An Automation-Friendly Set Theory for the B Method

Guillaume Bury[1], Simon Cruanes[2], David Delahaye[3](✉), and Pierre-Louis Euvrard[3]

[1] LSV, ENS Paris-Saclay, Inria, Cachan, France
Guillaume.Bury@inria.fr

[2] Aesthetic Integration, Austin, TX, USA
simon@aestheticintegration.com

[3] LIRMM, Université de Montpellier, CNRS, Montpellier, France
{David.Delahaye,Pierre-Louis.Euvrard}@lirmm.fr

Abstract. We propose an automation-friendly set theory for the B method. This theory is expressed using first order logic extended to polymorphic types and rewriting. Rewriting is introduced along the lines of deduction modulo theory, where axioms are turned into rewrite rules over both propositions and terms. We also provide experimental results of several tools able to deal with polymorphism and rewriting over a benchmark of problems in pure set theory (i.e. without arithmetic).

Keywords: B method · Set theory · Automated deduction
Polymorphic types · Rewriting

1 Introduction

In this paper, we present the set theory of the B method [1] using polymorphic types and rewriting. Expressed this way, this theory has the benefit of being quite automatable for several reasons. In particular, the use of polymorphism allows us to make the theory more synthetic by removing some typing predicates, which therefore improves proof search. As for rewriting, it is introduced along the lines of deduction modulo theory [5], where axioms are turned into rewrite rules over both propositions and terms. Deduction modulo theory has proved to be also very useful to improve proof search when integrated to usual automated proof techniques. In this paper, we also aim to advertise that more and more automated tools are able to deal with polymorphic types and rewriting, and we provide some experimental results involving the latest versions of these tools.

M. Butler et al. (Eds.): ABZ 2018, LNCS 10817, pp. 409–414, 2018.
https://doi.org/10.1007/978-3-319-91271-4_32

Axioms of Set Theory

$$(x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} s \times_{\alpha_1,\alpha_2} t \longrightarrow x \in_{\alpha_1} s \wedge y \in_{\alpha_2} t$$
$$s \in_{\mathsf{set}(\alpha)} \mathbb{P}_\alpha(t) \longrightarrow \forall x : \alpha. x \in_\alpha s \Rightarrow x \in_\alpha t$$
$$s =_{\mathsf{set}(\alpha)} t \longrightarrow \forall x : \alpha. x \in_\alpha s \Leftrightarrow x \in_\alpha t$$

Set Inclusion

$$s \subseteq_\alpha t \longrightarrow s \in_{\mathsf{set}(\alpha)} \mathbb{P}_\alpha(t) \qquad s \subset_\alpha t \longrightarrow s \subseteq_\alpha t \wedge s \neq_{\mathsf{set}(\alpha)} t$$

Derived Constructs

$$x \in_\alpha s \cup_\alpha t \longrightarrow x \in_\alpha s \vee x \in_\alpha t \qquad x \in_\alpha s \cap_\alpha t \longrightarrow x \in_\alpha s \wedge x \in_\alpha t$$
$$x \in s -_\alpha t \longrightarrow x \in_\alpha s \wedge x \notin_\alpha t \qquad x \in_\alpha \emptyset_\alpha \longrightarrow \bot$$
$$x \in_\alpha \{a\}_\alpha \longrightarrow x =_\alpha a \qquad \mathbb{P}_{1\ \alpha}(s) \longrightarrow \mathbb{P}_\alpha(s) -_\alpha \{\emptyset_\alpha\}_{\mathsf{set}(\alpha)}$$

Binary Relation Constructs: First Series

$$p \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} u \leftrightarrow_{\alpha_1,\alpha_2} v \longrightarrow$$
$$\qquad \forall x : \alpha_1. \forall y : \alpha_2. (x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \Rightarrow x \in_{\alpha_1} u \wedge y \in_{\alpha_2} v$$
$$(y,x) \in_{\mathsf{tup}(\alpha_2,\alpha_1)} p^{-1}_{\alpha_1,\alpha_2} \longrightarrow (x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p$$
$$x \in_{\alpha_1} \mathsf{dom}_{\alpha_1,\alpha_2}(p) \longrightarrow \exists b : \alpha_2. (x,b) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p$$
$$x \in_{\alpha_2} \mathsf{ran}_{\alpha_1,\alpha_2}(p) \longrightarrow \exists a : \alpha_1. (a,x) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p$$
$$(x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_3)} p ;_{\alpha_1,\alpha_2,\alpha_3} q \longrightarrow \exists b : \alpha_2. (x,b) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \wedge (b,y) \in_{\mathsf{tup}(\alpha_2,\alpha_3)} q$$
$$q \circ_{\alpha_1,\alpha_2,\alpha_3} p \longrightarrow p ;_{\alpha_1,\alpha_2,\alpha_3} q$$
$$(x,y) \in_{\mathsf{tup}(\alpha,\alpha)} \mathsf{id}_\alpha(u) \longrightarrow x \in_\alpha u \wedge x =_\alpha y$$
$$(x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} s \lhd_{\alpha_1,\alpha_2} p \longrightarrow (x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \wedge x \in_{\alpha_1} s$$
$$(x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \rhd_{\alpha_1,\alpha_2} t \longrightarrow (x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \wedge y \in_{\alpha_2} t$$
$$(x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} s \mathrel{\triangleleft\!\!\!-}_{\alpha_1,\alpha_2} p \longrightarrow (x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \wedge x \notin_{\alpha_1} s$$
$$(x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \mathrel{\triangleright\!\!\!-}_{\alpha_1,\alpha_2} t \longrightarrow (x,y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p \wedge y \notin_{\alpha_2} t$$

