Tableaux Modulo Theories Using Superdeduction

Mélanie Jacquel, Karim Berkani, David Delahaye, Catherine Dubois

Tableaux Modulo Theories using Superdeduction

Mélanie Jacquel[1], Karim Berkani[1], David Delahaye[2], and Catherine Dubois[3]

[1] Siemens IC-MOL, Châtillon, France,
Melanie.Jacquel@siemens.com
Karim.Berkani@siemens.com
[2] Cedric/Cnam/Inria, Paris, France,
David.Delahaye@cnam.fr
[3] Cedric/ENSIIE/Inria, Évry, France,
dubois@ensiie.fr

Abstract. We propose a method that allows us to develop tableaux modulo theories using the principles of superdeduction, among which the theory is used to enrich the deduction system with new deduction rules. This method is presented in the framework of the Zenon automated theorem prover, and is applied to the set theory of the B method. This allows us to provide another prover to Atelier B, which can be used to verify B proof rules in particular. We also propose some benchmarks, in which this prover is able to automatically verify a part of the rules coming from the database maintained by Siemens IC-MOL. Finally, we describe another extension of Zenon with superdeduction, which is able to deal with any first order theory, and provide a benchmark coming from the TPTP library, which contains a large set of first order problems.

Keywords: Automated Deduction, Tableaux, Superdeduction, Zenon, Set Theory, B Method, First Order Theories.

1 Introduction

Reasoning modulo a theory like arithmetic in first order logic may appear as a complex task, since even simple formulas become difficult to be proved (using natural deduction or sequent calculus), and the corresponding proofs are hardly readable. This is actually due to the fact that Gentzen's deductive systems, even if more elaborate than systems à la Hilbert, consist of low-level languages for deduction. Over the last few years, several approaches have been developed to palliate this problem, and for instance, deduction modulo [13] and superdeduction [6], which respectively focus on the computational and deductive parts of a theory, can be considered as steps toward high-level deductive languages.

Superdeduction has been adapted for usual deductive systems, such as natural deduction or sequent calculus (see [6], for example). In these systems, we have to deal with some common permutability problems of automated proof search, since superdeduction systems are in fact embedding a part of compiled automated deduction. As a consequence, superdeduction can be naturally integrated into automated deduction methods, and we propose to do so in the framework

of the tableau method (this corresponds to an alternative approach to the work described in [4], where a tableau method for deduction modulo is proposed).

The integration of superdeduction into the tableau method is actually motivated by an experiment that is managed by Siemens IC-MOL regarding the verification of B proof rules [15]. The B method [1], or B for short, allows engineers to develop software with high guarantees of confidence; more precisely it allows them to build correct by design software. B is a formal method based on set theory and theorem proving, and which relies on a refinement-based development process. The Atelier B environment [9] is a platform that supports B and offers, among other tools, both automated and interactive provers. In practice, to ensure the global correctness of formalized applications, the user must discharge proof obligations. These proof obligations may be proved automatically, but otherwise, they have to be handled manually either by using the interactive prover, or by adding new proof rules that the automated prover can exploit. These new proof rules can be seen as axioms and must be verified by other means, otherwise the global correctness may be endangered.

In [15], we develop an approach based on the use of the Zenon automated theorem prover [5], which relies on classical first order logic with equality and applies the tableau method as proof search. In this context, the choice of Zenon is strongly influenced by its ability of producing comprehensible proof traces under the form of Coq proofs [24] in particular. The method used in this approach consists in first normalizing the formulas to be proved, in order to obtain first order logic formulas containing only the membership set operator, and then calling Zenon on these new formulas. This experiment gives satisfactory results in the sense that it can prove a significant part of the rules coming from the database maintained by Siemens IC-MOL (we can deal with about 1,400 rules, 1,100 of which can be proved automatically, over a total of 5,300 rules). However, this approach is not complete (after the normalization, Zenon proves the formulas without any axiom of set theory, while some instantiations may require to be normalized), and suffers from efficiency issues (due to the preliminary normalization). To deal with these problems, the idea developed in this paper is to integrate the axioms and constructs of the B set theory into the Zenon proof search method by means of superdeduction rules. This integration can be concretely achieved thanks to the extension mechanism offered by Zenon, which allows us to extend its core of deductive rules to match specific requirements.

In this paper, we also propose another extension of Zenon with superdeduction, which is able to deal with any first order theory, and which must be seen as a generalization of the extension of Zenon with superdeduction and for the B set theory that has been described previously. This other extension of Zenon has been developed as a tool called Super Zenon [16], where each theory is analyzed to determine the axioms which can be turned into superdeduction rules, and these superdeduction rules are automatically computed on the fly to enrich the deductive kernel of Zenon.

The paper is organized as follows: in Sec. 2, we start by briefly reviewing some related work; in Secs. 3 and 4, we then respectively introduce superdeduction and

present the computation of superdeduction rules from axioms in the framework of the tableau method used by Zenon; next, we explain, in Sec. 5, the superdeduction rules corresponding to the set theory which the B method relies on, and describe, in Sec. 6, the implementation of our extension of Zenon for the B set theory and provide some comparative benchmarks concerning the verification of B proof rules coming from the database maintained by Siemens IC-MOL; thereafter, in Sec. 7, we present the other extension of Zenon with superdeduction, which is able to deal with any first order theory, and also provide a comparative benchmark coming from the TPTP library, which contains a large set of first order problems in particular; finally, in Sec. 8, we discuss more generally the use of superdeduction for proof search in axiomatic theories.

2 Related Work

In the framework of verification of B (or Event-B) proof obligations, and in addition to B dedicated provers, such as the main prover of Atelier B, some similar work has been done to use external provers, relying on either SMT (Satisfiability Modulo Theories) solvers like veriT [14], or proof assistants like Coq [10] or Isabelle/HOL [21]. However, veriT does not provide proof traces, and automation is poor in the considered proof assistants, even though dedicated tactics are offered. More generally, there are also some provers covering set theory, such as Muscadet [19] or Theorema [25], but they do not provide traces either.

Regarding the ability of reasoning modulo theories, there exist some alternative approaches, such as the technology of SMT solvers. This technology is based on a SAT solver and a combination of several decision procedures for several theories. There are several methods to combine the considered decision procedures, such as the Nelson-Oppen [18] or Shostak [22] methods. The approach used by SMT solvers is actually restrictive in the sense that the considered theories have to be decidable. To preserve this decidability, the combination methods also impose some restrictions over the theories to be combined, which must have disjoint signatures. The techniques of deduction modulo [13] and superdeduction [6], proposed in this paper in particular, are more general approaches to integrate theories into proof search methods since there is no constraint over the theories to be integrated, which may be decidable or not.

3 Rationale for Superdeduction

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. Superdeduction [6] is actually a variant of deduction modulo, where axioms are used to enrich the deduction system with new deduction rules, which are called superdeduction rules. For example, considering the definition of inclusion in set theory $\forall a \forall b\ ((a \subseteq b) \Leftrightarrow (\forall x\ (x \in a \Rightarrow x \in b)))$, the proof of $A \subseteq A$ in sequent calculus has the following form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, x \in A \vdash A \subseteq A, x \in A}\,\text{Ax}}{\ldots \vdash A \subseteq A, x \in A \Rightarrow x \in A}\,\Rightarrow\text{R}}{\ldots \vdash A \subseteq A, \forall x\ (x \in A \Rightarrow x \in A)}\,\forall\text{R} \quad \dfrac{}{\ldots, A \subseteq A \vdash A \subseteq A}\,\text{Ax}}{\ldots, (\forall x\ (x \in A \Rightarrow x \in A)) \Rightarrow A \subseteq A \vdash A \subseteq A}\,\Rightarrow\text{L}}{A \subseteq A \Leftrightarrow (\forall x\ (x \in A \Rightarrow x \in A)) \vdash A \subseteq A}\,\wedge\text{L}}{\forall a \forall b\ ((a \subseteq b) \Leftrightarrow (\forall x\ (x \in a \Rightarrow x \in b))) \vdash A \subseteq A}\,\forall\text{L} \times 2$$

In deduction modulo, the axiom of inclusion can be seen as a computation rule and therefore replaced by the rewrite rule $a \subseteq b \rightarrow \forall x\ (x \in a \Rightarrow x \in b)$. The previous proof is then transformed as follows:

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

It can be noticed that the obtained proof is much simpler than the one completed using sequent calculus. In addition to simplicity, deduction modulo also allows us for unbounded proof size speed-up [7].

