PROTEIN: A *PRO*ver with a *T*heory *E*xtension *IN*terface*

Peter Baumgartner and Ulrich Furbach

Universiät Koblenz
Institut für Informatik
Rheinau 1
56075 Koblenz, Germany

Tel.: +49–261–9119–426, +49–261–9119–433
E-mail: {peter,uli}@informatik.uni-koblenz.de

Abstract. PROTEIN (*PRO*ver with a *T*heory *E*xtension *IN*terface) is a PTTP-based first order theorem prover over built-in theories. Besides various standard-refinements known for model elimination, PROTEIN also offers a variant of model elimination for case-based reasoning and which does not need contrapositives.

PROTEIN is a complete theorem prover for first order clause logic. It is characterized by the following features:

- PROTEIN is based on the *PTTP implementation technique* [Sti88] for model elimination [Lov69].
- PROTEIN offers alternative inference rules for *case analysis* [Lov87, BF93]. In this setting no contrapositives are needed, and hence the system is well suited as an interpreter for disjunctive logic programming.
- PROTEIN includes *theory reasoning* [Sti85, Bau92, Bau94] in a very general way. An auxiliary program can be used to derive a suitable background reasoner from a given Horn theory in a fully automatic way.
- PROTEIN includes several *calculus refinements and flags*.

The idea of the *PTTP implementation technique* ("Prolog Technology Theorem Prover") [Sti88] is to view Prolog as an "almost complete" theorem prover, which has to be extended by only a few ingredients in order to handle the non-Horn case. By this technique the WAM-technology and other benefits of optimizing Prolog compilers are accessible to theorem proving. A disadvantage of PTTP, according to Stickel ([Sti90]), is that "the high inference rate can be overwhelmed by its exponential search space" and therefore PTTP might be well suited for easy problems whereas "it is unsuitable for many difficult theorems for which more conventional theorem provers have demonstrated success".

Our system proves that PTTP can be used even for many challenging problems from the theorem proving literature if PTTP is understood only as a kernel system, which has to be augmented by additional features, like theory handling or case analysis. This is done in PROTEIN.

* This research was sponsored by DFG within the "Schwerpunktprogramm *Deduktion*".

The *case-analysis* style of reasoning came up with various (non-theory) calculi which do not need all contrapositives ([Lov87, Pla88]). A detailed comparison of those calculi can be found in [RL92].

In [BF93] we have made a small change to model elimination which also avoids contrapositives and has some distinguished features. This modification of model elimination is called restart model elimination; its distinguished feature are, first, that it can be very easily implemented within a PTTP-framework, and, second, the better informed search due to additional ancestor literals.

Theory reasoning was introduced by M. Stickel within the resolution calculus[Sti85]; for model elimination it is defined an investigated in [Bau92].

Technically, *theory reasoning* means to relieve a calculus from explicit reasoning in some domain (e.g. equality, partial orders, taxonomic reasoning) by taking apart the domain knowledge and treating it by special inference rules. In an implementation, this results in a universal "foreground" reasoner that calls a specialized "background" reasoner for theory reasoning. See [BFP92] for an overview.

Fortunately, the calculus' features "case-analysis reasoning" and "theory reasoning" are fairly compatible with model elimination [Bau94]. In [BF93, BF94] we have shown that "case analysis" – in the non-theory setting – requires only a small change to the calculus. PROTEIN is the respective implementation for theory reasoning.

Furthermore PROTEIN includes several *calculus refinements and flags* such as unit-lemmas, factorisation, ground reduction steps (reduction steps not affecting variables can be handled irrevocably in the proof search) and regularity[2].

Theory Reasoning with Completed Theories

Theory reasoning comes in two variants: *total* and *partial* theory reasoning. *Total* theory reasoning generalizes the idea of finding complementary literals in inferences (e.g. resolution) to a semantic level. For example, in theory resolution the foreground reasoner may select from some clauses the literal set $\{a < b, b < c, c < a\}$, pass it to the background reasoner[3] which in turn should discover that this set is contradictory.

The problem with total theory reasoning is that in general it is undecidable which literals constitute a contradictory set. As a solution, partial theory reasoning tries to break the "big" total steps into more manageable, *decidable*, smaller steps. In the example, the background reasoner might be passed $\{a < b, b < c\}$, compute the new subgoal $a < c$ and return it to the foreground reasoner. In the next step, the foreground reasoner might call the background reasoner with $\{a < c, c < a\}$ again, which detects a trivial contradiction and thus concludes this chain.

