# From Axioms to Rewriting Rules

Guillaume Burel[1,3] and Guillaume Rousseau[2,3]

[1] ENSIIE/Laboratoire Samovar, Télécom SudParis and CNRS
guillaume.burel@ensiie.fr
[2] ENS de Lyon guillaume.rousseau@ens-lyon.fr
[3] Université Paris-Saclay, ENS Paris-Saclay, CNRS, Inria, Laboratoire Spécification et Vérification, 94235, Cachan, France

**Abstract.** Deduction Modulo Theory is a generic framework to describe proofs in a theory better than using raw axioms. This is done by presenting the theory through a congruence over propositions that is most often defined by means of rules rewriting terms and propositions. It has been shown that such representations of theories preserve good properties of axiom-free deductive systems, that they can lead to theoretical proof-length speed-ups, and that they actually improve automated proof search. In this paper, we are interested in transforming an axiomatic theory into a rewriting system so that it can be used in Deduction Modulo Theory. We design several techniques to automatically orient axioms into rewriting systems, some of them being complete, some being merely heuristics. Using automated theorem provers featuring Deduction Modulo Theory, namely iProverModulo, Zipperposition and ArchSAT, we perform experiments to compare these techniques. These experiments confirm the practical interest of using rewriting rules instead of axioms.

**Keywords:** automated deduction, proof theory, theory reasoning, rewriting, refinements of resolution

## 1 Introduction

Proofs are rarely built without context: mathematical theorems are proved for instance in set theory, or in arithmetic; program correctness may use pointer arithmetic or the theories associated to the data structures of the program (chained lists, arrays, etc.); theories can also model characteristics of encryption functions to prove security properties. Some proving contexts can also be seen as theories. For instance, the Sledgehammer tactic of Isabelle sends to automated provers the goal to be proved with a number of related lemmas that can be used as axioms, and that form therefore a specific theory in which the goal must be proved. Therefore, it is essential to develop methods that are adapted to search for proofs in theories. For instance, SMT provers provide efficient tools. Nevertheless, they are restricted to some particular theories, such as linear arithmetic or arrays. We would like to have a generic and automated way of obtaining efficient methods for a given theory, provided it is consistent. A naive idea is to use an axiomatic presentation of the theory, but it is now folklore that this is not efficient enough. The theory should therefore be presented in a more effective manner. One solution is, starting from the axiomatic presentation, to automatically design a deductive system that is adapted to the theory. Negri and von Plato [30] turn variable-free axioms into non-logical deduction rules that are added to a sequent calculus. Similarly, Ciabattoni, Galatos, and Terui [15] transforms a large class of axioms into inference rules in sequent and hypersequent calculi. Deduction Modulo Theory, introduced by Dowek, Hardin, and Kirchner [23], is a bit different: it presents the theory as computation, by means of a rewriting system, and the inference rules of an existing deductive system (natural deduction, sequent calculus, etc.) are applied modulo the congruence associated with this rewriting system. Deduction Modulo Theory can theoretically lead to unbounded proof-length speed-ups [7], and we have shown [8] that presenting theories as rewriting systems improves indeed the search for proofs in those theories. Deduction Modulo Theory is featured in at least four automated theorem provers : `iProverModulo` [8, 10] (https://github.com/gburel/iProverModulo), `Zenon Modulo` [17, 10] (https://github.com/Deducteam/zenon_modulo), `Zipperposition` (https://github.com/sneeuwballen/zipperposition) and `ArchSAT` (https://github.com/Gbury/archsat).

If one wants these presentations to behave well, they should have the following proof-theoretical property: the cut rule must be admissible. Indeed, in the usual setting, cut admissibility implies the consistency of the theory, the subformula property (to find a proof, one can restrict oneself to the subformulas of the formula to be proved), the existence of proof normal forms, etc. Furthermore, in Deduction Modulo Theory, this property is equivalent to the completeness of the various derived proof-search methods [23, 4, 21, 5]. For all systems produced by Negri and von Plato [30] and Ciabattoni et al. [15], because of restrictions on the form of the theories, cut admissibility holds. However, in Deduction Modulo Theory, it depends on the considered rewriting system. We therefore would like, given a consistent theory, a way to present it as a rewriting system such that cut admissibility holds in Deduction Modulo Theory, and we would like this process to be automated. This question has been tackled by various works. A presentation as a rewriting system with cut admissibility was designed specially for particular theories, such as Peano arithmetic by Dowek and Werner [25], simple type theory by Dowek, Hardin, and Kirchner [22], and Zermelo set theory by Dowek and Miquel [24]. When we experimented with our integration of a proof search method based on Deduction Modulo Theory into an existing prover [8], we had to design such a rewriting system by hand for each theory we considered, which led us to restrict ourselves to only five theories. Dowek [20] designed a systematic way of transforming a consistent *propositional* theory into such a rewriting system, using a model of the theory. Together with Kirchner [2010], we gave a semi-algorithm that can handle any first-order theory: first, it produces a rewriting system that corresponds to the theory; second, it completes the rewriting system to ensure cut admissibility. It is the second part that may not terminate. In [6], we show for how any first-order theory can always be presented as a rewriting system with cut admissibility. This was done by developing a characterization by Burel and Dowek [12] of an extension of the resolution method based on Deduction Modulo Theory as a combination of the set-of-support strategy of Wos, Robinson, and Carson [34] and of selection of literals. In [9], we propose another approach by first saturating the set of axioms w.r.t. the superposition calculus, and then obtaining a rewriting system from the saturated set that is guaranteed to ensure cut admissibility.

Although these two methods are perfectly valid from a theoretical point of view, they suffer from drawbacks that hinder their use in practice. The first one produces many rewriting rules, so that its benefit w.r.t. using axioms could be not so significant. The second one cannot always be employed, since the saturation of the set of axioms may not terminate. We therefore designed several methods to orient the axioms; some of them are based on these theoretical works, some others are base on heuristics. We implemented these methods in a tool called `autotheo`. We have performed an experiment to compare them, using three provers implementing a resolution method based on Deduction Modulo Theory : `iProverModulo`, a prover combining resolution and instantiation-generation; `Zipperposition`, a prover implementing the superposition calculus; and `ArchSAT`, a SMT solver based on McSat. We could not used `Zenon Modulo` because it does not handle *polarized* Deduction Modulo Theory, which is the flavour of Deduction Modulo Theory that is output by `autotheo`.

In the two next sections, we briefly present Deduction Modulo Theory and refinements of resolution. Section 4 describes how a theory can be presented as a rewriting system, and why cut admissibility is implied by the consistency of the theory, or by the saturation of the set of axioms. We then describe in Section 5 the different practical algorithms that can be used to perform this task, and we perform an experiment to compare them. We conclude by discussing further works.

## 2 Deduction Modulo Theory

We use standard definitions for terms, predicates, propositions (with connectives $\neg, \Rightarrow, \wedge, \vee$ and quantifiers $\forall, \exists$), sequents, substitutions, term rewriting rules and term rewriting, as can be found in [1, 26] . The substitution of a variable $x$ by a term $t$ in a term or a proposition $A$ is denoted by $\{t/x\}A$, and more generally the application of a substitution $\sigma$ in a term or a proposition $A$ by $\sigma A$. A term $t$ can be narrowed into $s$ using substitution $\sigma$ at position $\mathfrak{p}$ ($t \overset{\mathfrak{p},\sigma}{\rightsquigarrow} s$) if $\sigma t$ can be rewritten into $s$ using substitution $\sigma$ at position $\mathfrak{p}$. A literal is an atomic proposition or the negation of an atomic proposition. A proposition is in clausal form if it is the universal quantification of a disjunction of literals $\forall x_1, \dots, x_n.\ L_1 \vee \dots \vee L_p$ where $x_1, \dots, x_n$ are the free variables of $L_1, \dots, L_p$. In the following, we will often omit the quantifications, and we will identify propositions in clausal form with clauses (i.e. set of literals) as if $\vee$ were associative, commutative and idempotent. The symbol $\Box$ represents the empty clause. The polarity of a position in a proposition can be defined as follows: the root is positive, and the polarity switches when going under a $\neg$ or on the left of a $\Rightarrow$.

In Deduction Modulo Theory, term rewriting and narrowing is extended to propositions by congruence on the proposition structure. In addition, there are also proposition rewriting rules whose left-hand side is an atomic proposition and whose right-hand side can be any proposition. Such rules can also be applied to non-atomic propositions by congruence on the proposition structure. We call a rewriting system the combination of a term rewriting system and a proposition rewriting system. Given a rewriting system $\mathcal{R}$, we denote by $A \underset{\mathcal{R}}{\longrightarrow} B$ the fact that $A$ is rewritten in one step into $B$, either by a term rewriting rule or by a proposition rewriting rule, and by $A \underset{\mathcal{R}}{\leadsto} B$ the fact that $A$ is narrowed to $B$. $\overset{*}{\underset{\mathcal{R}}{\longrightarrow}}$ is the reflexive transitive closure of $\underset{\mathcal{R}}{\longrightarrow}$. Deduction Modulo Theory consists in applying the inference rules of an existing proof system modulo such a rewriting system. This leads for instance to the asymmetric sequent calculus modulo of Dowek [19], some of whose rules are presented in Figure 1.

