Soundly Proving B Method Formulæ Using Typed Sequent Calculus

Conference Paper *in* Lecture Notes in Computer Science · October 2016

DOI: 10.1007/978-3-319-46750-4_12

1 author:

Pierre Halmagrand
Électricité de France (EDF)
8 PUBLICATIONS **93** CITATIONS

SEE PROFILE

Soundly Proving B Method Formulæ Using Typed Sequent Calculus *

Pierre Halmagrand pierre.halmagrand@inria.fr

Cnam / Inria / ENS Cachan, Paris, France

Abstract. The B Method is a formal method mainly used in the railway industry to specify and develop safety-critical software. To guarantee the consistency of a B project, one decisive challenge is to show correct a large amount of proof obligations, which are mathematical formulæ expressed in a classical set theory extended with a specific type system. To improve automated theorem proving in the B Method, we propose to use a first-order sequent calculus extended with a polymorphic type system, which is in particular the output proof-format of the tableau-based automated theorem prover Zenon. After stating some modifications of the B syntax and defining a sound elimination of comprehension sets, we propose a translation of B formulæ into a polymorphic first-order logic format. Then, we introduce the typed sequent calculus used by Zenon, and show that Zenon proofs can be translated to proofs of the initial B formulæ in the B proof system.

1 Introduction

Automated transport systems have spread in many cities during last decades, becoming a leading sector for the development of highly trusted software using formal methods. The B Method [1] is a formal method mainly used in the railway industry to specify and develop safety-critical software. It allows the development of correct-by-construction programs, thanks to a refinement process from an abstract specification to a deterministic implementation of the program. The soundness of the refinement steps depends on the validity of logical formulæ called proof obligations, expressed in a specific typed set theory. Common industrial projects using the B Method generate thousands of proof obligations, thereby relying on automated tools to discharge as many as possible proof obligations. A specific tool, called Atelier B [18], designed to implement the B Method and provided with a theorem prover, helps users verify the validity of proof obligations, automatically or interactively.

Improving the automated verification of proof obligations is a crucial task. The BWare research project [10] proposed to use external automated provers, like first-order Automated Theorem Provers (ATPs) and Satisfiability Modulo Theory (SMT) solvers, by building a common platform to run these tools. This

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

platform, based on the software verification tool Why3 [3], requires proof obligations to be encoded in its native language called WhyML. Mentré *et al.* [16] proposed a translator program called bpo2why to address this issue. This tool focuses on the translation of B proof obligations output by Atelier B into the WhyML language. Besides, the B set theory is defined directly in WhyML. Then, Why3 uses specific drivers to translate proof obligations and the theory from WhyML to the specific format of each automated tool.

The first-order ATP Zenon [6] [9], based on the tableau method and recently extended to deal with polymorphic types, has been used to prove B proof obligations in the BWare project and obtained good experimental results, compared to the regular version of Zenon and other automated deduction tools as well [7]. One important feature of Zenon is to be a certifying prover [8], in the sense that it generates proof certificates, *i.e.* proof objects that can be verified by external proof checkers. It relies on an encoding of the output proof-format of Zenon, a typed sequent calculus called LLproof, into the proof checker Dedukti [5], a tool designed to be a universal backend to certify and share proofs coming from automated or interactive provers. These proof certificates allow us to be very confident about the soundness of the proofs produced by Zenon.

An issue about using Zenon to verify B proof obligations arises in the upstream translation chain, from B to Zenon input format. There is currently no formal guarantee that this chain is sound, and it would be a tremendous work to formalize the several steps represented by bpo2why and Why3. Instead, we decide to confirm the soundness of using Zenon to prove B proof obligations by formalizing a more general and direct translation from the B logic into a polymorphic first-order logic (PFOL for short), close to the Zenon input format. This translation is to be proven sound, in the sense that if Zenon finds a proof then it can be turned into a proof of the initial B formula. A solution is to show a logical equivalence between Zenon and B proof system.

The B set theory is provided with a specific type system, expressed using set constructs, resulting in a lack of separation in a B formula between typing and set reasoning. To help us embed B typing constraints into PFOL, we define a procedure to annotate B variables with their types, using the type-checking algorithm of the B Method. The interpretation of these types will then be given by the translation function into PFOL. Axioms and hypotheses are generalized by translating B types to (universally quantified) type variables in PFOL. In contrast, types coming from the formula to be proved are interpreted as type constants in PFOL. In addition, we define the reverse translation from PFOL to B, letting us to reword the initial B formula. Thanks to this reverse translation and the derivations of Zenon inference rules expressed using the B proof system, we can translate Zenon proofs into B proofs, guaranteeing the soundness of our translation.

The concerns about the confidence given to ATP in the case of B proof have been resolved using the alternative approach of a certified prover and relying on a deep embedding of the B logic into the interactive prover Coq, by Jaeger *et al.* [14]. It has also been studied in the context of Event-B by Schmalz in [17].

The problem of type inference in the B Method was studied in other contexts, see for instance [4] for an embedding into PVS, and [13] for Coq.

This paper is organized as follows: in Sec. 2, we introduce the B Method syntax, proof system and type system, then we introduce the type annotation procedure; in Sec. 3, we present the polymorphic first-order logic and the typed sequent calculus used by Zenon; in Sec. 4, we give the translation used to encode B formulæ into PFOL; finally, in Sec. 5, we present the translation of proofs expressed in the sequent calculus of Zenon into B proofs.

2 The B Set Theory

In this section, we present the core logic and theory of the B Method. We first introduce the syntax, the proof system, the set theory and the typing rules of the B Method. Then, we introduce a procedure to annotate variables with their corresponding types. Finally, we present an elimination procedure of comprehension sets.

2.1 Syntax, Proof System and Set Theory

The presentation below follows faithfully the first two chapters of the B-Book [1] dealing with mathematical reasoning and set theory.

Syntax. The syntax of the B Method is made of four syntactic categories, for formulæ, expressions, variables and sets. A formula, or predicate, P is built from the logical connectives conjunction, implication and negation and the universal quantification. A formula may also be the result of a substitution in a formula, the equality between two expressions and the membership to a set. An expression E may be a variable, the result of a substitution in an expression, an ordered pair, an arbitrary element in a set or a set. A variable x is either an identifier or a list of variables. Finally, a set s is built using the elementary set constructs, *i.e.* the cartesian product, the powerset and the comprehension set, or may be the set BIG, a given infinite set.

$$
\begin{array}{lcl}
P & ::= & P_1 \wedge P_2 \mid P_1 \Rightarrow P_2 \mid \neg P \mid \forall x \cdot P \mid [x := E]P \mid E_1 = E_2 \mid E \in s \\
E & ::= & x \mid [x := E_1]E_2 \mid E_1, E_2 \mid \mathsf{choice}(s) \mid s \\
x & ::= & identifier \mid x_1, x_2 \\
s & ::= & s_1 \times s_2 \mid \mathbb{P}(s) \mid \{x \mid P\} \mid \mathsf{BIG}
\end{array}
$$

Proof System. In Fig. 1, we present the proof system of the B Method. This is an adaptation of classical natural deduction for the B syntax. It should be noted that the B-Book proposes some rules to define the notion of non-freeness of a variable x in a formula P, denoted by $x \backslash P$. Since these rules are standard, we omit them here.

In the following, if x is a variable and Γ a set of formulæ, $x \backslash \Gamma$ means that $x \backslash H$ for each H of Γ; if Γ' is another set of formulæ, $\Gamma \sqsubset \Gamma'$ means that Γ is included in Γ'; and if P is a formula, $P \sqsubseteq \Gamma$ means that P occurs in Γ.