PROTEIN currently offers the following: for *total* theory reasoning the input clauses may contain a call to a Prolog-predicate. The theory then is implicitly defined via the enumeration of all answer substitutions to the Prolog-predicate. For the sake of

[2] Regularity means that proof attempts which repeat the same subgoal along a branch can be discarded; the regularity restriction has to be relaxed for the case-analysis variant.

[3] assume that $<$ is interpreted as a strict ordering

classification we have here a total theory-extension step with a theory-complementary set consisting of exactly one literal. As a possible application we think of *reasoning by examples* [KMS93].

The implementation of *partial* theory reasoning is currently tailored for the method of *linearizing completion* [Bau93]. Linearizing completion is a saturation technique that transforms a given Horn clause set $\mathcal{T}$ into a "completed" set, which admits (in resolution terminology) both linear and unit-resulting proofs for $\mathcal{T}$-unsatisfiable literal sets. Such a system then can be used as complete background reasoner for partial theory model elimination.

Implementation and Practical Experiments

Both PROTEIN and a tool for linearizing completion are implemented in ECLiPSe, ECRC's Prolog dialect. ECLiPSe extends Prolog by various features, with the most relevant ones for us being *sound unification* and *delayed subgoals*. While the use of sound unification is obvious, the delayed subgoal mechanism is used to implement the regularity restriction. Regularity is realized by delaying a set of constraints, where each constraint states the syntactical inequality of two path literals. A delayed constraint is checked (waked up) each time a change to one of its variable occurs.

We ran several examples known from the literature with PROTEIN and a high-performance model elimination prover. Table 1 contains the runtime results (in seconds), obtained on a SPARC 10/40. The first four columns refer to different versions of PROTEIN. Column 5 contains data for Setheo [LSBB92] in its latest version (Version 3.0).

PROTEIN was run in default mode, except where indicated in Table 1. In default mode it includes the regularity restriction and the ground-reduction refinement. Setheo was also run in its default mode, which then makes use of the following refinements and constraints: subgoal reordering, purity, anti-lemmas, regularity, tautology and subsumption.

The example referred to as *Non-obvious* is taken from the October 1986 Newsletter of the *Association of Automated Reasoning*[4]. The selected theory here consists of a transitive and symmetric relation p and a transitive relation q. In the *Graph* example a graph with a transitive and symmetric reachability relation is traversed. The *Eder*-examples are described in [LSBB92]. The *Bledsoe* examples are the first two of the five given in [Ble90]. The example referred to by $x \neq 0 \rightarrow x^2 > 0$ is to prove this theorem (x is universally quantified) from calculus. Case analysis is carried out according to the axiom $X > 0 \quad \vee \quad X = 0 \quad \vee \quad -X > 0$.

For the theory variants of PROTEIN, the background calculus was obtained completely automatical by the linearizing completion tool in a preprocessing phase. For the theories we have selected appropriate Horn-subsets of the input clauses. The runtime of the linearizing completion tool was sufficiently small and need not be mentioned. In case linearizing completion would yield an infinite inference system for background

[4] Entries such as `MSC/MSC006-1` refer to the respective TPTP-names [SSY94]. All examples were drawn from that problem library without modification — only the theory part had to be selected by hand.

Example	ME	Restart-ME	TME	Restart-TME	Setheo
Non-obvious MSC/MSC006-1	0.3	2.7	1.6	1.1 (0.15 [1])	0.5
Eder45	0.7	3.4 (1.8 [1])	-	-	1.0
Eder58	22.0	110	-	-	32
Graph	10.8	∞	0.2	7.0 (0.8 [2])	3.0
$x \neq 0 \rightarrow x^2 > 0$	2.4	0.7	2.2	0.6	0.8
Pelletier 48 SYN/SYN071-1	5.9	1.2 (0.6 [2])	0.4	0.9 (0.1 [1,2])	0.2
Pelletier 49 SYN/SYN072-1	∞	297	1.6	1.5	∞
Pelletier 55 PUZ/PUZ001-2	392	∞	21	254	3.5
Lion&Unicorn PUZ/PUZ005-1	588	∞	21	∞	47
Bledsoe 1 ANA/ANA003-4	∞	7.4 [1,3]	-	-	87
Bledsoe 2 ANA/ANA004-4	∞	32 [1,3]	-	-	∞
Wos 4 GRP/GRP008-1	22	20	0.3	26	13
Wos 10 GRP/GRP001-1	∞	-	14	-	850
Wos 11 GRP/GRP013-1	9.6	-	1.1	-	0.7
Wos 15 GRP/GRP035-3	384	-	47	-	478
Wos 16 GRP/GRP036-3	302	-	0.02	-	13
Wos 17 GRP/GRP037-3	∞	-	0.1	-	23

