Automated Deduction in the B Set Theory using Deduction Modulo[⋆]

Guillaume Burel[1], David Delahaye[2], Damien Doligez[3], Pierre Halmagrand[2], and Olivier Hermant[4]

[1] Cedric/ENSIIE/Inria, Évry, France,
guillaume.burel@ensiie.fr
[2] Cedric/Cnam/Inria, Paris, France,
David.Delahaye@cnam.fr
Pierre.Halmagrand@inria.fr
[3] Inria, Paris, France,
Damien.Doligez@inria.fr
[4] CRI, MINES ParisTech, Fontainebleau, France,
olivier.hermant@mines-paristech.fr

Abstract. We introduce a new encoding of the set theory of the B method based on deduction modulo. The theory of deduction modulo is an extension of predicate calculus that includes rewriting on both terms and propositions, which is well suited for proof search in axiomatic theories, as it turns many axioms into rewrite rules. We also present Zenon Modulo and iProver Modulo, two automated theorem provers that rely on deduction modulo and that are intended to deal with this B set theory modulo. These two tools have backends based on the Dedukti universal proof checker, which also relies on deduction modulo, and which allow us to certify the correctness of the proofs produced by these two tools. Finally, we provide some experimental results obtained on a benchmark consisting of derived properties of the B set theory, which show a significant gain of the tools based on deduction modulo compared to other first order automated theorem provers. In addition, to show the effectiveness of our approach, we describe an example of proof, whose proof is only found by the tools based on deduction modulo.

Keywords: Automated Theorem Proving, Deduction Modulo, Set Theory, B Method, Zenon Modulo, iProver Modulo, Dedukti.

1 Introduction

Reasoning within theories has become a crucial point in automated theorem proving, whether the theory under consideration is decidable or not. Actually, a theory, commonly formulated as a collection of axioms, is often necessary to specify, in a concise and understandable way, the properties of objects that we

[⋆] This work is supported by the BWare project [10] (ANR-12-INSE-0010) funded by the INS programme of the French National Research Agency (ANR).

are reasoning about in software proofs, such as lists or arrays. Each theory has its own features and specificities, but a small number of them appear recurrently, among which arithmetic and set theory. For example, the heart of the B method relies on a customized set theory [1], and this theory is supported by some tool sets, such as Atelier B, which are successfully used in industry to specify and build, by stepwise refinements, software that is correct by design.

The Atelier B tool set still lacks automation: it comes with built-in automated theorem provers, but in general, a lot of proof obligations, often generated during the refinement process, are left to the user, who must discharge them using the interactive theorem prover or by means of new proof rules that can be exploited by the automated theorem provers and that must be verified at a later stage. Due to the large practical impact of the B method, the BWare project [10] has committed itself to solve this issue, by providing a proof platform with several automated theorem provers that aim to support the verification of proof obligations coming from the development of industrial applications.

When dealing with proofs, leaving the axioms and definitions of set theory wandering among the hypotheses is not a reasonable option: first, it induces a combinatorial explosion in the proof search space and second, axioms do not bear any specific meaning that an automated theorem prover can take advantage of. To avoid these drawbacks, we propose to follow the lines of deduction modulo [13] and replace axioms by rewrite rules. Deduction modulo is a framework that combines a first-order proof system with rewrite rules on terms and on propositions. This latter possibility is a characteristic feature: with propositional rewrite rules, we can go beyond pure first order reasoning without using any axiom. As a counterpart, we must take great care of remaining in a theory that enjoys some properties, such as consistency, cut elimination, and where proof search methods are complete.

In this paper, we propose a new encoding of the set theory of the B method based on deduction modulo, which is formulated as a rewrite system rather than a set of axioms. As deduction modulo has been successfully applied to well-known proof search methods in classical first order logic, such as the tableau and resolution methods [4, 13], we also provide this new encoding with two automated theorem provers that are based on these methods and that rely on deduction modulo. These tools are Zenon Modulo [9] and iProver Modulo [6], which are respectively based on tableaux and resolution, and which are respectively extensions of the Zenon [5] and iProver [14] automated theorem provers to deduction modulo. In addition, both of these tools have backends based on the Dedukti universal proof checker [3], which also relies on deduction modulo, and which allow us to certify the correctness of the proofs produced by these two tools.

The paper is organized as follows: in Sec. 2, after an overview of deduction modulo, we introduce our formulation of the B set theory as a rewrite system; we then present, in Sec. 3, the Zenon Modulo and iProver Modulo automated theorem provers together with their backends based on Dedukti; finally, in Sec. 4, we describe our implementation and the experimental results obtained on a benchmark consisting of derived properties of the B set theory.

2 A Theory Modulo for the B Set Theory

In this section, we give an overview of deduction modulo and present the theory modulo that has been elaborated for the set theory of the B method and that is intended to be used by the two automated theorem provers described in Sec. 3.

2.1 From Axioms to Deduction Modulo

Deduction modulo [13] focuses on the computational part of a theory, where axioms are transformed into rewrite rules, which induces a congruence over propositions, and where reasoning is performed modulo this congruence. For example, considering the inclusion in set theory $\forall X, Y\ (X \subseteq Y \Leftrightarrow \forall x\ (x \in X \Rightarrow x \in Y))$, the proof of $A \subseteq A$ in sequent calculus has the following form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, x \in A \vdash A \subseteq A, x \in A}\,\text{Ax}}{\ldots \vdash A \subseteq A, x \in A \Rightarrow x \in A}\,\Rightarrow\!\text{R}}{\ldots \vdash A \subseteq A, \forall x\ (x \in A \Rightarrow x \in A)}\,\forall\text{R} \qquad \dfrac{}{\ldots, A \subseteq A \vdash A \subseteq A}\,\text{Ax}}{\ldots, \forall x\ (x \in A \Rightarrow x \in A) \Rightarrow A \subseteq A \vdash A \subseteq A}\,\Rightarrow\!\text{L}}{A \subseteq A \Leftrightarrow \forall x\ (x \in A \Rightarrow x \in A) \vdash A \subseteq A}\,\wedge\text{L}}{\forall X, Y\ (X \subseteq Y \Leftrightarrow \forall x\ (x \in X \Rightarrow x \in Y)) \vdash A \subseteq A}\,\forall\text{L} \times 2$$

In deduction modulo, the previous axiom of inclusion is replaced by the rewrite rule $X \subseteq Y \longrightarrow \forall x\ (x \in X \Rightarrow x \in Y)$, and the proof of $A \subseteq A$ becomes:

$$\dfrac{\dfrac{\dfrac{}{x \in A \vdash x \in A}\,\text{Ax}}{\vdash x \in A \Rightarrow x \in A}\,\Rightarrow\!\text{R}}{\vdash A \subseteq A}\,\forall\text{R},\ A \subseteq A \longrightarrow \forall x\ (x \in A \Rightarrow x \in A)$$

where it can be seen that computation (rewriting) is interleaved with the deduction rules and has been discharged from the proof tree. The resulting proof is simpler than the proof with axioms and in addition, shorter.

