proven as follow: Assume the left side $H \models_{\mathsf{FO}} F$. By expanding $\models_{\mathsf{FO}}$ this can be expressed as: For all first-order formulas K it holds that if $F \models K$, then $H \models K$. Hence, for all $j \in \mathbb{N}_0$ it holds that if $F \models G_j$, then $H \models G_j$. With (b) it follows that for all $j \in \mathbb{N}_0$ it holds that $H \models G_j$, that is, the right side. The right-to-left direction of the equivalence to show can be proven as follows: From (h) it follows that for all first-order formulas K it holds that if $F \models K$, then there exists a $k \in \mathbb{N}_0$ such that $G_k \models K$. Hence, if for all $i \in \mathbb{N}_0$ it holds that $H \models G_j$, then for all first-order formulas K such that $F \models K$ it holds that $H \models K$. By contracting into $\models_{\mathsf{FO}}$, the latter statement can be expressed as: If for all $j \in \mathbb{N}_0$ it holds that $H \models G_j$, then for $H \models_{\mathsf{FO}} F$, that is, the right-to-left direction of the equivalence to show. □

The formulas G_i whose existence is claimed by Theorem 11 are constructed from F and i as follows: The first-order quantifiers in F, which are universal, are eliminated by expansion with respect to the members of $\mathcal{C}$ and i additional individual symbols $u_1, \ldots, u_i$. Then a resultant of the obtained formula, that is, of the second-order quantifier prefix of F applied to the quantifier-free expansion, is computed. The formula G_i is then obtained by prefixing that resultant, which is quantifier-free, with existential first-order quantifiers upon $u_1, \ldots, u_i$.

By property (b), with increasing i the formulas G_i get (not necessarily strictly) stronger, and all the formulas G_i are entailed by F.

Property (c) holds invariantly for all $i \in \mathbb{N}_0$: Formula G_i "under domain closure with i existential individuals" (that is, conjoined with $\mathsf{DCA}^i_{\mathcal{C}}$) is equivalent to F under domain closure with the same number of existential individuals. This property is used in the proofs of (d), (e), and (g).

Property (d), which follows from (c), shows that with increasing i the formulas G_i under domain closure with i existential objects get (not necessarily strictly) *weaker*, conversely to the formulas G_i themselves, as shown with (b).

Properties (e) and (f) show necessary and sufficient conditions for the existence of a resultant of F, and in case of existence give a resultant. The first of these, (e), states that F has a resultant that is a universal relational formula with quantifier prefix length i if and only if G_i is equivalent to F. This property follows from (c) and Prop. 7. With Prop. 9 it leads to (f), which states that F has a resultant if and only if it is equivalent to G_k for some $k \in \mathbb{N}_0$. The

right-to-left directions of the respective equivalences $G_i \equiv F$ and $G_k \equiv F$ are immediate from (b), such that the existence of a resultant can be also characterized with just the entailments $G_i \models F$ and $G_k \models F$, respectively, instead. These entailments have F on their right side, a second-order formula, such that, differently from first-order logic, there is in general no algorithm that halts if and only if such an entailment holds.

Property (g) shows that F and G_i have the same $\forall_R$-formulas with quantifier prefix length i as consequences. This follows from (c) and Prop. 6. It is used together with the strengthened Craig interpolation property Prop. 8 to prove (h), by which any first-order consequence of F is a consequence of some G_k, and, moreover, such an index k can be effectively computed from F and the given consequence. Property (h) is applied to prove (i) and (j).

Properties (i) and (j) show certain settings where co-recursive enumerability with respect to equivalence and entailment, respectively, of the second-order formula F can be established. Property (i) states that the set of (the index numbers i of) the formulas G_i that are different from F with respect to their first-order consequences is recursively enumerable. This means that there is an algorithm that halts for given i if and only if G_i is *not* a formula with the same first-order consequences as F. If F has a resultant, this is equivalent to the statement that G_i *is not a resultant of* F. Property (i) is proven by giving such an algorithm and showing its correctness with referring to (b) and (h). By (b) the negated equivalence $G_i \not\equiv_{\mathsf{FO}} F$ can also be expressed as the negated entailment $G_i \not\models_{\mathsf{FO}} F$, which in the case where F has a resultant is equivalent to $G_i \not\models F$. From the perspective of trying to find a resultant of F or, more generally, a formula with the same first-order consequences as F, the property (i) only justifies a method to exclude failing candidate formulas G_i.

Property (j) states that the set of (the code numbers in some arithmetization of syntax of) the first-order formulas that do *not* entail F with respect to its first-order consequences is recursively enumerable. This means that there is an algorithm that halts for a given first-order formula H and only if $H \not\models_{\mathsf{FO}} F$. Like (i), this property is proven by giving such an algorithm and showing its correctness with referring to (b) and (h). The property justifies a method that detects for a given first-order formula H in the case where F has a resultant that F is *not* a consequence of H, and in the case where F has no resultant that *not* all first-order consequences of F are included in the consequences of H.

5 Discussion and Open Issues

In this section, Craig's work [6] on recursive *bases* is briefly compared, attempts are made to place the results of Theorem 11 in the context of applications of second-order quantifier elimination, open issues are shown, and potential directions for further research are indicated.

