HAL
open science

Using Deduction Modulo in Set Theory

Pierre Halmagrand

To cite this version:

Pierre Halmagrand. Using Deduction Modulo in Set Theory. SETS14, 1st International Workshop about Sets and Tools, Jun 2014, Toulouse, France. pp.12. hal-01100512

HAL Id: hal-01100512

https://inria.hal.science/hal-01100512v1

Submitted on 6 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Using Deduction Modulo in Set Theory *

Pierre Halmagrand

Cedric/Cnam/Inria, Paris, France
Pierre.Halmagrand@inria.fr

Abstract. We present some improvements of Zenon Modulo and the application of this tool to sets of problems coming from set theory. Zenon Modulo is an extension of the tableau-based first order automated theorem prover Zenon to deduction modulo. Deduction modulo is an extension of predicate calculus, which allows us to rewrite terms as well as propositions, and which is well-suited for proof-search in axiomatic theories, as it turns axioms into rewrite rules. The improvements discussed here consist in a better heuristic to automatically build rewrite systems given a set of axioms, and some optimizations in the rewriting process used during the proof search. We also present some updated results obtained on benchmarks provided by the TPTP library for set theory categories. Finally, we discuss some recent work about the application of our tool to the B method set theory, in particular the way we treat equality and the comprehension scheme.

Keywords: Automated Theorem Proving, Tableaux, Deduction Modulo, Rewriting, Set Theory, B Method, Zenon Modulo.

1 Introduction

The development of large-scale industrial projects based on formal software is constrained by the efficiency of verification tools that themselves rely on automated theorem provers to mechanize a maximal part of the formalization process. Therefore, to allow a wider dissemination of these techniques, a particular attention must be paid to the development of automated theorem prover. Since many formal developments may be based on specific theories, like the widely used B method [1] which relies on a particular typed set theory, we must pay attention to reason within axiomatic theories. A solution to improve verification tools is to combine different approaches in automated deduction and specialize them for a specific theory, like SMT solvers.

The BWare project [17] aims to provide a generic platform relying on different theorem provers (first order provers and SMT solvers) to verify proof obligations coming from the development of industrial applications using the B method.

* This work is supported by the BWare project [17] (ANR-12-INSE-0010) funded by the INS programme of the French National Research Agency (ANR).

As part of this project, the development of Zenon Modulo [10] is driven by its application to set theory.

Our approach is to extend the tableau-based first order automated theorem prover Zenon [6] to deduction modulo [11]. Deduction modulo is an extension of predicate calculus, which allows to rewrite terms as well as propositions. This is well suited for proof-search in axiomatic theories, as it turns axioms into rewrite rules. For instance, we can express Zermelo set theory without any axiom [12], and turn non-deterministic proof-search among axioms into computations, which reduces the proof-search space. Moreover, Zenon Modulo, like Zenon, adopts a certifying approach and produces proof certificates that can be checked by external proof checkers. Since the proofs produced by Zenon Modulo should keep the advantage of conciseness of deduction modulo, we do not want to record all the computational steps of rewriting [9]. As a result, in order to verify such proofs, we use Dedukti [4], a simple proof checker based on the $\lambda\Pi$-calculus modulo which can deal with rewriting.

We also present an updated heuristic, used by Zenon Modulo to automatically transform axioms into rewrite rules. The new heuristic captures more axioms and guarantees to have a more meaningful rewrite system and impacts both rewrite rules over terms and over propositions. We should notice here that this heuristic has been developed to be used as a preprocessing tool that allows a fully automatic use of Zenon Modulo on problems in the TPTP format. A second option is to give manually the rewrite system to Zenon Modulo. This second option is the one chosen for the B method set theory since there is a limited number of axioms and definitions, and we can afford to design a customized rewrite system.

To assess our approach, we apply this new heuristic to the SET and SEU categories of the first-order problems of the TPTP library [16] which deal with set theory, and compare results obtained by Zenon and Zenon Modulo with both heuristics. In particular, we exhibit some examples of difficult problems that can be proved by our tool using the new heuristic.

Finally, we discuss the application of Zenon Modulo to the B method set theory [1]. We will especially focus on equality and the comprehension scheme in first order logic.