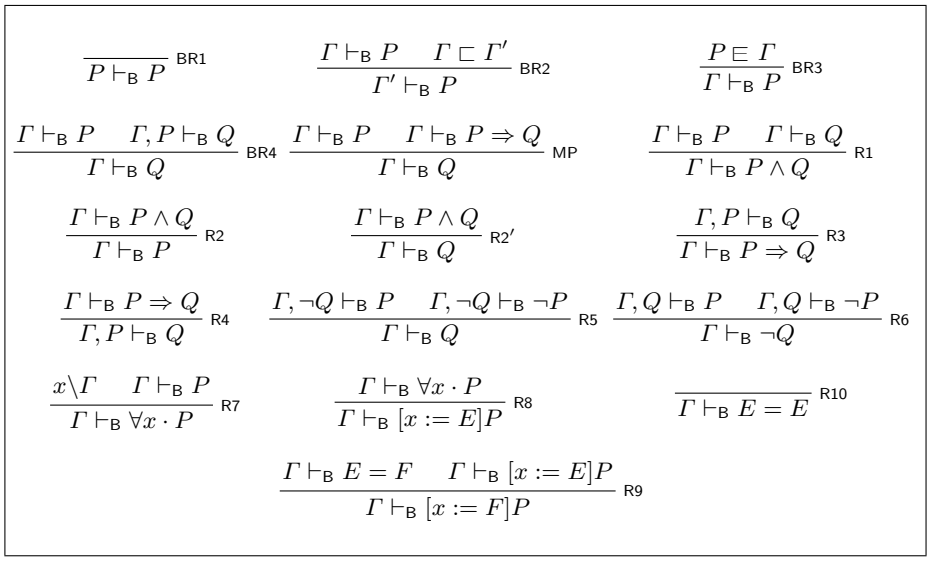

Fig. 1. The Proof System of the B Method

Set Theory. As presented in the B-Book, the B Method set theory is a simplification of classical set theory. Some common axioms, like the foundation axiom, are not needed in this context (see Sec. 2.2), leading to a theory made only of six axioms. Actually, axioms presented below are axiom schemata that have to be instantiated with some proper expressions. The first column represents non-freeness proviso.

	$E, F \in s \times t \Leftrightarrow (E \in s \wedge F \in t)$	SET1
$x \backslash (s,t)$	$s \in \mathbb{P}(t) \Leftrightarrow \forall x \cdot (x \in s \Rightarrow x \in t)$	SET2
$x \backslash s$	$E \in \{x \mid x \in s \wedge P\} \Leftrightarrow (E \in s \wedge [x := E]P)$	SET3
$x \backslash (s,t)$	$\forall x \cdot (x \in s \Leftrightarrow x \in t) \Rightarrow s = t$	SET4
$x \backslash s$	$\exists x \cdot (x \in s) \Rightarrow \mathsf{choice}(s) \in s$	SET5
	$\mathsf{infinite}(\mathsf{BIG})$	SET6

Remark 1. The B-Book defines rewrite rules for secondary common constructs:

$$P \vee Q \rightarrow \neg P \Rightarrow Q \qquad P \Leftrightarrow Q \rightarrow (P \Rightarrow Q) \wedge (Q \Rightarrow P) \qquad \exists x \cdot P \rightarrow \neg \forall x \cdot \neg P$$
$$s \subseteq t \rightarrow s \in \mathbb{P}(t) \qquad s \subset t \rightarrow s \subseteq t \wedge s \neq t$$

2.2 Type System

The B Method set theory differs from other ones, like the Zermelo-Fraenkel set theory. The main difference consists in the addition of typing constraints to expressions, and the application of a type-checking procedure before proving. This avoids ill-formed formulæ such as $\exists x \cdot (x \in x)$, whose negation is provable in ZF, due to the foundation axiom, unlike for the B Method.

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P) \quad \Delta \vdash_{\mathsf{tc}} \mathsf{ch}(Q)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P \wedge Q)}\ \mathsf{T1} \qquad \frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P) \quad \Delta \vdash_{\mathsf{tc}} \mathsf{ch}(Q)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P \Rightarrow Q)}\ \mathsf{T2}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\neg P)}\ \mathsf{T3} \qquad \frac{x \backslash s \quad x \backslash \Delta \quad \Delta, x \in s \vdash_{\mathsf{tc}} \mathsf{ch}(P)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\forall x \cdot (x \in s \Rightarrow P))}\ \mathsf{T4}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\forall x \cdot (x \in s \Rightarrow \forall y \cdot (y \in t \Rightarrow P)))}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\forall (x,y) \cdot (x, y \in s \times t \Rightarrow P))}\ \mathsf{T5} \qquad \frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\forall x \cdot (P \Rightarrow (Q \wedge R)))}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\forall x \cdot ((P \wedge Q) \Rightarrow R))}\ \mathsf{T6}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(E) \equiv \mathsf{ty}(F)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(E = F)}\ \mathsf{T7} \qquad \frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(E) \equiv \mathsf{su}(s)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(E \in s)}\ \mathsf{T8}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \equiv \mathsf{su}(t)}{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(s \subseteq t)}\ \mathsf{T8'} \qquad \frac{x \in s \sqsubseteq \Delta \quad \Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(x) \equiv U}\ \mathsf{T9}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(E) \times \mathsf{ty}(F) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(E, F) \equiv U}\ \mathsf{T10} \qquad \frac{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(\mathsf{choice}(s)) \equiv U}\ \mathsf{T11}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathbb{P}(\mathsf{su}(s)) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(s) \equiv U}\ \mathsf{T12} \qquad \frac{x \in s \sqsubseteq \Delta \quad \Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \equiv \mathbb{P}(U)}{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(x) \equiv U}\ \mathsf{T13}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \times \mathsf{su}(t) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(s \times t) \equiv U}\ \mathsf{T14} \qquad \frac{\Delta \vdash_{\mathsf{tc}} \mathbb{P}(\mathsf{su}(s)) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(\mathbb{P}(s)) \equiv U}\ \mathsf{T15}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(\forall x \cdot (x \in s \Rightarrow P)) \quad \Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(\{x \mid x \in s \wedge P\}) \equiv U}\ \mathsf{T16} \qquad \frac{\mathsf{gi}(I) \sqsubseteq \Delta \quad \Delta \vdash_{\mathsf{tc}} I \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(I) \equiv U}\ \mathsf{T17}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(s) \equiv \mathbb{P}(U)}{\Delta \vdash_{\mathsf{tc}} \mathsf{su}(\mathsf{choice}(s)) \equiv U}\ \mathsf{T18} \qquad \frac{\Delta \vdash_{\mathsf{tc}} T \equiv U}{\Delta \vdash_{\mathsf{tc}} \mathbb{P}(T) \equiv \mathbb{P}(U)}\ \mathsf{T19}$$

$$\frac{\Delta \vdash_{\mathsf{tc}} T \equiv U \quad \Delta \vdash_{\mathsf{tc}} V \equiv W}{\Delta \vdash_{\mathsf{tc}} T \times V \equiv U \times W}\ \mathsf{T20} \qquad \frac{\mathsf{gi}(I) \sqsubseteq \Delta}{\Delta \vdash_{\mathsf{tc}} I \equiv I}\ \mathsf{T21}$$

Fig. 2. The Type System of the B Method

The typing discipline proposed relies on the monotonicity of set inclusion. For instance, if we have an expression E and two sets s and t such that $E \in s$ and $s \subseteq t$, then $E \in t$. Going further with another set u such that $t \subseteq u$, we have then $E \in u$. The idea, as explained in the B-Book, is that, given a formula to be type checked, there exists an upper limit for such set containment. This upper limit is called the super-set of s and the type of E. Then, if u is the super-set of s, we obtain the typing information $E \in u$ and $s \in \mathbb{P}(u)$.

Type checking is performed by applying, in a backward way and following the numerical order, the inference rules presented in Fig. 2. Rules dealing with the right-hand side of a typing equivalence $\equiv$ are named with the same number

primed, for T9 to T18. If this decision procedure terminates and does not fail, then the formula is said to be well-typed. This procedure uses two syntactic categories $Type$ and $Type\_Pred$:

$$\begin{array}{rcl} Type & ::= & \mathsf{type}(E) \mid \mathsf{super}(s) \mid Type \times Type \mid \mathbb{P}(Type) \mid identifier \\ Type\_Pred & ::= & \mathsf{check}(P) \mid Type \equiv Type \end{array}$$

In the following, we use ty, su and ch as abbreviations for the keywords type, super and check respectively. As a consequence, the type of an expression E may be either an identifier (see the notion of given set below), the powerset of a type or the cartesian product of two types; and for the particular case of sets, the type of a set is necessarily the powerset of a type.

A type-checking sequent like $\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P)$ means that, within the environment Δ, the formula P is well-typed. The environment Δ is made of atomic formulæ of the form $x \in s$, where x is non-free in s. All free variables in P have to be associated with some atomic formula in Δ. The only exception is for variables in P representing some abstract given sets, introduced at a meta-level discourse like: "Given a set s ...". Such a given set s, which will be used to type other sets, is introduced in the environment Δ by the keyword $\mathsf{given}(s)$ ($\mathsf{gi}(s)$ for short), telling us that s is free in the formula to be type-checked, and has the specific property $\mathsf{su}(s) = s$.