Comparison to Craig's Construction of Recursive Bases. The setting in [6] is more general and comprehensive: First-order formulas under existential second-order quantification are considered without assuming syntactic restrictions and also the case without equality is considered. Our syntactic restriction allows an apparently simpler construction of the approximation formulas G_i for

which monotonicity (i.e., $G_0 \models\!\!\models G_1 \models\!\!\models G_2 \models\!\!\models \ldots$) directly follows. The restriction also allows to derive further properties of the approximation formulas: They are universal and the length of their quantifier prefix is related to that of entailed universal first-order consequences of the second-order formula. Decidability of specific entailment problems, which is implied by the syntactic restrictions, leads to co-recursive enumerability of certain sets.

Our construction of an approximation formula G_i for a second-order formula $F = \exists \boldsymbol{p}\, F'$ where F' is first-order involves the construction of a first-order prefix $\boldsymbol{Q}_i$ and of an intermediate quantifier-free formula F'_i such that $F = \exists \boldsymbol{p}\, F' \models \exists \boldsymbol{p}\boldsymbol{Q} F'_i \models \boldsymbol{Q}\exists \boldsymbol{p} F'_i \equiv G_i$. The setting in [6] is the same, except that the first entailment is replaced by an equivalence, that is, it holds that $\exists \boldsymbol{p}\, F' \equiv \exists \boldsymbol{p}\boldsymbol{Q} F'_i$. Actually, the techniques in [6] seem even to preserve $F' \equiv \boldsymbol{Q} F'_i$. Of course, more weakly constrained transformations, in our case mainly justified through the use of domain closure axioms, are favorable. However, it remains to be investigated in how far the weaker constraints are made possible through the restricted formula class and whether there are associated complexity properties.

Entailments Involving Existential Second-Order Formulas. Property (j) of Theorem 11 can be considered in the context of mechanical verification and falsification (that is, existence of an algorithm that terminates in case a statement does hold or does not hold, respectively) of entailments of the forms $F \models H$ and $H \models F$, where F is a second-order formula with an existential second-order quantifier prefix applied to a first-order formula and H is a first-order formula.

An application of the first form $F \models H$ is to verify that a first-order formula A is semantically independent from predicates $p_1 \ldots, p_n$, which can be expressed as $\exists p_1 \ldots \exists p_n\, A \models A$. The first form $F \models H$ is straightforwardly accessible to *verification*: The set $\{i \mid i \in \mathbb{N} \text{ and } F \models H_i\}$, where $\{H_1, H_2, H_3, \ldots\}$ is the set of all first-order formulas, is recursively enumerable, which follows from recursive enumerability of the set of (the code numbers of) the valid first-order formulas. The entailment $F \models H_i$ is equivalent to the first-order entailment $F' \models H_i$, where F' is obtained from F by dropping the second-order prefix and renaming the quantified predicates with fresh symbols. The entailment $F' \models H_i$ can then be expressed as validity of $F' \rightarrow H_i$. Moreover, if F and H_i are restricted such that $F' \wedge \neg H_i$ belong to a decidable formula class, then $\{i \mid i \in \mathbb{N} \text{ and } F \models H_i\}$ is recursive, allowing then also to *falsify* entailments of the form $F \models H$.

An application of the second form $H \models F$ is expressing for first-order formulas A and B that $A \wedge B$ is a conservative extension of A as the entailment $A \models \exists p_1 \ldots \exists p_n (A \wedge B)$, where $p_1, \ldots, p_n$ are the predicates that occur in B but not in A. Justified by property (j) of Theorem 11, the second form $H \models F$ (if restricted to $\exists\forall_{\mathrm{R}}$-formulas H and $\exists\forall$-formulas F) can be mechanically *falsified*.

To sum up, entailments of the form $F \models H$ can always be verified and for formula classes that lead to decidable formulas $F \wedge \neg H$ also be falsified. Property (j) of Theorem 11 extends this, by establishing that also entailments of the form $H \models F$ can be falsified, for certain formula classes.

Approximate Resultants with Respect to Quantifier Prefix Length. By property (g) of Theorem 11, the $\exists\forall_{\mathrm{R}}$-formula F and the formulas G_i constructed from it have the same $\forall_{\mathrm{R}}$-formulas *with quantifier prefix length* i as

consequences. In a sense, the formulas G_i can be considered as capturing the first-order semantics of F "up to quantifier prefix length i". This suggests to consider G_i as resultant of a generalized form of elimination where not just the existentially quantified predicates are "forgotten", but also the part of the formula's meaning that would be only expressible with quantifier prefix length $> i$. Exploring this idea is an open issue.

Showing Non-Recursiveness. Properties (i) and (j) of Theorem 11 show co-recursive enumerability of certain sets related to second-order quantifier elimination. It remains to consider the question whether this is the strongest recursiveness property that can be asserted about these sets, that is, to show whether they are actually *not recursive*. It is expected that this holds because the formula $\exists f\,(x \wedge \neg fy \wedge \forall u \forall v\,(\neg fu \vee fv \vee \neg nuv)$ used in [1] to show non-existence of a resultant is actually an $\exists\forall_{\mathrm{R}}$-formula.

Potential Approaches for Strengthening the Co-Recursive Enumerability. Of course, it would be of interest, not just for practical application, to strengthen the co-recursive enumerability of finding resultants and verifying entailments shown with properties (i) and (j) of Theorem 11 to recursiveness, at least for special cases. So far, this is an open issue. A direction for further investigation might be trying to determine for certain $\exists\forall_{\mathrm{R}}$-formulas the maximally required quantifier prefix length of the resultant. A further direction could be ensuring that a semantic fixed point G_k of the sequence $G_0, G_1, G_2, \ldots$ subsumes all G_i with $i \geq k$, that is, for all $i \geq k$ it holds that $G_i \equiv G_k$. This would follow, for example, if for all $i, j \in \mathbb{N}_0$ it holds that if $G_i \equiv G_j$, then $G_{i+1} \equiv G_{j+1}$. A third direction would be trying to express *for all $i \geq k$ it holds that $G_i \equiv G_k$* in some algorithmically verifiable way.