2.2 Rules for a B Set Theory Modulo

Let us first introduce the notion of class rewrite system as follows (in this definition, FV refers to the function that returns the set of free variables of a formula):

Definition 1 (Class Rewrite System). *A term rewrite rule is a pair of terms denoted by $l \longrightarrow r$, where $\text{FV}(r) \subseteq \text{FV}(l)$. An equational axiom is a pair of terms denoted by $l = r$. A proposition rewrite rule is a pair of propositions denoted by $l \longrightarrow r$, where l is an atomic proposition and r is an arbitrary proposition, and where $\text{FV}(r) \subseteq \text{FV}(l)$.*

A class rewrite system is a pair, denoted by $\mathcal{RE}$, consisting of:

- *$\mathcal{R}$: a set of proposition rewrite rules;*
- *$\mathcal{E}$: a set of term rewrite rules and equational axioms.*

Given a class rewrite system $\mathcal{RE}$, the relations $=_{\mathcal{E}}$ and $=_{\mathcal{RE}}$ denote the congruences generated respectively by the sets $\mathcal{E}$ and $\mathcal{R} \cup \mathcal{E}$.

Expressing the B set theory in this framework therefore amounts to build a class rewrite system $\mathcal{RE}$ for this theory. To do so, we turn whenever possible the axioms and definitions of Chap. 2 of the B-Book [1] into rewrite rules to build the sets $\mathcal{R}$ and $\mathcal{E}$ (in the same way than in the previous example regarding the inclusion in set theory). It turns out that we do not need equational axioms, so that $\mathcal{RE}$ is a pure rewrite system. The resulting theory is summarized in Fig. 1, where π_1 and π_2 are the projectors for tuples (these constructs do not exist in the initial theory of the B-Book and have been added to ease automated deduction when dealing with relations), and where we omit the BIG set (an arbitrary infinite set, only used to build natural numbers in the foundational theory), the sets defined in extension, and the $\mathbb{P}_1$ construct (whose definition uses a set defined in extension).

As can be observed, we only consider first order constructs of the B set theory. This means that we do not deal with sets defined by comprehension (Axiom SET 3 of the B-Book) or lambda abstractions. As for the set constructs using comprehension, we transform them into comprehension-free axioms. For example, given the sets s, t, and u, the following definition of the union construct:

$$s \cup t \mathrel{\hat{=}} \{a \mid a \in u \wedge (a \in s \vee a \in t)\}$$

is transformed into an axiom as follows:

$$x \in s \cup t \Leftrightarrow x \in u \wedge (x \in s \vee x \in t)$$

which is then turned into the following rewrite rule:

$$x \in s \cup t \longrightarrow x \in u \wedge (x \in s \vee x \in t)$$

Another remark is that the typing information of the axioms and definitions was removed when it appears to be redundant, which allows us to make automated deduction easier. For instance, this is the case for the union construct whose definition is given under the assumptions that the sets s and t are included in the set u (which means that s and t are of type u). Under these assumptions, it seems clear that the set of elements a s.t. $a \in u \wedge (a \in s \vee a \in t)$ is equal to the set of elements a s.t. $a \in s \vee a \in t$ (where the explicit condition $a \in u$ has been removed). The rewrite rule for the union construct is then simply as follows:

$$x \in s \cup t \longrightarrow x \in s \vee x \in t$$

It should also be noted that the initial definition of function evaluation in the B-Book is a conditional definition. Until recently, it was not possible to handle this kind of definition in deduction modulo as it involves conditional rewrite rules over terms. However, a recent work [?] shows how to consider such conditional rewrite rules, but has not been implemented yet for the two automated theorem provers based on deduction modulo and described in Sec. 3. This definition is therefore currently formulated as the following regular axiom:

$$\forall s, t, f, x \ (f \in s \mathrel{+\!\!\!\rightarrow} t \wedge x \in \mathsf{dom}(f) \Rightarrow f(x) = \mathsf{choice}(f[\{x\}]))$$

Tuple Operations

$\pi_1(x,y) \longrightarrow x$

$\pi_2(x,y) \longrightarrow y$

$(x,y) = z \longrightarrow \pi_1 z = x \wedge \pi_2 z = y$

$z = (x,y) \longrightarrow \pi_1 z = x \wedge \pi_2 z = y$

Axioms of Set Theory

$x \in s \times t \longrightarrow \pi_1 x \in s \wedge \pi_2 x \in t$

$s = t \longrightarrow \forall x\ (x \in s \Leftrightarrow x \in t)$

$s \in \mathbb{P}(t) \longrightarrow \forall x\ (x \in s \Rightarrow x \in t)$

$\mathsf{choice}(s) \in s \longrightarrow \exists x\ (x \in s)$

Set Inclusion

$s \subseteq t \longrightarrow s \in \mathbb{P}(t)$

$s \subset t \longrightarrow s \subseteq t \wedge s \neq t$

Derived Constructs

$x \in s \cup t \longrightarrow x \in s \vee x \in t$

$x \in s - t \longrightarrow x \in s \wedge x \notin t$

$x \in \emptyset \longrightarrow \bot$

$x \in \{a\} \longrightarrow x = a$

$x \in s \cap t \longrightarrow x \in s \wedge x \in t$

$x \in \{E, L\} \longrightarrow x = E \vee x \in \{L\}$

$\mathbb{P}_1(s) \longrightarrow \mathbb{P}(s) - \{\emptyset\}$

Binary Relation Constructs: First Series

$p \in u \leftrightarrow v \longrightarrow \forall x, y\ ((x,y) \in p \Rightarrow x \in u \wedge y \in v)$

$x \in \mathsf{dom}(p) \longrightarrow \exists b\ ((x,b) \in p)$

$x \in p;q \longrightarrow \exists b\ ((\pi_1 x, b) \in p \wedge (b, \pi_2 x) \in q)$