*Example 1.* Consider the rewriting rule $A \subseteq B \to \forall x.\ x \in A \Rightarrow x \in B$. We can build the following proof of the transitivity of the inclusion in the asymmetric sequent calculus modulo this rule:

$$\vdash\forall\ \dfrac{\vdash\Rightarrow\ \dfrac{\forall\vdash\ \dfrac{\Rightarrow\vdash\ \dfrac{\forall\vdash\ \dfrac{\Rightarrow\vdash\ \dfrac{\overset{\frown}{\vdash}\ \dfrac{}{x \in C \vdash x \in C} \qquad \overset{\frown}{\vdash}\ \dfrac{}{x \in B \vdash x \in B}}{x \in B \Rightarrow x \in C, x \in B \vdash x \in C}}{B \subseteq C, x \in B \vdash x \in C} \qquad \overset{\frown}{\vdash}\ \dfrac{}{x \in A \vdash x \in A}}{x \in A \Rightarrow x \in B, B \subseteq C, x \in A \vdash x \in C}}{A \subseteq B, B \subseteq C, x \in A \vdash x \in C}}{A \subseteq B, B \subseteq C \vdash x \in A \Rightarrow x \in C}}{A \subseteq B, B \subseteq C \vdash A \subseteq C}$$

Rewriting rules can be applied indifferently to the left- or the right-hand side of a sequent. Consequently, they can be considered semantically as an equivalence between their left- and right-hand sides. To be able to consider implications, a polarized version of Deduction Modulo Theory was introduced by Dowek [18]. Proposition rewriting rules are tagged with a polarity $+$ or $-$; they are then called polarized rewriting rules. A proposition $A$ is rewritten positively into a proposition $B$ ($A \longrightarrow^{+} B$) if it is rewritten by a positive rule at a positive position or by a negative rule at a negative position. It is rewritten negatively ($A \longrightarrow^{-} B$) if it is rewritten by a positive rule at a negative position or by a negative rule at a positive position. Intuitively, a positive rule $A \to^{+} B$ (resp. a negative rule $B \to^{-} A$) corresponds to an implication $B \Rightarrow A$. Term rewriting rules (but not proposition rewriting rules) are considered as both positive and negative. $\overset{*}{\longrightarrow}{}^{\pm}$ is the reflexive transitive closure of $\longrightarrow^{\pm}$. This gives the polarized sequent calculus modulo, some of whose rules are presented in Figure 2.

*Example 2.* Consider the polarized rewriting system

$$\begin{aligned} A \subseteq B &\to^{-} \forall x.\ x \in A \Rightarrow x \in B \\ A \subseteq B &\to^{+} \neg \mathit{diff}(A, B) \in A \\ A \subseteq B &\to^{+} \mathit{diff}(A, B) \in B \end{aligned}$$

$$\overset{\frown}{\vdash}\ \frac{}{\Gamma, A \vdash B, \Delta}\ A \xrightarrow[\mathcal{R}]{*} C \xleftarrow[\mathcal{R}]{*} B \qquad\qquad \underset{\smile}{\vdash}\ \frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash \Delta}\ A \xleftarrow[\mathcal{R}]{*} C \xrightarrow[\mathcal{R}]{*} B$$

$$\Rightarrow\vdash\ \frac{\Gamma, B \vdash \Delta \qquad \Gamma \vdash A, \Delta}{\Gamma, C \vdash \Delta}\ C \xrightarrow[\mathcal{R}]{*} A \Rightarrow B \qquad\qquad \vdash\Rightarrow\ \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash C, \Delta}\ C \xrightarrow[\mathcal{R}]{*} A \Rightarrow B$$

$$\forall\vdash\ \frac{\Gamma, \{t/x\}A \vdash \Delta}{\Gamma, B \vdash \Delta}\ B \xrightarrow[\mathcal{R}]{*} \forall x.\ A \qquad\qquad \vdash\forall\ \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash B, \Delta}\ \begin{array}{l} B \xrightarrow[\mathcal{R}]{*} \forall x.\ A \\ x \text{ not free in } \Gamma, \Delta \end{array}$$

**Fig. 1.** Some inference rules of the Asymmetric Sequent Calculus Modulo $\mathcal{R}$

$$\overset{\frown}{\vdash}\ \frac{}{\Gamma, A \vdash B, \Delta}\ A \xrightarrow[\mathcal{R}]{*} {}^{-}C\ {}^{+} \xleftarrow[\mathcal{R}]{*} B \qquad\qquad \underset{\smile}{\vdash}\ \frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash \Delta}\ A\ {}^{-} \xleftarrow[\mathcal{R}]{*} C \xrightarrow[\mathcal{R}]{*} {}^{+}B$$

$$\Rightarrow\vdash\ \frac{\Gamma, B \vdash \Delta \qquad \Gamma \vdash A, \Delta}{\Gamma, C \vdash \Delta}\ C \xrightarrow[\mathcal{R}]{*} {}^{-}A \Rightarrow B \qquad\qquad \vdash\neg\ \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash B, \Delta}\ B \xrightarrow[\mathcal{R}]{*} {}^{+}\neg A$$

$$\forall\vdash\ \frac{\Gamma, \{t/x\}A \vdash \Delta}{\Gamma, B \vdash \Delta}\ B \xrightarrow[\mathcal{R}]{*} {}^{-}\forall x.\ A \qquad\qquad \vdash\because\ \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash C, \Delta}\ \begin{array}{l} C \xrightarrow[\mathcal{R}]{*} {}^{+}A \\ C \xrightarrow[\mathcal{R}]{*} {}^{+}B \end{array}$$

**Fig. 2.** Some inference rules of the Polarized Sequent Calculus Modulo $\mathcal{R}$

(*diff* can be seen as the Skolem symbol introduced by the CNF transformation of the definition of the subset relation.) We can build the following proof of the transitivity of the inclusion in the polarized sequent calculus modulo this system:

$$\vdash\because\ \cfrac{\vdash\neg\ \cfrac{\forall\vdash\ \cfrac{\Rightarrow\vdash\ \cfrac{\forall\vdash\ \cfrac{\Rightarrow\vdash\ \cfrac{\overset{\frown}{\vdash}\ \cfrac{}{\mathit{diff}(A,C) \in C \vdash A \subseteq C} \qquad \overset{\frown}{\vdash}\ \cfrac{}{\mathit{diff}(A,C) \in B \vdash \mathit{diff}(A,C) \in B}}{\mathit{diff}(A,C) \in B \Rightarrow \mathit{diff}(A,C) \in C, \mathit{diff}(A,C) \in B \vdash A \subseteq C}}{B \subseteq C, \mathit{diff}(A,C) \in B \vdash A \subseteq C} \qquad \overset{\frown}{\vdash}\ \cfrac{}{\mathit{diff}(A,C) \in A \vdash \mathit{diff}(A,C) \in A}}{\mathit{diff}(A,C) \in A \Rightarrow \mathit{diff}(A,C) \in B, B \subseteq C, \mathit{diff}(A,C) \in A \vdash A \subseteq C}}{A \subseteq B, B \subseteq C, \mathit{diff}(A,C) \in A \vdash A \subseteq C}}{A \subseteq B, B \subseteq C \vdash A \subseteq C, A \subseteq C}}{A \subseteq B, B \subseteq C \vdash A \subseteq C}$$

To a rewriting system $\mathcal{R}$ corresponds a theory, which is the set of formulas that can be proved in the sequent calculus modulo $\mathcal{R}$. It was proved that this theory can always be presented by a traditional set of axioms, which is then called by [23] a compatible presentation. In this paper, we are concerned with the converse direction: is it possible to present any axiomatic first-order theory by a rewriting system? In [13, Corollary 25], we answered positively: it is possible to transform any first-order theory into a rewriting system. However, this rewriting system may not have all the good properties that ensure that deduction modulo behaves well, in particular the admissibility of the cut rule.