Possibly Generalization to Further Formula Classes. Theorem 11 takes decidability of the Bernays-Schönfinkel-Ramsey class ($\exists\forall_{\mathrm{R}}$-formulas) as basis to derive co-recursive enumerability of problems related to computing elimination resultants of existential second-order quantifiers upon universal relational formulas ($\exists\forall$-formulas). This raises the question, whether the techniques applied there can be generalized to further formula classes.

A straightforward transfer appears to be the computation of resultants of formulas of the Bernays-Schönfinkel-Ramsey class under existential second-order quantification, by switching the existential second- and first-order quantifier prefixes and then considering elimination of the inner $\exists\forall_{\mathrm{R}}$-formula. However, with this approach a further source of failure to find resultants sneaks in: There are $\exists\exists\forall_{\mathrm{R}}$-formulas that have a resultant, but where after the suggested quantifier switching the obtained inner $\exists\forall_{\mathrm{R}}$-formula does not have a resultant. As an example, consider a formula $\exists p\,(x = a \vee F)$ that does not have a resultant. However, $\exists p\,\exists x\,(x = a \vee F)$ clearly has for arbitrary formulas F the resultant $\top$.

Avoiding Explicit Ground Expansion. The construction of the approximation formulas G_i described in the proof of Theorem 11 involves ground expansion of the input formula F. As in instance-based theorem proving, such expansions are useful as a conceptual construct, but their construction should in practice usually be avoided as much as possible. With respect to elimination, a poten-

tial approach might be the use of quantifier relativizations to compactly express formulas equivalent to expansions. Possibly the resultant required in the construction of G_i can then be computed by applying the elimination method for single ground atoms described in [13] to the finite number of atoms upon which the quantifiers are relativized. Also techniques of [6], where quantified formulas are duplicated by conjoining copies of them, but without performing instantiation, might be relevant here.

6 Conclusion

We have considered second-order quantifier elimination for a class of relational formulas characterized by a restriction of the quantifier prefix: existential predicate quantifiers followed by universal individual quantifiers. The main original motivation was to transfer instance-based techniques from automated theorem proving to second-order quantifier elimination. The technical result, however, does not indicate an immediate possibility for such a transfer, but gives some insight into the elimination problem for this class: The set of elimination resultants of a given formula and the set of formulas entailing the given second-order formula of that class is co-recursively enumerable. Candidate resultants can be generated, and there is an algorithm that halts on exactly those candidates that are *not* a resultant. Similarly, there is a method to detect that a given first-order formula does *not* entail the given second-order formula. By comparing formulas with respect to their first-order consequences, it is possible to express the respective theorem statements in a generalized way that applies to given second-order formulas independently of whether they have a resultant. These results were proven on the basis of small number of formula-based tools used in automated deduction. Actually, the results and involved constructions might be seen as a specialization to a formula class of Craig's setting of determining recursive *bases* for subtheories of first-order formulas. The hope is that some inspiration and material for further investigation of "eliminability", that is, existence of a resultant, or, more generally, of a formula that is equivalent with respect to first-order consequences, is provided.

Acknowledgments. This work was supported by DFG grant WE 5641/1-1.

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Math. Ann. 110, 390–413 (1935)
2. Baumgartner, P., Thorstensen, E.: Instance based methods – A brief overview. KI 24(1), 35–42 (Apr 2010)
3. Behmann, H.: Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. Math. Ann. 86(3–4), 163–229 (1922)
4. Börger, E., E., Grädel, Gurevich, Y.: The Classical Decision Problem. Springer (1997)
5. Conradie, W.: On the strength and scope of DLS. J. Applied Non-Classical Logic 16(3–4), 279–296 (2006)
6. Craig, W.: Bases for first-order theories and subtheories. J. Symb. Log. 25(2), 97–142 (1960)

7. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Autom. Reasoning 18(3), 297–338 (1997)
8. Doherty, P., Łukaszewicz, W., Szałas, A.: General domain circumscription and its effective reductions. Fundamenta Informaticae 36, 23–55 (1998)
9. Eberhard, S., Hetzl, S., Weller, D.: Boolean unification with predicates. J. Logic and Computation 27(1), 109–128 (2017)
10. Fermüller, C., Leitsch, A., Hustadt, U., Tammet, T.: Resolution decision procedures. In: Robinson, A., Voronkov, A. (eds.) Handb. of Autom. Reasoning, vol. 2, pp. 1793–1849. Elsevier (2001)
11. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer, 2nd edn. (1995)
12. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications (2008)
13. Lin, F., Reiter, R.: Forget It! In: Working Notes, AAAI Fall Symposium on Relevance. pp. 154–159 (1994)
14. Löwenheim, L.: Über Möglichkeiten im Relativkalkül. Math. Ann. 76, 447–470 (1915)
15. Löwenheim, L.: Funktionalgleichungen im Gebietekalkül und Umformungsmöglichkeiten im Relativkalkül. History of Philosophy and Logic 28, 305–336 (2007), assumed to be written in 1935 [20]
16. Reiter, R.: Deductive quastion-answering on relational databases. In: Gallaire, H., Minker, J. (eds.) Logic and Databases, pp. 149–178. Plenum Press, New York (1978)
17. Schröder, E.: Vorlesungen über die Algebra der Logik, vol. 1. Teubner (1890)
18. Skolem, T.: Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme welche gewisse Klassen von Aussagen betreffen. Videnskapsselskapets Skrifter I. Mat.-Nat. Klasse(3) (1919)
19. Smullyan, R.M.: First-Order Logic. Springer, New York (1968), also republished with corrections by Dover publications, New York, 1995
20. Thiel, C.: A short introduction to Löwenheim's life and work and to a hitherto unknown paper. History of Philosophy and Logic 28, 289–302 (2007)
21. Van Benthem, J., Doets, K.: Higher-order logic. In: Handbook of Philosophical Logic, vol. 1, pp. 189–243. Springer, second edn. (2001)
22. Wernhard, C.: Heinrich Behmann's contributions to second-order quantifier elimination. Tech. Rep. KRR 15–05, TU Dresden (2015)
23. Wernhard, C.: Second-order quantifier elimination on relational monadic formulas – A basic method and some less expected applications. In: TABLEAUX 2015. LNCS (LNAI), vol. 9323, pp. 249–265. Springer (2015)
24. Wernhard, C.: The Boolean solution problem from the perspective of predicate logic. In: FroCoS 2017. LNCS (LNAI), vol. 10483, pp. 333–350 (2017)