$x \in \mathsf{id}(u) \longrightarrow x \in u \times u \wedge \pi_1 x = \pi_2 x$

$x \in p \rhd t \longrightarrow x \in p \wedge \pi_2 x \in t$

$x \in p \mathbin{\rhd\!\!\!-} t \longrightarrow x \in p \wedge \pi_2 x \notin t$

$x \in p^{-1} \longrightarrow (\pi_2 x, \pi_1 x) \in p$

$x \in \mathsf{ran}(p) \longrightarrow \exists a\ ((a,x) \in p)$

$q \circ p \longrightarrow p;q$

$x \in s \lhd p \longrightarrow x \in p \wedge \pi_1 x \in s$

$x \in s \mathbin{-\!\!\!\lhd} p \longrightarrow x \in p \wedge \pi_1 x \notin s$

Binary Relation Constructs: Second Series

$x \in p[w] \longrightarrow \exists a\ (a \in w \wedge (a,x) \in p)$

$x \in q \mathbin{<\!\!+} p \longrightarrow (x \in q \wedge \pi_1 x \notin \mathsf{dom}(p)) \vee x \in p$

$x \in f \otimes g \longrightarrow (\pi_1 x, \pi_1 \pi_2 x) \in f \wedge (\pi_1 x, \pi_2 \pi_2 x) \in g$

$x \in \mathsf{prj}_1(s,t) \longrightarrow x \in (s \times t) \times s \wedge \pi_2 x = \pi_1 \pi_1 x$

$x \in \mathsf{prj}_2(s,t) \longrightarrow x \in (s \times t) \times t \wedge \pi_2 x = \pi_2 \pi_1 x$

$x \in h \| k \longrightarrow (\pi_1 \pi_1 x, \pi_1 \pi_2 x) \in h \wedge (\pi_2 \pi_1 x, \pi_2 \pi_2 x) \in k$

Function Constructs: First Series

$x \in s \mathrel{+\!\!\!\!\rightarrow} t \longrightarrow x \in s \leftrightarrow t \wedge (x^{-1}; x) \subseteq \mathsf{id}(t)$

$x \in s \rightarrow t \longrightarrow x \in s \mathrel{+\!\!\!\!\rightarrow} t \wedge \mathsf{dom}(x) = s$

$x \in s \mathrel{\rightarrowtail\!\!\!\!\!+} t \longrightarrow x \in s \mathrel{+\!\!\!\!\rightarrow} t \wedge x^{-1} \in t \mathrel{+\!\!\!\!\rightarrow} s$

$x \in s \rightarrowtail t \longrightarrow x \in s \mathrel{\rightarrowtail\!\!\!\!\!+} t \wedge x \in s \rightarrow t$

$x \in s \mathrel{+\!\!\!\!\twoheadrightarrow} t \longrightarrow x \in s \mathrel{+\!\!\!\!\rightarrow} t \wedge \mathsf{ran}(x) = t$

$x \in s \twoheadrightarrow t \longrightarrow x \in s \mathrel{+\!\!\!\!\twoheadrightarrow} t \wedge x \in s \rightarrow t$

$x \in s \mathrel{\rightarrowtail\!\!\!\!\!+\!\!\!\!\twoheadrightarrow} t \longrightarrow x \in s \mathrel{\rightarrowtail\!\!\!\!\!+} t \wedge x \in s \mathrel{+\!\!\!\!\twoheadrightarrow} t$

$x \in s \mathrel{\rightarrowtail\!\!\!\!\twoheadrightarrow} t \longrightarrow x \in s \rightarrowtail t \wedge x \in s \twoheadrightarrow t$

Function Constructs: Second Series

$\forall s, t, f, x\ (f \in s \mathrel{+\!\!\!\!\rightarrow} t \wedge x \in \mathsf{dom}(f) \Rightarrow f(x) = \mathsf{choice}(f[\{x\}]))$

Fig. 1. Rules of the B Set Theory Modulo

This definition must be combined with the axiom of choice (Axiom SET 5), otherwise the term $\mathsf{choice}(f[\{x\}])$ is uninterpreted. However, to use the axiom of choice, we must introduce the proposition $\mathsf{choice}(f[\{x\}]) \in f[\{x\}]$, which allows us to get information over the term $\mathsf{choice}(f[\{x\}])$. This proposition is then proved by means of the axiom of choice, and is generally introduced by a cut (see the proof of Property 2.5.2 in the B-Book for example). The introduction of this cut tends to make automated deduction difficult, and the automated theorem provers should be customized to systematically make this cut once the term $\mathsf{choice}(f[\{x\}])$ has been introduced by a function evaluation. Other alternatives could be to use axioms that are more appropriate for automated deduction (like Properties 2.5.2 and 2.5.4 in the B-Book for instance).

To sum up, our formulation of the B set theory quite sticks to the B-Book, although it is admittedly different in some points. This deserves some comments since we need to justify the equivalence of both formulations (for the BIG and extension-free fragment), at least at the informal level. What we need is inter-derivability in both theories, which can be reduced to cross-derivability of the initial axioms and definitions in our system (completeness) and the propositions associated to our rewrite rules in the initial system (correctness).

It is a mere check that the minor variations over tuples and function evaluation are harmless, the main point being comprehension. Dropping this axiom is hazardous only for completeness, the correctness of the "inlined" rewrite rules that we propose follows from immediate applications of the very comprehension axiom. As for completeness, without further considerations, our formulation might suffer from incompleteness: sometimes, it might be necessary to introduce a set defined by comprehension, like for instance in Cantor's proof that "a set is not in bijection with its power set". However, this kind of smart trick is seldom required in practice, and we can leave this question to a later stage of this work.

3 Use of Deduction Modulo Based Proof Systems

In this section, we present Zenon Modulo and iProver Modulo, two automated theorem provers that are compatible with deduction modulo and that are intended to deal with the B set theory modulo described in Sec. 2. We also introduce the backends of these tools, which are based on the Dedukti universal proof checker relying on deduction modulo as well, and which allow us to ensure the correctness of the proofs produced by these tools.

3.1 The Zenon Modulo Automated Theorem Prover

Zenon Modulo is an adaptation to deduction modulo of Zenon [5], a tableau-based automated theorem prover for first order classical logic with equality. This adaptation is described in [9] and mainly consists in extending the usual rules of Zenon by allowing them to work modulo the congruence relation over propositions induced by rewriting. This extension is partially inspired by the presentation of tableaux modulo in [4].