Example 1. Given two sets s and t, the formula:

$$\forall (a,b) \cdot (a,b \in \mathbb{P}(s \times t) \times \mathbb{P}(s \times t) \Rightarrow \{x \mid x \in a \land x \in b\} \subseteq s \times t)$$

will be used as a running example in this paper. We want to verify that this formula is well-typed, *i.e.* verify that the following sequent is satisfied:

$$\begin{array}{l} \mathsf{gi}(s), \mathsf{gi}(t) \vdash_{\mathsf{tc}} \\ \quad \mathsf{ch}(\forall (a,b) \cdot (a,b \in \mathbb{P}(s \times t) \times \mathbb{P}(s \times t) \Rightarrow \{x \mid x \in a \land x \in b\} \subseteq s \times t)) \end{array}$$

By applying the rules of Fig. 2, we obtain the following typing derivation (due to the large size of the tree, we present only the names of rules, starting from the left with T5):

$$\mathsf{T5}\text{-}\mathsf{T4}\text{-}\mathsf{T4}\text{-}\mathsf{T8'}\text{-}\mathsf{T14'}\text{-}\mathsf{T16}\text{-}\left\{ \begin{array}{l} \mathsf{T4}\text{-}\mathsf{T8}\text{-}\mathsf{T9}\text{-}\mathsf{T13}\text{-}\mathsf{T15}\text{-}\mathsf{T19}\text{-}\mathsf{T13'}\text{-}\mathsf{T15'}\text{-}\mathsf{T19}\text{-}\mathsf{T14}\text{-}\mathsf{T14'}\text{-}\mathsf{T20}\text{-}\left\{ \begin{array}{l} \mathsf{T17}\text{-}\mathsf{T17'}\text{-}\mathsf{T21} \\ \mathsf{T17}\text{-}\mathsf{T17'}\text{-}\mathsf{T21} \end{array} \right. \\ \mathsf{T13}\text{-}\mathsf{T15}\text{-}\mathsf{T19}\text{-}\mathsf{T14}\text{-}\mathsf{T20}\text{-}\left\{ \begin{array}{l} \mathsf{T17}\text{-}\mathsf{T17'}\text{-}\mathsf{T21} \\ \mathsf{T17}\text{-}\mathsf{T17'}\text{-}\mathsf{T21} \end{array} \right. \end{array} \right.$$

2.3 Type Annotation

In the B syntax presented in Sec. 2.1, there are two constructs which introduce new bound variables: universal quantification $\forall x \cdot P$ and comprehension set $\{x \mid P\}$. It should be noted that the typing rules T4 and T16 dealing with these two syntactical constructs use the specific forms $\forall x \cdot x \in s \Rightarrow P$ and $\{x \mid x \in$

$s \wedge P\}$. Membership $x \in s$ is used to type the bound variable x. Unfortunately, typing information is hidden at a set theoretic level. There is no clear distinction between sets and types in the B Method.

For the translation function presented in Sec. 4.2, we want to distinguish the notion of types from the one of sets. We introduce a new syntactic category T for types:

$$T \ ::= \ identifier \mid T_1 \times T_2 \mid \mathbb{P}(T)$$

And we introduce the notation x^T meaning that the variable x has type T.

We now present a procedure to annotate variables with their type. Once the type-checking of a formula is done, the typing tree has environments Δ at each node, and in particular at leaves, following the syntax:

$$\Delta \ ::= \ \varnothing \mid \Delta, \mathsf{gi}(s) \mid \Delta, x \in s$$

In addition, Δ is augmented only by rule T4: if a formula $x \in s$ is added, then s has to be already associated in Δ (in particular because of rules T9 and T13), as a given set or in a formula like $s \in t$ for some already associated set t.

The annotation procedure transforms all the leaf environments Δ, *i.e.* the environments of the leaves, into annotated environments $\Delta^\star$, where all variables and given sets are annotated with their type, then uses these annotated environments to rebuild the typing tree of the (annotated) initial formula in a forward way. It should be noted that in a formula $x \in s$, the set s may be a composition of the two type constructors $\times$ and $\mathbb{P}$. We denote this kind of composition by a function symbol f with an arity n. Here is the syntax for $\Delta^\star$:

$$\Delta^\star \ ::= \ \varnothing \mid \Delta^\star, \mathsf{gi}(s^{\mathbb{P}(s)}) \mid \Delta^\star, x^{f(T_1,\ldots,T_n)} \in f(s_1^{\mathbb{P}(T_1)}, \ldots, s_n^{\mathbb{P}(T_n)})$$

We can now introduce the annotation procedure:

1. For all the leaf environments Δ:
   1.1. For all $\mathsf{gi}(s)$, we annotate s by its type $\mathbb{P}(s)$, and then substitute all occurrences of s in Δ by $s^{\mathbb{P}(s)}$;
   1.2. Following the introduction order in Δ, for all $x \in f(s_1^{\mathbb{P}(T_1)}, \ldots, s_n^{\mathbb{P}(T_n)})$, we annotate x with its type $f(T_1, \ldots, T_n)$, and we substitute all occurrences of x in Δ by $x^{f(T_1,\ldots,T_n)}$;
2. Rebuild the (annotated) initial formula by applying the type-checking tree in a forward way, *i.e.* from the leaves to the root.

In the following, we denote by $P^\star$ the formula P where all variables are annotated. We extend this notation to sets of formulæ Γ, and expressions E.

Proposition 1. *The annotation is sound.*
We have, for a variable x, an expression E and a formula P:

1. *If x^T is associated in $\Delta^\star$, $\Delta^\star \vdash_{\mathsf{tc}} \mathsf{ty}(x^T) \equiv T$;*
2. *If $\Delta \vdash_{\mathsf{tc}} \mathsf{ty}(E) \equiv U$, then $\Delta^\star \vdash_{\mathsf{tc}} \mathsf{ty}(E^\star) \equiv U$;*
3. *If $\Delta \vdash_{\mathsf{tc}} \mathsf{ch}(P)$, then $\Delta^\star \vdash_{\mathsf{tc}} \mathsf{ch}(P^\star)$.*

The B proof system of Fig. 1 is neutral with respect to variable annotation, so it is always possible to apply the same proof derivation to an annotated formula. The provability of well-typed formulæ is then preserved: $\Gamma \vdash_{\mathsf{B}} P$ if and only if $\Gamma^\star \vdash_{\mathsf{B}} P^\star$.

Finally, we take the universal closure of all free variables corresponding to given sets. To lighten the presentation in examples, we annotate only the first occurrence of a variable.

Example 2. Going back to the running example, we obtained the following environment Δ for the leave of the upper branch:

$$\mathsf{gi}(s), \mathsf{gi}(t), a \in \mathbb{P}(s \times t), b \in \mathbb{P}(s \times t), x \in a$$

It leads to the annotated environment $\Delta^\star$:

$$\mathsf{gi}(s^{\mathbb{P}(s)}), \mathsf{gi}(t^{\mathbb{P}(t)}), a^{\mathbb{P}(s\times t)} \in \mathbb{P}(s \times t), b^{\mathbb{P}(s\times t)} \in \mathbb{P}(s \times t), x^{s\times t} \in a$$

Finally, we obtain the annotated formula:

$$\begin{array}{l}\forall s^{\mathbb{P}(s)} \cdot (\forall t^{\mathbb{P}(t)} \cdot (\forall (a^{\mathbb{P}(s\times t)}, b^{\mathbb{P}(s\times t)}) \cdot \\ \quad (a, b \in \mathbb{P}(s \times t) \times \mathbb{P}(s \times t) \Rightarrow \{x^{s\times t} \mid x \in a \wedge x \in b\} \subseteq s \times t)))\end{array}$$

2.4 The Annotated Set Theory

Axioms SET5 and SET6 are introduced in the B Method set theory for theoretical reasons, like building natural numbers, and are never used in practice, in particular in proof obligations. So, we remove them from this work.