The Boolean Solution Problem from the Perspective of Predicate Logic (Abstract)

Christoph Wernhard

TU Dresden, Germany

Finding solution values for unknowns in Boolean equations was, along with second-order quantifier elimination, a principal reasoning mode in the *Algebra of Logic* of the 19th century. Schröder [19] investigated it as *Auflösungsproblem* (*solution problem*). It is closely related to the modern notion of Boolean unification. For a given formula that contains unknowns formulas are sought such that after substituting the unknowns with them the given formula becomes valid or, dually, unsatisfiable. Of interest are also most general solutions, condensed representations of all solution substitutions. A central technique there is the *method of successive eliminations*, which traces back to Boole. Schröder investigated *reproductive solutions* as most general solutions, anticipating the concept of *most general unifier*.

A comprehensive modern formalization based on this material, along with historic remarks, is presented by Rudeanu [17] in the framework of Boolean algebra. In automated reasoning variants of these techniques have been considered mainly in the late 80s and early 90s with the motivation to enrich Prolog and constraint processing by Boolean unification with respect to propositional formulas handled as terms [14,6,15,16,10,11]. The Π_2^P-completeness of Boolean unification with constants was proven only later in [10,11] and seemingly independently in [1]. Schröder's results were developed further by Löwenheim [12,13]. A generalization of Boole's method beyond propositional logic to relational monadic formulas has been presented by Behmann in the early 1950s [3,4]. Recently the complexity of Boolean unification in a predicate logic setting has been investigated for some formula classes, in particular for quantifier-free first-order formulas [8]. A brief discussion of Boolean reasoning in comparison with predicate logic can be found in [5].

Here we remodel the solution problem formally along with basic classical results and some new generalizations in the framework of first-order logic extended by second-order quantification. The main thesis of this work is that it is possible and useful to apply second-order quantification consequently throughout the formalization. What otherwise would require meta-level notation is then expressed just with formulas. As will be shown, classical results can be reproduced in this framework in a way such that applicability beyond propositional logic, possible algorithmic variations, as well as connections with second-order quantifier elimination and Craig interpolation become visible.

The envisaged application scenario is to let solving "solution problems", or Boolean equation solving, on the basis of predicate logic join reasoning modes

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

like second-order quantifier elimination (or "forgetting"), Craig interpolation and abduction to support the mechanized reasoning about relationships between theories and the extraction or synthesis of subtheories with given properties. On the practical side, the aim is to relate it to reasoning techniques such as Craig interpolation on the basis of first-order provers, SAT and QBF solving, and second-order quantifier elimination based on resolution [9] and the Ackermann approach [7]. Numerous applications of Boolean equation solving in various fields are summarized in [18, Chap. 14]. Applications in automated theorem proving and proof compression are mentioned in [8, Sect. 7]. The prevention of certain redundancies has been described as application of (concept) unification in description logics [2]. Here the synthesis of definitional equivalences is sketched as an application.

The material underlying the workshop presentation has in part been published as [20] and is described comprehensively in the report [21].

Acknowledgments. The author thanks anonymous reviewers for their helpful comments. This work was supported by DFG grant WE 5641/1-1.

References

1. Baader, F.: On the complexity of Boolean unification. Inf. Process. Lett. 67(4), 215–220 (Aug 1998)
2. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. Symb. Comput. 31, 277–305 (2001)
3. Behmann, H.: Das Auflösungsproblem in der Klassenlogik. Archiv für Philosophie 4(1), 97–109 (1950), (First of two parts, also published in *Archiv für mathematische Logik und Grundlagenforschung*, 1.1 (1950), pp. 17-29)
4. Behmann, H.: Das Auflösungsproblem in der Klassenlogik. Archiv für Philosophie 4(2), 193–211 (1951), (Second of two parts, also published in *Archiv für mathematische Logik und Grundlagenforschung*, 1.2 (1951), pp. 33-51)
5. Brown, F.M.: Boolean Reasoning. Dover Publications, second edn. (2003)
6. Büttner, W., Simonis, H.: Embedding Boolean expressions into logic programming. J. Symb. Comput. 4(2), 191–205 (1987)
7. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Autom. Reasoning 18(3), 297–338 (1997)
8. Eberhard, S., Hetzl, S., Weller, D.: Boolean unification with predicates. J. Logic and Computation 27(1), 109–128 (2017)
9. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: KR'92. pp. 425–435. Morgan Kaufmann (1992)
10. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z.: Constraint query languages. In: PODS'90. pp. 299–313. ACM Press (1990)
11. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z.: Constraint query languages. J. Comput. Syst. Sci. 51(1), 26–52 (1995)
12. Löwenheim, L.: Über das Auflösungsproblem im logischen Klassenkalkül. In: Sitzungsberichte der Berliner Mathematischen Gesellschaft. vol. 7, pp. 89–94. Teubner (1908)
13. Löwenheim, L.: Über die Auflösung von Gleichungen im logischen Gebietekalkül. Math. Ann. 68, 169–207 (1910)