This paper is organized as follows: in Sec. 2, we first introduce the basic notions of deduction modulo and its application to Zenon Modulo; we then present in Sec. 3 the new heuristic for Zenon Modulo; we expose in Sec. 4 the experimental results obtained on the benchmarks provided by the TPTP library; finally, in Sec. 5, we discuss the ongoing work on the B set theory.

2 Zenon with Deduction Modulo

Zenon Modulo [10] is an extension of the tableau-based first order automated theorem prover Zenon [6] to deduction modulo [11]. This extension is partially inspired by the presentation of tableaux modulo in [5]. 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 can be closed by applying a closure rule, the tree itself is said closed, and this closed tree is a proof of the goal.

Deduction modulo [11] extends the predicate calculus by introducing a congruence relation over propositions and the ability to perform conversion between propositions that are equivalent modulo the congruence. Given a set of axioms, the congruence is generated by the rewrite system coming from the transformation of axioms into rewrite rules. One of the major interests of deduction modulo lies in reasoning modulo this congruence, since it reduces the search space by removal of axioms from the context. For example, considering the following definition of inclusion in set theory: $\forall X, Y\ (X \subseteq Y \Leftrightarrow \forall x\ (x \in X \Rightarrow x \in Y))$

The proof of $A \subseteq A$ produced by Zenon has the following shape:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg(A \subseteq A),\ \forall X, Y\ (X \subseteq Y \Leftrightarrow \forall x\ (x \in X \Rightarrow x \in Y))}{\forall Y\ (A \subseteq Y \Leftrightarrow \forall x\ (x \in A \Rightarrow x \in Y))}\ \gamma_{\forall\text{inst}}}{A \subseteq A \Leftrightarrow \forall x\ (x \in A \Rightarrow x \in A)}\ \gamma_{\forall\text{inst}}}{\neg\forall x\ (x \in A \Rightarrow x \in A)}\ \beta_{\Leftrightarrow} \qquad \dfrac{A \subseteq A}{\odot}\ \odot}{\neg(\tau \in A \Rightarrow \tau \in A)}\ \delta_{\neg\forall}}{\tau \in A,\ \neg(\tau \in A)}\ \alpha_{\neg\Rightarrow}}{\odot}\ \odot$$

where $\tau = \epsilon x.\neg(x \in A \Rightarrow x \in A)$ is a Hilbert epsilon-term formed with a variable x and the formula $\neg(x \in A \Rightarrow x \in A)$, which is the formula existentially quantified over. The Hilbert epsilon-term denotes the term that satisfies the formula, seen as a predicate on x, if it exists [18]. It can be seen as a Skolem term that records the formula itself.

Deduction modulo replaces the axiom by the rewrite rule, where X and Y are variables: $X \subseteq Y \longrightarrow \forall x\ (x \in X \Rightarrow x \in Y)$

The proof produced by Zenon Modulo is then:

$$\dfrac{\dfrac{\dfrac{\neg(A \subseteq A)}{\neg(\tau \in A \Rightarrow \tau \in A)}\ \delta_{\neg\forall},\ A \subseteq A \longrightarrow \forall x\ (x \in A \Rightarrow x \in A)}{\tau \in A,\ \neg(\tau \in A)}\ \alpha_{\neg\Rightarrow}}{\odot}\ \odot$$

where $\tau = \epsilon x.\neg(x \in A \Rightarrow x \in A)$.

It can be seen that computations are interleaved with the deduction rules. It can be noticed that the proof is much simpler than the one produced by Zenon. In addition to simplicity, deduction modulo also allows for unbounded proof size reduction [7].

3 Building Theories Modulo

Dealing with large sets of axioms is quite difficult when building theories modulo. In the TPTP library [16], the first order problems category has almost 7,000,000 axioms and the subset of set theory problems has more than 170,000 axioms. Since we cannot manually build rewrite systems for such large sets of axioms, we develop a heuristic to automatically turn axioms into rewrite rules.