The cut rule is admissible in the sequent calculus modulo $\mathcal{R}$ if, whenever a sequent can be proved in it, then it can be proved without using the cut rule

($\underset{\smile}{\vdash}$ in Figure 1 and 2). Abusing terminology, we say that a rewriting system $\mathcal{R}$ admits cut if the cut rule is admissible in the sequent calculus modulo $\mathcal{R}$. The admissibility of the cut rule has a strong proof-theoretical as well as practical importance: it involves that normal forms exist for proofs; it implies the consistency of the theory associated to $\mathcal{R}$; it is equivalent to the completeness of the proof search procedures based on Deduction Modulo Theory $\mathcal{R}$ (such as ENAR by Dowek et al. [23], extending the resolution method, and TaMed by Bonichon and Hermant [4], extending the tableau method); etc. Cut admissibility can also be seen as the completeness of the cut-free sequent calculus w.r.t. the sequent calculus with cuts. In [13], to ensure the cut admissibility, we designed a procedure that completes the rewriting system. However, this procedure may not terminate (and produces too many rules in practice). In this paper, we propose another method to transform an axiomatic presentation of a theory into a cut-admitting rewriting system, that works for any finitely presented and consistent first-order theory.

## 3 Resolution Calculi

We briefly recall the resolution calculus and two refinements, namely the set-of-support strategy and ordered resolution with selection, before presenting the extension of resolution with Deduction Modulo Theory. A derivation in resolution [31] tries to refute a set of clauses by inferring new clauses by means of the two following inference rules (where $P$ and $Q$ are atoms, whereas $L$ and $K$ are literals), until the empty clause is derived.

$$\textsf{Resolution}\ \frac{P \vee C \qquad \neg Q \vee D}{\sigma(C \vee D)}\ \sigma = mgu(P, Q) \qquad \textsf{Factoring}\ \frac{L \vee K \vee C}{\sigma(L \vee C)}\ \sigma = mgu(L, K)$$

### 3.1 Set-of-Support Strategy

The set-of-support strategy for resolution, introduced by Wos et al. [34], consists in restricting the clauses on which resolution can be applied. The input set of clauses is separated into a theory $\Gamma$ and a set of support $\Delta$. At least one of the clauses on which resolution is applied must be in the set of support, and the generated clause is put into the set of support. If the theory $\Gamma$ is assumed to be consistent, this strategy is complete: if $\Gamma, \Delta$ is a unsatisfiable set of clauses, the empty clause can be derived from it using the set-of-support strategy. The set-of-support strategy can therefore be seen as proving a formula $\neg\Delta$ in a theory $\Gamma$ without trying to find a contradiction in $\Gamma$ because $\Gamma$ is assumed to be consistent.

### 3.2 Ordered Resolution with Selection

Ordered resolution with selection [3] (ORS($\succ, S$)) is another refinement of resolution parametrized by an Noetherian ordering $\succ$ on atoms which is stable under substitution and total on ground atoms, and by a selection function $S$ that associates to each clause a subset of the negative literals of this clause. It consists in restricting the literals on which resolution can be applied: if $S(C)$ is not empty, then only the literals in $S(C)$ can be used; otherwise, only the maximal literals w.r.t. $\succ$ can be used. We will therefore say that a literal is selected in a clause $C$ if it is in $S(C)$ or if $S(C)$ is empty and the literal is maximal in $C$. Ordered resolution with selection is refutationally complete whatever ordering or selection function are used.

### 3.3 ([Ordered] Polarized) Resolution Modulo

An extension of resolution based on Deduction Modulo Theory, named Extended Narrowing and Resolution (ENAR), was defined by Dowek et al. [23]. ENAR is a family of resolution calculi, each parametrized by a rewriting system $\mathcal{R}$.[4] It consists in adding a new inference rule, called Extended Narrowing, which produces the clauses obtained by narrowing a clause by $\mathcal{R}$. Since narrowing a clause with a proposition rewriting rule can produce a formula which is not in clausal normal form, the latter has to be computed to find the generated clauses. The Extended Narrowing rule is therefore:

$$\textsf{Ext. Narr.}\ \frac{C}{D}\ C \underset{\mathcal{R}}{\leadsto} A,\ D \in \mathcal{C}\ell(A)$$

where $\mathcal{C}\ell(A)$ is the set of clauses of the clausal normal form of $A$.

We say that ENAR for $\mathcal{R}$ is complete if, whenever $\vdash A$ can be proved in the sequent calculus modulo $\mathcal{R}$, the empty clause can be derived from $\mathcal{C}\ell(\neg A)$ in ENAR for $\mathcal{R}$. Hermant [28] proved that the empty clause can be derived from $\mathcal{C}\ell(\neg A)$ in ENAR for $\mathcal{R}$ if and only if $\vdash A$ can be proved *without cut* in the sequent calculus modulo $\mathcal{R}$. This implies that ENAR for a rewriting system $\mathcal{R}$ is complete if and only if the sequent calculus modulo $\mathcal{R}$ admits cut.

In ENAR, formulas have to be put in clausal normal form dynamically, which may require fresh Skolem symbols each time. To avoid this, Dowek [21] introduced the Polarized Resolution Modulo (PRM). As ENAR, this is a family of resolution calculi parametrized by a rewriting system, but this system is assumed to be polarized, and clausal, i.e., each negative rule is of the form $P \to^- C$, and each positive rule is of the form $P \to^+ \neg C$, where $C$ is in clausal form. In that case, the Extended Narrowing rule becomes:

$$\textsf{Ext. Narr.}^-\ \frac{P \vee C}{\sigma(D \vee C)}\ \sigma = mgu(P,Q),\ Q \to^- D \in \mathcal{R}$$

$$\textsf{Ext. Narr.}^+\ \frac{\neg Q \vee D}{\sigma(C \vee D)}\ \sigma = mgu(P,Q),\ P \to^+ \neg C \in \mathcal{R}$$

Gao [27] proved that any rewriting system admitting cut can be transformed into an equivalent polarized and clausal one, so that PRM can be applied whenever ENAR can.

[4] ENAR is originally parametrized by a rewriting system $\mathcal{R}$ and an equational theory $\mathcal{E}$, and the unification in the Resolution, Factoring and Extended Narrowing rules is performed modulo the equational theory $\mathcal{E}$. To keep it simple, we choose not to consider equational theories in this paper.

PRM can itself be restricted using orderings, as in Ordered Resolution. In Orderered Polarized Resolution Modulo (OPRM($\succ$), 5), the inference rules of PRM can only be applied to literals that are maximal w.r.t. the ordering $\succ$. It has been proved that completeness of OPRM($\succ$) is equivalent to the completeness of PRM (and therefore to the cut admissibility of the rewriting system), whatever ordering is used. This means that if a rewriting system admits cut, it can be used in OPRM($\succ$) even if $\succ$ is not compatible with the rewriting system, that is, it orients some rules in the wrong direction. Note that completeness of OPRM with selection of negative literals is still an open question.

### 3.4 Polarized Rewriting Rules and One-Way Clauses

To each polarized clausal rewriting rule can be associated a clause in which one literal is selected. This clause is called a *one-way* clause by Dowek [21]. For instance, to $P \rightarrow^- C$ is associated $\underline{\neg P} \vee C$, and to $P \rightarrow^+ \neg C$ is associated $\underline{P} \vee C$ (the selected literals are underlined). Conversely, to a clause and a literal occurrence in this clause can be associated a polarized clausal rewriting rule: to $\underline{P} \vee C$ is associated $P \rightarrow^+ \neg C$, and to $\underline{\neg P} \vee C$ is associated $P \rightarrow^- C$. It is worth remarking that applying Extended Narrowing on a clause $C$ with a polarized clausal rule $R$ leads to the same clause as applying Resolution on $C$ and the one-way clause corresponding to $R$. Thus, polarized rewriting rules can be seen as special clauses with the following properties:

- only the selected literal can be used to resolve a one-way clause;
- two one-way clauses cannot be resolved together.

The results of this paper exploit this isomorphism between polarized clausal rewriting rules and one-way clauses.

# 4 Cut-Admitting Presentations of Theories

### 4.1 Simulating set of support

We suppose that the theory is presented by means of a set of clauses. If not, it has to be transformed into clausal normal form using standard techniques.