14. Martin, U., Nipkow, T.: Unification in Boolean rings. In: CADE-8. LNCS (LNAI), vol. 230, pp. 506–513. Springer (1986)
15. Martin, U., Nipkow, T.: Unification in Boolean rings. J. Autom. Reasoning 4(4), 381–396 (1988)
16. Martin, U., Nipkow, T.: Boolean unification – The story so far. J. Symb. Comput. 7, 275–293 (1989)
17. Rudeanu, S.: Boolean Functions and Equations. Elsevier (1974)
18. Rudeanu, S.: Lattice Functions and Equations. Springer (2001)
19. Schröder, E.: Vorlesungen über die Algebra der Logik. Teubner (vol. 1, 1890; vol. 2, pt. 1, 1891; vol. 2, pt. 2, 1905; vol. 3, 1895)
20. Wernhard, C.: The Boolean solution problem from the perspective of predicate logic. In: FroCoS 2017. LNCS (LNAI), vol. 10483, pp. 333–350 (2017)
21. Wernhard, C.: The Boolean solution problem from the perspective of predicate logic – extended version. Tech. Rep. KRR 17–01, TU Dresden (2017), `https://arxiv.org/abs/1706.08329`

Early Steps of Second-Order Quantifier Elimination beyond the Monadic Case: The Correspondence between Heinrich Behmann and Wilhelm Ackermann 1928–1934 (Abstract)

Christoph Wernhard

TU Dresden, Germany

This presentation focuses on the span between two early seminal papers on second-order quantifier elimination on the basis of first-order logic: Heinrich Behmann's Habilitation thesis *Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem* (*Contributions to the algebra of logic, in particular to the decision problem*), published in 1922 as [4], and Wilhelm Ackermann's *Untersuchungen über das Eliminationsproblem der mathematischen Logik* (*Investigations on the elimination problem of mathematical logic*) from 1935 [2].

Behmann developed in [4] a method to decide relational monadic formulas (that is, first-order formulas with only unary predicates and no functions other than constants, also known as *Löwenheim class*) that actually proceeds by performing second-order quantifier elimination with a technique that improves Schröder's *rough-and-ready resultant* (*Resultante aus dem Rohen*) [22,10]. If all predicates are existentially quantified, then elimination yields either a truth value constant or a formula that just expresses with counting quantifiers a cardinality constraint on the domain. Although technically related to earlier works by Löwenheim [16] and Skolem [23,24], Behmann's presentation appears quite modern from the view of computational logic: He shows a method that proceeds by equivalence preserving formula rewriting until a normal form is achieved in which second-order subformulas have a certain shape for which the elimination result is known [27,26].

Ackermann laid in [2] the foundation for the two major modern paradigms of second-order quantifier elimination, the resolution-based approach [12], and the so-called *direct* or *Ackermann approach* [11,13,21], which is like Behmann's method based on formula rewriting until second-order subformulas have a certain shape for which the elimination result is known, however, now based on more powerful equivalences of second- to first-order formulas, such as *Ackermann's Lemma*. Another result of Ackermann's paper was a proof that second-order quantifier elimination on the basis of first-order logic does not succeed in general.

As documented by letters and manuscripts in Behmann's scientific bequest [6], between 1922 and 1935 Behmann and Ackermann both thought about possibilities to extend elimination to formulas with predicates of arity two or more. Behmann gave in 1926 at the *Jahresversammlung der Deutschen Mathematiker-*

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

Vereinigung a talk on *the decision problem and the logic of relations* (*Entscheidungsproblem und Logik der Beziehungen*), where he falsely claimed a positive result. Its abstract, published as [5], aroused the curiosity of Ackermann, who wrote in August 1928 to Behmann, initiating a correspondence that lasted to November 1928 and comprises five letters. Among the issues discussed were forms of what today is called Skolemization. Their correspondence concerning elimination was resumed in 1934 with a letter sent by Behmann upon receiving the offprint of [2], where he suggests a graphical presentation of the resolution-based elimination method by Ackermann [2], and Ackermann's reply, where, aside of technical issues, he gratuitously acknowledges that Behmann's work [4], at its time, was for him the impetus to investigate the elimination problem more closely. Their correspondence, as far as archived in [6], then only continues in January 1953, with five more letters until December 1955, in which different topics are discussed.