The heuristic described below is based on the following guidelines. We never introduce a free variable that was not bound at the head of the formula by a universal quantifier. For rewrite rules over terms, we want to capture equality properties of terms, except axioms expressing commutativity for obvious termination reasons. We also excluded to rewrite a variable to a more complicated term for the same reason. For rewrite rules over propositions, we select axioms expressing equivalence properties of propositions. The left hand side of the rewrite rule must be an atom, or its negation, as required by deduction modulo [11], and the right hand side may be any proposition. We also handle cases where the atomic proposition is an equality, in this case we require that at least one side of the equality is a function symbol. Moreover, axioms with just an atomic proposition, or its negation, are turned to rewrite to true, and false respectively. Finally, conjunctions of propositions that satisfy one of the previous conditions, are separated into multiple rewrite rules, accordingly to the logical equivalence of $\forall \bar{x}\ \varphi \wedge \psi$ and $\forall \bar{x}\ \varphi \wedge \forall \bar{x}\ \psi$, where φ and ψ are two arbitrary formulas.

In the following, t_1 and t_2 are terms, P is an atomic formula that is not an equation, and φ is an arbitrary formula. $\mathrm{FV}(\varphi)$ refers to the set of free variables of φ. Here are the shapes of axioms that are handled by our heuristic and the corresponding rewrite rule that is generated:

- $\forall \bar{x}\ t_1 = t_2$
  - If $\mathrm{FV}(t_2) \subseteq \mathrm{FV}(t_1)$ and t_1 is not a variable, then the term rewrite rule $t_1 \longrightarrow t_2$ is generated;
  - Otherwise, if $\mathrm{FV}(t_1) \subseteq \mathrm{FV}(t_2)$ and t_2 is not a variable, then the term rewrite rule $t_2 \longrightarrow t_1$ is generated;
  - In addition, all axioms expressing the commutativity of a given symbol are excluded.

- $\forall \bar{x}\ P$ (resp. $\forall \bar{x}\ \neg P$)
  - The proposition rewrite rule $P \longrightarrow \top$ is generated (resp. $P \longrightarrow \bot$).

- $\forall \bar{x}\ P \Leftrightarrow \varphi$ (resp. $\forall \bar{x}\ \neg P \Leftrightarrow \varphi$)
  - If $\mathrm{FV}(\varphi) \subseteq \mathrm{FV}(P)$, then the proposition rewrite rule $P \longrightarrow \varphi$ is generated (resp. $P \longrightarrow \neg\varphi$);
  - Otherwise, if φ is a literal and $\mathrm{FV}(P) \subseteq \mathrm{FV}(\varphi)$, then we apply the heuristic to the formula $\forall \bar{x}\ \varphi \Leftrightarrow P$ (resp. $\forall \bar{x}\ \varphi \Leftrightarrow \neg P$).

- $\forall \bar{x}\ (t_1 = t_2) \Leftrightarrow \varphi$ (resp. $\forall \bar{x}\ \neg(t_1 = t_2) \Leftrightarrow \varphi$)
  - If $\mathrm{FV}(\varphi) \subseteq \mathrm{FV}(t_1) \cup \mathrm{FV}(t_2)$ and at least one of the two terms t_1 and t_2 is not a variable, then the proposition rewrite rule $(t_1 = t_2) \longrightarrow \varphi$ is generated (resp. $(t_1 = t_2) \longrightarrow \neg\varphi$).

The main difference between this new heuristic and the previous one presented in [10] is the last item of the list above. With this new rule, we can transform into rewrite rules axioms which express an equivalence between equality of two terms and a proposition. An example of a proof using such rewrite rule is given in Sec. 4. In addition, we improve the generation of rewrite rules over terms in order to exclude to rewrite variables, and we deal with conjunctions of propositions that satisfy one the conditions listed above.

4 Experimental Results

4.1 TPTP Benchmarks

The TPTP library (v6.0.0) provides a large set of standard benchmark examples for automated theorem proving systems [16]. To assess our approach, we perform experiments over all the first order problems coming from the TPTP library and dealing with set theory. This leads to select the SET and SEU categories that have respectively 462 and 900 problems. We compare Zenon, the old version of Zenon Modulo and the new version of Zenon Modulo which uses the heuristic described in Sec. 3. This experiment was done on an Intel Core i7-4770 3.40GHz computer, with a memory limit of 1GB and a timeout of 300s.