Superdeduction proposes to go further than deduction modulo precisely when the considered axiom defines a predicate P with an equivalence $\forall \bar{x}\ (P \Leftrightarrow \varphi)$. While deduction modulo replaces the axiom by a rewrite rule, superdeduction adds to this transformation the decomposition of the connectives occurring in this definition. This corresponds to an extension of Prawitz's folding (resp. unfolding) rules [20], where a maximum of connectives of the definition are introduced (resp. eliminated). Superdeduction may be seen as the alliance of deduction modulo with focusing, which is a technique initially introduced in the framework of linear logic [3]. For the axiom of inclusion, the proposed superdeduction rule is therefore the following (there is also a corresponding left rule):

$$\dfrac{\Gamma, x \in a \vdash x \in b, \Delta}{\Gamma \vdash a \subseteq b, \Delta}\,\text{IncR},\ x \notin \Gamma, \Delta$$

Hence, proving $A \subseteq A$ with this new rule can be performed as follows:

$$\dfrac{\dfrac{}{x \in A \vdash x \in A}\,\text{Ax}}{\vdash A \subseteq A}\,\text{IncR}$$

This new proof is not only simpler and shorter than in deduction modulo, but also follows a natural human reasoning scheme usually used in mathematics (in the same vein, see [11] for more details about how superdeduction allows us to recover intuition from automated proofs).

4 From Axioms to Superdeduction Rules

As mentioned previously, reasoning modulo a theory in a tableau method using superdeduction requires to generate new deduction rules from some axioms of

the theory. The axioms which can be considered for superdeduction are of the form $\forall \bar{x}\ (P \Leftrightarrow \varphi)$, where P is atomic. This specific form of axiom allows us to introduce an orientation of the axiom from P to φ, and we introduce the notion of proposition rewrite rule (this notion appears in [6], from which we borrow the following definition and notation):

Definition 1 (Proposition Rewrite Rule). *The notation $R : P \to \varphi$ denotes the axiom $\forall \bar{x}\ (P \Leftrightarrow \varphi)$, where R is the name of the rule, P an atomic proposition, φ a proposition, and $\bar{x}$ the free variables of P and φ.*

It should be noted that P may contain first order terms and therefore that such an axiom is not just a definition. For instance, $x \in a \cap b \to x \in a \wedge x \in b$ (where $a \cap b$ is a first order term) is a proposition rewrite rule.

As said in the introduction, one of our main objectives is to develop a proof search procedure for the set theory of the B method using the Zenon automated theorem prover [5]. In the following, we will thus consider the tableau method used by Zenon as the framework in which superdeduction rules will be generated from proposition rewrite rules.

The proof search rules of Zenon are described in detail in [5] and summarized in Fig. 1 (for the sake of simplification, we have omitted the unfolding and extension rules), where ϵ is Hilbert's operator ($\epsilon(x).P(x)$ means some x that satisfies $P(x)$, and is considered as a term), capital letters are used for metavariables, and R_r, R_s, R_t, and R_{ts} are respectively reflexive, symmetric, transitive, and transitive-symmetric relations (the corresponding rules also apply to the equality in particular). As hinted by the use of Hilbert's operator, the δ-rules are handled by means of ϵ-terms rather than using Skolemization. What we call here metavariables are often named free variables in the tableau-related literature; they are not used as variables as they are never substituted. The proof search rules are applied with the normal 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 an application of a closure rule), the tree is closed, and this closed tree is a proof of the goal. Note that this algorithm is applied in strict depth-first order: we close the current branch before starting work on another branch. Moreover, we work in a non-destructive way: working on one branch will never change the formulas of any other branch. We divide these rules into five distinct classes to be used for a more efficient proof search. This extends the usual sets of rules dealing with $\alpha, \beta, \delta, \gamma$-formulas and closure ($\odot$) with the specific rules of Zenon. We list below the five sets of rules and their elements:

α	$\alpha_{\neg\vee}, \alpha_{\wedge}, \alpha_{\neg\Rightarrow}, \alpha_{\neg\neg}, \neg_{\text{refl}}$
β	$\beta_{\vee}, \beta_{\neg\wedge}, \beta_{\Rightarrow}, \beta_{\Leftrightarrow}, \beta_{\neg\Leftrightarrow}$, pred, fun, sym, trans$*$
δ	$\delta_{\exists}, \delta_{\neg\forall}$
γ	$\gamma_{\forall M}, \gamma_{\neg\exists M}, \gamma_{\forall\text{inst}}, \gamma_{\neg\exists\text{inst}}$
$\odot$	$\odot_{\top}, \odot_{\bot}, \odot, \odot_r, \odot_s$

where "trans$*$" gathers all the transitivity rules.

Closure and Cut Rules

$$\frac{\bot}{\odot}\ \odot_{\bot} \qquad \frac{\neg\top}{\odot}\ \odot_{\neg\top} \qquad \frac{}{P \mid \neg P}\ \text{cut}$$

$$\frac{\neg R_r(t,t)}{\odot}\ \odot_r \qquad \frac{P \qquad \neg P}{\odot}\ \odot \qquad \frac{R_s(a,b) \qquad \neg R_s(b,a)}{\odot}\ \odot_s$$

Analytic Rules

$$\frac{\neg\neg P}{P}\ \alpha_{\neg\neg} \qquad \frac{P \Leftrightarrow Q}{\neg P, \neg Q \mid P, Q}\ \beta_{\Leftrightarrow} \qquad \frac{\neg(P \Leftrightarrow Q)}{\neg P, Q \mid P, \neg Q}\ \beta_{\neg\Leftrightarrow}$$

$$\frac{P \wedge Q}{P, Q}\ \alpha_{\wedge} \qquad \frac{\neg(P \vee Q)}{\neg P, \neg Q}\ \alpha_{\neg\vee} \qquad \frac{\neg(P \Rightarrow Q)}{P, \neg Q}\ \alpha_{\neg\Rightarrow}$$

$$\frac{P \vee Q}{P \mid Q}\ \beta_{\vee} \qquad \frac{\neg(P \wedge Q)}{\neg P \mid \neg Q}\ \beta_{\neg\wedge} \qquad \frac{P \Rightarrow Q}{\neg P \mid Q}\ \beta_{\Rightarrow}$$

$$\frac{\exists x\ P(x)}{P(\epsilon(x).P(x))}\ \delta_{\exists} \qquad \frac{\neg\forall x\ P(x)}{\neg P(\epsilon(x).\neg P(x))}\ \delta_{\neg\forall}$$

γ-Rules

$$\frac{\forall x\ P(x)}{P(X)}\ \gamma_{\forall M} \qquad \frac{\neg\exists x\ P(x)}{\neg P(X)}\ \gamma_{\neg\exists M}$$

$$\frac{\forall x\ P(x)}{P(t)}\ \gamma_{\forall\text{inst}} \qquad \frac{\neg\exists x\ P(x)}{\neg P(t)}\ \gamma_{\neg\exists\text{inst}}$$