Apparently, there are very few works that are concerned with the history of second-order quantifier elimination. There is a paper by Craig [10] explicitly dedicated to that subject, with emphasis on Schröder's work, and in [19] Ackermann's results from [2] are discussed and explicitly related to modern approaches. A variant of Behmann's method from [4] is provided along with extensive historic remarks by Church [9, §49]. Further accounts of Behmann's early work with main focus on the Hilbert school and the decision problem (Behmann's talk on 10 May 1921 at the *Mathematische Gesellschaft* on the topic of [4] seems the first documented use of the term *Entscheidungsproblem* (*decision problem*) [28]) can be found in [17,28,18]. Behmann's scientific bequest [6] has been registered in [8], and before in [15]. His correspondence with Gödel has been published in [14]. A further archive source is his personal file as university professor [7], where excerpts have been published in [20]. Aside of the correspondence with Ackermann, also Behmann's correspondences with Russell, Carnap, Scholz and Church touch topics related to elimination. Letters from Ackermann's correspondences published in [1] give further hints on the "pre-history" of Ackermann's paper [2]: He sent the manuscript in 1933 to Bernays, who recommended it to Hilbert for publication and sent six large pages with remarks to Ackermann.

The historical-technical perspective on the archived correspondences and manuscripts provides interesting insight into the development of modern logic, including, in particular, computational logic. Often past technical results and methods that got out of sight turn out to be relevant for the ongoing discourse, as, for example, shown in [25,19] for results from [2], or in [27] for results from [4] and [3].

The workshop presentation is based on parts IV and V of the report [26].

Acknowledgments. This work was supported by DFG grant WE 5641/1-1.

References

1. Ackermann, H.R.: Aus dem Briefwechsel Wilhelm Ackermanns. History and Philosophy of Logic (4), 181–202 (1983)
2. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Mathematische Annalen 110, 390–413 (1935)
3. Ackermann, W.: Zum Eliminationsproblem der mathematischen Logik. Mathematische Annalen 111, 61–63 (1935)
4. Behmann, H.: Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. Mathematische Annalen 86(3–4), 163–229 (1922)
5. Behmann, H.: Entscheidungsproblem und Logik der Beziehungen. In: Jahresbericht der Deutschen Mathematiker-Vereinigung. vol. 36, no. 2. Abt. Heft 1/4, pp. 17–18 (1927)
6. Behmann, H. (bestandsbildner): Nachlass Heinrich Johann Behmann, Staatsbibliothek zu Berlin – Preußischer Kulturbesitz, Handschriftenabteilung, Nachlass 335
7. Behmann, H. (bestandsbildner): Personalakte Heinrich Behmann, Archiv der Martin-Luther-Universität Halle-Wittenberg, PA 4295
8. Bernhard, P., Thiel, C.: Der wissenschaftliche Nachlass Heinrich Behmanns: ein Verzeichnis seines Bestandes am 20.1.2000 (2000), Printed working copy with handwritten corrections and updates in Staatsbibliothek Berlin, Handschriftenabteilung. Second printed copy in [6, Kasten 1].
9. Church, A.: Introduction to Mathematical Logic, vol. I. Princeton University Press, Princeton, NJ (1956)
10. Craig, W.: Elimination problems in logic: A brief history. Synthese (164), 321–332 (2008)
11. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. Journal of Automated Reasoning 18(3), 297–338 (1997)
12. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: KR'92. pp. 425–435. Morgan Kaufmann, San Francisco, CA (1992)
13. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications, London (2008)
14. Gödel, K.: Collected Works, vol. IV: Selected Correspondence, A-G. Oxford University Press, Oxford (2003)
15. Haas, G., Stemmler, E.: Der Nachlaß Heinrich Behmanns (1891–1970). Gesamtverzeichnis. Aachener Schriften zur Wissenschaftstheorie, Logik und Logikgeschichte, RWTH Aachen (1981)
16. Löwenheim, L.: Über Möglichkeiten im Relativkalkül. Mathematische Annalen 76(4), 447–470 (1915)
17. Mancosu, P.: Between Russell and Hilbert: Behmann on the foundations of mathematics. The Bulletin of Symbolic Logic 5(3), 303–330 (1999)
18. Mancosu, P., Zach, R.: Heinrich Behmann's 1921 lecture on the algebra of logic and the decision problem. The Bulletin of Symbolic Logic 21(2), 164–187 (2015)
19. Nonnengart, A., Ohlbach, H.J., Szałas, A.: Elimination of predicate quantifiers. In: Ohlbach, H.J., Reyle, U. (eds.) Logic, Language and Reasoning, Trends in Logic, vol. 5, pp. 149–171. Springer, Berlin (1999)
20. Schenk, G., Mayer, R. (eds.): Philosophisches Denken in Halle: Personen und Texte, vol. 2: Beförderer der Logik. Schenk, Halle (Saale) (2002)
21. Schmidt, R.A.: The Ackermann approach for modal logic, correspondence theory and second-order reduction. Journal of Applied Logic 10(1), 52–74 (2012)

22. Schröder, E.: Vorlesungen über die Algebra der Logik, vol. 1. Teubner, Leipzig (1890)
23. Skolem, T.: Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme welche gewisse Klassen von Aussagen betreffen. Videnskapsselskapets Skrifter I. Mat.-Nat. Klasse(3) (1919)
24. Skolem, T.: Logisch-Kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theoreme über dichte Mengen. Videnskapsselskapets Skrifter I. Mat.-Nat. Klasse(4) (1920)
25. Szałas, A.: On the correspondence between modal and classical logic: An automated approach. Journal of Logic and Computation 3, 605–620 (1993)
26. Wernhard, C.: Heinrich Behmann's contributions to second-order quantifier elimination from the view of computational logic. Tech. Rep. KRR 15-05, TU Dresden (2015), http://cs.christophwernhard.com/papers/behmann.pdf
27. Wernhard, C.: Second-order quantifier elimination on relational monadic formulas – a basic method and some less expected applications. In: TABLEAUX 2015. LNCS (LNAI), vol. 9323, pp. 249–265. Springer, Berlin (2015)
28. Zach, R.: Completeness before Post: Bernays, Hilbert, and the development of propositional logic. The Bulletin of Symbolic Logic 5(3), 331–366 (1999)