We now define the annotated version of the axioms presented in Sec. 2.1. In addition, we take the universal closure for all free variables.

$$\begin{array}{ll}\forall s^{\mathbb{P}(s)} \cdot (\forall t^{\mathbb{P}(t)} \cdot (\forall x^s \cdot (\forall y^t \cdot (x, y \in s \times t \Leftrightarrow (x \in s \wedge y \in t))))) & \textsf{SET1} \\ \forall s^{\mathbb{P}(s)} \cdot (\forall t^{\mathbb{P}(s)} \cdot (s \in \mathbb{P}(t) \Leftrightarrow \forall x^s \cdot (x \in s \Rightarrow x \in t))) & \textsf{SET2} \\ \forall s^{\mathbb{P}(s)} \cdot (\forall y^s \cdot (y \in \{x^s \mid x \in s \wedge P\} \Leftrightarrow (y \in s \wedge [x := y]P))) & \textsf{SET3} \\ \forall s^{\mathbb{P}(s)} \cdot (\forall t^{\mathbb{P}(s)} \cdot (\forall x^s \cdot (x \in s \Leftrightarrow x \in t) \Rightarrow s = t)) & \textsf{SET4}\end{array}$$

2.5 Skolemization of Comprehension Sets

We propose an elimination procedure of comprehension sets inside formulæ, based on the definition of new function symbols. The idea to skolemize comprehension sets is not new, see for instance [12]. In an expression, when meeting a set u of the shape: $u = \{x^T \mid P(x, s_1^{T_1}, \ldots, s_n^{T_n})\}$ we apply the following procedure:

1. Define a fresh function symbol $f^{\mathbb{P}(T)}$ of arity n and annotated by $\mathbb{P}(T)$;
2. Add to the B set theory, the axiom:

$$\forall s_1^{T_1} \cdot (\ldots \cdot (\forall s_n^{T_n} \cdot (\forall x^T \cdot (x \in f^{\mathbb{P}(T)}(s_1, \ldots, s_n) \Leftrightarrow P(x, s_1, \ldots, s_n)))))$$

3. Replace all the occurrences of u by $f^{\mathbb{P}(T)}(s_1, \ldots, s_n)$.

Remark 2. This skolemization procedure is sound (the new axiom is an instance of axiom SET3), but not complete (it is no more possible to define a set by comprehension during proof search).

Example 3. Applying skolemization to the running example leads to add the following axiom to the theory:

$$\forall a^{\mathbb{P}(s\times t)} \cdot (\forall b^{\mathbb{P}(s\times t)} \cdot (\forall x^{s\times t} \cdot (x \in f^{\mathbb{P}(s\times t)}(a,b) \Leftrightarrow x \in a \wedge x \in b)))$$

And we obtain the skolemized formula:

$$\forall s^{\mathbb{P}(s)} \cdot (\forall t^{\mathbb{P}(t)} \cdot \\ (\forall (a^{\mathbb{P}(s\times t)}, b^{\mathbb{P}(s\times t)}) \cdot (a, b \in \mathbb{P}(s\times t) \times \mathbb{P}(s\times t) \Rightarrow f^{\mathbb{P}(s\times t)}(a,b) \subseteq s \times t)))$$

2.6 Updated Syntax and Proof System

To conclude this section, we present the new version of the B syntax, with annotated variables, function symbols and without comprehension sets, choice function and BIG. In addition, we suppose that expressions are normalized in the sense that substitutions are reduced, as it is for proof obligations, so we remove substitutions from the syntax. We also merge the two categories for expressions and sets in a single category called E. Finally, we introduce $\bot := P \wedge \neg P$ and $\top := \neg\bot$, where P is a fixed formula.

$$\begin{array}{lcl} T & ::= & identifier \mid T_1 \times T_2 \mid \mathbb{P}(T) \\ P & ::= & \bot \mid \top \mid P_1 \wedge P_2 \mid P_1 \Rightarrow P_2 \mid \neg P \mid \forall x \cdot P \mid E_1 = E_2 \mid E_1 \in E_2 \\ E & ::= & x \mid E_1, E_2 \mid E_1 \times E_2 \mid \mathbb{P}(E) \mid f^{\mathbb{P}(T)}(E_1, \ldots, E_n) \\ x & ::= & identifier \mid x^T \mid x_1^{T_1}, x_2^{T_2} \end{array}$$

Finally, we enrich the B proof system of Fig. 1 with the two basic rules BR5 and BR6 dealing with $\bot$ and $\top$:

$$\frac{}{\Gamma, \bot \vdash_B Q}\ \text{BR5} \quad := \quad \dfrac{\dfrac{\dfrac{}{\Gamma, P\wedge\neg P, \neg Q \vdash_B P\wedge\neg P}\ \text{BR3}}{\Gamma, P\wedge\neg P, \neg Q \vdash_B P}\ \text{R2} \qquad \dfrac{\dfrac{}{\Gamma, P\wedge\neg P, \neg Q \vdash_B P\wedge\neg P}\ \text{BR3}}{\Gamma, P\wedge\neg P, \neg Q \vdash_B \neg P}\ \text{R2}'}{\Gamma, P\wedge\neg P \vdash_B Q}\ \text{R5}$$

$$\frac{}{\Gamma \vdash_B \top}\ \text{BR6} \quad := \quad \dfrac{\dfrac{}{\Gamma, \bot \vdash_B Q}\ \text{BR5} \qquad \dfrac{}{\Gamma, \bot \vdash_B \neg Q}\ \text{BR5}}{\Gamma \vdash_B \neg\bot}\ \text{R6}$$

3 LLproof: Typed Sequent Calculus of Zenon

3.1 Polymorphic First-Order Logic

We present in this section the polymorphic first-order logic, PFOL for short, used by the sequent calculus proof system LLproof. This presentation is highly inspired by [2].

A polymorphic signature is a triple $\Sigma = (\mathcal{K}, \mathcal{F}, \mathcal{P})$, where $\mathcal{K}$, $\mathcal{F}$ and $\mathcal{P}$ are countable sets of respectively type constructors k with their arity m, denoted $k :: m$, function symbols f and predicate symbols P with their type signature σ.

$$\sigma ::= f : \Pi\alpha_1 \ldots \alpha_m.\tau_1 \to \ldots \to \tau_n \to \tau \mid P : \Pi\alpha_1 \ldots \alpha_m.\tau_1 \to \ldots \to \tau_n \to o$$

where $\alpha_1 \ldots \alpha_m$ are the m first arguments of f or P and correspond to the type parameters; $\tau_1, \ldots, \tau_n$ are the following n arguments of f or P and correspond to the types of the term parameters; τ is the return type of f and o is the return pseudo-type of predicates P (but it is not a type of the language).

The syntax of PFOL is made of types, terms, formulæ and polymorphic formulæ. A type τ is either a type variable α or the application of a type constructor k. A term e is either a variable x or the application of a function symbol f to types and terms. A formula φ is inductively built from $\bot$, $\top$, conjunction, implication, negation, universal quantification over (term) variable, equality between terms and application of a predicate symbol. A polymorphic formula φ_α is a universal quantification over type variable. The typing rules of PFOL are standard and can be found in [2]. In the following, we may omit the m first type arguments for function and predicate symbols when they are clear from the context.

$$\begin{array}{rcl}
\tau & ::= & \alpha \mid k(\tau_1, \ldots, \tau_m) \\
e & ::= & x \mid f(\tau_1, \ldots, \tau_m; e_1, \ldots, e_n) \\
\varphi & ::= & \bot \mid \top \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \neg\varphi \mid \forall x : \tau.\varphi \mid e_1 =_\tau e_2 \\
& \mid & P(\tau_1, \ldots, \tau_m; e_1, \ldots, e_n) \\
\varphi_\alpha & ::= & \forall\alpha.\varphi_\alpha \mid \forall\alpha.\varphi
\end{array}$$

3.2 The Typed Sequent Calculus Proof System LLproof

In Fig. 3, we present the typed sequent calculus LLproof used by the automated theorem prover Zenon to output proofs. This sequent calculus is close to a tableau method proof system; we are looking for a contradiction, given the negation of the goal as an hypothesis. All formulæ are on the left hand side of the sequent, and the negation of the goal has to be unsatisfiable. In addition, the contraction rule is always applied, leading to a growing context Γ.

This presentation differs with the one in [8], which also introduces the proof system LLproof and its embedding into the proof-checker Dedukti. We remove the rules for equivalence and existential quantification, because these constructs are defined using other ones in the B Method (see Sec. 2.1). Moreover, we replace all rules from the category Special Rules by the new one Subst, since the Subst rule is easier to translate and can be used to define other Special rules [8].

The rules $\forall$ and $\neg\forall$ dealing with quantification over variables both get a side condition about the type of the chosen instance.

Rule $\forall_{\mathsf{type}}$ is applied to instantiate the type variables in axioms with the closed types coming from the translation of the proof obligation to be proved.