**Definition 1.** *Given a set of clauses $\Gamma$, we define the polarized rewriting system $\mathcal{R}_\Gamma$ consisting of, for each clause $C$ in $\Gamma$ and each literal $L$ in $C$,*

- *if $L = P$ is positive, a positive rewriting rule $P \rightarrow^+ \neg\forall x_1, \dots, x_n.\, L_1 \vee \dots \vee L_m$ where $x_1, \dots, x_n$ are the free variables of $C$ that are not free in $P$ and $L_1, \dots, L_m$ are the literals of $C$ different from $P$;*
- *if $L = \neg P$ is negative, a negative rewriting rule $P \rightarrow^- \forall x_1, \dots, x_n.\, L_1 \vee \dots \vee L_m$ where $x_1, \dots, x_n$ are the free variables of $C$ that are not free in $P$ and $L_1, \dots, L_m$ are the literals of $C$ different from $\neg P$.*

*Example 3.* Let $\Gamma$ be the set of clauses corresponding to the definition of the inclusion:

$$\begin{aligned}\neg A \subseteq B \vee \neg(X \in A) \vee X \in B\\ A \subseteq B \vee \mathit{diff}(A,B) \in A\\ A \subseteq B \vee \neg(\mathit{diff}(A,B) \in B)\end{aligned}$$

Then $\mathcal{R}_\Gamma$ is

$$\begin{aligned} A \subseteq B &\rightarrow^- \forall x.\ \neg x \in A \vee x \in B\\ X \in A &\rightarrow^- \forall b.\ \neg A \subseteq b \vee X \in b\\ X \in B &\rightarrow^+ \neg\forall a.\ \neg a \subseteq B \vee X \in a\\ A \subseteq B &\rightarrow^+ \neg \mathit{diff}(A,B) \in A\\ \mathit{diff}(A,B) \in A &\rightarrow^+ \neg A \subseteq B\\ A \subseteq B &\rightarrow^+ \neg\neg \mathit{diff}(A,B) \in B\\ \mathit{diff}(A,B) \in B &\rightarrow^- A \subseteq B\end{aligned}$$

*Remark 1.* The number of rewriting rules in $\mathcal{R}_\Gamma$ is equal to the number of literal occurrences in $\Gamma$.

Using such a rewriting system, one can simulate the set-of-support strategy. Hence, completeness holds as well as cut admissibility.

**Theorem 1 ([6, Theorem 14]).** *The consistency of a finite set of clauses $\Gamma$ implies the admissibility of the cut rule in the polarized sequent calculus modulo $\mathcal{R}_\Gamma$.*

## 4.2 Cut admissibility through saturation

To reduce the number of rules, it is possible to associate a polarized rewriting system to a set of clauses for ordered resolution with selection by considering as left-hand sides only the literals that are selected in a clause. Thus, we would not produce a rule for each literal but only for those that are in $S(C)$ or that are maximal if $S(C)$ is empty.

*Example 4.* We consider the example of the inclusion again, with an ordering such that literals with $\subseteq$ are greater than literals with $\in$. The resulting rewriting system is reduced to

$$\begin{aligned} A \subseteq B &\rightarrow^- \forall x.\ \neg x \in A \vee x \in B\\ A \subseteq B &\rightarrow^+ \neg \mathit{diff}(A,B) \in A\\ A \subseteq B &\rightarrow^+ \neg\neg \mathit{diff}(A,B) \in B\end{aligned}$$

However, ordered resolution with selection is not compatible with the set-of-support strategy, in the sense that their combination jeopardizes completeness. Nevertheless, a sufficient condition to ensure the completeness is the saturation of the set of clauses used as complement of the set of support (i.e. the theory): the clauses that can be inferred from it must either be in it or be redundant (i.e. they must be semantically implied by smaller clauses).

**Theorem 2 ([9, Theorem 7]).** *The saturation of a finite set of clauses $\Gamma$ w.r.t. ORS($\succ, S$) implies the admissibility of the cut rule in the polarized sequent calculus modulo the rewrite system $\mathcal{R}_\Gamma$ restricted to rules whose left-hand side is selected (or maximal if none are selected) w.r.t. $S$ and $\succ$.*

Since saturation implies satisfiability of the set of clauses (if it does not contain the empty clause), which is undecidable, saturating a set of clauses may not terminate. However, it can be semi-automated. First-order automated theorem provers like SPASS by Weidenbach, Dimova, Fietzke, Kumar, Suda, and Wischnewski [33] actually work by trying to saturate the input set of clauses, unless the empty clause is derived. Running SPASS on the example above (with precedence $\subseteq\ >\ \in\ > \mathit{diff}$ and $\subseteq$ and $\in$ dominant predicates), the saturation generates two new clauses

$$\begin{gathered}\underline{\neg X \in A} \vee \mathit{diff}(A,B) \in A \vee X \in B\\ \underline{\neg \mathit{diff}(A,B) \in B} \vee \neg X \in A \vee X \in B\end{gathered}$$

The following rewriting system therefore admits cuts:

$$\begin{aligned} A \subseteq B &\rightarrow^- \forall x.\ \neg x \in A \vee x \in B\\ A \subseteq B &\rightarrow^+ \neg \mathit{diff}(A,B) \in A\\ A \subseteq B &\rightarrow^+ \neg\neg \mathit{diff}(A,B) \in B\\ X \in A &\rightarrow^- \forall x.\ \mathit{diff}(A,B) \in A \vee x \in B\\ \mathit{diff}(A,B) \in B &\rightarrow^- \forall x.\ \neg X \in A \vee X \in B\end{aligned}$$

## 5 Experimental Comparison of Orientation Techniques

We have compared several techniques that transform a set of axioms into a rewriting system. Two of them, being based on Theorems 1 and 2, are therefore proved to be complete, in the sense that the resulting rewriting system admits cut. The other ones are merely heuristics.

### 5.1 Description of the different techniques

We compared six different ways of transforming a set of axioms into a rewriting system.

*ClausalAll:* The set of axioms is put in clausal normal form (using the prover E of Schultz [32]), and it is transformed into a rewriting system as described in Definition 1.

*Sat:* The set of axioms is saturated using E, and it is then transformed into a rewriting system restricted to the selected literals, as in Theorem 2. Of course, the saturation may not terminate, so this technique does not always succeed.

*Equiv(ClausalAll):* Depending on their shape, axioms are transformed into rewriting rules:

| Axioms | give rewrite rules |
|---|---|
| $\forall \overline{x}.\ t = u$ | $t \to u$ provided $FV(u) \subseteq FV(t)$ |
| $\forall \overline{x}.\ P \Rightarrow A$ | $P \to^- \forall \overline{y}.\ C$ for all $C \in \mathcal{C}\ell(A)$, with $\overline{y} = FV(C) \setminus FV(P)$ |
| $\forall \overline{x}.\ A \Rightarrow P$ | $P \to^+ \neg\forall \overline{y}.\ C$ for all $C \in \mathcal{C}\ell(\neg A)$, with $\overline{y} = FV(C) \setminus FV(P)$ |
| $\forall \overline{x}.\ P \Leftrightarrow A$ <br> $\forall \overline{x}.\ A \Leftrightarrow P$ | the same as those of $\forall \overline{x}.\ P \Rightarrow A$ and $\forall \overline{x}.\ A \Rightarrow P$ |
| $\forall \overline{x}.\ \neg P \Rightarrow A$ | $P \to^+ \neg\forall \overline{y}.\ C$ for all $C \in \mathcal{C}\ell(A)$, with $\overline{y} = FV(C) \setminus FV(P)$ |
| $\forall \overline{x}.\ A \Rightarrow \neg P$ | $P \to^- \forall \overline{y}.\ C$ for all $C \in \mathcal{C}\ell(\neg A)$, with $\overline{y} = FV(C) \setminus FV(P)$ |
| $\forall \overline{x}.\ \neg P \Leftrightarrow A$ <br> $\forall \overline{x}.\ A \Leftrightarrow \neg P$ | the same as those of $\forall \overline{x}.\ \neg P \Rightarrow A$ and $\forall \overline{x}.\ A \Rightarrow \neg P$ |
| $\forall \overline{x}.\ P$ | $P \to^+ \neg\bot$ |
| $\forall \overline{x}.\ \neg P$ | $P \to^- \bot$ |
| $\forall \overline{x}.\ A \wedge B$ | the same as those of $\forall \overline{x}.\ A$ and $\forall \overline{x}.\ B$ |

where $P$ is atomic, and $FV(A)$ is the set of free variables of $A$. Axioms not of these shapes are transformed using the ClausalAll technique above.

*Equiv(Id):* This is the same technique as Equiv(ClausallAll), except that axioms that are not of a recognized shape are not transformed into rewrite rules at all and are used as normal formulas.

*Presat(ClausalAll):* Since saturation may not terminate, we can decide to stop it after a certain amount of clauses have been generated. In this technique, we saturate the set of axioms using `E` until 100 clauses have been processed. These processed clauses are transformed into a rewriting system restricted to the selected literals as in Sat. The unprocessed clauses generated during the saturation are transformed using the ClausalAll technique.

*Presat(Id):* This is the same technique as Presat(ClausallAll), except that unprocessed clauses are not transformed into rewrite rules and are used as normal clauses.

## 5.2 `autotheo`

We have implemented a tool called `autotheo` that automatically transforms the set of axioms of a problem into a rewriting system using one of these techniques. This tool takes as input a problem in TPTP format. It outputs a new problem in which part of the axioms have been replaced by rewriting rules. This output can either be in the `Zipperposition` format[5], which is also understood by `ArchSAT`; or in the TPTP format with the following convention: if a clause appears with the axiom role, then it should be understood as the rewriting rule corresponding

[5] https://github.com/sneeuwballen/zipperposition#native-syntax

to the one-way clause where the first literal is selected. This is the convention used by `iProverModulo` with the flag `--modulo true`.