Fig. 1. Rewriting rules of the B set theory (part 1)

2 A Set Theory with Polymorphism and Rewriting for B

In the following, we consider the pure set theory part of the B method, i.e. the material introduced in Chap. 2 of the B-Book [1]. This part of the B theory is suitable as it can be easily turned into a theory that is compatible with deduction modulo theory, i.e. where a large part of axioms can be turned into rewrite rules. We therefore transform whenever possible the axioms and definitions into rewrite rules. The resulting theory is summarized in Figs. 1 and 2, where we omit the set BIG and the sets defined in extension.

As can be seen, the proposed theory is typed, using first order logic extended to polymorphic types à la ML, through a type system in the spirit of [2]. This extension to polymorphic types offers more flexibility, and allows us to deal with

Binary Relation Constructs: Second Series

$$x \in_{\alpha_2} p[w]_{\alpha_1,\alpha_2} \longrightarrow \exists a : \alpha_1 . a \in_{\alpha_1} w \wedge (a, x) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p$$
$$(x, y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} q \mathbin{\lhd\!\!+}_{\alpha_1,\alpha_2} p \longrightarrow$$
$$\quad ((x, y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} q \wedge x \notin_{\alpha_1} \mathsf{dom}_{\alpha_1,\alpha_2}(p)) \vee (x, y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} p$$
$$(x, (y, z)) \in_{\mathsf{tup}(\alpha_1,\mathsf{tup}(\alpha_2,\alpha_3))} f \otimes_{\alpha_1,\alpha_2,\alpha_3} g \longrightarrow (x, y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} f \wedge (x, z) \in_{\mathsf{tup}(\alpha_1,\alpha_3)} g$$
$$((x, y), z) \in_{\mathsf{tup}(\mathsf{tup}(\alpha_1,\alpha_2),\alpha_1)} \mathsf{prj_1}_{\ \alpha_1,\alpha_2}(s, t) \longrightarrow$$
$$\quad ((x, y), z) \in_{\mathsf{tup}(\mathsf{tup}(\alpha_1,\alpha_2),\alpha_1)} (s \times_{\alpha_1,\alpha_2} t) \times_{\mathsf{tup}(\alpha_1,\alpha_2),\alpha_1} s \wedge x =_{\alpha_1} z$$
$$((x, y), z) \in_{\mathsf{tup}(\mathsf{tup}(\alpha_1,\alpha_2),\alpha_2)} \mathsf{prj_2}_{\ \alpha_1,\alpha_2}(s, t) \longrightarrow$$
$$\quad ((x, y), z) \in_{\mathsf{tup}(\mathsf{tup}(\alpha_1,\alpha_2),\alpha_1)} (s \times_{\alpha_1,\alpha_2} t) \times_{\mathsf{tup}(\alpha_1,\alpha_2),\alpha_1} t \wedge y =_{\alpha_1} z$$
$$((x, y), (z, w)) \in_{\mathsf{tup}(\mathsf{tup}(\alpha_1,\alpha_3),\mathsf{tup}(\alpha_2,\alpha_4))} h||_{\alpha_1,\alpha_2,\alpha_3,\alpha_4} k \longrightarrow$$
$$\quad (x, z) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} h \wedge (y, w) \in_{\mathsf{tup}(\alpha_3,\alpha_4)} k$$

Function Constructs: First Series

$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\rightarrow}_{\alpha_1,\alpha_2} t \longrightarrow f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \leftrightarrow_{\alpha_1,\alpha_2} t \wedge$$
$$\quad \forall x : \alpha_1 . \forall y, z : \alpha_2 . (x, y) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} f \wedge (x, z) \in_{\mathsf{tup}(\alpha_1,\alpha_2)} f \Rightarrow y =_{\alpha_2} z$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \rightarrow_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\rightarrow}_{\alpha_1,\alpha_2} t \wedge \mathsf{dom}_{\alpha_1,\alpha_2}(f) =_{\mathsf{set}(\alpha_1)} s$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{\rightarrowtail\!\!\!\!\!+}_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\rightarrow}_{\alpha_1,\alpha_2} t \wedge f^{-1}_{\alpha_1,\alpha_2} \in_{\mathsf{set}(\mathsf{tup}(\alpha_2,\alpha_1))} t \mathrel{+\!\!\!\rightarrow}_{\alpha_2,\alpha_1} s$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \rightarrowtail_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{\rightarrowtail\!\!\!\!\!+}_{\alpha_1,\alpha_2} t \wedge f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \rightarrow_{\alpha_1,\alpha_2} t$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\twoheadrightarrow}_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\rightarrow}_{\alpha_1,\alpha_2} t \wedge \mathsf{ran}_{\alpha_1,\alpha_2}(f) =_{\mathsf{set}(\alpha_2)} t$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \twoheadrightarrow_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\twoheadrightarrow}_{\alpha_1,\alpha_2} t \wedge f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \rightarrow_{\alpha_1,\alpha_2} t$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{\rightarrowtail\!\!\!\!\!+\!\!\!\twoheadrightarrow}_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{\rightarrowtail\!\!\!\!\!+}_{\alpha_1,\alpha_2} t \wedge f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{+\!\!\!\twoheadrightarrow}_{\alpha_1,\alpha_2} t$$
$$f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \mathrel{\rightarrowtail\!\!\!\!\twoheadrightarrow}_{\alpha_1,\alpha_2} t \longrightarrow$$
$$\quad f \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \rightarrowtail_{\alpha_1,\alpha_2} t \wedge x \in_{\mathsf{set}(\mathsf{tup}(\alpha_1,\alpha_2))} s \twoheadrightarrow_{\alpha_1,\alpha_2} t$$