Relational Rules

$$\frac{P(t_1,...,t_n) \qquad \neg P(s_1,..,s_n)}{t_1 \neq s_1 \mid ... \mid t_n \neq s_n}\ \text{pred} \qquad \frac{f(t_1,...,t_n) \neq f(s_1,...,s_n)}{t_1 \neq s_1 \mid ... \mid t_n \neq s_n}\ \text{fun}$$

$$\frac{R_s(s,t) \qquad \neg R_s(u,v)}{t \neq u \mid s \neq v}\ \text{sym} \qquad \frac{\neg R_r(s,t)}{s \neq t}\ \neg_{\text{refl}}$$

$$\frac{R_t(s,t) \qquad \neg R_t(u,v)}{u \neq s, \neg R_t(u,s) \mid t \neq v, \neg R_t(t,v)}\ \text{trans}$$

$$\frac{R_{ts}(s,t) \qquad \neg R_{ts}(u,v)}{v \neq s, \neg R_{ts}(v,s) \mid t \neq u, \neg R_{ts}(t,u)}\ \text{transsym}$$

$$\frac{s = t \qquad \neg R_t(u,v)}{u \neq s, \neg R_t(u,s) \mid \neg R_t(u,s), \neg R_t(t,v) \mid t \neq v, \neg R_t(t,v)}\ \text{transeq}$$

$$\frac{s = t \qquad \neg R_{ts}(u,v)}{v \neq s, \neg R_{ts}(v,s) \mid \neg R_{ts}(v,s), \neg R_{ts}(t,u) \mid t \neq u, \neg R_{ts}(t,u)}\ \text{transeqsym}$$

Fig. 1. Proof Search Rules of Zenon

Let us now describe how the computation of superdeduction rules for Zenon is performed from a given proposition rewrite rule.

Definition 2 (Computation of Superdeduction Rules). *Let $\mathcal{S}$ be a set of rules composed by the subset of the proof search rules of Zenon formed of the closure rules, the analytic rules, as well as the $\gamma_{\forall M}$ and $\gamma_{\neg\exists M}$ rules. Given a proposition rewrite rule $R : P \to \varphi$, two superdeduction rules (a positive one R and a negative one $\neg R$) are generated in the following way:*

1. *To get the positive rule R, initialize the procedure with the formula φ. Next, apply the rules of $\mathcal{S}$ until there is no open leaf anymore on which they can be applied. Then, collect the premises and the conclusion, and replace φ by P to obtain the positive rule R.*
2. *To get the negative rule $\neg R$, initialize the procedure with the formula $\neg\varphi$. Next, apply the rules of $\mathcal{S}$ until there is no open leaf anymore on which they can be applied. Then, collect the premises and the conclusion, and replace $\neg\varphi$ by $\neg P$ to obtain the negative rule $\neg R$.*

If the rule R (resp. $\neg R$) involves metavariables, an instantiation rule R_{inst} (resp. $\neg R_{\text{inst}}$) is added, where one or several metavariables can be instantiated.

Integrating these new deduction rules to the proof search rules of Zenon is trivially correct as they are generated from a subset of the rules of Zenon, while the considered subset of rules only consists of reversible rules, which is a necessary condition for completeness.

Let us illustrate the computation of superdeduction rules from a proposition rewrite rule with the example of the set inclusion.

Example 1 (Set Inclusion). From the definition of the set inclusion, we introduce the proposition rewrite rule $\text{Inc} : a \subseteq b \to \forall x\ (x \in a \Rightarrow x \in b)$, and the corresponding superdeduction rules Inc and $\neg$Inc are generated as follows:

$$\dfrac{\dfrac{\forall x\ (x \in a \Rightarrow x \in b)}{X \in a \Rightarrow X \in b}\,\gamma_{\forall M}}{X \notin a \mid X \in b}\,\beta_{\Rightarrow} \qquad \dfrac{\dfrac{\neg\forall x\ (x \in a \Rightarrow x \in b)}{\neg(\epsilon_x \in a \Rightarrow \epsilon_x \in b)}\,\delta_{\neg\forall}}{\epsilon_x \in a, \epsilon_x \notin b}\,\alpha_{\neg\Rightarrow}$$

where $\epsilon_x = \epsilon(x).\neg(x \in a \Rightarrow x \in b)$.

The resulting superdeduction rules are then the following:

$$\dfrac{a \subseteq b}{X \notin a \mid X \in b}\,\text{Inc} \qquad \dfrac{a \subseteq b}{t \notin a \mid t \in b}\,\text{Inc}_{\text{inst}} \qquad \dfrac{a \not\subseteq b}{\epsilon_x \in a, \epsilon_x \notin b}\,\neg\text{Inc}$$

5 Superdeduction Rules for the B Set Theory

The purpose of this section is to build a superdeduction system for the B set theory, which can be used for the verification of proof rules of Atelier B, such as the rules coming from the database maintained by Siemens IC-MOL (see Sec. 6). To do so, the idea is to apply the computational algorithm of superdeduction rules described in Sec. 4 to the several axioms of the B set theory.

5.1 The B Set Theory

As said in the introduction, the B method [1] aims to assist experts to develop certified software. The initial step is defined with abstract properties of a model. Several steps of property refinement are then applied until the release of the complete software. A refinement step is characterized by adding details on the software behavior under construction. For each step, generated proof obligations must be demonstrated to ensure the global correctness of the formalization.

The B method is based on a typed set theory. There are two rule systems: one for demonstrating that a formula is well-typed, and one for demonstrating that a formula is a logical consequence of a set of axioms. The main aim of the type system is to avoid inconsistent formulas, such as Russell's paradox for example. The B proof system is based on a sequent calculus with equality. Six axioms define the basic operators and the extensionality which, in turn, defines the equality of two sets. In addition, the other operators ($\cup$, $\cap$, etc.) are defined using the previous basic ones. Fig. 2 gathers a part of the axioms and constructs of the B set theory, where BIG is an infinite set (mostly only used to build natural numbers in the foundational theory). In this figure, we only consider the four first axioms of the B set theory, as we do not need the two remaining axioms in the rules that we want to verify (see Sec. 6). Regarding functions and due to space restrictions, we only present the notion of partial function, even though we can deal with the other function constructs in our superdeduction system. Compared to [1], all type information has been removed from the axioms and constructs. This can be done since typechecking is performed before proving a formula, and the pieces of type information dispersed throughout the axioms and constructs are useless when building a proof (see [2] for details regarding the modularity between the type and proof systems). Finally, we do not deal with substitution explicitly (see the axiom for comprehension sets, for example), as we do not intend to manage rules involving explicit substitutions (see Sec. 6).