On the input file corresponding to Example 1:

```
fof(inclusion_def, axiom,
  ! [X,Y] : (subseteq(A,B) <=> (! [Z] : (in(Z, A) => in(Z,B))))).
fof(h1, hypothesis, subseteq(a,b)).
fof(h2, hypothesis, subseteq(b,c)).
fof(g, conjecture, subseteq(a,c)).
```

with the Equiv(ClausalAll) heuristic, `autotheo` produces the following files in `Zipperposition` format:

```
val iota : type.
val a :  iota.
val b :  iota.
val c :  iota.
val zf_in : iota -> iota ->  prop.
val sk1_esk1_2 : iota -> iota ->  iota.
val subseteq : iota -> iota ->  prop.
rewrite[name c_0_11_0] forall (X2 X3 : iota). (subseteq (X3) (X2) =>
  (forall (X1 : iota). (zf_in (X1) (X2)) || (~ (zf_in (X1) (X3))))).
rewrite[name c_0_10_0] forall (X1 X2 : iota). (~ (subseteq (X2) (X1)) =>
  (zf_in (sk1_esk1_2 (X1) (X2)) (X2))).
rewrite[name c_0_9_0] forall (X1 X2 : iota). (~ (subseteq (X2) (X1)) =>
  (~ (zf_in (sk1_esk1_2 (X1) (X2)) (X1)))).
goal[name g] subseteq (a) (c).
assert[name h2] subseteq (b) (c).
assert[name h1] subseteq (a) (b).
```

and in TPTP format for `iProverModulo`:

```
cnf(c_0_11_0,axiom,~subseteq(X3,X2)|in(X1,X2)|~in(X1,X3)).
cnf(c_0_10_0,axiom,subseteq(X2,X1)|in(sk1_esk1_2(X1,X2),X2)).
cnf(c_0_9_0,axiom,subseteq(X2,X1)|~in(sk1_esk1_2(X1,X2),X1)).
cnf(c_0_12,negated_conjecture,~subseteq(a,c)).
cnf(c_0_13,hypothesis,subseteq(b,c)).
cnf(c_0_14,hypothesis,subseteq(a,b)).
```

`autotheo` consists of ∼2000 lines of OCaml code. It depends on the prover `E` to put formulas in clausal normal form and to saturate set of clauses. `autotheo` is distributed as part of `iProverModulo`, even if it can be compiled and used independently from it. It is available on GitHub (`https://github.com/gburel/iProverModulo/tree/master/autotheo`).

## 5.3 Experiment

We compared the orientation techniques on all first-order problems of the TPTP library v.7.1.0, i.e. 16079 problems using `cnf` and `fof` syntax. For each problem,

**Table 1.** Number of problems solved by the prover with the given orientation heuristic. (In parenthesis, number of problems solved by the prover uniquely with the given heuristic.) For "Any of the above", number of problems solved by the prover by at least one of the orientation techniques. (In parenthesis, number of problems solved by the prover by at least one of the orientation techniques but not without rewriting.)

| | `ArchSAT` | `iProverModulo` | `Zipperposition` |
|---|---|---|---|
| Equiv(ClausalAll) | 1674 (14) | 3684 (83) | 1162 (22) |
| ClausalAll | 1480 (27) | 3879 (49) | 1040 (31) |
| Equiv(Id) | 1542 (22) | 3911 (76) | 3054 (198) |
| Presat(Id) | 1707 (80) | 3915 (266) | 1857 (47) |
| Presat(ClausalAll) | 1653 (65) | 3604 (27) | 1233 (23) |
| Any of the above | 2443 (663) | 5002 (1162) | 3305 (564) |
| Without rewriting | 1473 (340) | 3849 (357) | 5678 (2316) |

and for each tested heuristic, we first transform the set of axioms of the problem into a rewriting system using `autotheo`, with a timeout of 10s. We then try to prove the resulting problem with `ArchSAT` (v0.1), `iProverModulo` (v0.7+0.3) and `Zipperposition` (v1.2), with a timeout of 180s and a memory limit of 1GB. Experiments were performed on virtualized Intel Core Processor (Haswell, no TSX) 2.30 GHz CPUs, on Inria's OpenStack platform Gulliver.

The results are summarized in Table 1. We do not report the results for the Sat heuristic because it succeeds at orienting a problem only for a few cases in the given time. We can remark that `ArchSAT` always proves more problems when it is given a problem with axioms oriented as rewriting rules, compared to using the axiom as is. This may be explained by the way SMT solvers work: one needs to find the good instantiations for first-order axioms, whereas rewriting rules are applied just by matching. For `iProverModulo`, the benefit of using rewriting rules depends of the chosen heuristic. However, the number problems solved by at least one of the heuristic is much larger than without rewriting rules. The results are different for `Zipperposition`, where using axioms turns out to be better. Only the Equiv(Id) strategy works relatively well, but this may be due to the fact that on problems where there are no formula on which the Equiv heuristic can be applied, this strategy is equivalent to using the original problem file. This is the case in particular for problems already in clausal normal form, where only clauses with a single literal can be turned into a rewriting rules. We need to investigate further why using rewriting rules in `Zipperposition` does not work so well. It seems that for some of the problems, it is due to the fact that `Zipperposition` assumes the rewriting system to be confluent, and therefore only computes one normal form. Another explanation is that rewriting was only added lately in `Zipperposition`, and perhaps it does not combine well enough with the rest of the proving process, in particular the fact that first-order predicates are transformed into Boolean equations.

Looking at the problems solved uniquely by a given heuristic, we can note that each heuristic has some interest. This suggests to use of the schedule that

**Table 2.** Number of problems without equality symbol solved by the prover with or without orienting axioms into rewriting rules. (In parenthesis, number of problems solved by the prover uniquely with or without.)

| | `ArchSAT` | `iProverModulo` | `Zipperposition` |
|---|---|---|---|
| With at least one heuristic | 272 (90) | 896 (46) | 734 (35) |
| Without rewriting | 150 (7) | 850 (0) | 656 (43) |

runs the prover successively on each orientation, or to run several instances of the prover for each orientation in parallel.

It can be interesting to compare how the results vary depending on the domain of the TPTP library. Table 3 in the appendix details the results on each domain, distinguishing problems in clausal normal form (`-`) and general first-order formulas (`+`). For instance, using rewriting is always better (even using `Zipperposition`) in the Semantic Web (SWB) and in the Syntactic (SYN and SYO) domains. Putting aside `Zipperposition`, orienting theories leads to a large gain in the General Algebra (ALG), Analysis (ANA), Boolean Algebra (BOO), Category Theory (CAT), Commonsense Reasoning (CSR), Geometry (GEO), Groups (GRP), Henkin Models (HEN), Hardware Verification (HWV), Lattices (LAT), Logic Calculi (LCL), Number Theory (NUM), and Rings (RNG) domains. This suggests that such domains contain theories that can be nicely oriented. On the Set Theory (SET, SEU, and SEV) and Software Verification (SWV and WSW) domains, `iProverModulo` is much better with oriented theories than without rewriting rules, but this result cannot be observed in the two other provers. This is a bit striking, since Deduction Modulo Theory is believed to work well on problems in set theory; in particular, it is used in Zenon Modulo to prove proof obligations stemming from industrial problems using Atelier B, a tool founded on a set theory (namely B Set Theory) [14].

If one restricts oneself to problems without equality, results are even better, even for `Zipperposition`, as shown in Table 2. In particular, each problem solved by `iProverModulo` without rewriting can also be proved by using at least one of the orientation heuristics. The difference of results without and with the equality predicate may indicate that the interaction between the rewriting rules and the equality predicate needs to be better understood, in particular in the context of the superposition calculus.

## 6 Conclusion and Further Work

In this paper, we have presented several methods to turn axioms into rewriting systems usable in Deduction Modulo Theory. For some of them, it is proved that the resulting proof calculus is complete, whereas some other are heuristics depending on the shape of the formulas. We have presented the results of experiments comparing these techniques. For those, we used three provers implementing Deduction Modulo Theory, namely `ArchSAT`, `iProverModulo`, and

Zipperposition. Whereas the results are somehow mediocre for Zipperposition, they are much better for ArchSAT and iProverModulo, and they indicate that automatically orienting theories to use them in these provers is actually feasible. No orientation techniques outperform the others, which suggests to use them concurrently when solving a problem. This work therefore constitutes an important step towards the automatic production of provers adapted to a given theory.