The results are summarized in Tab. 1. This table has three columns: the first one provides the number of problems proved by Zenon for each category, while the other two show the results for the old and new versions of Zenon Modulo. In addition, for both old and new versions of Zenon Modulo, we provide some detailed results with profit and loss between provers.

From the results of Tab. 1, we observe that the new version of Zenon Modulo proves more problems than Zenon and the old version of Zenon Modulo for both SET and SEU categories. The total profit of 5 problems in SET category (resulting from the gain of 12 problems and the loss of 7) may seem low, regarding to the difference of 73 problems between Zenon and the old version of Zenon Modulo, but we have gained some very difficult problems, as shown below. The improvement is more significant than for the SEU category. We notice a net profit of 8 problems, coming from a gain of 26 problems and a loss of 18. We also show an example from the SEU category below.

The verification by Dedukti of proofs produced by Zenon Modulo is almost total. Over the 227 proofs in SET category, 224 are correctly verified. The three missing proofs are due to termination issues of the translation of the proof from Zenon Modulo to Dedukti by the backend of Zenon Modulo. In SEU category, all the 110 proofs are declared well typed by Dedukti.

TPTP Category	Zenon	Zenon Modulo (Old)		Zenon Modulo (New Version)		
	Total	Total	vs. Zenon	Total	vs. Zenon	vs. Old
SET 462 prob.	149	222	+86 -13	227	+91 -13	+12 -7
SEU 900 prob.	96	102	+14 -8	110	+32 -18	+26 -18

Table 1. Experimental Results over the TPTP Library

4.2 Analysis of Two Proofs From Set Theory

The TPTP library provides a ranking system to evaluate problems. This note, between 0 and 1, expresses the percentage of automated theorem provers that are not able to prove the problem. A note of 0 means that the problem is trivial (every prover solves it), and a note of 1 means that all provers fail.

According to the TPTP ranking, Zenon Modulo is able to prove some quite difficult problems. The hardest problem is SET817+4, with a ranking of 0.97 in TPTP v6.0.0 (solved only by Muscadet [15]), and which neither Zenon or the previous version of Zenon Modulo [10] was able to prove. It states that the intersection of all elements of a non-empty set of ordinal numbers is an ordinal. Among the 8 axioms needed by Zenon Modulo to solve this problem, 7 are transformed into rewrite rules by our heuristic. The proof of this problem is too big to be displayed here.

An other example coming from the set theory, and that bodes well for application to the B method set theory, is the problem SEU194+1. This problem, with a ranking of 0.70, and which neither Zenon or the previous version of Zenon Modulo [10] was able to prove, states that for any set s, if p is a relation, then the domain of the restriction of p to s is equal to the intersection of the domain of p and the set s. Among the twenty eight axioms provided with the conjecture, Zenon Modulo needs only two axioms to solve it, of which one is turned into a rewrite rule. In the following, s, t and u are sets, p is a relation, a is an element of a set, rel is a predicate for relations, dom a function that returns the domain of a relation, and rest a function that returns the restriction of a relation.

- Conjecture:
  $\forall s, p\ (\mathsf{rel}(p) \Rightarrow \mathsf{dom}(\mathsf{rest}(p, s)) = \mathsf{dom}(p) \cap s)$
- Axiom:
  $\forall a, s, p\ (\mathsf{rel}(p) \Rightarrow (a \in \mathsf{dom}(\mathsf{rest}(p, s)) \Leftrightarrow (a \in s) \wedge (a \in \mathsf{dom}(p))))$
- Rewrite rule:
  $u = s \cap t \longrightarrow \forall a\ (a \in u \Leftrightarrow ((a \in s) \wedge (a \in t)))$

Zenon Modulo produces the proof of Fig. 1, presented in a format combining deduction steps in solid line and rewriting steps in dashed line. In addition, we omit some unnecessary formulas resulting from the application of $\beta_{\Leftrightarrow}$ rules to lighten the presentation.

