Polarized Rewriting and Tableaux in B Set Theory

Olivier Hermant

Polarized Rewriting and Tableaux in B Set Theory

Olivier Hermant

MINES ParisTech, PSL University, France
olivier.hermant@mines-paristech.fr

Abstract. We propose and extension of the tableau-based first-order automated theorem prover Zenon Modulo to polarized rewriting. We introduce the framework and explain the potential benefits. The first target is an industrial benchmark composed of B Set Theory problems.

1 Introduction

The B Method set theory [1] has been extensively used for 20 years by the railway industry in France to develop certified correct-by-construction software. Recently, the BWare [18] project has tackled the issue of automatically proving the thousands of *proof obligations* generated by the development process.

Zenon Modulo [11] is one of the tools developed to this aim. Originally a tableau-based prover, Zenon [4] is used for instance by TLA+ [10] and FoCaLiZe [14]. To help manage the axioms of set theory, but also the uncountable derived constructs definitions (e.g. inclusion, union, functions), we deemed useful to not let nonlogical axioms wander as formulas: a prover would easily get lost by decomposing one or another axiom in an unorganized fashion.

We replaced them with rewrite rules, turning Zenon into an implementation of Deduction Modulo Theory [3,5,13], which allows rewriting on terms and *formulas*. Additionally, we equipped it with ML polymorphic types and arithmetic [8]. On the BWare benchmark, the success rate was raised from 2.5% to 95%.

We propose to extend Zenon Modulo with *polarized* rewriting, a more permissive rewrite relation. We first introduce the framework, then we discuss examples and the pros and cons of the approach. There is currently no implementation, essentially because this is a perfect match for an intern or a PhD student.

2 Polarized Tableaux Modulo Theory

We assume familiarity with first-order logic and at least one deduction system. Tableaux calculus is a refutational calculus, thus, to show F under the assumptions Γ, we refute $\Gamma, \neg F$. The first-order tableaux rules are recalled in Figure 1, see textbooks [16] for details. The rules have the following characteristics:

- as customary, they are presented in a top-down fashion.
- Formulas are not in negation normal form, rules are duplicated.

– A branch may be closed, denoted $\odot$, if we find on it (including internal nodes) an occurrence of some F and its negation, or an explicit contradiction. A tableau is a proof iff each branch is closed.
– α-rules are for non-branching connectives rules and β-rules for branching ones, δ-rules are for quantifier rules introducing a fresh constant c and γ-rules for those introducing any term.