5.2 Generating the Superdeduction Rules

To generate the superdeduction rules corresponding to the axioms and constructs defined in Fig. 2, we use the algorithm described in Def. 2 of Sec. 4, and we must therefore identify the several proposition rewrite rules. On the one hand, the axioms are all of the form $P_i \Leftrightarrow Q_i$, and the associated proposition rewrite rules are $R_i : P_i \rightarrow Q_i$, where each axiom is oriented from left to right. On the other hand, the constructs are expressed by the definitions $E_i \triangleq F_i$, where E_i and F_i are expressions, and the corresponding proposition rewrite rules are $R_i : x \in E_i \rightarrow x \in F_i$; if E_i and F_i represent relations, the proposition rewrite rules are $R_i : (x, y) \in E_i \rightarrow (x, y) \in F_i$, where we consider that relations are defined by means of ordered pairs. The superdeduction rules generated from this set of proposition rewrite rules are described in Figs. 3 and 4. Due to space restrictions, we do not include the definitions of $\subseteq$ (as well as $\subset$), and $\circ$ in these figures, but their rules can be respectively deduced from the rules of $\mathbb{P}$ and ";". For the same reasons, we do not describe the instantiation rules associated with

Axioms

$$(x, y) \in a \times b \Leftrightarrow x \in a \wedge y \in b$$
$$a \in \mathbb{P}(b) \Leftrightarrow \forall x \ (x \in a \Rightarrow x \in b)$$
$$x \in \{ \ y \mid P(y) \ \} \Leftrightarrow P(x)$$
$$a = b \Leftrightarrow \forall x \ (x \in a \Leftrightarrow x \in b)$$

Derived Constructs

$$a \subseteq b \Leftrightarrow a \in \mathbb{P}(b) \qquad a \subset b \Leftrightarrow a \subseteq b \wedge a \neq b$$
$$a \cup b \triangleq \{ \ x \mid x \in a \vee x \in b \ \} \qquad a \cap b \triangleq \{ \ x \mid x \in a \wedge x \in b \ \}$$
$$a - b \triangleq \{ \ x \mid x \in a \wedge x \notin b \ \} \qquad \emptyset \triangleq \mathrm{BIG} - \mathrm{BIG}$$
$$\{ \ e_1, \ldots, e_n \ \} \triangleq \{ \ x \mid x = e_1 \ \} \cup \ldots \cup \{ \ x \mid x = e_n \ \}$$

Binary Relation Constructs: First Series

$$a \leftrightarrow b \triangleq \mathbb{P}(a \times b) \qquad a^{-1} \triangleq \{ \ (y, x) \mid (x, y) \in a \ \}$$
$$\mathrm{dom}(a) \triangleq \{ \ x \mid \exists y \ (x, y) \in a \ \} \qquad \mathrm{ran}(a) \triangleq \mathrm{dom}(a^{-1})$$
$$a; b \triangleq \{ \ (x, z) \mid \exists y \ ((x, y) \in a \wedge (y, z) \in b \ \} \qquad a \circ b \triangleq b; a$$
$$\mathrm{id}(a) \triangleq \{ \ (x, y) \mid (x, y) \in a \times a \wedge x = y \ \}$$
$$a \lhd b \triangleq \mathrm{id}(a); b \qquad a \rhd b \triangleq a; \mathrm{id}(b)$$
$$a \lhd\!\!\!\cdot \ b \triangleq (\mathrm{dom}(b) - a) \lhd b \qquad a \rhd\!\!\!\cdot \ b \triangleq a \rhd (\mathrm{ran}(a) - b)$$

Binary Relation Constructs: Second Series

$$a[b] \triangleq \mathrm{ran}(b \lhd a) \qquad a \leftarrow\!\!\!\cdot \ b \triangleq (\mathrm{dom}(b) \lhd\!\!\!\cdot \ a) \cup b$$
$$a \otimes b \triangleq \{ \ (x, (y, z)) \mid (x, y) \in a \wedge (x, z) \in b \ \}$$
$$\mathrm{prj}_1(a, b) \triangleq (\mathrm{id}(a) \otimes (a \times b))^{-1} \qquad \mathrm{prj}_2(a, b) \triangleq ((b \times a) \otimes \mathrm{id}(b))^{-1}$$
$$a \parallel b \triangleq \{ \ (x, y), (z, t) \mid (x, z) \in a \wedge (y, t) \in b \ \}$$

Function Constructs: First Series

$$a \nrightarrow b \triangleq \{ \ t \mid t \in a \leftrightarrow b \wedge \forall (x, y, z) \ ((x, y) \in t \wedge (x, z) \in t \Rightarrow y = z) \ \}$$

Fig. 2. Axioms and Constructs of the B Set Theory

each rule involving metavariables. It should also be noted that the computation of these superdeduction rules goes further than the one proposed in Sec. 4, since given a proposition rewrite rule $R : P \rightarrow Q$, we apply to Q not only all the rules considered by Definition 2, but also the new generated superdeduction rules (except the rules for the extensional equality, in order to benefit from the dedicated rules of Zenon for equality) whenever applicable.

Regarding the superdeduction rules for the constructs, we can notice that given a definition $E \triangleq F$, the generation of superdeduction rules is performed from the proposition rewrite rule $R : x \in E \rightarrow x \in F$, and we can therefore wonder if completeness can still be ensured. In particular, we have to show that $P(E) \Leftrightarrow P(F)$, where P is a predicate symbol. This is actually possible using

Rules for Axioms

$$\frac{(x,y) \in a \times b}{x \in a, y \in b}\ \times \qquad \frac{a \in \mathbb{P}(b)}{X \notin a \mid X \in b}\ \mathbb{P} \qquad \frac{x \in \{\ y \mid P(y)\ \}}{P(x)}\ \{|\}$$

$$\frac{(x,y) \notin a \times b}{x \notin a \mid y \notin b}\ \neg\times \qquad \frac{a \notin \mathbb{P}(b)}{\epsilon_x \in a, \epsilon_x \notin b}\ \neg\mathbb{P} \quad \text{with } \epsilon_x = \epsilon(x).\neg(x \in a \Rightarrow x \in b) \qquad \frac{x \notin \{\ y \mid P(y)\ \}}{\neg P(x)}\ \neg\{|\}$$

$$\frac{a = b}{X \notin a, X \notin b \mid X \in a, X \in b}\ = \qquad \frac{a \neq b}{\epsilon_x \notin a, \epsilon_x \in b \mid \epsilon_x \in a, \epsilon_x \notin b}\ \neq \quad \text{with } \epsilon_x = \epsilon(x).\neg(x \in a \Leftrightarrow x \in b)$$

Rules for Derived Constructs

$$\frac{x \in a \cup b}{x \in a \mid x \in b}\ \cup \qquad \frac{x \in a \cap b}{x \in a, x \in b}\ \cap \qquad \frac{x \in a - b}{x \in a, x \notin b}\ -$$

$$\frac{x \notin a \cup b}{x \notin a, x \notin b}\ \neg\cup \qquad \frac{x \notin a \cap b}{x \notin a \mid x \notin b}\ \neg\cap \qquad \frac{x \notin a - b}{x \notin a \mid x \in b}\ \neg-$$

$$\frac{x \in \{\ e_1, \ldots, e_n\ \}}{x = e_1 \mid \ldots \mid x = e_1}\ \{\} \qquad \frac{x \notin \{\ e_1, \ldots, e_n\ \}}{x \neq e_1, \ldots, x \neq e_n}\ \neg\{\} \qquad \frac{x \in \emptyset}{\odot}\ \emptyset$$

Rules for Binary Relation Constructs: First Series

$$\frac{(x,y) \in a^{-1}}{(y,x) \in a}\ a^{-1} \qquad \frac{x \in \mathrm{dom}(a)}{(x,\epsilon_y) \in a}\ \mathrm{dom} \quad \text{with } \epsilon_y = \epsilon(y).((x,y) \in a) \qquad \frac{y \in \mathrm{ran}(a)}{(\epsilon_x, y) \in a}\ \mathrm{ran} \quad \text{with } \epsilon_x = \epsilon(x).((x,y) \in a)$$

$$\frac{(x,y) \notin a^{-1}}{(y,x) \notin a}\ \neg a^{-1} \qquad \frac{x \notin \mathrm{dom}(a)}{(x,Y) \notin a}\ \neg\mathrm{dom} \qquad \frac{y \notin \mathrm{ran}(a)}{(X,y) \notin a}\ \neg\mathrm{ran}$$

$$\frac{(x,z) \in a;b}{(x,\epsilon_y) \in a,\ (\epsilon_y, z) \in b}\ ; \quad \text{with } \epsilon_y = \epsilon(y).((x,y) \in a \wedge (y,z) \in b) \qquad \frac{(x,z) \notin a;b}{(x,Y) \notin a \mid (Y,z) \notin b}\ \neg;$$

$$\frac{(x,y) \in \mathrm{id}(a)}{x = y,\ x \in a,\ y \in a}\ id \qquad \frac{(x,y) \notin \mathrm{id}(a)}{x \neq y \mid x \notin a \mid y \notin a}\ \neg id$$

$$\frac{(x,y) \in a \lhd b}{(x,y) \in b,\ x \in a}\ \lhd \qquad \frac{(x,y) \notin a \lhd b}{(x,y) \notin b \mid x \notin a}\ \neg\lhd$$