Entries: Numbers: runtimes (seconds) – ∞ no proof within reasonable time bound – "-" Not applicable –
Remarks: 1 – With selection function, 2 – With (back) factoring, 3 – With Lemmas

Fig. 1. Runtime Results for various provers: *ME* – plain model elimination version of PROTEIN; *Restart-ME* – case-analysis style reasoning; *TME* and *Restart-TME* – respective versions with theory reasoning extensions.

reasoning – in the Wos examples from group theory – a finite approximation was used. The selected theory consists here of equality (except f-substituivity) and the associativity of the group operation.

Concerning the search strategy we used iterative deepening with the costs of extension steps uniformly set to 1. The same costs are used for case analysis steps.

The runtime results suggest to us that indeed both variants — restart vs. non-restart – are valuable: every variant obtains proof not obtainable to the other variant. Furthermore, the application of theory reasoning very often helps to find a proof more quickly. If no suitable theory is at hand, the "empty" theory can be used which instantiates TME (resp.

Restart-TME) to ME (resp. Restart-ME).

In order to obtain the PROTEIN system please contact
`peter@informatik.uni-koblenz.de`.

References

[Bau92] P. Baumgartner. A Model Elimination Calculus with Built-in Theories. In H.-J. Ohlbach, editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 30–42. Springer, 1992. LNAI 671.

[Bau93] P. Baumgartner. Linear Completion: Combining the Linear and the Unit-Resulting Restrictions. Research Report 9/93, University of Koblenz, 1993.

[Bau94] P. Baumgartner. Refinements of Theory Model Elimination and a Variant without Contrapositives. In *Proc. ECAI'94*, 1994. (to appear).

[BF93] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives and its Application to PTTP. Fachbericht Informatik 12/93, Universität Koblenz, 1993. (short version in Proc. CADE-12).

[BF94] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives. In *Proc. 12th International Conference on Automated Deduction*. Springer, 1994. (in this volume).

[BFP92] P. Baumgartner, U. Furbach, and U. Petermann. A Unified Approach to Theory Reasoning. Forschungsbericht 15/92, University of Koblenz, 1992. (submitted to Journal of Automated Reasoning).

[Ble90] W. W. Bledsoe. Challenge Problems in Elementary Calculus. *Journal of Automated Reasoning*, 6:341–359, 1990.

[KMS93] Manfred Kerber, Erica Melis, and Jörg Siekmann. Reasoning with Assertions and Examples. Technical Report SEKI Report SR-93-10, Universität des Saarlandes, Fachbereich Informatik, 1993.

[Lov69] D. Loveland. A Simplified Version for the Model Elimination Theorem Proving Procedure. *JACM*, 16(3), 1969.

[Lov87] D.W. Loveland. Near-Horn Prolog. In J.-L. Lassez, editor, *Proc. of the 4th Int. Conf. on Logic Programming*, pages 456–469. The MIT Press, 1987.

[LSBB92] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performace Theorem Prover. *Journal of Automated Reasoning*, 8(2), 1992.

[Pla88] D. Plaisted. Non-Horn Clause Logic Programming Without Contrapositives. *Journal of Automated Reasoning*, 4:287–325, 1988.

[RL92] D. W. Reed and D. W. Loveland. A Comparison of Three Prolog Extensions. *Journal of Logic Programming*, 12:25–50, 1992.

[SSY94] G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In *Proc. CADE-12*. Springer, 1994.

[Sti85] M.E. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[Sti88] M. Stickel. A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler. *Journal of Automated Reasoning*, 4:353–380, 1988.

[Sti90] M. Stickel. A Prolog Technology Theorem Prover. In M.E. Stickel, editor, *Proc CADE 10, LNCS 449*, pages 673–675. Springer, 1990.