$$
\begin{array}{c}
\neg(\forall s, p\ (\mathsf{rel}(p) \Rightarrow \mathsf{dom}(\mathsf{rest}(p,s)) = \mathsf{dom}(p) \cap s)), \\
\forall a, s, p\ (\mathsf{rel}(p) \Rightarrow (a \in \mathsf{dom}(\mathsf{rest}(p,s)) \Leftrightarrow a \in s \wedge a \in \mathsf{dom}(p))) \\
\hline
\neg(\mathsf{rel}(\tau_2) \Rightarrow \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) = \mathsf{dom}(\tau_2) \cap \tau_1) \quad \delta_{\neg\forall} \times 2 \\
\hline
\mathsf{rel}(\tau_2),\ \neg(\mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) = \mathsf{dom}(\tau_2) \cap \tau_1) \quad \alpha_{\neg\Rightarrow} \\
\hline
\neg(\forall a\ (a \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow a \in \mathsf{dom}(\tau_2) \wedge a \in \tau_1)) \quad \textit{rewrite} \\
\hline
\neg(\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \mathsf{dom}(\tau_2) \wedge \tau_3 \in \tau_1) \quad \delta_{\neg\forall} \\
\hline
\Pi_1 \qquad \begin{array}{c} \tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)),\ \neg(\tau_3 \in \mathsf{dom}(\tau_2) \wedge \tau_3 \in \tau_1) \quad \beta_{\neg\Leftrightarrow} \\ \hline \Pi_2 \qquad\qquad \Pi_3 \end{array} \quad \beta_{\neg\wedge}
\end{array}
$$

$$
\begin{array}{c}
\Pi_1 \\
\hline
\neg(\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1))),\ \tau_3 \in \mathsf{dom}(\tau_2) \wedge \tau_3 \in \tau_1 \quad \alpha_{\wedge} \\
\hline
\tau_3 \in \mathsf{dom}(\tau_2),\ \tau_3 \in \tau_1 \quad \gamma_{\forall} \times 3 \\
\hline
\mathsf{rel}(\tau_2) \Rightarrow (\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2)) \quad \beta_{\Rightarrow} \\
\hline
\begin{array}{c} \neg(\mathsf{rel}(\tau_2)) \\ \hline \odot \end{array} \odot \qquad \begin{array}{c} \tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \begin{array}{c} \neg(\tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2)) \\ \hline \begin{array}{c} \neg(\tau_3 \in \tau_1) \\ \hline \odot \end{array} \odot \qquad \begin{array}{c} \neg(\tau_3 \in \mathsf{dom}(\tau_2)) \\ \hline \odot \end{array} \odot \end{array} \beta_{\neg\wedge} \qquad \begin{array}{c} \tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \\ \hline \odot \end{array} \odot \end{array} \beta_{\Leftrightarrow}
\end{array}
$$

$$
\begin{array}{c}
\Pi_2 \\
\hline
\neg(\tau_3 \in \mathsf{dom}(\tau_2)) \\
\hline
\mathsf{rel}(\tau_2) \Rightarrow (\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2)) \quad \gamma_{\forall} \times 3 \\
\hline
\begin{array}{c} \neg(\mathsf{rel}(\tau_2)) \\ \hline \odot \end{array} \odot \qquad \begin{array}{c} \tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \begin{array}{c} \neg(\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1))) \\ \hline \odot \end{array} \odot \qquad \begin{array}{c} \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \begin{array}{c} \tau_3 \in \tau_1,\ \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \odot \end{array} \odot \end{array} \alpha_{\wedge} \end{array} \beta_{\Leftrightarrow} \quad \beta_{\Rightarrow}
\end{array}
$$

$$
\begin{array}{c}
\Pi_3 \\
\hline
\neg(\tau_3 \in \tau_1) \\
\hline
\mathsf{rel}(\tau_2) \Rightarrow (\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2)) \quad \gamma_{\forall} \times 3 \\
\hline
\begin{array}{c} \neg(\mathsf{rel}(\tau_2)) \\ \hline \odot \end{array} \odot \qquad \begin{array}{c} \tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1)) \Leftrightarrow \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \begin{array}{c} \neg(\tau_3 \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1))) \\ \hline \odot \end{array} \odot \qquad \begin{array}{c} \tau_3 \in \tau_1 \wedge \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \begin{array}{c} \tau_3 \in \tau_1,\ \tau_3 \in \mathsf{dom}(\tau_2) \\ \hline \odot \end{array} \odot \end{array} \alpha_{\wedge} \end{array} \beta_{\Leftrightarrow} \quad \beta_{\Rightarrow}
\end{array}
$$