Fig. 3. Superdeduction Rules for the B Set Theory (Part 1)

$$\frac{(x,y) \in a \lhd b}{(x,y) \in b,\ x \notin a}\ \lhd \qquad \frac{(x,y) \in a \rhd b}{(x,y) \in a,\ y \in b}\ \rhd \qquad \frac{(x,y) \in a \bowtie b}{(x,y) \in a,\ y \notin b}\ \bowtie$$

$$\frac{(x,y) \notin a \lhd b}{(x,y) \notin b \mid x \in a}\ \neg\lhd \qquad \frac{(x,y) \notin a \rhd b}{(x,y) \notin a \mid y \notin b}\ \neg\rhd \qquad \frac{(x,y) \notin a \bowtie b}{(x,y) \notin a \mid y \in b}\ \neg\bowtie$$

Rules for Binary Relation Constructs: Second Series

$$\frac{y \in a[b]}{\epsilon_x \in b, (\epsilon_x, y) \in a}\ a[b] \quad \text{with } \epsilon_x = \epsilon(x).(x \in b \wedge (x,y) \in a) \qquad \frac{(x,y) \in a \lhd\!\!\cdot\ b}{(x,y) \in a,\ (x,Y) \notin b \mid (x,y) \in b}\ \lhd\!\!\cdot$$

$$\frac{y \notin a[b]}{X \notin b \mid (X,y) \notin a}\ \neg a[b] \qquad \frac{(x,y) \notin a \lhd\!\!\cdot\ b}{(x,y) \notin a,\ (x,y) \notin b \mid (x,\epsilon_y) \in b,\ (x,y) \notin b}\ \neg\lhd\!\!\cdot \quad \text{with } \epsilon_y = \epsilon(y).((x,y) \in b)$$

$$\frac{(x,(y,z)) \in a \otimes b}{(x,y) \in a,\ (x,z) \in b}\ \otimes \qquad \frac{((x,y),z) \in \mathrm{prj}_1(a,b)}{x \in a,\ y \in b,\ z \in a,\ z = x}\ \mathrm{prj}_1$$

$$\frac{(x,(y,z)) \in a \otimes b}{(x,y) \notin a \mid (x,z) \notin b}\ \neg\otimes \qquad \frac{((x,y),z) \notin \mathrm{prj}_1(a,b)}{x \notin a \mid y \notin b \mid z \notin a \mid z \neq x}\ \neg\mathrm{prj}_1$$

$$\frac{((x,y),z) \in \mathrm{prj}_2(a,b)}{x \in a,\ y \in b,\ z \in b,\ z = y}\ \mathrm{prj}_2 \qquad \frac{(x,y),(z,t)) \in a \parallel b}{(x,z) \in a,\ (y,t) \in b}\ \parallel$$

$$\frac{((x,y),z) \notin \mathrm{prj}_2(a,b)}{x \notin a \mid y \notin b \mid z \notin b \mid z \neq y}\ \neg\mathrm{prj}_2 \qquad \frac{(x,y),(z,t)) \notin a \parallel b}{(x,z) \notin a \mid (y,t) \notin b}\ \neg\parallel$$

Rules for Functions

$$\frac{a \in b \nrightarrow c}{\begin{array}{c} T \notin a, (X,Y) \notin a \mid T \notin a, (X,Z) \notin a \mid T \notin a, Y = Z \mid \\ \Delta, (X,Y) \notin a \mid \Delta, (X,Z) \notin a \mid \Delta, Y = Z \end{array}}\ \nrightarrow$$
$$\text{with } \Delta \equiv T = (\epsilon_x, \epsilon_y), \epsilon_x \in b, \epsilon_y \in c$$
$$\text{where } \epsilon_x = \epsilon(x).\exists y\ (T = (x,y) \wedge (x,y) \in b \times c)$$
$$\epsilon_y = \epsilon(y).(T = (\epsilon_x, y) \wedge (\epsilon_x, y) \in b \times c)$$

$$\frac{a \notin b \nrightarrow c}{\begin{array}{c} \epsilon'_x \in a, \epsilon'_x \neq (Y,Z) \mid \epsilon'_x \in a, Y \notin b \mid \epsilon'_x \in a, Z \notin c \mid \\ (\epsilon_x, \epsilon_y) \in a,\ (\epsilon_x, \epsilon_z) \in a,\ \epsilon_y \neq \epsilon_z \end{array}}\ \neg\nrightarrow$$
$$\text{with } \epsilon'_x = \epsilon(x).\neg(x \in a \Rightarrow x \in b \times c)$$
$$\epsilon_x = \epsilon(x).\neg(\forall y \forall z\ ((x,y) \in a \wedge (x,z) \in a \Rightarrow y = z))$$
$$\epsilon_y = \epsilon(y).\neg(\forall z\ ((\epsilon_x, y) \in a \wedge (\epsilon_x, z) \in a \Rightarrow y = z))$$
$$\epsilon_z = \epsilon(z).\neg((\epsilon_x, \epsilon_y) \in a \wedge (\epsilon_x, z) \in a \Rightarrow \epsilon_y = z)$$

Fig. 4. Superdeduction Rules for the B Set Theory (Part 2)

the extensional equality (rules = and $\neq$) together with the substitution rules (rules pred and fun) as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(P(E) \Leftrightarrow P(F))}{\neg P(E), P(F)}\ \text{pred}}{E \neq F}\ \neq}{\epsilon_x \notin E, \epsilon_x \in F \quad\quad \epsilon_x \in E, \epsilon_x \notin F}}{\begin{array}{cc}\vdots & \vdots \\ \odot & \odot\end{array}} \qquad \dfrac{\dfrac{\neg(P(E) \Leftrightarrow P(F))}{P(E), \neg P(F)}\ \beta_{\neg\Leftrightarrow}}{\begin{array}{c}\vdots \\ \odot\end{array}}$$

where $\epsilon_x = \epsilon(x).\neg(x \in E \Leftrightarrow x \in F)$, and where the superdeduction rules for $R : x \in E \rightarrow x \in F$ can be respectively applied on $\epsilon_x \in E$ and $\epsilon_x \notin E$.

5.3 Dealing with Relations

As mentioned previously, the superdeduction rules for relations are generated from proposition rewrite rules of the form $R : (x, y) \in E \rightarrow (x, y) \in F$, where E and F are relations, and the roots of these rules are therefore either $(x, y) \in E$, or $(x, y) \notin E$. As a consequence, these rules cannot be applied to formulas in which pairs are not explicit, such as $x \in E$ or $x \notin E$ for example. To deal with this problem, the idea is to add another proposition rewrite rule for the product, i.e. the rule $R : x \in a \times b \rightarrow \exists y \exists z\ (x = (y, z) \wedge (y, z) \in a \times b)$, which generates the two following superdeduction rules:

$$\dfrac{x \in a \times b}{x = (\epsilon_y, \epsilon_z), \epsilon_y \in a, \epsilon_z \in b}\ \times^* \qquad \dfrac{x \notin a \times b}{x \neq (Y, Z) \mid Y \notin a \mid Z \notin b}\ \neg\times^*$$

where the first rule introduces $\epsilon_y = \epsilon(y).(\exists z\ (x = (y, z) \wedge (y, z) \in a \times b))$, and $\epsilon_z = \epsilon(z).(x = (\epsilon_y, z) \wedge (\epsilon_y, z) \in a \times b))$.