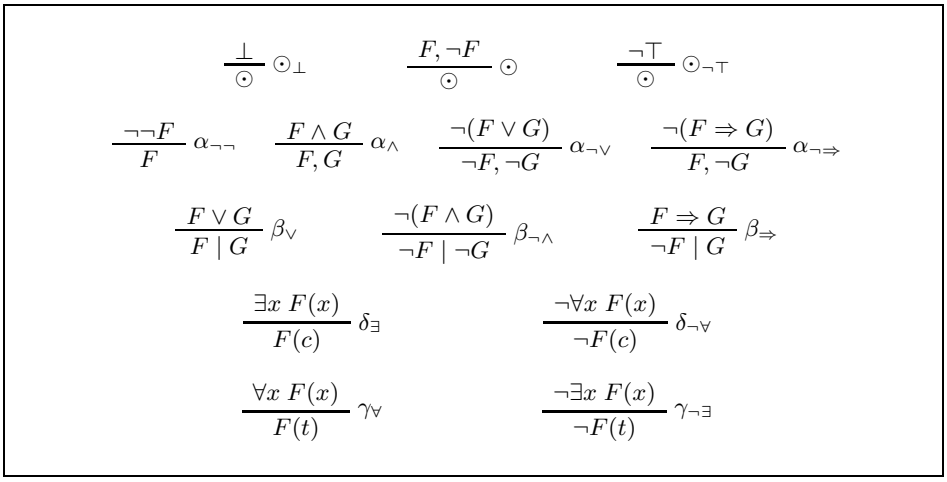


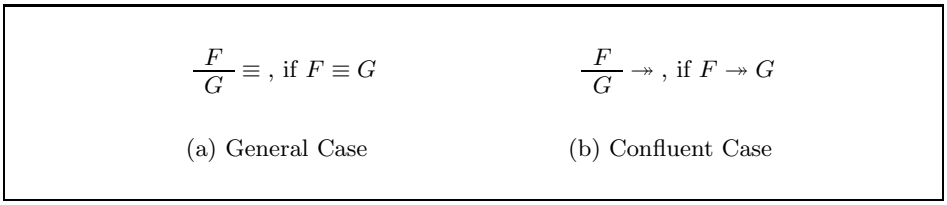
Fig. 1: Tableaux Rules

Tableaux Modulo Theory [3] extends tableaux with a set of rewrite rules $\mathcal{R}$. A rewrite rule is a pair of terms, $l \to r$, where the variables of r appear in l. Given a set $\mathcal{R}$, a term t rewrites into u, denoted $t \to u$, if there is a rule $l \to r \in \mathcal{R}$ and a substitution σ, such that there is an occurrence of $l\sigma$ in t, and u is t where that occurrence has been replaced with $r\sigma$. In other words, $\to$ is the closure of $\mathcal{R}$ by substitution and the subterm relation. The transitive closure of $\to$ is denoted $\twoheadrightarrow$ and its further reflexive-symmetric closure is $\equiv$, which is a congruence.

Deduction Modulo Theory also allows rewrite rules on *formulas*, provided the left member P of such a rule $P \to F$ is atomic. The relations $\to, \twoheadrightarrow, \equiv$ on formulas embed their counterparts on the subterms of the formulas.

Tableaux can be extended to rewriting with the addition of a rule allowing to convert any formula with $\equiv$, as in Figure 2a. When rewriting is confluent, we can orient this rule as in Figure 2b. In practice, Deduction Modulo Theory-based automated theorem provers [6] implement this last rule, which is a way to decide $\equiv$ when confluence holds. Other presentations exist [3,5,8].

The calculus of Zenon Modulo [8] enjoys meta-variables, Hilbert's ϵ operator, reasoning over reflexive/transitive/symmetric relations, an equality predicate, ML-polymorphic types, and, of course, rewriting. The simpler case of Figure 1 is sufficient here, as we focus on rewriting, that we now extend to polarity.



(a) General Case (b) Confluent Case

Fig. 2: The Additional Rule of Tableaux Modulo Theory

Definition 1 (Polarity of an Occurrence). *The occurrence of a formula F in a formula G is positive (resp. negative) iff*

- *G is F,*
- *G is $G_1 \wedge G_2$, $G_1 \vee G_2$, $\forall x G_1$, $\forall x G_1$ or $H \Rightarrow G_1$ and the occurence of F in G_1 or G_2 is positive (resp. negative),*
- *G is $\neg G_1$ or $G_1 \Rightarrow H$ and the occurrence of F in G_1 is negative (resp. positive).*

Now, we consider two (proposition) rewrite systems $\mathcal{R}^+ \cup \mathcal{R}^-$.

Definition 2 (Polarized Rewrite Relation). *Let F and G be two formulas. $F \rightarrow_+ G$ iff $F \rightarrow G$ with a term rewrite rule or there exists a positive (resp. negative) occurrence H in F, a substitution σ, and a rule $l \rightarrow r \in \mathcal{R}^+$ (resp. $\mathcal{R}^-$), such that $H = l\sigma$ and G is F where H has been replaced with $r\sigma$.*

$F \rightarrow_- G$ iff $\neg F \rightarrow_+ \neg G$, that is to say we exchange $\mathcal{R}^+$ and $\mathcal{R}^-$ above.

We denote by $\twoheadrightarrow_+$ and $\twoheadrightarrow_-$ the reflexive-transitive closures of $\rightarrow_+$ and $\rightarrow_-$, respectively. Defining $\equiv_+$ and $\equiv_-$ is more delicate and unlikely to be useful practice. Polarized Tableaux combine the rules of Figure 1 and Figure 3.

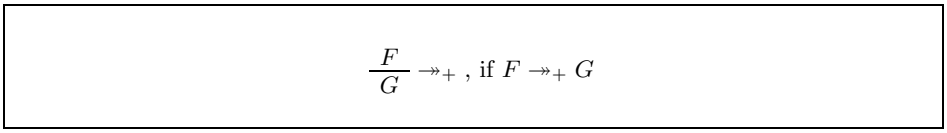


Fig. 3: The Additional Rule of Polarized Tableaux Modulo Theory

3 Implementation

Zenon Modulo rewrites only literals, in a forward fashion. This is a further restriction of Figure 2b and it relies on termination of term rewriting and on confluence of the whole rewriting. Otherwise, completeness of the proof search fails. The heuristic is, each time we meet a literal, to:

1. normalize the terms it contains;

2. rewrite the literal itself (if there is an applicable rewrite rule) on *one step*;
3. if the formula is in normal form or compound, stop, otherwise repeat.