Algorithmic Correspondence and Canonicity for Possibility Semantics (Abstract)

Zhiguang Zhao

Delft University of Technology, Netherlands

Unified Correspondence. Correspondence and completeness theory have a long history in modal logic, and they are referred to as the "three pillars of wisdom supporting the edifice of modal logic" [22, page 331] together with duality theory. Dating back to [20,21], the Sahlqvist theorem gives a syntactic definition of a class of modal formulas, the *Sahlqvist class*, each member of which defines an elementary (i.e. first-order definable) class of Kripke frames and is canonical. Since modal logic on the frame level is essentially second-order, computing the first-order correspondence of a modal formula is a kind of second-order quantifier elimination.

Recently, a uniform and modular theory which subsumes the above results and extends them to logics with a *non-classical* propositional base has emerged, and has been dubbed *unified correspondence* [5]. It is built on duality-theoretic insights [9] and uniformly exports the state-of-the-art in Sahlqvist theory from normal modal logic to a wide range of logics which include, among others, intuitionistic and distributive and general (non-distributive) lattice-based (modal) logics [6,8], non-normal (regular) modal logics based on distributive lattices of arbitrary modal signature [19], hybrid logics [12], many valued logics [16] and bi-intuitionistic and lattice-based modal mu-calculus [1,3,2]. Unified correspondence theory has two components: the first one is a very general syntactic definition of Sahlqvist and inductive formulas, which applies uniformly to each logical signature and is given purely in terms of the order-theoretic properties of the algebraic interpretations of the logical connectives; the second one is the Ackermann lemma based algorithm ALBA, which is a generalization of SQEMA based on order-theoretic and algebraic insights, which effectively computes first-order correspondents of input formulas/inequalities, and is guaranteed to succeed on the Sahlqvist and inductive classes of formulas/inequalities. The algorithm aims at eliminating all propositional variables, which are, on the relational semantics side, second-order variables, and rewrite the formula into a quasi-inequality which contains only nominals and co-nominals, which are, on the relational semantics side, essentially first-order. In this sense, unified correspondence theory is essentially second-order quantifier elimination on the algebraic side.

The breadth of this work has stimulated many and varied applications. Some are closely related to the core concerns of the theory itself, such as understanding the relationship between different methodologies for obtaining canonicity results [18,7], the phenomenon of pseudocorrespondence [10], and the investigation of

In: P. Koopmann, S. Rudolph, R. Schmidt, C. Wernhard (eds.): *SOQE 2017 – Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics, Dresden, Germany, December 6–8, 2017*, published at `http://ceur-ws.org`.

the extent to which the Sahlqvist theory of classes of normal distributive lattice expansions can be reduced to the Sahlqvist theory of normal Boolean algebra expansions, by means of Gödel-type translations [11]. Other, possibly surprising applications include the dual characterizations of classes of finite lattices [13], the identification of the syntactic shape of axioms which can be translated into structural rules of a proper display calculus [14] and of internal Gentzen calculi for the logics of strict implication [17], and the epistemic interpretation of lattice-based modal logic in terms of categorization theory in management science [4]. These and other results (cf. [9]) form the body of a theory called unified correspondence [5], a framework within which correspondence results can be formulated and proved abstracting away from specific logical signatures, using only the order-theoretic properties of the algebraic interpretations of logical connectives.

Possibility Semantics. Possibility semantics for modal logic is a generalization of standard Kripke semantics. In this semantics, a possibility frame has a refinement relation which is a partial order between states, in addition to the accessibility relation for modalities. From an algebraic perspective, full possibility frames are dually equivalent to complete Boolean algebras with complete operators which are not necessarily atomic, while filter-descriptive possibility frames are dually equivalent to Boolean algebras with operators.

In recent years, the theoretic study of possibility semantics has received more attention. In [23], Yamamoto investigates the correspondence theory in possibility semantics in a frame-theoretic way and prove a Sahlqvist-type correspondence theorem over full possibility frames, which are the possibility semantic counterpart of Kripke frames, using insights from the algebraic understanding of possibility semantics. In [15, Theorem 7.20], it is shown that all inductive formulas are filter-canonical and hence every normal modal logic axiomatized by inductive formulas is sound and complete with respect to its canonical full possibility frame. However, the correspondence result for inductive formulas is still missing, as well as the correspondence result over filter-descriptive possibility frames (see [15, page 103]) and soundness and completeness with respect to the corresponding elementary class of full possibility frames. The present paper aims at giving a closer look at the aforementioned unsolved problems using the algebraic and order-theoretic insights from a current ongoing research project, namely *unified correspondence*.