For each relation p, we also need to add another proposition rewrite rule of the form $R : x \in p \rightarrow \exists y \exists z\ (x = (y, z) \wedge (y, z) \in p)$, and then generate the corresponding superdeduction rules. For example, for the inverse relation, we obtain the rules as follows:

$$\dfrac{x \in a^{-1}}{x = (\epsilon_y, \epsilon_z), (\epsilon_z, \epsilon_y) \in a}\ {a^{-1}}^* \qquad \dfrac{x \notin a^{-1}}{x \neq (Y, Z) \mid (Z, Y) \notin a}\ \neg {a^{-1}}^*$$

where the first rule introduces $\epsilon_y = \epsilon(y).(\exists z\ (x = (y, z) \wedge (y, z) \in a^{-1}))$, and $\epsilon_z = \epsilon(z).(x = (\epsilon_y, z) \wedge (\epsilon_y, z) \in a^{-1})$.

With these new rules, we can prove $p \subseteq a \times b \Rightarrow x \in p \Rightarrow x \in (p^{-1})^{-1}$ in the following way (we start the proof after the series of $\alpha_{\neg\Rightarrow}$ has been applied, and we do not detail the definitions of the involved ϵ-terms):

$$\dfrac{\dfrac{p \subseteq a \times b, x \in p, x \notin (p^{-1})^{-1}}{X \notin p}\subseteq \qquad \dfrac{\dfrac{X \in a \times b}{X = (\epsilon_y, \epsilon_z), \epsilon_y \in a, \epsilon_z \in b}\times^* \qquad \dfrac{\dfrac{(Z,Y) \notin p^{-1}}{(Y,Z) \notin p}\neg a^{-1}}{\dfrac{x \neq (Y,Z)}{\vdots}\quad \dfrac{p \neq p}{\odot}}\begin{array}{l}\neg a^{-1^*}\\ \text{pred}\\ \odot_r\end{array}}{\vdots}}{\odot}$$

where X, Y, and Z are respectively instantiated by x, ϵ_y, and ϵ_z.

6 Implementation and Benchmarks

The implementation of our extension of Zenon for the B set theory described in Sec. 5 has been possible thanks to the ability of Zenon to extend its core of deductive rules to match specific requirements like superdeduction. Concretely, this extension is an OCaml file in which the superdeduction rules of Figs. 3 and 4 are implemented (about 2,000 lines of code). This file is loaded through command-line options when Zenon is started, along with a Coq file containing the translation of the superdeduction rules and used to generate Coq proofs.

As said in the introduction, one of the main motivations for this extension of Zenon for the B set theory is to verify B proof rules of Atelier B [9], and in particular rules coming from the database maintained by Siemens IC-MOL. Concretely, a rule is mainly a set formula with guards, which are used to control the application of the rule (verifying that a given hypothesis is in the context, for example). The verification of a rule has been formally described in [15], and consists in verifying that the rule is well-typed, well-defined, and that it can be derived using the rules of the B proof system (see [1]). Over the last few years, Siemens IC-MOL has developed a formal and mechanized environment, named BCARe, which is dedicated to the rule verification, and which relies in particular on a deep embedding of the B set theory in Coq, called BCoq. In this environment, our extension of Zenon is supposed to deal with the last step of verification of a rule, i.e. the derivation of the rule within the B proof system.

Regarding benchmarks, we consider a selection of B proof rules coming from the database maintained by Siemens IC-MOL. This selection actually consists of well-typed and well-defined rules, which involve all the B set constructs currently handled by the implementation of our extension of Zenon, i.e. all the constructs of Chap. 2 of the B-Book [1] until the override construct (noted $\lhd\!\!+$). This represents a subset of 1,397 rules, and we propose two benchmarks whose results are gathered in Figs. 5 and 6.

The first benchmark aims to compare our extension of Zenon with the approach coming from another experiment described in [15]. In this approach, Zenon is simply used as a first order theorem prover (no set theory is added),

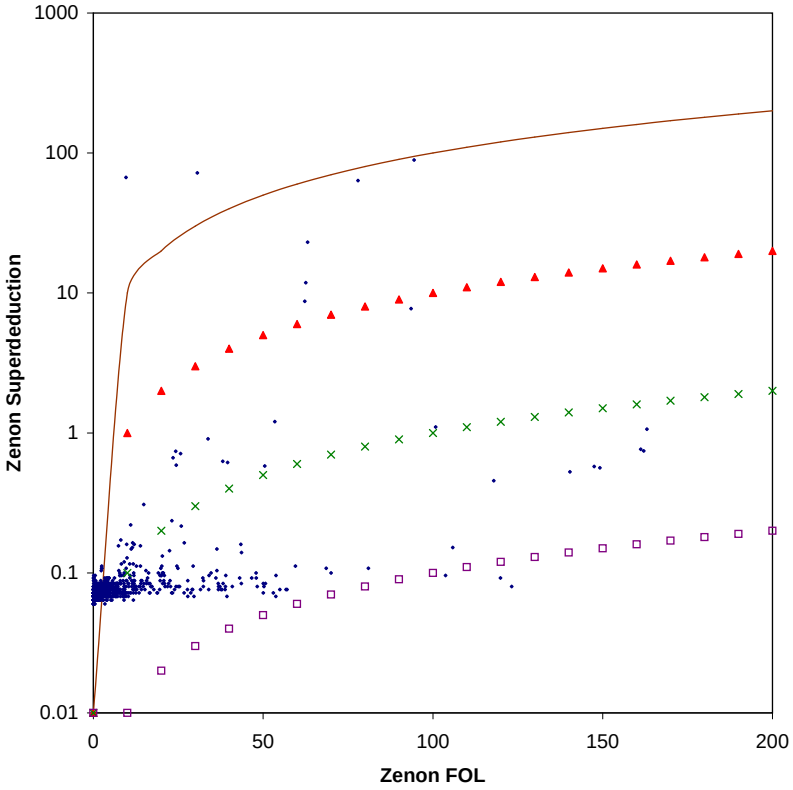

Fig. 5. Proof Time Comparative Benchmark

and the set formulas must be preliminarily normalized in order to obtain first order logic formulas containing only the $\in$ set operator (which is considered as an uninterpreted predicate symbol). This pre-normalization is performed in Coq (within the BCoq embedding), before calling Zenon. Over the 1,397 selected rules, our extension of Zenon is able to prove 1,340 rules (96%), while our initial approach can deal with 1,145 rules (82%), which represents an increase of 195 rules (14%). The difference of proved rules mostly comes from ineffective parts of the implementation of our initial approach, which still have to be optimized, and is not due to the incompleteness of this approach (which is not pointed out over the considered subset of rules).