where :
$\tau_1 = \epsilon(s).\neg(\mathsf{rel}(p) \Rightarrow \mathsf{dom}(\mathsf{rest}(p,s)) = \mathsf{dom}(p) \cap s)$
$\tau_2 = \epsilon(p).\neg(\mathsf{rel}(p) \Rightarrow \mathsf{dom}(\mathsf{rest}(p,\tau_1)) = \mathsf{dom}(p) \cap \tau_1)$
$\tau_3 = \epsilon(a).\neg(a \in \mathsf{dom}(\mathsf{rest}(\tau_2,\tau_1) \Leftrightarrow a \in \mathsf{dom}(\tau_2) \wedge a \in \tau_1))$

Fig. 1. Proof of Problem SEU194+1

5 Application to the B Method Set Theory

The BWare project [17] aims to provide a generic platform based on Why3 [3] relying on different deduction tools, such as Alt-Ergo [2], iProver Modulo [8], Super Zenon [13] and Zenon Modulo [10], in order to verify proof obligations coming from the development of industrial applications using the B method. Since B proof obligations are translated into the input language of Why3 [14], the B method set theory has been axiomatized in the WhyML language.

Building the B set theory modulo consists mainly in turning six axioms and many derived constructs into rewrite rules. The first two axioms, dealing with membership of an ordered pair in a cartesian product and the membership of a set in the power-set, can be easily turned into rewrite rules. The third axiom, defining the comprehension scheme, is removed due to its high-order definition, we present below how to deal with derived constructs defined with the comprehension scheme. The fourth axiom is the extensionality axiom and states that two sets s and t are equal if being a member of s is equivalent to be a member of t. This is not the only property of the equality symbol, as we will see below. Finally, the last two axioms, the axiom of choice and the existence of an infinite set are easy to deal with.

Following the notations of the B-Book, s and t are sets, E and F some expressions, x a variable, $\mathbb{P}(t)$ the power-set of the set t and `BIG` the constant infinite set. Here is the set of rewrite rules generated from the axioms:

$$\begin{cases} (E,F) \in (s \times t) \longrightarrow (E \in s \wedge F \in t) & \text{(pair)} \\ s \in \mathbb{P}(t) \longrightarrow \forall x\ (x \in s \Rightarrow x \in t) & \text{(power)} \\ s = t \longrightarrow \forall x\ (x \in s \Leftrightarrow x \in t) & \text{(extensionality)} \\ \texttt{choice}(s) \in s \longrightarrow \exists x\ (x \in s) & \text{(choice)} \\ \texttt{infinite}(\texttt{BIG}) \longrightarrow \top & \text{(infinite)} \end{cases}$$

Fig. 2. Expression of the Axioms of the B Set Theory as a Rewrite System

5.1 Removal of the Comprehension Scheme

In the B-book, the comprehension scheme is used to define non-primitive symbols. For instance, the union of two sets is defined as follows:

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

where u is a set, and s and t and two subsets of u.

Since we have dismissed the comprehension scheme, we expand the above definition by directly defining membership to the union, thereby removing the use of comprehension:

$$x \in s \cup t \longrightarrow x \in s \vee x \in t \qquad \text{(union)}$$

This rewrite rule, combined with extensionality, is equivalent to the previous definition of the union. Handling the other non-primitive symbols, like intersection of sets, inverse of a relation or also the identity relation, in this systematic way, allows a total removal of the comprehension scheme used to define derived constructs in the B-Book. Unfortunately, this method do not permit us to manage user-defined sets using the comprehension scheme for the moment.