*Equality.* The orientation techniques presented in this paper are primarily designed for first-order logic without equality. However, theories are often presented in first-order logic with equality. Adding the axioms for equality (reflexivity, symmetry, transitivity and congruence w.r.t. the function symbols and the predicates) and transforming them as presented in this paper is a theoretical way to obtain presentations of such theories. However, it does not take into account the specificity of equality, and the way it can be integrated into a deduction system thanks to Deduction Modulo Theory. A first improvement is to put the equational axioms into an equational theory modulo which rewriting and unification is performed (see footnote page 7). Nevertheless, existing provers perform unification and rewriting modulo only for specific equational theories, such as commutativity of a function symbol. Only such axioms should therefore be presented this way. The other equational axioms should be transformed into *term* rewriting rules. It remains to be proved that using term rewriting rules for equational axioms and proposition rewriting rules as obtained as in this paper for the other axioms is complete. We conjecture that it is the case as long as the term rewriting system is confluent and commutes with the proposition rewriting system. The confluence of the term rewriting system could be ensured by the standard completion of Knuth and Bendix [29].

The next step is to design proof-search procedures based on deduction modulo for first-order logic with equality. A good candidate would be an extension of the superposition calculus [2] with an Extended Narrowing rule, but we currently do not know if cut admissibility is enough to prove its completeness. In particular, we do not know how to take this assumption into account in a completeness proof based on saturation, the kind of proof usually used for the completeness of superposition.

*Combination with other kinds of presentations.* In this paper, we have shown how to present any first-order theories as rewriting systems. However, for some specific theories, rewriting is probably not the best way to present them. For instance, to search for proofs in linear arithmetic, it is probably more efficient to use a combination with the simplex method than to use a rewriting system for linear arithmetic. Therefore, we would like to investigate how it could be possible to combine Deduction Modulo Theory with other ways to present theories. A first lead would be to study canonized rewriting [16], where (ground) rewriting is combined with Shostak theories to get SMT solvers modulo AC. Then, we would need a way to recognize theories during proof search to trigger the most appropriate method [11].

# Bibliography

[1] Baader, F., Nipkow, T.: Term *Re*writing and *all That*. Cambridge University Press (1998)

[2] Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. Journal of Logic and Computation 4(3), 1–31 (1994)

[3] Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 19–99. Elsevier and MIT Press (2001)

[4] Bonichon, R., Hermant, O.: A semantic completeness proof for TaMed. In: Hermann, M., Voronkov, A. (eds.) LPAR. LNCS, vol. 4246, pp. 167–181. Springer (2006)

[5] Burel, G.: Embedding deduction modulo into a prover. In: Dawar, A., Veith, H. (eds.) CSL. LNCS, vol. 6247, pp. 155–169. Springer (2010)

[6] Burel, G.: Consistency implies cut admissibility. In: Faure, G., Lengrand, S., Mahboubi, A. (eds.) International Workshop on Proof-Search in Axiomatic Theories and Type Theories (2011), https://hal.inria.fr/inria-00614040

[7] Burel, G.: Efficiently simulating higher-order arithmetic by a first-order theory modulo. Logical Methods in Computer Science 7(1:3), 1–31 (2011)

[8] Burel, G.: Experimenting with deduction modulo. In: Sofronie-Stokkermans, V., Bjørner, N. (eds.) CADE. LNCS, vol. 6803, pp. 162–176. Springer (2011)

[9] Burel, G.: Cut admissibility by saturation. In: Dowek, G. (ed.) RTA-TLCA. LNCS, vol. 8560, pp. 124–138. Springer (2014)

[10] Burel, G., Bury, G., Cauderlier, R., Delahaye, D., Halmagrand, P., Hermant, O.: First-order automated reasoning with theories: When deduction modulo theory meets practice. Journal of Automated Reasoning (2019)

[11] Burel, G., Cruanes, S.: Detection of first-order axiomatic theories. In: Fontaine, P., Ringeissen, C., Schmidt, R. (eds.) FroCoS. LNAI, vol. 8152, pp. 229–244. Springer (2013)

[12] Burel, G., Dowek, G.: How can we prove that a proof search method is not an instance of another? In: LFMTP. pp. 84–87. ACM International Conference Proceeding Series, ACM (2009)

[13] Burel, G., Kirchner, C.: Regaining cut admissibility in deduction modulo using abstract completion. Information and Computation 208(2), 140–164 (2010)

[14] Bury, G., Delahaye, D., Doligez, D., Halmagrand, P., Hermant, O.: Automated deduction in the B set theory using typed proof search and deduction modulo. In: Fehnker, A., McIver, A., Sutcliffe, G., Voronkov, A. (eds.) LPAR short presentations. EPiC Series in Computing, vol. 35, pp. 42–58. EasyChair (2015), http://www.easychair.org/publications/volume/LPAR-20

[15] Ciabattoni, A., Galatos, N., Terui, K.: From axioms to analytic rules in nonclassical logics. In: Pfenning, F. (ed.) LICS. IEEE Computer Society (2008)

[16] Conchon, S., Contejean, E., Iguernelala, M.: Canonized rewriting and ground AC completion modulo Shostak theories. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, Springer (2011)

[17] Delahaye, D., Doligez, D., Gilbert, F., Halmagrand, P., Hermant, O.: Zenon modulo: When Achilles outruns the tortoise using deduction modulo. In: McMillan, K.L., Middeldorp, A., Voronkov, A. (eds.) LPAR. LNCS, vol. 8312, pp. 274–290. Springer (2013)

[18] Dowek, G.: What is a theory? In: Alt, H., Ferreira, A. (eds.) STACS. LNCS, vol. 2285, pp. 50–64. Springer (2002)

[19] Dowek, G.: Confluence as a cut elimination property. In: Nieuwenhuis, R. (ed.) RTA. LNCS, vol. 2706, pp. 2–13. Springer (2003)

[20] Dowek, G.: What do we know when we know that a theory is consistent? In: Nieuwenhuis, R. (ed.) CADE. LNCS, vol. 3632, pp. 1–6. Springer (2005)

[21] Dowek, G.: Polarized resolution modulo. In: Calude, C.S., Sassone, V. (eds.) IFIP TCS. IFIP AICT, vol. 323, pp. 182–196. Springer (2010)

[22] Dowek, G., Hardin, T., Kirchner, C.: HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. Mathematical Structures in Computer Science 11(1), 1–25 (2001)

[23] Dowek, G., Hardin, T., Kirchner, C.: Theorem proving modulo. Journal of Automated Reasoning 31(1), 33–72 (2003)

[24] Dowek, G., Miquel, A.: Cut elimination for Zermelo's set theory (2006), available on authors' web page

[25] Dowek, G., Werner, B.: Arithmetic as a theory modulo. In: Giesl, J. (ed.) RTA. LNCS, vol. 3467, pp. 423–437. Springer (2005)

[26] Gallier, J.H.: Logic for Computer Science: Foundations of Automatic Theorem Proving, Computer Science and Technology Series, vol. 5. Harper & Row, New York (1986)

[27] Gao, J.: Clausal presentation of theories in deduction modulo. In: International Workshop on Proof-Search in Axiomatic Theories and Type Theories 2011 (2011), http://hal.inria.fr/inria-00614251/en/

[28] Hermant, O.: Resolution is cut-free. Journal of Automated Reasoning 44(3), 245–276 (2009)

[29] Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press, Oxford (1970)

[30] Negri, S., von Plato, J.: Cut elimination in the presence of axioms. The Bulletin of Symbolic Logic 4(4), 418–435 (1998)

[31] Robinson, J.A.: A machine-oriented logic based on the resolution principle. Journal of the ACM 12, 23–41 (1965)

[32] Schultz, S.: System description: E 0.81. In: Basin, D.A., Rusinowitch, M. (eds.) IJCAR. LNCS, vol. 3097, pp. 223–228. Springer (2004)

[33] Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) CADE. LNCS, vol. 5663, pp. 140–145. Springer (2009)

[34] Wos, L., Robinson, G.A., Carson, D.F.: Efficiency and completeness of the set of support strategy in theorem proving. J. ACM 12(4), 536–541 (1965)