The graph of Fig. 5 presents a comparison of both approaches in terms of proof time (run on an Intel Core i5-2500K 3.30GHz/12GB computer) for a subset of the 1,397 selected rules, where both approaches succeed in finding a proof (the time measurement includes the compilation of Coq proofs generated by Zenon), i.e. for 1,145 rules. In this figure, a point represents the result for a rule, and the x/y-axes respectively correspond to the Zenon approach with pre-normalization

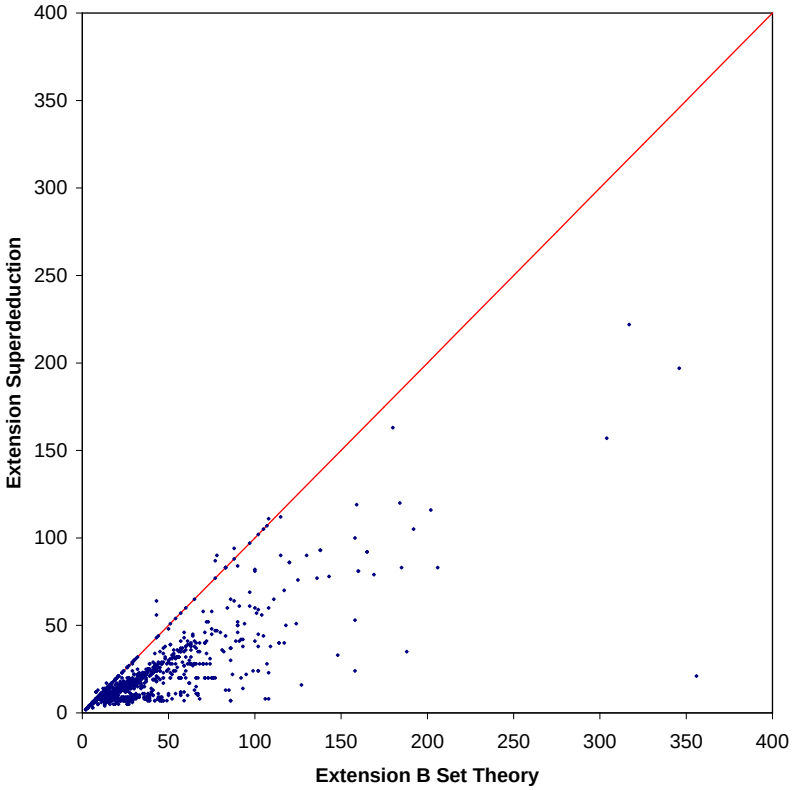

Fig. 6. Proof Size Comparative Benchmark

of the formulas and to our extension of Zenon using superdeduction. For each point under the triangled curve, our extension based on superdeduction is at least 10 times faster than our initial approach with pre-normalization of the formulas; under the crossed curve, it is 100 times faster, and under the squared curve, it is 1,000 times faster. We can observe that there is a lot of rules for which the proof is 10 or 100 times faster with superdeduction, and on average, the superdeduction proofs are obtained 67 times faster (the best ratio is 1,540). These results are quite satisfactory in the sense that our extension based on superdeduction solves the inefficiency issues of our initial approach (mainly due to the pre-normalization, which is highly time-consuming).

We propose a second benchmark whose purpose is to emphasize the proof size speed-up offered by superdeduction. In this benchmark, the idea is to compare our extension of Zenon using superdeduction with another extension of Zenon for the B set theory, where the proposition rewrite rules are not computed into superdeduction rules, but just unfolded/folded (like in Prawitz's initial approach [20]). The comparison consists in computing the number of proof nodes

of each proof generated by Zenon. We consider a subset of 1,340 rules, i.e. the rules for which both extensions succeed in finding a proof. The results are summarized by the graph of Fig. 6, where a point represents the result for a rule, and where the x/y-axes respectively correspond to the extension without and with superdeduction. As can be seen, the major part of proofs in superdeduction are shorter than the proofs where the rules have not been computed and on average, the former have 1.6 times less proof nodes than the latter (the best ratio is 6.25).

The previous benchmarks tend to show that the use of our extension of Zenon for the B set theory and based on superdeduction is quite effective in practice, and very promising in terms of scalability. In particular, this could be also applied to the verification of B proof obligations (which involve much larger formulas than B proof rules).

7 A Generic Implementation for First Order Theories

In this section, we present another extension of Zenon with superdeduction, which is able to deal with any first order theory. In this extension, the theory is analyzed to determine the axioms that can be turned into superdeduction rules, and these superdeduction rules are automatically computed on the fly to enrich the deductive kernel of Zenon. We also provide a comparative benchmark coming from the TPTP library, which contains a large set of first order problems, and which is usually used to test the implementations of automated theorem provers.

7.1 From Theories to Superdeduction Systems

This extension of Zenon is actually a generalization of the previous one dedicated to the B set theory, where superdeduction rules are automatically computed on the fly. In the previous extension, superdeduction rules are hard-coded since the B set theory is a higher order theory due to one of the axioms of the theory (the comprehension scheme), and we have to deal with this axiom specifically in the implementation of Zenon. Even though some techniques exist to handle higher order theories as first order theories (like the theory of classes, for example), a hard-coding of these theories may be preferred as these techniques unfortunately tend to increase the entropy of the proof search. In addition, in the previous extension, some of the superdeduction rules must be manually generated as they must be shrewdly tuned (ordering the several branches of the rules, for instance) to make the tool efficient. The new extension of Zenon dealing with any first order theory has been developed as a tool called Super Zenon [16], where each theory is analyzed to determine the axioms that are candidates to be turned into superdeduction rules. As said in Sec. 4, axioms of the form $\forall \bar{x}\ (P \Leftrightarrow \varphi)$, where P is atomic, can be transformed, but we can actually deal with more axioms, in particular with axioms of the form $\forall \bar{x}\ (P \Rightarrow \varphi)$, which is actually equivalent to perform polarized deduction modulo as introduced in [12]. Here is the exhaustive list of axioms that can be handled by Super Zenon, as well as the corresponding superdeduction rules that can be generated (in the following, P and P' are atomic, and φ is an arbitrary formula):

- Axiom of the form $\forall \bar{x}\ (P \Leftrightarrow \varphi)$: we consider the proposition rewrite rule $R : P \to \varphi$, and the two superdeduction rules R and $\neg R$ are generated;
- Axiom of the form $\forall \bar{x}\ (P \Rightarrow P')$: we consider the proposition rewrite rules $R : P \to P'$ and $R' : \neg P' \to \neg P$, and only the superdeduction rules R and R' are generated;
- Axiom of the form $\forall \bar{x}\ (P \Rightarrow \varphi)$: we consider the proposition rewrite rule $R : P \to \varphi$, and only the superdeduction rule R is generated;
- Axiom of the form $\forall \bar{x}\ (\varphi \Rightarrow P)$: we consider the proposition rewrite rule $R : \neg P \to \neg\varphi$, and only the superdeduction rule R is generated;
- Axiom of the form $\forall \bar{x}\ P$: we consider the degenerated proposition rewrite rule $R : \neg P \to \bot$, and only the superdeduction rule R is generated.

The axioms of the theory that are not of these forms are left as regular axioms. An axiom that is of one of these forms is also left as a regular axiom if the conclusion of one of the generated superdeduction rules (i.e. the top formula of one of these rules) unifies with the conclusion of an already computed superdeduction rule (in this case, the theory is actually non-deterministic, and we try to minimize this source of non-determinism by dividing these incriminated axioms among the sets of superdeduction rules and regular axioms). An axiom that is of one of these forms is still left as a regular axiom if P is an equality (as we do not want to interfere with the specific management of equality by the kernel of Zenon). Finally, for axioms of the form $\forall \bar{x}\ (P \Rightarrow P')$, we also consider the proposition rewrite rule that corresponds to the converse of the initial formula; this actually allows us to keep cut-free completeness in this particular case.