Closure and Quantifier-free Rules

$$\frac{}{\Gamma, \bot \vdash_{\mathsf{LL}} \bot}\ \bot \qquad \frac{}{\Gamma, \neg\top \vdash_{\mathsf{LL}} \bot}\ \neg\top \qquad \frac{}{\Gamma, t =_\tau u, u \neq_\tau t \vdash_{\mathsf{LL}} \bot}\ \text{Sym}$$

$$\frac{}{\Gamma, P, \neg P \vdash_{\mathsf{LL}} \bot}\ \text{Ax} \qquad \frac{}{\Gamma, t \neq_\tau t \vdash_{\mathsf{LL}} \bot}\ \neq \qquad \frac{\Gamma, P \vdash_{\mathsf{LL}} \bot \quad \Gamma, \neg P \vdash_{\mathsf{LL}} \bot}{\Gamma \vdash_{\mathsf{LL}} \bot}\ \text{Cut}$$

$$\frac{\Gamma, \neg\neg P, P \vdash_{\mathsf{LL}} \bot}{\Gamma, \neg\neg P \vdash_{\mathsf{LL}} \bot}\ \neg\neg \qquad \frac{\Gamma, P \wedge Q, P, Q \vdash_{\mathsf{LL}} \bot}{\Gamma, P \wedge Q \vdash_{\mathsf{LL}} \bot}\ \wedge \qquad \frac{\Gamma, \neg(P \Rightarrow Q), P, \neg Q \vdash_{\mathsf{LL}} \bot}{\Gamma, \neg(P \Rightarrow Q) \vdash_{\mathsf{LL}} \bot}\ \neg\Rightarrow$$

$$\frac{\Gamma, P \Rightarrow Q, \neg P \vdash_{\mathsf{LL}} \bot \quad \Gamma, P \Rightarrow Q, Q \vdash_{\mathsf{LL}} \bot}{\Gamma, P \Rightarrow Q \vdash_{\mathsf{LL}} \bot}\ \Rightarrow$$

$$\frac{\Gamma, \neg(P \wedge Q), \neg P \vdash_{\mathsf{LL}} \bot \quad \Gamma, \neg(P \wedge Q), \neg Q \vdash_{\mathsf{LL}} \bot}{\Gamma, \neg(P \wedge Q) \vdash_{\mathsf{LL}} \bot}\ \neg\wedge$$

Quantifier Rules Over Variables

$$\frac{\Gamma, \neg\forall x : \tau.\ P(x), \neg P(c) \vdash_{\mathsf{LL}} \bot}{\Gamma, \neg\forall x : \tau.\ P(x) \vdash_{\mathsf{LL}} \bot}\ \neg\forall$$

where $c : \tau$ is a fresh constant

$$\frac{\Gamma, \forall x : \tau.\ P(x), P(t) \vdash_{\mathsf{LL}} \bot}{\Gamma, \forall x : \tau.\ P(x) \vdash_{\mathsf{LL}} \bot}\ \forall$$

where $t : \tau$ is any closed term

Quantifier Rules Over Type Variables

$$\frac{\Gamma, \forall \alpha.\ P(\alpha), P(\tau) \vdash_{\mathsf{LL}} \bot}{\Gamma, \forall \alpha.\ P(\alpha) \vdash_{\mathsf{LL}} \bot}\ \forall_{\mathsf{type}}$$

where τ is any closed type

Special Rule

$$\frac{\Gamma, P(t), t \neq_\tau u \vdash_{\mathsf{LL}} \bot \quad \Gamma, P(t), P(u) \vdash_{\mathsf{LL}} \bot}{\Gamma, P(t) \vdash_{\mathsf{LL}} \bot}\ \text{Subst}$$

Fig. 3. The Typed Sequent Calculus LLproof

4 Translation of B Formulæ into PFOL

4.1 Type Signatures of Primitive Constructs

We start by defining a general skeleton for the type signatures of the B basic constructs. We introduce two type constructors Set and Pair corresponding respectively to the B type constructors $\mathbb{P}$ and $\times$. Then, we can define the function symbols (-,-) for ordered pair, $\mathbb{P}$(-) for powerset and - $\times$ - for product set. Finally, we define two predicate symbols for membership and equality. For easier reading, we use an infix notation with type arguments subscripted. For instance, - $\in_\alpha$ - corresponds to $\in (\alpha, -, -)$.

$$\mathcal{T}_{\mathsf{ske}} := \begin{cases} \mathsf{Set}(\text{-}) :: 1,\ \mathsf{Pair}(\text{-},\text{-}) :: 2 \\ (\text{-},\text{-})_{\alpha_1,\alpha_2} \quad : \Pi\alpha_1\alpha_2.\ \alpha_1 \to \alpha_2 \to \mathsf{Pair}(\alpha_1,\alpha_2) \\ \mathbb{P}_\alpha(\text{-}) \qquad : \Pi\alpha.\ \mathsf{Set}(\alpha) \to \mathsf{Set}(\mathsf{Set}(\alpha)) \\ \text{-}\ \times_{\alpha_1,\alpha_2}\ \text{-} : \Pi\alpha_1\alpha_2.\ \mathsf{Set}(\alpha_1) \to \mathsf{Set}(\alpha_2) \to \mathsf{Set}(\mathsf{Pair}(\alpha_1,\alpha_2)) \\ \text{-}\ \in_\alpha\ \text{-} \quad : \Pi\alpha.\ \alpha \to \mathsf{Set}(\alpha) \to o \\ \text{-}\ =_\alpha\ \text{-} \quad : \Pi\alpha.\ \alpha \to \alpha \to o \end{cases}$$

4.2 Translating Formulæ from B to PFOL

We present in Fig. 4 the translation function of B formulæ into PFOL formulæ. This translation, denoted $\langle P \rangle$ for some B formula P, is made of the three translations $\langle T \rangle_{\mathsf{t}}$ for types, $\langle P \rangle_{\mathsf{f}}$ for formulæ and $\langle E \rangle_{\mathsf{e}}$ for expressions, and a function $\theta(E)$ that returns the PFOL type of a B expression E.

One important point in this embedding is the interpretation given to B type identifiers coming from the type annotation procedure (see Sec. 2.3). We interpret B type identifiers coming from axioms and hypotheses as type variables (and take the universal closure with respect to them), and B type identifiers of the formula to prove (also called goal) as new constants, *i.e.* nullary type constructors. This allows us to get polymorphic axioms in PFOL and a monomorphic/many-sorted goal. To achieve this, we add to all B formulæ to translate a flag ax for axioms and hypotheses and gl for the goal.

Before presenting the three translation functions, we have to define a function called $Sig(f(\ldots))$, where f is a B function symbol coming from the skolemization of comprehension sets (see Sec. 2.5), that returns the type signature of f. Let $FV(e)$ be the set of free variables of an expression e.

$$Sig(f^{\mathbb{P}(T)}(E_1,\ldots,E_n)) = \underset{\alpha \in FV_1^n(\theta(E_i))}{\Pi} \boldsymbol{\alpha}.\ \theta(E_1) \to \ldots \to \theta(E_n) \to \theta(\mathbb{P}(T))$$

During the translation procedure, we carry a target PFOL theory $\mathcal{T}$ composed by the skeleton $\mathcal{T}_{\mathsf{ske}}$ defined in Sec.4.1, previously translated formulæ, new type constructors and new type signatures. Also, for each formula to be translated, we carry a PFOL local context Δ of bound variables and their type, and a set Ω of pairs of B type identifiers and their corresponding PFOL types, *i.e.* type variables for axioms and type constants for goals.

Example 4. Continuing with the running example, we first translate axioms SET1, SET2 and SET4, then the axiom coming from the skolemization, and finally the goal. To lighten the presentation, we omit the subscripted type arguments of function and predicate symbols of $\mathcal{T}_{\mathsf{ske}}$ and we factorize the symbol $\forall$. The three set theory axioms become:

$\forall\alpha_1,\alpha_2.\ \forall s : \mathsf{Set}(\alpha_1), t : \mathsf{Set}(\alpha_2), x : \alpha_1, y : \alpha_2.\ (x,y) \in s \times t \Leftrightarrow (x \in s \wedge y \in t)$
$\forall\alpha.\ \forall s : \mathsf{Set}(\alpha), t : \mathsf{Set}(\alpha).\ s \in \mathbb{P}(t) \Leftrightarrow \forall x : \alpha.\ x \in s \Rightarrow x \in t$
$\forall\alpha.\ \forall s : \mathsf{Set}(\alpha), t : \mathsf{Set}(\alpha).\ (\forall x : \alpha.\ x \in s \Leftrightarrow x \in t) \Rightarrow s = t$