The adaptation of proof search rules of Zenon to a class rewrite system $\mathcal{RE}$ is summarized in Fig. 2 (for the sake of simplicity, the unfolding and extension rules are omitted, and due to space restrictions, the relational rules are not shown but can be found in [9]), where the "|" symbol is used to separate the propositions of two distinct nodes to be created, ϵ is Hilbert's operator ($\epsilon(x).P(x)$ is a term meaning some x that satisfies $P(x)$), and capital letters are used for metavariables. As hinted by the use of Hilbert's operator, ϵ-terms play the role of Skolem terms. What we call here metavariables are often named free variables in the tableau-related literature. However, those are not used as variables as they are never substituted, and do not even help to generate a global constraint closing all the branches of the tableau; metavariables are instead used as clues (through unification attempts) for the "real" instantiation rules $\gamma\forall_{inst}/\gamma\neg\exists_{inst}$. The proof search rules are applied with the usual tableau method: starting from the negation of the goal, apply the rules in a top-down fashion to build a tree. When all branches are closed (i.e. end with a closure rule), the tree is closed, and this closed tree is a proof of the goal. This algorithm is applied in strict depth-first order: we close the current branch before starting working on another branch. Moreover, we work in a non-destructive way: working on a branch will never change the propositions of another branch.

To find instantiation clues, Zenon Modulo uses the following method: it looks for two formulas P and Q s.t. $P =_{\mathcal{RE}} P'$, $Q =_{\mathcal{RE}} \neg Q'$, and there exists a metavariable unifier σ s.t. $\sigma(P') =_{\mathcal{E}} \sigma(Q')$. This instantiation search is also extended to propositional narrowing (even if we are not trying to close a branch): we look for a formula P and a substitution σ s.t. $P =_{\mathcal{RE}} P'$, and there exist $P'_{|\omega}$ and a rule $l \longrightarrow r$ of $\mathcal{RE}$ s.t. $\sigma(P'_{|\omega}) =_{\mathcal{E}} \sigma(l)$ (where $P'_{|\omega}$ is the term or proposition at the occurrence ω in the proposition P').

The current implementation of Zenon Modulo[5] is still in an early stage of development. In this implementation, the class rewrite system $\mathcal{RE}$ is assumed to be a pure rewrite system, i.e. there are only rewrite rules and no equational axiom in $\mathcal{E}$. In addition, the rewrite system $\mathcal{RE}$ is assumed to be confluent and (weakly) terminating, and the relation $=_{\mathcal{RE}}$ is therefore decidable. The rewrite system $\mathcal{RE}$ is directly fed to Zenon Modulo through an appropriate extension of the TPTP input syntax [15].

3.2 The iProver Modulo Automated Theorem Prover

iProver is an automated theorem prover for first order logic developed by Korovin [14]. It based on a combination of the instantiation-generation method and the resolution method. iProver Modulo is an extension of this prover in which the resolution part has been upgraded into polarized resolution modulo [12], a proof search method based on deduction modulo. This provides a real improvement, as shown by experimenting with problems from the TPTP library [6].

The idea of polarized resolution modulo is to add to the resolution method the possibility to rewrite clauses using term but also proposition rewriting rules.

[5] See: `https://www.rocq.inria.fr/deducteam/ZenonModulo/`.

Closure and Cut Rules

$$\frac{P \qquad \neg Q}{\odot}\,\odot \ \text{if } P =_{\mathcal{RE}} Q \qquad\qquad \frac{}{P \mid \neg Q}\,\text{cut} \ \text{if } P =_{\mathcal{RE}} Q$$

$$\frac{P}{\odot}\,\odot_{\perp} \ \text{if } P =_{\mathcal{RE}} \perp \qquad\qquad \frac{\neg P}{\odot}\,\odot_{\neg\top} \ \text{if } P =_{\mathcal{RE}} \top$$

Analytic Rules

$$\frac{S}{P, Q}\,\alpha_{\wedge} \ \text{if } S =_{\mathcal{RE}} P \wedge Q \qquad\qquad \frac{\neg S}{\neg P \mid \neg Q}\,\beta_{\neg\wedge} \ \text{if } S =_{\mathcal{RE}} P \wedge Q$$

$$\frac{S}{P \mid Q}\,\beta_{\vee} \ \text{if } S =_{\mathcal{RE}} P \vee Q \qquad\qquad \frac{\neg S}{\neg P, \neg Q}\,\alpha_{\neg\vee} \ \text{if } S =_{\mathcal{RE}} P \vee Q$$

$$\frac{S}{\neg P \mid Q}\,\beta_{\Rightarrow} \ \text{if } S =_{\mathcal{RE}} P \Rightarrow Q \qquad\qquad \frac{\neg S}{P, \neg Q}\,\alpha_{\neg\Rightarrow} \ \text{if } S =_{\mathcal{RE}} P \Rightarrow Q$$

$$\frac{S}{\neg P, \neg Q \mid P, Q}\,\beta_{\Leftrightarrow} \ \text{if } S =_{\mathcal{RE}} P \Leftrightarrow Q \qquad\qquad \frac{\neg S}{\neg P, Q \mid P, \neg Q}\,\beta_{\neg\Leftrightarrow} \ \text{if } S =_{\mathcal{RE}} P \Leftrightarrow Q$$

$$\frac{\neg S}{P}\,\alpha_{\neg\neg} \ \text{if } S =_{\mathcal{RE}} \neg P$$

$$\frac{S}{P(\epsilon(x).P(x))}\,\delta_{\exists} \ \text{if } S =_{\mathcal{RE}} \exists x\ P(x) \qquad\qquad \frac{\neg S}{\neg P(\epsilon(x).\neg P(x))}\,\delta_{\neg\forall} \ \text{if } S =_{\mathcal{RE}} \forall x\ P(x)$$

γ-Rules

$$\frac{S}{P(X)}\,\gamma_{\forall M} \ \text{if } S =_{\mathcal{RE}} \forall x\ P(x) \qquad\qquad \frac{\neg S}{\neg P(X)}\,\gamma_{\neg\exists M} \ \text{if } S =_{\mathcal{RE}} \exists x\ P(x)$$

$$\frac{S}{P(t)}\,\gamma_{\forall\text{inst}} \ \text{if } S =_{\mathcal{RE}} \forall x\ P(x) \qquad\qquad \frac{\neg S}{\neg P(t)}\,\gamma_{\neg\exists\text{inst}} \ \text{if } S =_{\mathcal{RE}} \exists x\ P(x)$$