To get polarized rewriting it suffices to modify the second step into "rewrite positively if the literal is positive, negatively otherwise". The expected gain does not lie here, but in an *optimized preprocessing* for rules. Indeed [12], a polarized rule $P \rightarrow_+ F \in \mathcal{R}^+$ represents/can be represented as an axiom $\overline{\forall}(P \Rightarrow F)$ ($\overline{\forall}$ is the universal closure over the free variables). Similarly, a negative rewrite rule $P \twoheadrightarrow_- F \in \mathcal{R}^-$ is equivalent to the axiom $\overline{\forall}(F \Rightarrow P)$. In contrast, Deduction Modulo Theory's rewrite rules $P \rightarrow F$ are equivalent to $\overline{\forall}((P \Leftrightarrow F)$ [13]. Remind that we are discussing propositional rewrite rules, so P has to be atomic. Consequently, polarization offers the following improvements:

- more axioms correspond to rewrite rules, and this improves proof search [11]. Axioms of the form $\forall\overline{x}(P \Rightarrow A)$ and $\forall\overline{x}(A \Rightarrow P)$, with P atomic, become rules of $\mathcal{R}^+$ and $\mathcal{R}^-$, respectively. In classical logic, when P is a negated atom, we also get rewrite rules in $\mathcal{R}^-$ and $\mathcal{R}^+$, respectively.
- We can *Skolemize* rewrite rules. This has two benefits: first, less inference rules are necessary in the tableaux, and second, the Skolem term is *uniform*, while multiple applications of $\delta_\exists$ or $\delta_{\neg\forall}$ introduce different fresh symbols at each time. This also holds in the presence of meta-variables.

Skolemizing the rules is impossible in vanilla Deduction Modulo Theory, as rewriting applies at positive and negative occurrences. Therefore, we do not know in advance which quantifiers are positive and negative. To illustrate the difference, consider axioms of the type $\forall\overline{x}(P \Rightarrow A)$ and $\forall\overline{x}(A \Rightarrow P)$.

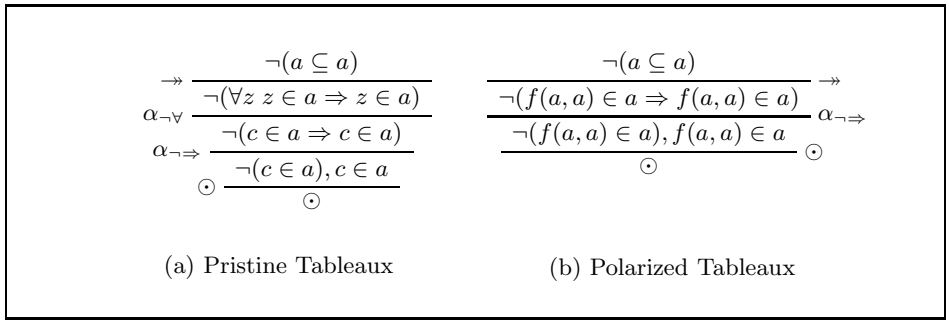
- In $\forall\overline{x}(P \Rightarrow A)$, we can replace all the positive existential and negative universal quantifiers of A by a Skolem function symbol.
- Similarly, in $\forall\overline{x}(A \Rightarrow P)$, we can replace all the positive universal and negative existential quantifiers of A by a Skolem function symbol.

The very same principle applies to polarized rewrite rules. We leave the study and the choice of the strategies for Skolemization [17] for a later stage. Both improvements can be applied to heuristics turning assumptions (of a given problem) into rewrite rules, and to hand-tuning of the rewrite rules of a specific theory, for instance B Method set theory.

4 Example

Let us consider the classical example of proving $a \subseteq a$ with the standard axiom of inclusion $\forall x \forall y\ x \subseteq y \Leftrightarrow (\forall z\ z \in x \Rightarrow z \in y)$. A usual tableau proof involves the succession of rules $\gamma_\forall$ (twice), $\alpha_\wedge$, $\beta_\Rightarrow$, $\delta_{\neg\forall}$ and $\alpha_{\neg\Rightarrow}$ on the axiom. Deduction Modulo Theory turns it into the rewrite rule $x \subseteq y \rightarrow (\forall z\ z \in x \Rightarrow z \in y)$, and yields the 3-rules axiomless proof of Figure 4a.

If we switch to Polarized Deduction Modulo Theory, we get the pair $\mathcal{R}^+ = \{x \subseteq y \rightarrow (\forall z z \in x \Rightarrow z \in y)\}$ and $\mathcal{R}^- = \{x \subseteq y \rightarrow (f(x,y) \in x \Rightarrow f(x,y) \in y)\}$. The proof of $a \subseteq a$ is one more step smaller, as shown in Figure 4b.



(a) Pristine Tableaux (b) Polarized Tableaux

Fig. 4: Proof of $a \subseteq a$ in Deduction Modulo Theory

5 Conclusion

We expect the polarized approach to give at least as efficient as Zenon Modulo itself. The proof-search algorithm needs only few changes, mostly in the rule pre-processing. The obtained rules contain less quantifiers, allowing for fewer rules in proof-search and potential earlier unification and branch closure, since using a rewrite rule several times now involves the *same* Skolem symbol.