$$
\begin{array}{l}
\theta(E) = \mathsf{match}\ E\ \mathsf{with} \\
\quad \mid x^T \qquad\qquad\qquad \to \Delta(x) \\
\quad \mid E_1, E_2 \qquad\qquad \to \mathsf{Pair}(\theta(E_1), \theta(E_2)) \\
\quad \mid E_1 \times E_2 \qquad\quad \to \mathsf{Set}(\mathsf{Pair}(\theta(E_1), \theta(E_2))) \\
\quad \mid \mathbb{P}(E) \qquad\qquad \to \mathsf{Set}(\theta(E)) \\
\quad \mid f^{\mathbb{P}(T)}(\ldots) \qquad \to \mathsf{Set}(\langle T \rangle_{\mathsf{t}}) \\
\\
\langle T \rangle_{\mathsf{t}} = \mathsf{match}\ T\ \mathsf{with} \\
\quad \mid id\ \mathsf{when}\ flag = ax \to \begin{cases} \mathsf{if}\ id \in \Omega\ \mathsf{then\ return}\ \Omega(id) \\ \mathsf{else}\ \Omega := \Omega, (id, \alpha_{id})\ \mathsf{return}\ \alpha_{id} \end{cases} \\
\quad \mid id\ \mathsf{when}\ flag = gl \to \begin{cases} \mathsf{if}\ id \in \Omega\ \mathsf{then\ return}\ \Omega(id) \\ \mathsf{else}\ \mathcal{T} := \mathcal{T}, k_{id} :: 0\ ;\ \Omega := \Omega, (id, k_{id})\ \mathsf{return}\ k_{id} \end{cases} \\
\quad \mid T_1 \times T_2 \qquad\quad \to \mathsf{Pair}(\langle T_1 \rangle_{\mathsf{t}}, \langle T_2 \rangle_{\mathsf{t}}) \\
\quad \mid \mathbb{P}(T) \qquad\qquad \to \mathsf{Set}(\langle T \rangle_{\mathsf{t}}) \\
\\
\langle P \rangle_{\mathsf{f}} = \mathsf{match}\ P\ \mathsf{with} \\
\quad \mid \bot \mid \top \qquad\qquad \to \bot \mid \top \\
\quad \mid P_1 \wedge P_2 \qquad\quad \to \langle P_1 \rangle_{\mathsf{f}} \wedge \langle P_2 \rangle_{\mathsf{f}} \\
\quad \mid P_1 \Rightarrow P_2 \qquad\quad \to \langle P_1 \rangle_{\mathsf{f}} \Rightarrow \langle P_2 \rangle_{\mathsf{f}} \\
\quad \mid \neg P \qquad\qquad\quad \to \neg \langle P \rangle_{\mathsf{f}} \\
\quad \mid \forall x^T \cdot P \qquad\quad \to \forall x : \langle T \rangle_{\mathsf{t}} \,.\, \langle P \rangle_{\mathsf{f}}\ \mathsf{and}\ \Delta := \Delta, x : \langle T \rangle_{\mathsf{t}} \\
\quad \mid \forall (x_1^{T_1}, x_2^{T_2}) \cdot P \to \begin{cases} \forall x_1 : \langle T_1 \rangle_{\mathsf{t}} \,.\, \forall x_2 : \langle T_2 \rangle_{\mathsf{t}} \,.\, \langle P \rangle_{\mathsf{f}} \\ \mathsf{and}\ \Delta := \Delta, x_1 : \langle T_1 \rangle_{\mathsf{t}}, x_2 : \langle T_2 \rangle_{\mathsf{t}} \end{cases} \\
\quad \mid E_1 = E_2 \qquad\quad \to \langle E_1 \rangle_{\mathsf{e}} =_{\theta(E_1)} \langle E_2 \rangle_{\mathsf{e}} \\
\quad \mid E_1 \in E_2 \qquad\quad \to \langle E_1 \rangle_{\mathsf{e}} \in_{\theta(E_1)} \langle E_2 \rangle_{\mathsf{e}} \\
\\
\langle E \rangle_{\mathsf{e}} = \mathsf{match}\ E\ \mathsf{with} \\
\quad \mid x^T \qquad\qquad\qquad \to x \\
\quad \mid E_1, E_2 \qquad\qquad \to (\langle E_1 \rangle_{\mathsf{e}}, \langle E_2 \rangle_{\mathsf{e}})_{\theta(E_1), \theta(E_2)} \\
\quad \mid E_1 \times E_2 \qquad\quad \to \langle E_1 \rangle_{\mathsf{e}} \times_{\tau_1, \tau_2} \langle E_2 \rangle_{\mathsf{e}}\ \mathsf{where} \begin{cases} \theta(E_1) = \mathsf{Set}(\tau_1) \\ \theta(E_2) = \mathsf{Set}(\tau_2) \end{cases} \\
\quad \mid \mathbb{P}(E) \qquad\qquad \to \mathbb{P}_\tau(\langle E \rangle_{\mathsf{e}})\ \mathsf{where}\ \theta(E) = \mathsf{Set}(\tau) \\
\quad \mid f^{\mathbb{P}(T)}(E_1, \ldots, E_n) \ \to \\
\qquad \mathsf{if}\ f : \Pi \alpha_1 \ldots \alpha_m \,.\, \tau_1 \to \ldots \to \tau_n \to \tau \notin \mathcal{T} \\
\qquad \mathsf{then}\ \mathcal{T} := \mathcal{T}, f : Sig(f^{\mathbb{P}(T)}(E_1, \ldots, E_n)) \\
\qquad \mathsf{return}\ f(\tau_1', \ldots, \tau_m'; \langle E_1 \rangle_{\mathsf{e}}, \ldots, \langle E_n \rangle_{\mathsf{e}})\ \mathsf{where} \begin{cases} \theta(E_1) = \tau_1(\tau_1', \ldots, \tau_m') \\ \cdots \\ \theta(E_n) = \tau_n(\tau_1', \ldots, \tau_m') \end{cases}
\end{array}
$$

Fig. 4. Translation from B to PFOL

The remainder of the theory, *i.e.* the signature of f, the axiom defining f and the declaration of the two type constants coming from the translation of the goal, is:

$$\begin{cases} k_1 :: 0,\ k_2 :: 0 \\ f : \Pi\alpha_1\alpha_2.\ \mathsf{Set}(\mathsf{Pair}(\alpha_1,\alpha_2)) \to \mathsf{Set}(\mathsf{Pair}(\alpha_1,\alpha_2)) \to \mathsf{Set}(\mathsf{Pair}(\alpha_1,\alpha_2)) \\ \forall\alpha_1,\alpha_2.\ \forall a : \mathsf{Set}(\mathsf{Pair}(\alpha_1,\alpha_2)), b : \mathsf{Set}(\mathsf{Pair}(\alpha_1,\alpha_2)), x : \mathsf{Pair}(\alpha_1,\alpha_2). \\ \qquad x \in f(a,b) \Leftrightarrow (x \in a \wedge x \in b) \end{cases}$$

Finally, the translation of the goal (we unfold the $\subseteq$ definition, see Sec. 2.1) is:

$$\begin{array}{l} \forall s : \mathsf{Set}(k_1), t : \mathsf{Set}(k_2), a : \mathsf{Set}(\mathsf{Pair}(k_1,k_2)), b : \mathsf{Set}(\mathsf{Pair}(k_1,k_2)). \\ \qquad (a,b) \in \mathbb{P}(s \times t) \times \mathbb{P}(s \times t) \Rightarrow f(a,b) \in \mathbb{P}(s \times t) \end{array}$$

5 Translating LLproof Proofs into B Proofs

In Fig. 5, we present the reverse translation, denoted $\langle\varphi\rangle^{-1}$, to translate monomorphic PFOL formulæ into B formulæ. This reverse translation is simpler than the one presented in Sec. 4.2 because we do not need to translate types, annotations for bound variables and function symbols not being necessary anymore.

$$\begin{array}{lll} \langle\varphi\rangle_{\mathsf{f}}^{-1} = \mathsf{match}\ \varphi\ \mathsf{with} & & \\ \quad \mid \bot \mid \top & \to \bot \mid \top & \\ \quad \mid \varphi_1 \wedge \varphi_2 & \to \langle\varphi_1\rangle_{\mathsf{f}}^{-1} \wedge \langle\varphi_2\rangle_{\mathsf{f}}^{-1} & \\ \quad \mid \varphi_1 \Rightarrow \varphi_2 & \to \langle\varphi_1\rangle_{\mathsf{f}}^{-1} \Rightarrow \langle\varphi_2\rangle_{\mathsf{f}}^{-1} & \\ \quad \mid \neg\varphi & \to \neg\,\langle\varphi\rangle_{\mathsf{f}}^{-1} & \\ \quad \mid \forall x : \tau.\ \varphi & \to \forall x \cdot \langle\varphi\rangle_{\mathsf{f}}^{-1} & \\ \quad \mid e_1 =_\tau e_2 & \to \langle e_1\rangle_{\mathsf{e}}^{-1} = \langle e_2\rangle_{\mathsf{e}}^{-1} & \\ \quad \mid e_1 \in_\tau e_2 & \to \langle e_1\rangle_{\mathsf{e}}^{-1} \in \langle e_2\rangle_{\mathsf{e}}^{-1} & \\ & & \\ \langle e\rangle_{\mathsf{e}}^{-1} = \mathsf{match}\ E\ \mathsf{with} & & \\ \quad \mid x & \to x & \\ \quad \mid (e_1,e_2)_{\tau_1,\tau_2} & \to \langle e_1\rangle_{\mathsf{e}}^{-1}, \langle e_2\rangle_{\mathsf{e}}^{-1} & \\ \quad \mid e_1 \times_{\tau_1,\tau_2} e_2 & \to \langle e_1\rangle_{\mathsf{e}}^{-1} \times \langle e_2\rangle_{\mathsf{e}}^{-1} & \\ \quad \mid \mathbb{P}_\tau(e) & \to \mathbb{P}(\langle e\rangle_{\mathsf{e}}^{-1}) & \\ \quad \mid f(\tau'_1,\ldots,\tau'_m;e_1,\ldots,e_n) & \to f(\langle e_1\rangle_{\mathsf{e}}^{-1},\ldots,\langle e_n\rangle_{\mathsf{e}}^{-1}) & \end{array}$$