Fig. 2. Deduction Modulo Rules for Zenon

Nevertheless, if any proposition rewriting rules were allowed, it would transform clauses, that is disjunctions of literals, into arbitrary formulas. To keep formulas in clausal form, polarized resolution modulo introduces the following restrictions:

- Rules are polarized: we consider two kinds of proposition rewriting rules, the positive ones and the negative ones (tagged respectively with $+$ and $-$). A positive (resp. negative) rule can only be applied at positive (resp. negative) positions in a formula. Let us recall that the root position of a formula is positive, and that polarity switches under a negation and at the left-hand side of an implication.
- Rules are clausal: a positive rule is of the form $P \longrightarrow^{+} \neg C$, and a negative rule of the form $P \longrightarrow^{-} C$, where in both cases C is in clausal form, which means that it is a universally quantified disjunction of literals.

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

$$\textsf{Ext. Narr.}^{-}\ \frac{P \vee C}{\sigma(D \vee C)}\ \begin{array}{l}\sigma = mgu(P,Q)\\ Q \longrightarrow^{-} D\end{array} \qquad \textsf{Ext. Narr.}^{+}\ \frac{\neg Q \vee D}{\sigma(C \vee D)}\ \begin{array}{l}\sigma = mgu(P,Q)\\ P \longrightarrow^{+} \neg C\end{array}$$

$$\textsf{Ext. Narr.}^{t}\ \frac{L \vee C}{\sigma(L' \vee C)}\ L\sigma \longrightarrow L' \text{ by a term rewrite rule}$$

Fig. 3. Inference Rules of Polarized Resolution Modulo

Then, polarized resolution modulo consists of the inference rules of Fig. 3.

Any proposition rewriting system can be turned into an equivalent one that is polarized and clausal. Intuitively, the right-hand side of the rewriting formulas must be put into clausal normal form using a standard algorithm. This is what we can do for the rewriting system for the B set theory presented in Sec. 2. For instance, the rule for the power set construct $s \in \mathbb{P}(t) \longrightarrow \forall x\ (x \in s \Rightarrow x \in t)$ is transformed into one negative rule $s \in \mathbb{P}(t) \longrightarrow^{-} \forall x\ (\neg x \in s \vee x \in t)$ and two positive rules $s \in \mathbb{P}(t) \longrightarrow^{+} \neg\neg sk(s,t) \in s$ and $s \in \mathbb{P}(t) \longrightarrow^{+} \neg sk(s,t) \in t$, where sk is a fresh function symbol.

More generally, any set of first order axioms can be turned into a polarized and clausal rewriting system such that polarized resolution modulo is complete for the theory defined by the axioms. A tool[6] has been developed, that automatically orients the axioms into a rewriting system usable by iProver Modulo. This tool can also be used to transform an arbitrary rewriting system into a polarized and clausal one, so that we use it for the B set theory.

3.3 The Dedukti Backends

Skeptically checking proofs imposes to carry in traces all the information needed by the proof checker to assert the validity of proofs. The naive idea to record each rewriting step as a special rule causes us to lose the conciseness provided by deduction modulo, because an arbitrary number of rewrite steps can occur between two consecutive proof nodes. To circumvent this problem, the proof checker must be able not to distinguish propositions or terms belonging to the same equivalence class modulo rewriting. The only information needed is then the set of rewrite rules, which cannot be bigger than the problem statement itself. The Dedukti universal proof checker [3], based on the $\lambda\Pi$-calculus modulo (a λ-calculus with dependent types and rewrite rules), meets this requirement in a simple way: the set of rewrite rules is given to Dedukti in the header of the proof, and the conversions modulo rewriting are performed on the nose.

As the logics of Zenon Modulo and Dedukti are respectively classical and constructive, a translation has been implemented. Since Dedukti offers many facilities

[6] See: `http://www.ensiie.fr/~guillaume.burel/empty_autotheo.html.en`.

to define shallow embeddings (thanks to rewrite rules), we opted for a double-negation translation, described in [9], rather than for a non-computational axiomatization of the excluded-middle law. Introducing double-negations at each place à la Kolmogorov is a linear solution with respect to proof size, but appeared impracticable. This is why we forged a translation that introduces as few double-negation as possible, improving over previously existing work by Gödel, Gentzen, and Kuroda. Moreover, we treat differently the formulas along the side of the sequent they appear, following the remark that left-rules are identical in classical and intuitionistic sequent calculi. Combined with the use of rewrite rules, this has an important impact on the size of the generated traces, and on the time that it takes to check them.

iProver Modulo is also able to output proof in Dedukti's format. More precisely, it can transform a derivation of the empty clause in polarized resolution modulo into a proof of false in Dedukti, following the ideas developed in [7]. It is worth noting that the encoding of resolution proofs is shallow, which means that it can be easily combined with other proofs in first order logic. Here again, the term rewriting system used in iProver Modulo is reused in Dedukti, which ensures that we keep the gain of performing proof search modulo rewriting, for proof checking. Nevertheless, iProver Modulo backend to Dedukti is not complete, because it lacks a translation of the transformation of the initial problem into clausal form. The main issue to solve this is that some steps of the transformation are not provable (in the sense that the set of formulas before the step is not logically equivalent to the set of formulas after), all that we know is that the set of formulas are equisatisfiable. This is in particular the case for the Skolemization.

4 Implementation and Experimental Results

In this section, we describe our implementation of the B set theory modulo, which has been introduced in Sec. 2. We also provide some experimental and comparative results obtained on a benchmark consisting of a large set of properties coming from the B-Book, and which involve several automated theorem provers, including Zenon Modulo and iProver Modulo, the two deduction modulo based tools introduced in Sec. 3. Finally, we present an example of proof, whose proof is only found by the tools based on deduction modulo.

