nanoCoP-Ω: A Non-Clausal Connection Prover with Arithmetic

Talk Abstract

Leo Repp[1], Mario Frank[2]

[1]TUD Dresden University of Technology, Andreas-Pfitzmann-Bau, Nöthnitzer Str. 46, 01187 Dresden, Saxony, Germany
[2]University of Potsdam, Institute of Computer Science, An der Bahn 2, 14476 Potsdam, Brandenburg, Germany

Abstract
Adding arithmetic and equalities to automated theorem provers is a complex process. This work presents nanoCoP-Ω, a new lean non-clausal connection calculus automated theorem prover for first-order logic with arithmetic and equality. It uses the Omega Library, which is based on Presburger arithmetic.

Keywords
non-clausal connection calculus, arithmetic, equality, presburger, omega test, nanoCoP, leanCoP

1. Prerequisites

The Connection Calculus was first described in [1, 2]. The automated proof systems leanCoP and nanoCoP are based on more recent descriptions from [3] and [4] respectively. These articles describe a complete and sound proof system for first order formulas without arithmetic and equalities that works roughly as follows.

A proof by contradiction is done by transforming the formula into a matrix. The structure of the matrix describes how the literals contained within the formula can be used in the proof of the said formula, i.e., it defines which literals can be used in reasoning together and which literals belong to different, orthogonal case distinctions of the proof.

An algorithm then searches for "connections" within the matrix: Complementary literals from the original formula that yield contradictions. Once a connection has been found for every possible case distinction of the proof, a proof by contradiction of the formula is complete.

Clausal versions of the connection calculus work on clausal form matrices. The conversion into clausal form provides a minor computational overhead and makes the structure of the original formula not recognizable when looking at the matrix. Both leanCoP and nanoCoP can generate a human-readable proof as justification. But as leanCoP operates on clausal form matrices its proofs tend to be less readable than nanoCoP's.

The Omega Library is a library developed by the Omega Project Team [5]. It utilizes the Omega Test [6] to test for satisfiability of equalities and inequalities connected by disjunctions

AReCCa'23: International Workshop on Automated Reasoning with Connection Calculi, September 18, 2023, Prague, Czech Republic
leo.repp.0410@gmail.com (L. Repp); mario.frank@uni-potsdam.de (M. Frank)
0000-0003-1021-6959 (L. Repp); 0000-0001-8888-7475 (M. Frank)

and conjunctions. As the Omega Test is based on Presburger arithmetic [7], the library is limited in the same way: Reasoning is limited to linear polynomials. For example, division and prime numbers cannot be defined since the arithmetic is limited to general addition and constant multiplication of integer values. This limited reasoning capacity allows for a completeness and decidability, which are desirable for automated proof systems. The Omega Library is utilized by leanCoP-Ω, first presented in [8], and nanoCoP-Ω, the result of this work.

Until now the connection calculus can only handle equalities uninterpreted, i.e. an equality containing arithmetic expressions cannot be processed. leanCoP-Ω introduced new methods for handling arithmetic expressions and (in-) equalities in connection calculi. Literals containing equalities can be proven by several methods: Firstly, with connections, just like regular literals. Secondly, by being trivial (i.e. solvable by Prolog) and thirdly, by constructing an equation system that is tested for validity by the Omega Library. Lastly it may be necessary to collect more literals containing equalities and then check the larger system for validity. These methods needed to be added to nanoCoP to create nanoCoP-Ω.

2. The nanoCoP-Ω Implementation

nanoCoP-Ω is an extension of nanoCoP by arithmetic and equality handling procedures. As such, it is a Prolog program that transforms the formula into a non-clausal matrix that stores variable dependencies, the number of submatrix instantiations and clause identifiers. Prolog's integrated search algorithm with backtracking and Prolog's logic programming paradigm then allows stating the proof rules of the non-clausal connection calculus in their desired order to construct a proof system.