Fig. 5. Translation from PFOL to B

Theorem 1. *For a set of B formulæ Γ and a B goal P, if there exists a LLproof proof of the sequent $\langle\Gamma\rangle, \langle\neg P\rangle \vdash_{\mathsf{LL}} \bot$, then there exists a set Γ' of monomorphic instances of $\langle\Gamma\rangle$, and a B proof of the sequent $\langle\Gamma'\rangle^{-1}, \neg P \vdash_{\mathsf{B}} \bot$.*

Proof. We present a sketch of the proof.

1. We show that if P is a B goal, then we have $\langle\langle P\rangle\rangle^{-1} \Leftrightarrow P$.

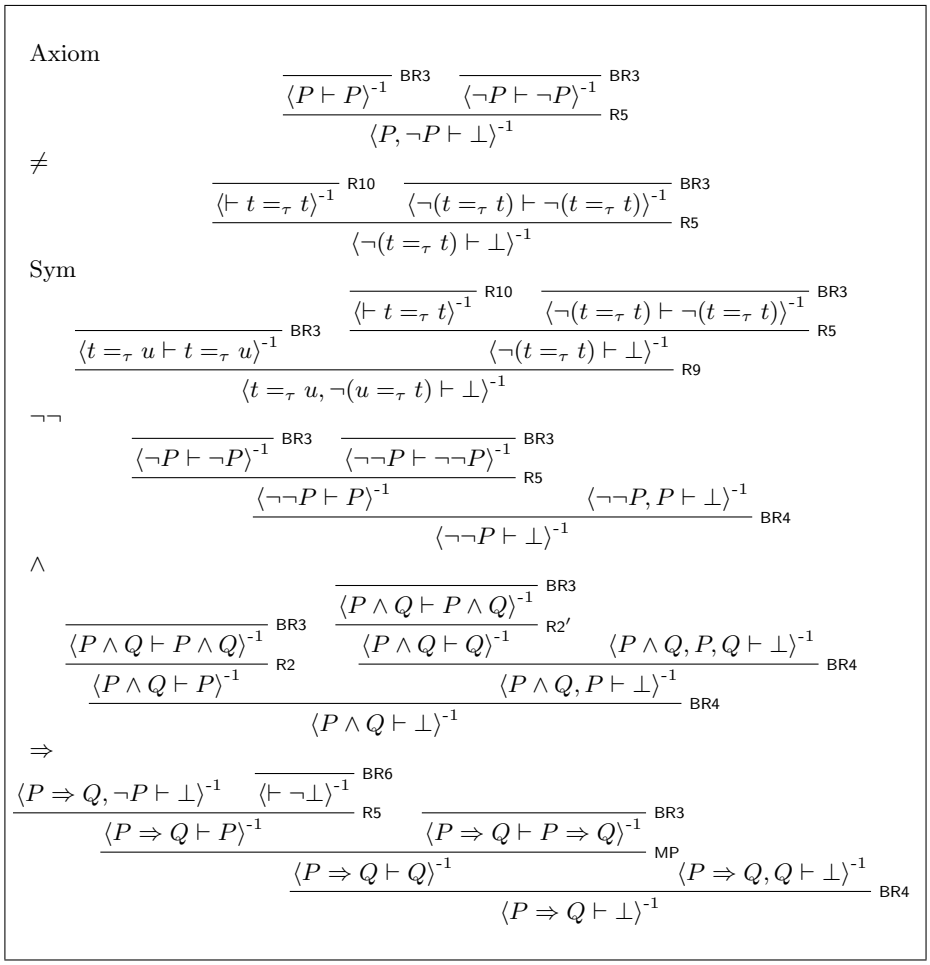

Fig. 6. Translations of LLproof Rules into B Proof System (part 1)

2. Given a proof Π of the sequent $\langle \Gamma \rangle, \langle \neg P \rangle \vdash_{\mathsf{LL}} \bot$, there exists a proof Π_{Kleene} of the sequent, starting with all applications of $\forall_{\mathsf{type}}$ rules on polymorphic formulæ, thanks to the permutation of inference rules in sequent calculus [15].
3. We take the subproof Π_{mono} of Π_{Kleene}, where we removed all the $\forall_{\mathsf{type}}$ nodes and the remaining polymorphic formulæ.
4. The set Γ' of monomorphic instances of $\langle \Gamma \rangle$ is made of the root node formulæ of Π_{mono}, except $\langle \neg P \rangle$.
5. We extend the reverse translation to LLproof sequents, $\langle P_1, \ldots, P_n \vdash_{\mathsf{LL}} Q \rangle^{-1} \rightarrow \langle P_1 \rangle^{-1}, \ldots, \langle P_n \rangle^{-1} \vdash_{\mathsf{B}} \langle Q \rangle^{-1}$, and to LLproof proof nodes in Fig. 6 and Fig. 7.
6. $\langle \Pi_{mono} \rangle^{-1}$ is a B proof of the sequent $\langle \Gamma' \rangle^{-1}, \neg P \vdash_{\mathsf{B}} \bot$.

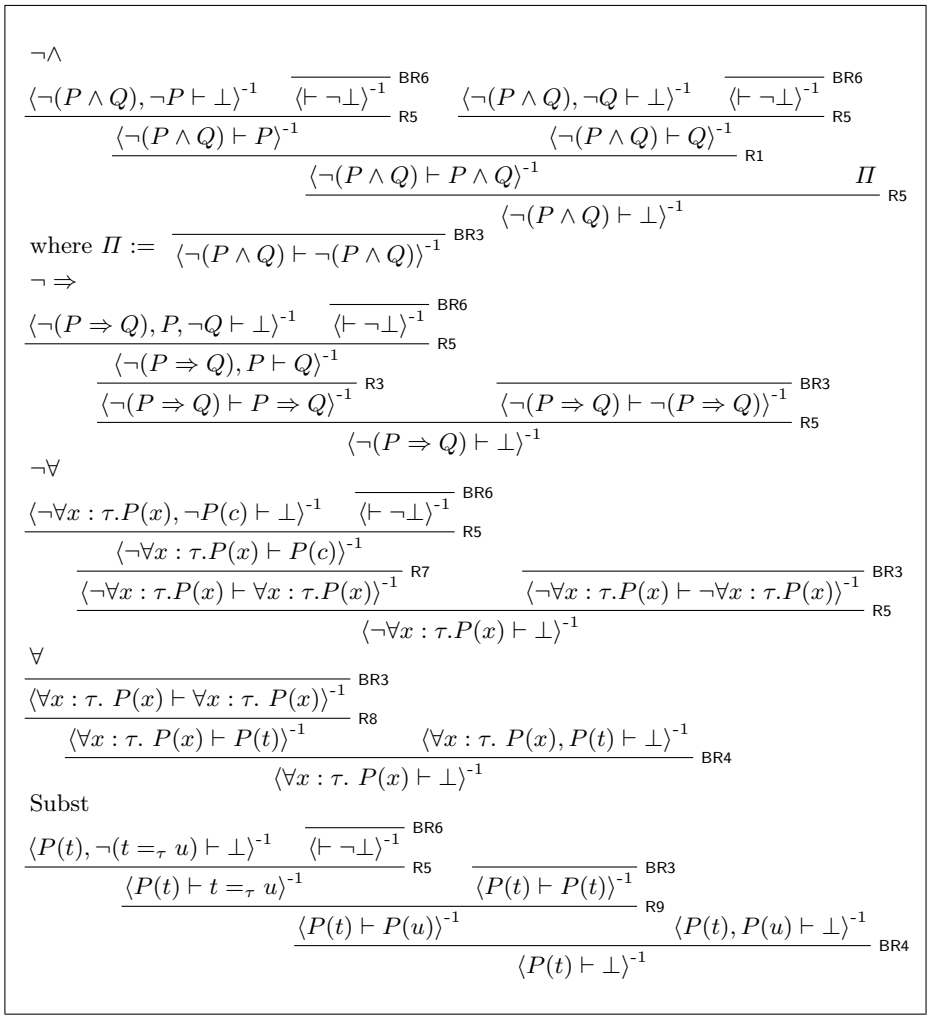

Fig. 7. Translation of LLproof Rules into B Proof System (part 2)