4.1 Implementation

Our implementation of the B set theory modulo is actually part of a larger development, which is the BWare platform. This platform, which is developed in the framework of the BWare project [10], aims to automatically verify proof obligations coming from Atelier B and works as shown in Fig. 4. The proof obligations, which are initially produced by Atelier B, are then translated by a specific tool into files for the Why3 platform [2] (a generic platform usually used for deductive program verification), which are compatible with a Why3 encoding of the B set theory. From these files, Why3 can produce (by means of appropriate drivers) the proof obligations for the automated theorem provers considered in the BWare project, using the TPTP format [15] for the Zenon Modulo and iProver Modulo first order theorem provers, and a native format for the Alt-Ergo SMT solver. This translation together with the encoding of the B set theory aims to generate valid statements that are appropriate for the automated theorem provers, i.e. whose proofs can be found by these provers. Finally, once proofs have been found by these tools, some of these provers can generate proof objects to be verified by proof checkers. This is the case of Zenon Modulo and iProver Modulo, which can produce proof objects for Dedukti, as presented in Sec. 3.

In this paper and w.r.t. the architecture of the BWare platform, our contribution consists of a Why3 encoding of the B set theory modulo introduced in Sec. 2. This encoding uses the programming and specification language of Why3, which is a typed first order language with polymorphic types in particular. As this language does not handle rewrite rules, the encoding is expressed as pure first order theory, and the corresponding axioms are transformed into rewrite rules directly by Zenon Modulo and iProver Modulo after the theory has been transformed into a TPTP theory by the appropriate driver of Why3. This pure first order encoding also allows us to compare our tools with other automated theorem provers that do not handle rewrite rules but only axioms (see the experimental results below). In addition, in order to be quite conform with our initial theory modulo, we have also customized the Why3 platform to disable the production of types in the TPTP files, which allows us to remove the first order encoding of Why3 types as they are useless in the expression of our theory.

4.2 Experimental Results

We propose a test[7] of our B set theory modulo and our automated theorem provers based on deduction modulo, respectively presented in Secs. 2 and 3. This test is performed on a benchmark consisting of all the properties of Chap. 2 of the B-Book [1], except those involving lambda abstractions or function evaluations (as said in Sec. 2, function evaluations actually require to be specifically handled by the downstream automated theorem provers, and currently, this has not been implemented). This leads to a set of 319 properties, mainly consisting of laws regarding membership, monotonicity, inclusion, and equality for every construct defined in the B set theory.

Regarding the tools involved in our test, we compare Zenon Modulo 0.1.2 and iProver Modulo v0.7+0.2 with their respective original versions Zenon 0.7.2 and iProver v0.7, as well as with Vampire 2.6 and E 1.8, two other first-order automated theorem provers that are usually considered as good contenders in this domain. The results of this test (run on an Intel i7-4770 3.40GHz computer, with a timeout of 60s and a memory limit of 1GiB) are summarized in Tab. 1. As can be observed, the tools based on deduction modulo not only prove more properties than all the other tools (more than 75%), but also provide a significant gain

[7] All the files of this benchmark, as well as Zenon Modulo and iProver Modulo, are available at: `http://cedric.cnam.fr/~delahaye/benchmark-iFM14.tgz`.

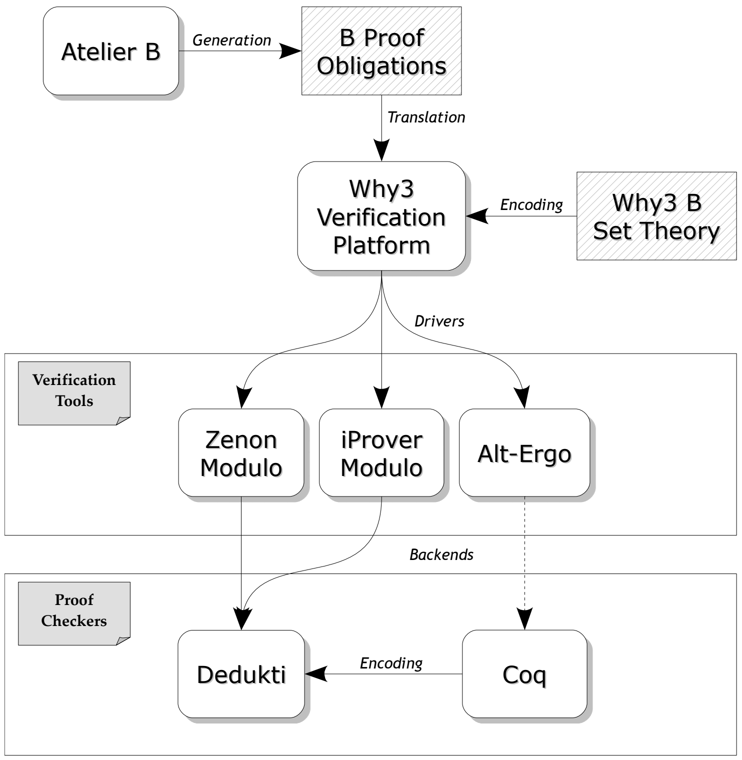

Fig. 4. The BWare Platform for the Automated Verification of B Proof Obligations

compared to the original tools that they extend (about 75% for Zenon Modulo and 56% for iProver Modulo). There is also a significant gain of these tools compared to the other first-order automated theorem provers considered in this test (more than 50% compared to Vampire and E). All these results tend to show how convenient it is to consider a theory as a system of rewrite rules, rather than under the form of a set of raw axioms. Regarding the soundness of the produced proofs, Dedukti is able to check with success all the 245 proofs found by Zenon Modulo, and 233 proofs (over 248 proofs) found by iProver Modulo (the other unverified proofs are due to failures of the backend).

4.3 An Example of Proof

Among the catalog of properties of the B-Book, most of them are provided without proofs, the B-Book proposing proofs only for those that are worthy of interest. To show the effectiveness of our tools, let us describe the proof produced by Zenon Modulo for Property 2.6.1 of the B-Book, which is a law concerning the subtraction of domains. This property states that, given two relations over

319 Prob.	Zenon	Zenon Modulo	iProver	iProver Modulo	Vampire	E
Proofs	6	245	68	248	76	48
Rate	1.9%	76.8%	21.3%	77.7%	23.8%	15%

Table 1. Experimental Results over the B-Book Properties

the same sets, the difference of their domains is included in the domain of their difference. More precisely, this property is expressed as follows:

$$\forall s, t, p, q\ (p \in s \leftrightarrow t \wedge q \in s \leftrightarrow t \Rightarrow \mathsf{dom}(p) - \mathsf{dom}(q) \subseteq \mathsf{dom}(p - q))$$

This property is proved by both Zenon Modulo and iProver Modulo, but by none of the other automated theorem provers considered in our previous test. When applied to this property, Zenon Modulo produces the proof of Fig. 5, which is expressed in a format combining deduction steps, in solid line and with labels referring to the rules of Fig. 2, and rewriting steps, in dashed line and with labels involving set constructs and referring to the rules of Fig. 1.