The computation of the sub-matrices needed after each proof step is a complex process based on the matrix' shape. Prolog's cut allows for the introduction of restricted backtracking. Expressions implementing iterative deepening are added to ensure completeness. The equality axioms are explicitly added to the formula. Unification with occurs check is done by using an internal Prolog predicate and the skolem form representation of the variables. A fixed strategy scheduling is used to iteratively invoke the proof search with different strategies.

The arithmetic and (in-) equality handling methods of leanCoP-Ω were added to nanoCoP with several adjustments to the different matrix representations required for non-clausal form matrices. The interface used to invoke the Omega Library from nanoCoP's Prolog code was adapted from leanCoP-Ω[9], which uses a shared object and Prolog's foreign language interface. A transformation due to different matrix representation was necessary again.

3. Evaluation

We compared the clausal leanCoP-Ω (as provided in [9]) to the non-clausal nanoCoP-Ω. The comparison was done with 1007 non-polymorphic typed first-order logic theorems from the TPTP library v8.0.0 [10]. The problems stem from multiple domains.

leanCoP-Ω has a success rate of 30.5% and nanoCoP-Ω has one of 25.5%. leanCoP-Ω solves 82 problems that nanoCoP-Ω does not solve and nanoCoP-Ω solves 32 problems that leanCoP-Ω does not. Both solvers performed well on problems from the ARI and NUM domains and leanCoP-Ω performs better than nanoCoP-Ω for problems from the SWW domain. nanoCoP-Ω is faster than leanCoP-Ω in all but 13 of 255 problems both systems solve, with an overall average speedup of approx. 2.3 and a larger speedup for small formulas.

We thus conclude that nanoCoP-Ω is a capable system that is especially fast for small problems.

4. Conclusion

We developed a proof system capable of handling first order logic with Presburger arithmetic. It is comparable in the number of solved problems to its predecessor leanCoP-Ω[8], while being considerably faster for many problems. As a non-clausal prover it provides more easily readable proofs.

There is potential for future work: The combination of leanCoP-Ω and nanoCoP-Ω could yield a more powerful prover and further optimizations within nanoCoP-Ω are possible. Furthermore, the interface provided in [9] together with transformation operations provided in nanoCoP-Ω can be a basis for extension by other arithmetic decision procedures.

References

[1] W. Bibel, Automated Theorem Proving, Vieweg, Braunschweig [u.a.], 1982. doi:10.1007/978-3-322-90100-2.

[2] W. Bibel, Matings in Matrices, Commun. ACM 26 (1983) 844—-852. doi:10.1145/182.183.

[3] J. Otten, W. Bibel, leanCoP: Lean Connection-Based Theorem Proving, J. Symb. Comput. 36 (2003) 139–161. doi:10.1016/S0747-7171(03)00037-3.

[4] J. Otten, A Non-clausal Connection Calculus, in: K. Brünnler, G. Metcalfe (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 226–241. doi:10.1007/978-3-642-22119-4_18.

[5] William Pugh and the entire Omega Project Team, The Omega Project: Frameworks and Algorithms for the Analysis and Transformation of Scientific Programs, 2010. URL: https://www.cs.umd.edu/projects/omega/.

[6] W. Pugh, The Omega Test: A Fast and Practical Integer Programming Algorithm for Dependence Analysis, in: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing, Supercomputing '91, Association for Computing Machinery, New York, NY, USA, 1991, p. 4–13. doi:10.1145/125826.125848.

[7] M. Presburger, Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt, in: Comptes-Rendus du ler Congrès des Mathématiciens des Pays Slaves, Warsaw, Poland, 1929, p. 92–101.

[8] J. Otten, H. Trölenberg, T. Raths, leanCoP-Ω, 2010. URL: https://www.tptp.org/CASC/J5/leanCoP-Omega---0.1.pdf.

[9] K. Münch, Entwicklung einer performanten und verlässlichen Schnittstelle für leanCoP-Ω, Master's thesis, Institute for Computer Science, University of Potsdam, Potsdam, Germany, 2021.

[10] G. Sutcliffe, The TPTP Problem Library and Associated Infrastructure, J. Autom. Reason. 59 (2017) 483–502. doi:10.1007/s10817-017-9407-7.