## A Appendix

**Table 3.** Number of problems solved by the prover with the given orientation heuristic, by domains.

| Problem | Zipperposition<br>ClausalAll | Equiv(ClausalAll) | Equiv(Id) | Presat(Id) | Presat(ClausalAll) | Any | No rewriting |
|---|---|---|---|---|---|---|---|
| ALG- | 1 | 0 | 16 | 25 | 1 | 36 | 43 |
| ANA- | 4 | 4 | 28 | 9 | 5 | 31 | 33 |
| BOO- | 0 | 0 | 13 | 0 | 0 | 13 | 45 |
| CAT- | 1 | 1 | 41 | 5 | 11 | 41 | 34 |
| COL- | 23 | 22 | 61 | 39 | 24 | 64 | 133 |
| COM- | 4 | 3 | 12 | 7 | 6 | 13 | 12 |
| FLD- | 0 | 0 | 56 | 1 | 0 | 56 | 66 |
| GEO- | 2 | 2 | 115 | 4 | 5 | 115 | 106 |
| GRA- | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| GRP- | 99 | 99 | 201 | 241 | 194 | 335 | 583 |
| HEN- | 0 | 0 | 54 | 1 | 0 | 54 | 44 |
| HWC- | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| HWV- | 2 | 1 | 14 | 1 | 1 | 15 | 64 |
| KRS- | 0 | 0 | 16 | 0 | 0 | 16 | 9 |
| LAT- | 9 | 3 | 30 | 28 | 9 | 39 | 58 |
| LCL- | 13 | 5 | 187 | 34 | 11 | 211 | 258 |
| LDA- | 0 | 0 | 0 | 2 | 0 | 2 | 6 |
| MGT- | 2 | 1 | 55 | 17 | 17 | 56 | 49 |
| MSC- | 3 | 3 | 12 | 4 | 4 | 12 | 10 |
| NLP- | 47 | 47 | 90 | 47 | 47 | 90 | 12 |
| NUM- | 1 | 3 | 19 | 1 | 1 | 21 | 19 |
| PLA- | 2 | 2 | 30 | 6 | 5 | 31 | 29 |
| PUZ- | 9 | 8 | 55 | 10 | 9 | 55 | 48 |
| REL- | 0 | 0 | 0 | 0 | 0 | 0 | 19 |
| RNG- | 0 | 0 | 7 | 0 | 0 | 7 | 23 |
| ROB- | 0 | 0 | 0 | 0 | 0 | 0 | 11 |
| SCT- | 0 | 0 | 1 | 1 | 2 | 3 | 18 |
| SET- | 9 | 12 | 135 | 37 | 12 | 154 | 205 |
| SEV- | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| SWB- | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SWC- | 0 | 0 | 233 | 71 | 0 | 0 | 237 |
| SWV- | 29 | 25 | 163 | 99 | 39 | 228 | 273 |
| SWW- | 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| SYN- | 335 | 335 | 563 | 509 | 341 | 572 | 486 |
| SYO- | 0 | 0 | 10 | 1 | 0 | 10 | 10 |
| TOP- | 0 | 0 | 4 | 2 | 0 | 4 | 4 |
| TOTAL- | 597 | 578 | 2224 | 1203 | 745 | 2287 | 2961 |
| AGT+ | 0 | 0 | 0 | 15 | 0 | 15 | 18 |
| ALG+ | 54 | 53 | 63 | 86 | 59 | 97 | 137 |
| CAT+ | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| COM+ | 0 | 0 | 9 | 5 | 2 | 14 | 28 |
| CSR+ | 1 | 16 | 16 | 37 | 5 | 44 | 49 |
| GEO+ | 0 | 1 | 19 | 4 | 8 | 27 | 279 |
| GRA+ | 0 | 0 | 5 | 0 | 0 | 5 | 11 |
| GRP+ | 0 | 1 | 2 | 2 | 1 | 3 | 31 |
| HAL+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HWV+ | 0 | 0 | 0 | 3 | 0 | 3 | 2 |
| KLE+ | 1 | 1 | 1 | 0 | 0 | 1 | 86 |
| KRS+ | 0 | 9 | 9 | 1 | 0 | 10 | 129 |
| LAT+ | 0 | 0 | 4 | 2 | 0 | 6 | 12 |
| LCL+ | 28 | 28 | 28 | 28 | 28 | 28 | 56 |
| MED+ | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| MGT+ | 2 | 13 | 17 | 17 | 17 | 23 | 49 |
| MSC+ | 0 | 0 | 1 | 0 | 0 | 1 | 5 |
| NLP+ | 25 | 27 | 27 | 25 | 25 | 27 | 10 |
| NUM+ | 2 | 5 | 54 | 16 | 3 | 65 | 318 |
| PLA+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PRO+ | 0 | 0 | 48 | 0 | 0 | 48 | 11 |
| PUZ+ | 5 | 5 | 8 | 6 | 6 | 8 | 18 |
| REL+ | 0 | 0 | 0 | 0 | 0 | 0 | 20 |
| RNG+ | 0 | 0 | 0 | 4 | 0 | 4 | 76 |
| SCT+ | 3 | 3 | 1 | 2 | 1 | 5 | 13 |
| SET+ | 8 | 21 | 28 | 12 | 11 | 36 | 254 |
| SEU+ | 4 | 4 | 7 | 10 | 4 | 17 | 273 |
| SEV+ | 0 | 0 | 2 | 0 | 0 | 2 | 1 |
| SWB+ | 0 | 0 | 66 | 5 | 0 | 68 | 48 |
| SWC+ | 0 | 73 | 73 | 49 | 0 | 90 | 240 |
| SWV+ | 4 | 20 | 20 | 10 | 6 | 23 | 203 |
| SWW+ | 6 | 6 | 4 | 11 | 9 | 23 | 44 |
| SYN+ | 300 | 298 | 308 | 302 | 302 | 314 | 281 |
| SYO+ | 0 | 0 | 10 | 1 | 0 | 10 | 4 |
| TOP+ | 0 | 0 | 0 | 1 | 1 | 1 | 7 |
| TOTAL+ | 443 | 584 | 830 | 654 | 488 | 1018 | 2717 |
| Problem | ClausalAll<br>Zipperposition | Equiv(ClausalAll) | Equiv(Id) | Presat(Id) | Presat(ClausalAll) | Any | No rewriting |
| TOTAL | 1040 | 1162 | 3054 | 1857 | 1233 | 3305 | 5678 |

| Problem | archSAT<br>ClausalAll | Equiv(ClausalAll) | Equiv(Id) | Presat(Id) | Presat(ClausalAll) | Any | No rewriting |
|---|---|---|---|---|---|---|---|
| ALG- | 10 | 8 | 10 | 13 | 9 | 16 | 7 |
| ANA- | 6 | 6 | 7 | 7 | 7 | 15 | 8 |
| BOO- | 3 | 3 | 3 | 28 | 36 | 37 | 0 |
| CAT- | 3 | 2 | 2 | 11 | 10 | 18 | 3 |
| COL- | 45 | 45 | 45 | 44 | 42 | 48 | 42 |
| COM- | 3 | 3 | 2 | 2 | 2 | 3 | 2 |
| FLD- | 8 | 8 | 6 | 6 | 2 | 8 | 7 |
| GEO- | 23 | 21 | 17 | 13 | 16 | 35 | 18 |
| GRA- | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| GRP- | 16 | 20 | 17 | 73 | 99 | 118 | 5 |
| HEN- | 5 | 5 | 2 | 15 | 10 | 20 | 2 |
| HWC- | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| HWV- | 63 | 59 | 37 | 45 | 49 | 76 | 6 |
| KRS- | 15 | 15 | 1 | 22 | 23 | 24 | 1 |
| LAT- | 30 | 27 | 31 | 29 | 29 | 37 | 25 |
| LCL- | 35 | 29 | 30 | 45 | 51 | 81 | 31 |
| LDA- | 0 | 0 | 0 | 2 | 2 | 2 | 2 |
| MGT- | 8 | 9 | 2 | 0 | 2 | 12 | 3 |
| MSC- | 5 | 5 | 3 | 9 | 9 | 11 | 3 |
| NLP- | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NUM- | 5 | 5 | 4 | 3 | 4 | 5 | 3 |
| PLA- | 2 | 2 | 0 | 1 | 1 | 2 | 0 |
| PUZ- | 38 | 38 | 23 | 42 | 41 | 49 | 21 |
| REL- | 2 | 2 | 2 | 1 | 1 | 3 | 0 |
| RNG- | 4 | 4 | 4 | 3 | 3 | 7 | 0 |
| ROB- | 1 | 1 | 1 | 0 | 1 | 1 | 2 |
| SCT- | 2 | 2 | 2 | 7 | 3 | 2 | 7 |
| SET- | 40 | 38 | 33 | 30 | 29 | 58 | 35 |
| SEV- | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| SWB- | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SWC- | 61 | 61 | 39 | 41 | 46 | 61 | 38 |
| SWV- | 69 | 58 | 59 | 119 | 119 | 151 | 68 |
| SWW- | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SYN- | 161 | 166 | 148 | 185 | 154 | 214 | 149 |
| SYO- | 0 | 1 | 1 | 4 | 9 | 10 | 2 |
| TOP- | 4 | 4 | 3 | 3 | 4 | 4 | 3 |
| TOTAL- | 671 | 651 | 537 | 806 | 816 | 1132 | 495 |
| AGT+ | 10 | 15 | 15 | 8 | 10 | 15 | 8 |
| ALG+ | 169 | 167 | 168 | 167 | 168 | 169 | 171 |
| CAT+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| COM+ | 0 | 30 | 30 | 2 | 2 | 32 | 6 |
| CSR+ | 108 | 120 | 103 | 110 | 110 | 164 | 35 |
| GEO+ | 49 | 129 | 136 | 53 | 52 | 148 | 93 |
| GRA+ | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| GRP+ | 2 | 2 | 2 | 6 | 1 | 8 | 1 |
| HAL+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HWV+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| KLE+ | 3 | 6 | 6 | 4 | 4 | 12 | 0 |
| KRS+ | 9 | 7 | 9 | 66 | 66 | 66 | 42 |
| LAT+ | 17 | 16 | 10 | 21 | 19 | 22 | 5 |
| LCL+ | 21 | 20 | 20 | 31 | 31 | 32 | 21 |
| MED+ | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| MGT+ | 22 | 30 | 20 | 21 | 20 | 32 | 10 |
| MSC+ | 3 | 4 | 4 | 6 | 5 | 7 | 3 |
| NLP+ | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| NUM+ | 10 | 59 | 60 | 19 | 18 | 71 | 62 |
| PLA+ | 6 | 6 | 6 | 6 | 6 | 6 | 0 |
| PRO+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PUZ+ | 17 | 20 | 20 | 19 | 20 | 22 | 17 |
| REL+ | 1 | 1 | 1 | 1 | 1 | 2 | 0 |
| RNG+ | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| SCT+ | 2 | 2 | 2 | 6 | 5 | 6 | 7 |
| SET+ | 28 | 32 | 34 | 26 | 22 | 49 | 57 |
| SEU+ | 17 | 14 | 13 | 25 | 11 | 40 | 70 |
| SEV+ | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| SWB+ | 24 | 22 | 22 | 25 | 28 | 34 | 14 |
| SWC+ | 52 | 81 | 80 | 81 | 40 | 104 | 46 |
| SWV+ | 64 | 66 | 66 | 5 | 5 | 67 | 121 |
| SWW+ | 6 | 6 | 6 | 6 | 7 | 7 | 7 |
| SYN+ | 164 | 164 | 164 | 177 | 177 | 179 | 173 |
| SYO+ | 1 | 2 | 3 | 7 | 7 | 10 | 2 |
| TOP+ | 3 | 1 | 1 | 2 | 1 | 3 | 2 |
| TOTAL+ | 809 | 1023 | 1005 | 901 | 837 | 1311 | 978 |
| Problem | ClausalAll<br>archSAT | Equiv(ClausalAll) | Equiv(Id) | Presat(Id) | Presat(ClausalAll) | Any | No rewriting |
| TOTAL | 1480 | 1674 | 1542 | 1707 | 1653 | 2443 | 1473 |