5 Conclusion

In this paper, we have introduced a new encoding of the set theory of the B method based on deduction modulo, and which is mainly expressed as a set of rewrite rules. We have also presented Zenon Modulo and iProver Modulo, two automated theorem provers that rely on deduction modulo and that are able to directly handle this encoding. These two tools have backends based on the Dedukti universal proof checker, which also relies on deduction modulo, and which allow us to certify the correctness of the proofs produced by these two tools. Finally, we have provided some experimental results on a benchmark consisting of derived properties of the B-Book, which show a significant gain of the tools based on deduction modulo compared to other regular first order automated theorem provers that handle the B set theory as a set of raw axioms rather than under the form of a rewrite system. In addition, to show the effectiveness of our approach, we have described an example of proof, whose proof is only found by the tools based on deduction modulo.

As future work, we first aim to improve our encoding of the B set theory based on deduction modulo so that Zenon Modulo and iProver Modulo are able to prove more properties automatically. Among these improvements, there is in particular the problem of function evaluation, which is currently managed as a pure axiom and which still needs attention to be smoothly integrated into our theory modulo. The comprehension scheme will also need to be revisited, and may require some special narrowing rules to ensure cut-free completeness. Indeed, we plan to study this kind of meta-theoretical properties concerning this B set theory modulo, once it has been fully designed. As there is no simple and

$$\frac{\neg\,(\forall s,t,p,q\ (p \in s \leftrightarrow t \wedge q \in s \leftrightarrow t \Rightarrow \mathsf{dom}(p) - \mathsf{dom}(q) \subseteq \mathsf{dom}(p-q)))}{\neg\,(\tau_3 \in \tau_1 \leftrightarrow \tau_2 \wedge \tau_4 \in \tau_1 \leftrightarrow \tau_2 \Rightarrow \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \subseteq \mathsf{dom}(\tau_3 - \tau_4))}\ \delta_{\neg\forall} \times 4$$

$$\frac{\neg\,(\tau_3 \in \tau_1 \leftrightarrow \tau_2 \wedge \tau_4 \in \tau_1 \leftrightarrow \tau_2 \Rightarrow \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \subseteq \mathsf{dom}(\tau_3 - \tau_4))}{\tau_3 \in \tau_1 \leftrightarrow \tau_2,\ \tau_4 \in \tau_1 \leftrightarrow \tau_2,\ \neg\,(\mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \subseteq \mathsf{dom}(\tau_3 - \tau_4))}\ \alpha_{\neg\Rightarrow},\ \alpha_{\wedge}$$

$$\frac{\tau_3 \in \tau_1 \leftrightarrow \tau_2,\ \tau_4 \in \tau_1 \leftrightarrow \tau_2,\ \neg\,(\mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \subseteq \mathsf{dom}(\tau_3 - \tau_4))}{\neg\,(\mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \in \mathbb{P}(\mathsf{dom}(\tau_3 - \tau_4)))}\ \mathit{rewrite}(\subseteq)$$

$$\frac{\neg\,(\mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \in \mathbb{P}(\mathsf{dom}(\tau_3 - \tau_4)))}{\neg\,(\forall x\ (x \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \Rightarrow x \in \mathsf{dom}(\tau_3 - \tau_4)))}\ \mathit{rewrite}(\mathbb{P})$$

$$\frac{\neg\,(\forall x\ (x \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \Rightarrow x \in \mathsf{dom}(\tau_3 - \tau_4)))}{\neg\,(\tau_5 \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \Rightarrow \tau_5 \in \mathsf{dom}(\tau_3 - \tau_4))}\ \delta_{\neg\forall}$$

$$\frac{\neg\,(\tau_5 \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \Rightarrow \tau_5 \in \mathsf{dom}(\tau_3 - \tau_4))}{\tau_5 \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4),\ \neg\,(\tau_5 \in \mathsf{dom}(\tau_3 - \tau_4))}\ \alpha_{\neg\Rightarrow}$$

$$\frac{\tau_5 \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4),\ \neg\,(\tau_5 \in \mathsf{dom}(\tau_3 - \tau_4))}{\tau_5 \in \mathsf{dom}(\tau_3) \wedge \neg\,(\tau_5 \in \mathsf{dom}(\tau_4))}\ \mathit{rewrite}(-)$$

$$\frac{\tau_5 \in \mathsf{dom}(\tau_3) \wedge \neg\,(\tau_5 \in \mathsf{dom}(\tau_4))}{\neg\,(\exists b\ ((\tau_5, b) \in \tau_3 - \tau_4))}\ \mathit{rewrite}(\mathsf{dom})$$

$$\frac{\neg\,(\exists b\ ((\tau_5, b) \in \tau_3 - \tau_4))}{\tau_5 \in \mathsf{dom}(\tau_3),\ \neg\,(\tau_5 \in \mathsf{dom}(\tau_4))}\ \alpha_{\wedge}$$

$$\frac{\tau_5 \in \mathsf{dom}(\tau_3),\ \neg\,(\tau_5 \in \mathsf{dom}(\tau_4))}{\exists b\ ((\tau_5, b) \in \tau_3),\ \neg\,(\exists b\ ((\tau_5, b) \in \tau_4))}\ \mathit{rewrite}(\mathsf{dom}) \times 2$$

$$\frac{\exists b\ ((\tau_5, b) \in \tau_3),\ \neg\,(\exists b\ ((\tau_5, b) \in \tau_4))}{(\tau_5, \tau_6) \in \tau_3}\ \delta_{\exists}$$

$$\frac{(\tau_5, \tau_6) \in \tau_3}{\neg\,((\tau_5, \tau_6) \in \tau_3 - \tau_4)}\ \gamma_{\neg\exists}$$

$$\frac{\neg\,((\tau_5, \tau_6) \in \tau_3 - \tau_4)}{\neg\,((\tau_5, \tau_6) \in \tau_3 \wedge \neg\,((\tau_5, \tau_6) \in \tau_4))}\ \mathit{rewrite}(-)$$