We give in Fig. 6 and Fig. 7 the translations for each LLproof proof node. Each node can be translated to a B derivation where all PFOL sequents are translated into B sequents, leading to a B proof tree. To lighten the presentation, we omit to indicate the context Γ and some useless formulæ (removable by applying BR2) on the left-hand side of sequents, and we use $\vdash$ for $\vdash_{\mathsf{LL}}$. For instance, the translation of the LLproof Axiom rule should be:

$$\dfrac{\dfrac{}{\langle \Gamma, P, \neg P, \neg\bot \vdash_{\mathsf{LL}} P\rangle^{-1}}\,\mathsf{BR3} \qquad \dfrac{}{\langle \Gamma, P, \neg P, \neg\bot \vdash_{\mathsf{LL}} \neg P\rangle^{-1}}\,\mathsf{BR3}}{\langle \Gamma, P, \neg P \vdash_{\mathsf{LL}} \bot\rangle^{-1}}\,\mathsf{R5}$$

Example 5. The proof of the running example is too big to be presented here. Instead, we present the proof translation for the following B formula, given s:

$$\forall x \cdot (x \in s \Rightarrow x \in s)$$

The latter leads to the PFOL formula, where k is a constant:

$$\forall s : \mathsf{Set}(k).\ \forall x : k.\ x \in s \Rightarrow x \in s$$

The LLproof proof is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{c_x \in_k c_s,\ c_x \notin_k c_s \vdash_{\mathsf{LL}} \bot}\,\mathrm{Ax}}{\neg(c_x \in_k c_s \Rightarrow c_x \in_k c_s) \vdash_{\mathsf{LL}} \bot}\,\neg\Rightarrow}{\neg\forall x : k.\ x \in_k c_s \Rightarrow x \in_k c_s \vdash_{\mathsf{LL}} \bot}\,\neg\forall}{\neg\forall s : \mathsf{Set}(k).\ \forall x : k.\ x \in_k s \Rightarrow x \in_k s \vdash_{\mathsf{LL}} \bot}\,\neg\forall$$

We obtain the B proof (we removed the universal quantification over the given set s, the first R5 node in the translation of $\neg\forall$, some useless formulæ on the left-hand side of sequents and used $\vdash$ for $\vdash_{\mathsf{B}}$, c for c_x and s for c_s):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{c \in s \vdash c \in s}\,\mathrm{BR3} \quad \dfrac{}{c \notin s \vdash c \notin s}\,\mathrm{BR3}}{c \in s, c \notin s \vdash \bot}\,\mathrm{R5} \quad \dfrac{}{\vdash \neg\bot}\,\mathrm{BR6}}{c \in s \vdash c \in s}\,\mathrm{R5}}{\vdash c \in s \Rightarrow c \in s}\,\mathrm{R3} \quad \dfrac{}{\neg(c \in s \Rightarrow c \in s) \vdash \neg(c \in s \Rightarrow c \in s)}\,\mathrm{BR3}}{\neg(c \in s \Rightarrow c \in s) \vdash \bot}\,\mathrm{R5} \quad \dfrac{}{\vdash \neg\bot}\,\mathrm{BR6}}{\vdash c \in s \Rightarrow c \in s}\,\mathrm{R5}}{\vdash \forall x \cdot (x \in s \Rightarrow x \in s)}\,\mathrm{R7}$$

6 Conclusion

Automated theorem provers are in general made of thousands lines of code, using elaborate decision procedures and specific heuristics. The confidence in such tools may therefore be questioned. The correctness of Zenon proofs is already guaranteed by the checking of proof certificates by an external proof checker. But to prove B proof obligations, Zenon relies on two external tools, bpo2why and Why3, to translate proof obligations into its input format, which raises the question whether the proof found still corresponds to a proof of the original statement.

In this paper, we have formalized a different and direct translation from the B Method to a polymorphic first-order logic. The main purpose of this work is not to replace bpo2why, but to validate the use of Zenon to prove B proof obligations. One of the most challenging part of this translation deals with the encoding of the B notion of types. Our solution to make the axioms polymorphic allows us to benefit from the flexibility of polymorphism. Furthermore, we showed that this translation is sound and gave a procedure to translate Zenon proofs in the B proof system.

As future work, we want to prove the soundness and completeness of the deduction modulo theory [11] extension of the proof system LLproof with regard to those of LLproof, in particular in the case of the B Method.

References

1. Abrial, J.R.: The B-Book, Assigning Programs to Meanings. Cambridge University Press (1996)
2. Blanchette, J.C., Böhme, S., Popescu, A., Smallbone, N.: Encoding Monomorphic and Polymorphic Types. In: Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2013. Springer (2013)
3. Bobot, F., Filliâtre, J.C., Marché, C., Paskevich, A.: Why3: Shepherd Your Herd of Provers. In: International Workshop on Intermediate Verification Languages (Boogie) (2011)
4. Bodeveix, J.P., Filali, M.: Type Synthesis in B and the Translation of B to PVS. In: Formal Specification and Development in Z and B, ZB. Springer (2002)
5. Boespflug, M., Carbonneaux, Q., Hermant, O.: The $\lambda\Pi$-Calculus Modulo as a Universal Proof Language. In: Proof Exchange for Theorem Proving (PxTP) (2012)
6. Bonichon, R., Delahaye, D., Doligez, D.: Zenon: An Extensible Automated Theorem Prover Producing Checkable Proofs. In: Logic for Programming Artificial Intelligence and Reasoning (LPAR). LNCS/LNAI, vol. 4790. Springer (2007)
7. Bury, G., Delahaye, D., Doligez, D., Halmagrand, P., Hermant, O.: Automated Deduction in the B Set Theory using Typed Proof Search and Deduction Modulo. In: LPAR 20 : 20th International Conference on Logic for Programming, Artificial Intelligence and Reasoning. Suva, Fiji (2015)
8. Cauderlier, R., Halmagrand, P.: Checking Zenon Modulo Proofs in Dedukti. In: Fourth Workshop on Proof eXchange for Theorem Proving (PxTP). Berlin, Germany (2015)
9. Delahaye, D., Doligez, D., Gilbert, F., Halmagrand, P., Hermant, O.: Zenon Modulo: When Achilles Outruns the Tortoise using Deduction Modulo. In: Logic for Programming Artificial Intelligence and Reasoning (LPAR). LNCS/ARCoSS, vol. 8312. Springer (2013)
10. Delahaye, D., Dubois, C., Marché, C., Mentré, D.: The BWare Project: Building a Proof Platform for the Automated Verification of B Proof Obligations. In: Abstract State Machines, Alloy, B, VDM, and Z (ABZ). LNCS, Springer (2014)
11. Dowek, G., Hardin, T., Kirchner, C.: Theorem Proving Modulo. Journal of Automated Reasoning (JAR) 31 (2003)
12. Dowek, G., Miquel, A.: Cut elimination for zermelo set theory. Archive for Mathematical Logic. Springer. Submitted (2007)
13. Jacquel, M., Berkani, K., Delahaye, D., Dubois, C.: Verifying B Proof Rules using Deep Embedding and Automated Theorem Proving. Software Engineering and Formal Methods 7041, 253–268 (2011)
14. Jaeger, E., Dubois, C.: Why Would You Trust B? In: Berlin, S. (ed.) Logic for Programming, Artificial Intelligence, and Reasoning. Lecture Notes in Computer Science, Yerevan, Armenia (2007)
15. Kleene, S.C.: Permutability Of Inferences In Gentzens Calculi LK And LJ. In: Bulletin Of The American Mathematical Society. vol. 57, pp. 485–485. Amer Mathematical Soc 201 Charles St, Providence, RI (1951)
16. Mentré, D., Marché, C., Filliâtre, J.C., Asuka, M.: Discharging Proof Obligations from Atelier B using Multiple Automated Provers. In: Abstract State Machines, Alloy, B, VDM, and Z (ABZ). LNCS, vol. 7316. Springer (2012)
17. Schmalz, M.: Formalizing the logic of event-B. Ph.D. thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20516, 2012 (2012)
18. ClearSy: Atelier B 4.1 (2013), `http://www.atelierb.eu/`