7.2 Benchmark from the TPTP Library

To assess the effectiveness of Super Zenon compared to the regular version of Zenon, we propose a benchmark that consists of problems coming from the TPTP library [23]. This library is a large collection of problems, which is used to test the implementations of automated theorem provers in particular. From this library, we consider a selection of problems that consists of the set of non-clausal first order problems, i.e. the problems of the FOF category in the TPTP library. The results of this experiment (run on an Intel Pentium D Xeon 3.60GHz/32GB computer) are summarized in Tab. 1, where both Zenon and Super Zenon are called over the problems of the FOF category. As can be observed, Super Zenon is able to prove more problems than Zenon in the whole FOF category with a significant rate (about 7%). This rate becomes even much better in some specific sub-categories of the FOF category, like in the SET sub-category (about 37%), which gathers problems of set theory. These results in the SET sub-category tend to confirm the other results described in Sec. 6 in the framework of the B method, and also tend to show that set theory is an appropriate theory to be handled by superdeduction and by deduction modulo more generally. In addition to the problems of the FOF category, Super Zenon has been also applied to the counter-satisfiable problems of the TPTP library, i.e. problems that are not valid, and Super Zenon does not find any proof for these problems, which allows us to have a relative confidence in the correctness of the implementation of this tool.

TPTP Category	Zenon	Super Zenon
FOF 6,644 problems	1,646	1,765 (7.2%)
SET 462 problems	147	202 (37.4%)

Table 1. Experimental Results over the TPTP Library

8 Superdeduction for Automated Deduction

Superdeduction can be seen as a generic method to integrate axiomatic first order theories into deductive formal proof systems, like sequent calculus, as well as into proof search methods that are very closed to deductive formal proof systems, like the tableau method, which is a proof search method in sequent calculus without cut. In particular, as seen previously, it has allowed us to smoothly integrate the set theory of the B method into the tableau method, which has been implemented as an extension of the Zenon automated theorem prover, and which has been used for the verification of B proof rules. The considered approach is actually so generic that it has been generalized into a new extension of Zenon, called Super Zenon, which is able to deal with any axiomatic first order theory. Thus, any user that aims to perform automated deduction in a given axiomatic first order theory must just provide the theory to Super Zenon, which is able to automatically integrate a part of this theory as superdeduction rules. The approach is even more generic since it can be actually implemented in any tool based on a tableau method. As for the application of superdeduction to other proof search methods, such as resolution for example, the approach is probably less straightforward since resolution proofs are actually far from proofs in usual deductive formal proof systems, like sequent calculus, which makes the introduction of superdeduction rules harder as these rules are pure deduction rules. However, some work has been done in this direction in the framework of deduction modulo [13] (whose superdeduction may be considered as a variant), with a concrete development, called iProver Modulo [8], implemented as an extension of iProver [17], which is a resolution-based automated theorem prover.

9 Conclusion

We have proposed a method that allows us to develop tableaux modulo theories using superdeduction. This method has been presented in the framework of the Zenon automated theorem prover, and applied to the set theory of the B method. This has allowed us to provide another prover to Atelier B, which can be used to verify B proof rules automatically. We have also proposed some benchmarks using rules coming from the database maintained by Siemens IC-MOL. These

benchmarks have emphasized significant speed-ups both in terms of proof time and proof size compared to previous and alternative approaches. Finally, we have described another extension of Zenon with superdeduction, called Super Zenon, which is able to deal with any first order theory. In this extension, the theory is analyzed to determine the axioms that can be turned into superdeduction rules, and these superdeduction rules are automatically computed on the fly to enrich the deductive kernel of Zenon. A comparative benchmark that consists of a large set of first order problems coming from the TPTP library has shown a significant improvement of Super Zenon over the regular version of Zenon.

As future work, we must extend our specific implementation realized for verifying B proof rules in order to deal with a larger set of rules coming from the database maintained by Siemens IC-MOL. More precisely, the next step is to consider functions, and the corresponding rules should point out the incompleteness of our initial approach that consists in pre-normalizing the set formulas. For instance, formulas such as $\{ (x, y) \mid x = y \} \in a \nrightarrow a \Rightarrow \exists f \ (f \in a \nrightarrow a \wedge (b, b) \in f)$ should be proved by our extension based on superdeduction, but not by our initial approach (since an additional normalization is required once f has been instantiated by the comprehension set in hypothesis). We also plan to enhance the heuristic used by Super Zenon to transform axioms into superdeduction rules in order to increase the improvement of Super Zenon over Zenon in the framework of generic theories. In addition, it should be worth building an heuristic with transformation rules that preserves some important properties (from the automated deduction point of view) of the initial axiomatic theories, such as cut-free completeness for example.

Acknowledgement. Many thanks to G. Burel and O. Hermant for their detailed comments on this paper, to G. Dowek for seminal discussions about this work, and to D. Doligez for his help in the integration of superdeduction into Zenon.

References

1. J.-R. Abrial. *The B-Book: Assigning Programs to Meanings.* Cambridge University Press, Cambridge (UK), 1996. ISBN 0-521-49619-5.
2. J.-R. Abrial and L. Mussat. On Using Conditional Definitions in Formal Theories. In *Formal Specification and Development in Z and B (ZB)*, volume 2272 of *LNCS*, pages 317–322, Grenoble (France), Jan. 2002. Springer.
3. J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation (JLC)*, 2(3):297–347, June 1992.
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. P. Brauner, C. Houtmann, and C. Kirchner. Principles of Superdeduction. In *Logic in Computer Science (LICS)*, pages 41–50, Wrocław (Poland), July 2007. IEEE Computer Society Press.
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. ClearSy. *Atelier B 4.1.1*, Dec. 2013. `http://www.atelierb.eu/`.
10. S. Colin, D. Petit, V. Poirriez, J. Rocheteau, R. Marcano, and G. Mariano. BRILLANT: An Open Source and XML-based platform for Rigourous Software Development. In *Software Engineering and Formal Methods (SEFM)*, pages 373–382, Koblenz (Germany), Sept. 2005. IEEE Computer Society Press.
11. D. Delahaye and M. Jacquel. Recovering Intuition from Automated Formal Proofs using Tableaux with Superdeduction. *Electronic Journal of Mathematics and Technology (eJMT)*, 7(2), Feb. 2013.
12. 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.
13. G. Dowek, T. Hardin, and C. Kirchner. Theorem Proving Modulo. *Journal of Automated Reasoning (JAR)*, 31(1):33–72, Sept. 2003.
14. D. Déharbe. Integration of SMT-Solvers in B and Event-B Development Environments. *Science of Computer Programming (SCP)*, 78(3):310–326, Mar. 2013.
15. M. Jacquel, K. Berkani, D. Delahaye, and C. Dubois. Verifying B Proof Rules using Deep Embedding and Automated Theorem Proving. *Software and Systems Modeling (SoSyM)*, pages 1–19, June 2013.
16. M. Jacquel and D. Delahaye. *Super Zenon, version 0.0.1*. Siemens and Cnam, May 2012. `http://cedric.cnam.fr/~delahaye/super-zenon/`.
17. 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.
18. G. Nelson and D. C. Oppen. Simplification by Cooperating Decision Procedures. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1(2):245–257, Oct. 1979.
19. D. Pastre. Strong and Weak Points of the Muscadet Theorem Prover – Examples from CASC-JC. *AI Communications*, 15(2–3):147–160, July 2002.
20. D. Prawitz. Natural Deduction. A Proof-Theoretical Study. *Stockholm Studies in Philosophy*, 3, 1965.
21. M. Schmalz. Term Rewriting in Logics of Partial Functions. In *International Conference on Formal Engineering Methods (ICFEM)*, volume 6991 of *LNCS*, pages 633–650, Durham (UK), Oct. 2011. Springer.
22. R. E. Shostak. Deciding Combinations of Theories. *Journal of the Association for Computing Machinery (JACM)*, 31(1):1–12, Jan. 1984.
23. 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.
24. The Coq Development Team. *Coq, version 8.4pl4*. Inria, May 2014. `http://coq.inria.fr/`.
25. W. Windsteiger. An Automated Prover for Zermelo-Fraenkel Set Theory in Theorema. *Journal of Symbolic Computation (JSC)*, 41(3–4):435–470, Mar. 2006.