$$\frac{\neg\,((\tau_5, \tau_6) \in \tau_3 \wedge \neg\,((\tau_5, \tau_6) \in \tau_4))}{\dfrac{\neg\,((\tau_5, \tau_6) \in \tau_3)}{\odot}\ \odot \qquad \dfrac{\neg\,(\neg\,((\tau_5, \tau_6) \in \tau_4))}{\dfrac{(\tau_5, \tau_6) \in \tau_4}{\dfrac{\neg\,((\tau_5, \tau_6) \in \tau_4)}{\odot}\ \odot}\ \gamma_{\neg\exists}}\ \alpha_{\neg\neg}}\ \beta_{\neg\wedge}$$

where :
$\tau_1 = \epsilon(s).\neg\,(p \in s \leftrightarrow t \wedge q \in s \leftrightarrow t \Rightarrow \mathsf{dom}(p) - \mathsf{dom}(q) \subseteq \mathsf{dom}(p-q))$
$\tau_2 = \epsilon(t).\neg\,(p \in \tau_1 \leftrightarrow t \wedge q \in \tau_1 \leftrightarrow t \Rightarrow \mathsf{dom}(p) - \mathsf{dom}(q) \subseteq \mathsf{dom}(p-q))$
$\tau_3 = \epsilon(p).\neg\,(p \in \tau_1 \leftrightarrow \tau_2 \wedge q \in \tau_1 \leftrightarrow \tau_2 \Rightarrow \mathsf{dom}(p) - \mathsf{dom}(q) \subseteq \mathsf{dom}(p-q))$
$\tau_4 = \epsilon(q).\neg\,(\tau_3 \in \tau_1 \leftrightarrow \tau_2 \wedge q \in \tau_1 \leftrightarrow \tau_2 \Rightarrow \mathsf{dom}(\tau_3) - \mathsf{dom}(q) \subseteq \mathsf{dom}(\tau_3 - q))$
$\tau_5 = \epsilon(x).\neg\,(x \in \mathsf{dom}(\tau_3) - \mathsf{dom}(\tau_4) \Rightarrow x \in \mathsf{dom}(\tau_3 - \tau_4))$
$\tau_6 = \epsilon(b).((\tau_5, b) \in \tau_3)$

Fig. 5. Proof of Property 2.6.1 of the B-Book by Zenon Modulo

syntactical way to ensure cut-free completeness of theories modulo, this proof will probably be an ad hoc proof. Finally, as this work is part of the BWare project, our ultimate goal is to apply Zenon Modulo and iProver Modulo together with our B set theory modulo to the verification of proof obligations coming from industrial applications. To do so, we will have to develop a translation from B proof obligations to our encoding expressed in the programming and specification language of Why3. Such a translation already exists but for another encoding of the B set theory in Why3 that is customized for the Alt-Ergo SMT solver (which is part of the automated theorem provers considered in the BWare project). With the current set of proof obligations available in the BWare project (more than 10,500 proof obligations), Alt-Ergo provides very promising results, as it is able to automatically discharge more than 98% of these proof obligations (see [8]). Our objective is to obtain similar results with Zenon Modulo and iProver Modulo.

References

1. J.-R. Abrial. *The B-Book, Assigning Programs to Meanings.* Cambridge University Press, Cambridge (UK), 1996. ISBN 0521496195.
2. F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich. Why3: Shepherd Your Herd of Provers. In *International Workshop on Intermediate Verification Languages (Boogie)*, 2011.
3. M. Boespflug, Q. Carbonneaux, and O. Hermant. The $\lambda\Pi$-Calculus Modulo as a Universal Proof Language. In *Proof Exchange for Theorem Proving (PxTP)*, pages 28–43, Manchester (UK), June 2012.
4. R. Bonichon. TaMeD: A Tableau Method for Deduction Modulo. In *International Joint Conference on Automated Reasoning (IJCAR)*, volume 3097 of *LNCS*, pages 445–459, Cork (Ireland), July 2004. Springer.
5. R. Bonichon, D. Delahaye, and D. Doligez. Zenon: An Extensible Automated Theorem Prover Producing Checkable Proofs. In *Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 4790 of *LNCS/LNAI*, pages 151–165, Yerevan (Armenia), Oct. 2007. Springer.
6. G. Burel. Experimenting with Deduction Modulo. In *Conference on Automated Deduction (CADE)*, volume 6803 of *LNCS/LNAI*, pages 162–176, Wrocław (Poland), July 2011. Springer.
7. G. Burel. A Shallow Embedding of Resolution and Superposition Proofs into the $\lambda\Pi$-Calculus Modulo. In *Proof Exchange for Theorem Proving (PxTP)*, volume 14 of *EPiC*. EasyChair, 2013.
8. S. Conchon and M. Iguernelala. Tuning the Alt-Ergo SMT Solver for B Proof Obligations. In *Abstract State Machines, Alloy, B, VDM, and Z (ABZ)*, LNCS, Toulouse (France), June 2014. Springer. To appear.
9. D. Delahaye, D. Doligez, F. Gilbert, P. Halmagrand, and O. Hermant. Zenon Modulo: When Achilles Outruns the Tortoise using Deduction Modulo. In *Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 8312 of *LNCS/ARCoSS*, pages 274–290, Stellenbosch (South Africa), Dec. 2013. Springer.
10. D. Delahaye, C. Dubois, C. Marché, and D. Mentré. The BWare Project: Building a Proof Platform for the Automated Verification of B Proof Obligations. In *Abstract State Machines, Alloy, B, VDM, and Z (ABZ)*, LNCS, Toulouse (France), June 2014. Springer. To appear.
11. G. Dowek. What is a Theory? In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285 of *LNCS*, pages 50–64, Antibes Juan-les-Pins (France), Mar. 2002. Springer.
12. G. Dowek. Polarized Resolution Modulo. In *Theoretical Computer Science (TCS)*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 182–196, Brisbane (Australia), Sept. 2010. Springer.
13. G. Dowek, T. Hardin, and C. Kirchner. Theorem Proving Modulo. *Journal of Automated Reasoning (JAR)*, 31(1):33–72, Sept. 2003.
14. K. Korovin. iProver – An Instantiation-Based Theorem Prover for First-Order Logic (System Description). In *International Joint Conference on Automated Reasoning (IJCAR)*, volume 5195 of *LNCS*, pages 292–298, Sydney (Australia), Aug. 2008. Springer.
15. G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning (JAR)*, 43(4):337–362, Dec. 2009.