Here is, as an example, the proof produced by Zenon Modulo for the commutativity of union:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg\forall A, B\ (A \cup B = B \cup A)}{\neg\forall B\ (\tau_1 \cup B = B \cup \tau_1)}\ \delta_{\neg\forall}}{\tau_1 \cup \tau_2 = \tau_2 \cup \tau_1}\ \delta_{\neg\forall}}{\neg\forall X\ (X \in \tau_1 \cup \tau_2 \Leftrightarrow X \in \tau_2 \cup \tau_1)}\ \text{extensionality}}{\neg(\tau_3 \in \tau_1 \cup \tau_2 \Leftrightarrow \tau_3 \in \tau_2 \cup \tau_1)}\ \delta_{\neg\forall}}{\Pi_1 \qquad \Pi_2}\ \beta_{\neg\Leftrightarrow}$$

where Π_1 and Π_2 are the following subtrees:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Pi_1}{\neg(\tau_3 \in \tau_1 \cup \tau_2),\ \tau_3 \in \tau_2 \cup \tau_1}}{\neg(\tau_3 \in \tau_1 \vee \tau_3 \in \tau_2),\ \tau_3 \in \tau_2 \vee \tau_3 \in \tau_1}\ \text{union} \times 2}{\neg(\tau_3 \in \tau_1),\ \neg(\tau_3 \in \tau_2)}\ \alpha_{\neg\vee}}{\dfrac{\tau_3 \in \tau_2}{\odot}\ \odot \qquad \dfrac{\tau_3 \in \tau_1}{\odot}\ \odot}\ \beta_{\vee}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Pi_2}{\tau_3 \in \tau_1 \cup \tau_2,\ \neg(\tau_3 \in \tau_2 \cup \tau_1)}}{\tau_3 \in \tau_1 \vee \tau_3 \in \tau_2,\ \neg(\tau_3 \in \tau_2 \vee \tau_3 \in \tau_1)}\ \text{union} \times 2}{\neg(\tau_3 \in \tau_2),\ \neg(\tau_3 \in \tau_1)}\ \alpha_{\neg\vee}}{\dfrac{\tau_3 \in \tau_1}{\odot}\ \odot \qquad \dfrac{\tau_3 \in \tau_2}{\odot}\ \odot}\ \beta_{\vee}$$

and where:

$$\begin{aligned}
\tau_1 &= \epsilon A.\neg(A \cup B = B \cup A)\\
\tau_2 &= \epsilon B.\neg(\tau_1 \cup B = B \cup \tau_1)\\
\tau_3 &= \epsilon X.\neg(X \in \tau_1 \cup \tau_2 \Leftrightarrow X \in \tau_2 \cup \tau_1)
\end{aligned}$$

We notice that the subtrees Π_1 and Π_2 are symmetric, the proof of the commutativity of union resulting of the commutativity of the disjunction.

5.2 Dealing with Equality

The set theory of the B-book relies heavily on a primitive notion of equality, introduced before axioms for the set theory [1]. The main property of the equality is substitutivity (*i.e.* an expression can be replaced by another one in a proposition provided they are equal). Starting from substitutivity and reflexivity of equality, other properties like symmetry and transitivity can be derived. Zenon already have rules that deal with this equational reasoning [6].

The second property of equality, introduced much later, and that has to deal directly with set theory is extensionality as shown in Fig. 2. Turning the axiom of extensionality into a rewrite rule allows Zenon Modulo to extend the equality symbol into membership equivalence at each step of the proof-search.

Choosing between equational reasoning and the extensionality properties of equality during proof-search may be decisive to find a proof. There are different solutions to deal with this problem. A first idea is to try both equational reasoning and extensionality every time we meet an equality symbol, but this solution may not be efficient since we duplicate the work for each equality symbol. The solution we are working on is to implement a heuristic into Zenon Modulo that decides to apply the extensionality rewrite rule, or to use the equational reasoning of Zenon, based on the shape of terms. For instance, if both sides of the equality are variables, we do not apply the extensionality rewrite rule, otherwise we use it.

6 Conclusion

We have presented some improvements of Zenon Modulo, in particular a new heuristic to transform axioms into rewrite rules over terms and propositions. This heuristic is used as a preprocessing tool to automatically build theories modulo given sets of axioms. We have also presented results obtained on set theory problems, coming from the benchmarks provided by the TPTP library. This experiment was performed using the new heuristic. In particular, we have shown that this new version proves some new difficult problems according to the TPTP ranking.