On the risk side, implicational axioms can now be turned into rewrite rules. This might be a threat to termination or confluence. A study of the theoretical framework, including models, is required.

Automated theorem provers are aggressively optimized tools, naturally lending themselves to bugs. This is why independent double checking facilities are important. Zenon Modulo is able to produce *proof-terms* or *proof certificates*, though it provides no rewrite steps explicitly, following Poincaré's Principle: computations (rewriting) in proofs give no insight, they can be quickly reconstructed (by the checker) at will and are to be left implicit. Such a clerk/expert distinction has for instance been studied in the Foundational Proof Certificate project [15], at the proof level, with the help of focusing [9].

On the BWare benchmark, all statements proved by Zenon Modulo [8], that do not involve arithmetic, are actually declared well-typed by Dedukti [2], a type checker based on an extension of Deduction Modulo Theory to dependent types. Dedukti's rewriting ability made extremely smooth the reconstruction of rewriting : there is essentially nothing to do but to declare the rules.

The challenge is to keep this skeptical double-checking approach viable. We may need a *depolarization* of the proofs, perhaps following [7], or an substantial extension of Dedukti and its foundations to polarized rewriting, perhaps with the help of subtyping.

References

1. Abrial, J.R.: The B-book: Assigning Programs to Meanings. Cambridge University Press, New York, NY, USA (1996)

2. Boespflug, M., Carbonneaux, Q., Hermant, O.: The $\lambda\Pi$-calculus modulo as a universal proof language. In: PxTP 2012, Proceedings. vol. 878, pp. 28–43. CEUR-WS.org (2012)
3. Bonichon, R.: TaMeD : A tableau method for deduction modulo. IJCAR 2004 (2004)
4. Bonichon, R., Delahaye, D., Doligez, D.: Zenon : An extensible automated theorem prover producing checkable proofs. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007, Yerevan, Armenia. LNCS, vol. 4790, pp. 151–165. Springer (2007)
5. Bonichon, R., Hermant, O.: A semantic completeness proof for tableaux modulo. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS, vol. 4246, pp. 167–181. Springer-Verlag, Phom Penh, Cambodia (2006)
6. Burel, G.: Embedding deduction modulo into a prover. In: Dawar, A., Veith, H. (eds.) CSL. LNCS, vol. 6247, pp. 155–169. Springer (2010)
7. Burel, G., Kirchner, C.: Regaining cut admissibility in deduction modulo using abstract completion. Information and Computation 208(2), 140–164 (2010)
8. 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 2015 – Short presentations, Suva, Fiji. EPiC Series in Computing, vol. 35, pp. 42–58. EasyChair (2015)
9. Chihani, Z., Miller, D., Renaud, F.: A semantic framework for proof evidence. J. Autom. Reasoning 59(3), 287–330 (2017)
10. Cousineau, D., Doligez, D., Lamport, L., Merz, S., Ricketts, D., Vanzetto, H.: TLA + proofs. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012, Paris, France. LNCS, vol. 7436, pp. 147–154. Springer (2012)
11. Delahaye, D., Doligez, D., Gilbert, F., Halmagrand, P., Hermant, O.: Zenon modulo: When achilles outruns the tortoise using deduction modulo. In: McMillan, K., Middledorp, A., Voronkov, A. (eds.) LPAR 2013. LNCS ARCoSS, vol. 8312, pp. 274–290. Springer (2013)
12. Dowek, G.: Polarized deduction modulo. In: IFIP Theoretical Computer Science (2010)
13. Dowek, G., Hardin, T., Kirchner, C.: Theorem proving modulo. Journal of Automated Reasoning 31, 33–72 (2003)
14. Dubois, C., Rioboo, R.: Verified functional iterators using the focalize environment. In: Giannakopoulou, D., Salaün, G. (eds.) SEFM 2014, Grenoble, France. Lecture Notes in Computer Science, vol. 8702, pp. 317–331. Springer (2014)
15. Miller, D.: Foundational proof certificates. In: Delahaye, D., Woltzenlogel Paleo, B. (eds.) All about Proofs, Proofs for All, Studies in Logic, Mathematical Logic and Foundations, vol. 55, chap. 8, pp. 150–163. College Publications (2015)
16. Nerode, A., Shore, R.A.: Logic for Applications. Springer (1993)
17. Nonnengart, A., Weidenbach, C.: Computing small clause normal forms. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 335–367. Elsevier and MIT Press (2001)
18. The BWare Project: (2012), http://bware.lri.fr/