| iProverModulo<br>ClausalAll | Equiv(Clausal | Equiv(Id) | Presat(Id) | Presat(Clausa | Any | No rewriting | Problem |
|---|---|---|---|---|---|---|---|
| 43 | 43 | 39 | 44 | 46 | 52 | 37 | ALG- |
| 19 | 18 | 16 | 23 | 24 | 29 | 18 | ANA- |
| 10 | 9 | 6 | 33 | 8 | 39 | 5 | BOO- |
| 19 | 19 | 22 | 27 | 28 | 35 | 14 | CAT- |
| 47 | 46 | 49 | 55 | 46 | 59 | 47 | COL- |
| 8 | 8 | 7 | 7 | 8 | 8 | 7 | COM- |
| 105 | 104 | 105 | 67 | 25 | 111 | 104 | FLD- |
| 74 | 74 | 73 | 66 | 59 | 86 | 65 | GEO- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | GRA- |
| 100 | 100 | 100 | 187 | 102 | 193 | 102 | GRP- |
| 19 | 19 | 18 | 27 | 11 | 36 | 18 | HEN- |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | HWC- |
| 45 | 47 | 46 | 23 | 23 | 49 | 26 | HWV- |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | KRS- |
| 26 | 38 | 36 | 55 | 40 | 66 | 17 | LAT- |
| 209 | 200 | 310 | 276 | 235 | 387 | 255 | LCL- |
| 0 | 0 | 0 | 5 | 3 | 5 | 0 | LDA- |
| 35 | 35 | 35 | 30 | 29 | 35 | 35 | MGT- |
| 12 | 12 | 12 | 12 | 12 | 12 | 12 | MSC- |
| 15 | 15 | 15 | 15 | 15 | 15 | 15 | NLP- |
| 32 | 32 | 25 | 23 | 24 | 33 | 24 | NUM- |
| 30 | 30 | 32 | 33 | 32 | 33 | 32 | PLA- |
| 56 | 57 | 56 | 55 | 55 | 58 | 54 | PUZ- |
| 0 | 0 | 0 | 4 | 1 | 4 | 0 | REL- |
| 18 | 18 | 16 | 13 | 12 | 21 | 12 | RNG- |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | ROB- |
| 23 | 19 | 18 | 16 | 21 | 24 | 23 | SCT- |
| 212 | 216 | 190 | 95 | 107 | 228 | 174 | SET- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | SEV- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | SWB- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | SWC- |
| 223 | 199 | 184 | 209 | 222 | 250 | 196 | SWV- |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | SWW- |
| 601 | 601 | 601 | 600 | 600 | 602 | 600 | SYN- |
| 3 | 4 | 13 | 14 | 10 | 15 | 12 | SYO- |
| 5 | 5 | 5 | 5 | 4 | 5 | 5 | TOP- |
| 2000 | 1980 | 2041 | 2032 | 1814 | 2504 | 1920 | TOTAL- |
| 17 | 9 | 9 | 18 | 18 | 18 | 18 | AGT+ |
| 14 | 19 | 13 | 22 | 20 | 23 | 13 | ALG+ |
| 2 | 2 | 2 | 0 | 0 | 2 | 2 | CAT+ |
| 14 | 10 | 12 | 16 | 16 | 20 | 15 | COM+ |
| 210 | 203 | 215 | 232 | 203 | 319 | 230 | CSR+ |
| 255 | 255 | 254 | 225 | 197 | 263 | 255 | GEO+ |
| 3 | 0 | 3 | 3 | 3 | 4 | 4 | GRA+ |
| 7 | 6 | 7 | 14 | 7 | 14 | 7 | GRP+ |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | HAL+ |
| 3 | 0 | 3 | 6 | 3 | 6 | 3 | HWV+ |
| 0 | 0 | 0 | 26 | 13 | 29 | 3 | KLE+ |
| 90 | 77 | 76 | 103 | 93 | 148 | 111 | KRS+ |
| 14 | 13 | 20 | 21 | 15 | 23 | 21 | LAT+ |
| 55 | 46 | 45 | 82 | 82 | 87 | 52 | LCL+ |
| 9 | 9 | 9 | 8 | 6 | 9 | 9 | MED+ |
| 36 | 35 | 36 | 29 | 29 | 36 | 36 | MGT+ |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | MSC+ |
| 15 | 15 | 15 | 15 | 15 | 15 | 15 | NLP+ |
| 164 | 126 | 154 | 170 | 168 | 231 | 163 | NUM+ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | PLA+ |
| 11 | 11 | 11 | 11 | 10 | 11 | 11 | PRO+ |
| 9 | 4 | 7 | 13 | 13 | 15 | 8 | PUZ+ |
| 0 | 0 | 0 | 5 | 1 | 5 | 0 | REL+ |
| 35 | 34 | 32 | 41 | 13 | 48 | 32 | RNG+ |
| 11 | 13 | 16 | 18 | 13 | 22 | 16 | SCT+ |
| 209 | 209 | 228 | 173 | 169 | 285 | 184 | SET+ |
| 160 | 146 | 165 | 188 | 170 | 252 | 171 | SEU+ |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | SEV+ |
| 34 | 20 | 32 | 45 | 44 | 51 | 38 | SWB+ |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | SWC+ |
| 172 | 134 | 174 | 79 | 158 | 194 | 176 | SWV+ |
| 41 | 25 | 48 | 34 | 25 | 77 | 49 | SWW+ |
| 272 | 267 | 269 | 269 | 269 | 273 | 272 | SYN+ |
| 5 | 4 | 4 | 6 | 6 | 6 | 4 | SYO+ |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | TOP+ |
| 1879 | 1704 | 1870 | 1883 | 1790 | 2498 | 1929 | TOTAL+ |
| ClausalAll<br>iProverModulo | Equiv(Clausal | Equiv(Id) | Presat(Id) | Presat(Clausa | Any | No rewriting | Problem |
| 3879 | 3684 | 3911 | 3915 | 3604 | 5002 | 3849 | TOTAL |

Spreadsheet file available at http://www.ensiie.fr/~guillaume.burel/download/IJCAR2020.ods