Methodology. Our contribution is methodological: we analyze the correspondence phenomenon in possibility semantics using the dual algebraic structures, namely complete (not necessarily atomic) Boolean algebras with complete operators, where the atoms are not always available. For the correspondence over full possibility frames, our strategy is to identify two different Boolean algebras with operators as the dual algebraic structures of the possibility frame, namely the Boolean algebra of regular open subsets $\mathbb{B}_{\mathsf{RO}}$ (when viewing the possibility frame as a possibility frame itself) and the Boolean algebra of arbitrary subsets

$\mathbb{B}_{\mathsf{Full}}$ (when viewing the possibility frame as a bimodal Kripke frame), where a canonical order-embedding map $e : \mathbb{B}_{\mathsf{RO}} \to \mathbb{B}_{\mathsf{Full}}$ can be defined. The embedding e preserves arbitrary meets, therefore a left adjoint $c : \mathbb{B}_{\mathsf{Full}} \to \mathbb{B}_{\mathsf{RO}}$ of e can be defined, which sends a subset X of the domain W of possibilities to the smallest regular open subset containing X. This left adjoint c plays an important role in the dual characterization of the interpretations of the expanded language, which form the ground of the regular open translation, i.e. the counterpart of standard translation in possibility semantics. When it comes to canonicity, we use the fact that filter-canonicity is equivalent to constructive canonicity [15, Theorem 5.46, 7.20], and prove a topological Ackermann lemma, which justifies the soundness of propositional variable elimination rules and forms the basis of the correspondence result with respect to the class of filter-descriptive frames as well as the canonicity and completeness result with respect to the corresponding class of full possibility frames.

References

1. W. Conradie and A. Craig. Canonicity results for mu-calculi: an algorithmic approach. *Journal of Logic and Computation*, Forthcoming. ArXiv preprint arXiv:1408.6367.
2. W. Conradie, A. Craig, A. Palmigiano, and Z. Zhao. Constructive canonicity for lattice-based fixed point logics. Submitted. ArXiv preprint arXiv:1603.06547.
3. W. Conradie, Y. Fomatati, A. Palmigiano, and S. Sourabh. Algorithmic correspondence for intuitionistic modal mu-calculus. *Theoretical Computer Science*, 564:30–62, 2015.
4. W. Conradie, S. Frittella, A. Palmigiano, M. Piazzai, A. Tzimoulis, and N. Wijnberg. Categories: How I learned to stop worrying and love two sorts. *Proceedings of WoLLIC 2016*, ArXiv preprint 1604.00777.
5. W. Conradie, S. Ghilardi, and A. Palmigiano. Unified correspondence. In A. Baltag and S. Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, volume 5 of *Outstanding Contributions to Logic*, pages 933–975. Springer International Publishing, 2014.
6. W. Conradie and A. Palmigiano. Algorithmic correspondence and canonicity for distributive modal logic. *Annals of Pure and Applied Logic*, 163(3):338 – 376, 2012.
7. W. Conradie and A. Palmigiano. Constructive canonicity of inductive inequalities. Submitted. ArXiv preprint 1603.08341.
8. W. Conradie and A. Palmigiano. Algorithmic correspondence and canonicity for non-distributive logics. Submitted. ArXiv preprint 1603.08515.
9. W. Conradie, A. Palmigiano, and S. Sourabh. Algebraic modal correspondence: Sahlqvist and beyond. Submitted. ArXiv preprint 1606.06881.
10. W. Conradie, A. Palmigiano, S. Sourabh, and Z. Zhao. Canonicity and relativized canonicity via pseudo-correspondence: an application of ALBA. Submitted. ArXiv preprint 1511.04271.
11. W. Conradie, A. Palmigiano, and Z. Zhao. Sahlqvist via translation. Submitted. ArXiv preprint 1603.08220.
12. W. Conradie and C. Robinson. On Sahlqvist theory for hybrid logic. *Journal of Logic and Computation*, 2015. doi: 10.1093/logcom/exv045.

13. S. Frittella, A. Palmigiano, and L. Santocanale. Dual characterizations for finite lattices via correspondence theory for monotone modal logic. *Journal of Logic and Computation*, 2016. doi:10.1093/logcom/exw011.
14. G. Greco, M. Ma, A. Palmigiano, A. Tzimoulis, and Z. Zhao. Unified correspondence as a proof-theoretic tool. *Journal of Logic and Computation*, 2016. doi: 10.1093/logcom/exw022. ArXiv preprint 1603.08204.
15. W. Holliday. Possibility frames and forcing for modal logic. *UC Berkeley Working Paper in Logic and the Methodology of Science*, June 2016. URL http://escholarship.org/uc/item/9v11r0dq.
16. C. le Roux. Correspondence theory in many-valued modal logics. Master's thesis, University of Johannesburg, South Africa, 2016.
17. M. Ma and Z. Zhao. Unified correspondence and proof theory for strict implication. *Journal of Logic and Computation*, 2016. doi: 10.1093/logcom/exw012. ArXiv preprint 1604.08822.
18. A. Palmigiano, S. Sourabh, and Z. Zhao. Jónsson-style canonicity for ALBA-inequalities. *Journal of Logic and Computation*, 2015. doi:10.1093/logcom/exv041.
19. A. Palmigiano, S. Sourabh, and Z. Zhao. Sahlqvist theory for impossible worlds. *Journal of Logic and Computation*, 2016. doi:10.1093/logcom/exw014.
20. H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logic. In *Studies in Logic and the Foundations of Mathematics*, volume 82, pages 110–143. 1975.
21. J. van Benthem. *Modal logic and classical logic*. Bibliopolis, 1983.
22. J. van Benthem. Correspondence theory. In D. M. Gabbay and F. Guenthner, editors, *Handbook of philosophical logic*, volume 3, pages 325–408. Kluwer Academic Publishers, 2001.
23. K. Yamamoto. Modal correspondence theory for possibility semantics. *UC Berkeley Working Paper in Logic and the Methodology of Science*, 2016. URL http://escholarship.org/uc/item/7t12914n.