We have also discussed some recent work about the application of our tool to the B method set theory. Since we want to use Zenon Modulo to verify proof obligations coming from the development of industrial applications using the B method, we have to build a B set theory modulo. We have described a method to remove the comprehension scheme used to define non-primitive symbols like union of sets. Finally, we have presented some ideas to handle equality in the B set theory.

As future work, we first aim to implement a heuristic into Zenon Modulo that handles the equality of B set theory. In particular, this heuristic should decide whether to apply the extensionality rewrite rule, or to use the equational reasoning of Zenon. To assess our work, we will first try to prove a large part of derived lemmas coming from the B-Book. Finally, we will apply Zenon Modulo to the set of proof obligations provided by the benchmarks of the BWare project.

Acknowledgement. Many thanks to O. Hermant, and D. Doligez for their detailed comments on this paper, and to the Deducteam Inria research team for the many interactions.

References

1. J.-R. Abrial. *The B-Book, Assigning Programs to Meanings.* Cambridge University Press, Cambridge (UK), 1996. ISBN 0521496195.
2. F. Bobot, S. Conchon, E. Contejean, and S. Lescuyer. Implementing Polymorphism in SMT solvers. In C. Barrett and L. de Moura, editors, *SMT 2008: 6th International Workshop on Satisfiability Modulo*, volume 367 of *ACM International Conference Proceedings Series*, pages 1–5, 2008.
3. F. Bobot, J.-C. Filliâtre, C. Marché, and A. Paskevich. Why3: Shepherd your herd of provers. In *Boogie 2011: First International Workshop on Intermediate Verification Languages*, pages 53–64, Wrocław, Poland, August 2011.
4. 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.
5. 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.
6. 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.
7. G. Burel. Efficiently Simulating Higher-Order Arithmetic by a First-Order Theory Modulo. *Logical Methods in Computer Science (LMCS)*, 7(1):1–31, Mar. 2011.
8. 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.
9. D. Delahaye, D. Doligez, F. Gilbert, P. Halmagrand, and O. Hermant. Proof Certification in Zenon Modulo: When Achilles Uses Deduction Modulo to Outrun the Tortoise with Shorter Steps. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *International Workshop on the Implementation of Logics (IWIL)*, Stellenbosch (South Africa), Dec. 2013. EasyChair.
10. D. Delahaye, D. Doligez, F. Gilbert, P. Halmagrand, and O. Hermant. Zenon Modulo: When Achilles Outruns the Tortoise using Deduction Modulo. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 8312 of *LNCS/ARCoSS*, pages 274–290, Stellenbosch (South Africa), Dec. 2013. Springer.
11. G. Dowek, T. Hardin, and C. Kirchner. Theorem Proving Modulo. *Journal of Automated Reasoning (JAR)*, 31(1):33–72, Sept. 2003.
12. G. Dowek and A. Miquel. Cut Elimination for Zermelo Set Theory. In *Archive for Mathematical Logic.* Springer. Submitted.
13. M. Jacquel, K. Berkani, D. Delahaye, and C. Dubois. Tableaux Modulo Theories using Superdeduction: An Application to the Verification of B Proof Rules with the Zenon Automated Theorem Prover. In *International Joint Conference on Automated Reasoning (IJCAR)*, volume 7364 of *LNCS*, pages 332–338, Manchester (UK), June 2012. Springer.

14. D. Mentré, C. Marché, J.-C. Filliâtre, and M. Asuka. Discharging proof obligations from Atelier B using multiple automated provers. In S. Reeves and E. Riccobene, editors, *ABZ'2012 - 3rd International Conference on Abstract State Machines, Alloy, B and Z*, volume 7316 of *Lecture Notes in Computer Science*, pages 238–251, Pisa, Italy, June 2012. Springer. `http://hal.inria.fr/hal-00681781/en/`.
15. D. Pastre. Muscadet 2.3: A knowledge-based theorem prover based on natural deduction. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning*, volume 2083 of *Lecture Notes in Computer Science*, pages 685–689. Springer Berlin Heidelberg, 2001.
16. 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.
17. The BWare Project, 2012. `http://bware.lri.fr/`.
18. C.-P. Wirth. Hilbert's epsilon as an operator of indefinite committed choice. *Journal of Applied Logic*, 6(3):287 – 317, 2008.