Fig. 2. Rewriting rules of the B set theory (part 2)

theories that rely on elaborate type systems, like the B set theory. The complete type system used here can be found in [3]. The type constructors, i.e. tup for tuples and set for sets, and type schemes of the considered set constructs are provided in Fig. 3 of Appendix A, where Type is the type of types and o the type of formulas. Type arguments are subscript annotations of the construct, and to improve readability, we remove the type annotations in tuples when they are redundant with the membership construct.

Table 1. Experimental results over the B set theory benchmark

319 problems	Zenon Modulo	ArchSAT	Zipperposition	Alt-Ergo
Proofs	138	272	306	232
Rate	43.3%	85.3%	95.9%	72.7%
Time(s)	2.86	268.69	109.88	8.42

3 Experimental Results

To test the previous theory, we consider 319 lemmas[1] coming from Chap. 2 of the B-Book [1]. As tools, we consider automated theorem provers able to deal with polymorphic types and rewriting natively. Our set of tools includes: Zenon Modulo (version 0.4.2), a tableau-based prover that is an extension of Zenon to deduction modulo theory; ArchSAT (development version[2]), a prover that combines a SAT solver with tableau calculus and rewriting; and Zipperposition (version 1.5), a prover based on superposition and rewriting. To show the impact of rewriting over the results, we also include the Alt-Ergo SMT solver (version 1.01), which deals with polymorphic types but not rewriting.

The experiment was run on an Intel Xeon E5-1650 v3 3.50 GHz computer, with a timeout of 90 s and a memory limit of 1 GiB. The results are summarized in Table 1. These results show the high performances, in terms of proved problems, obtained by the provers extended to rewriting, Zipperposition and ArchSAT in particular, compared to the SMT approach of Alt-Ergo. Looking at the cumulative times, Alt-Ergo is not really faster than Zipperposition and ArchSAT, which take more time to find few more difficult problems (with a timeout of 3 s, they respectively find 303 and 260 proofs in 17.61 s and 16.61 s, while Alt-Ergo finds the same number of proofs). The low results of Zenon Modulo are probably due to the fact that it uses a heuristic to transform the axioms into rewrite rules.

4 Conclusion

In light of the previous experimental results and as perspectives, we aim to apply our approach, consisting of a B set theory using polymorphic types and rewriting together with appropriate tools (Zenon Modulo, ArchSAT, and Zipperposition), to proof obligations coming from the formalization of real-world applications. In particular, we plan to use the benchmark provided by the industrial partners of the BWare project [4], which gathers about 13,000 proof obligations.

[1] The benchmark is available at: https://github.com/delahayd/bset.
[2] Git version 7720d8c, available at: https://gforge.inria.fr/projects/archsat.

A Typing of the B Set Theory

Type Constructors

$\mathsf{tup} : \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{Type}$ $\qquad$ $\mathsf{set} : \Pi\alpha : \mathsf{Type}.\mathsf{Type}$

Type Schemes of the Set Constructs

$- \in -$	$: \Pi\alpha : \mathsf{Type}.\alpha \to \mathsf{set}(\alpha) \to o$
$(-,-)$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\alpha_1 \to \alpha_2 \to \mathsf{tup}(\alpha_1, \alpha_2)$
$- \times -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\alpha_1) \to \mathsf{set}(\alpha_2) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2))$
$\mathbb{P}(-)$	$: \Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha) \to \mathsf{set}(\mathsf{set}(\alpha))$
$- = -$	$: \Pi\alpha : \mathsf{Type}.\alpha \to \alpha \to o$
BIG	$: \Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha)$
$- \subseteq -,\ - \subsetneq - :$	$\Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha) \to \mathsf{set}(\alpha) \to o$
$- \cup -,\ - \cap -,\ - - - :$	$\Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha) \to \mathsf{set}(\alpha) \to \mathsf{set}(\alpha)$
$\{-\}$	$: \Pi\alpha : \mathsf{Type}.\alpha \to \mathsf{set}(\alpha)$
$\emptyset$	$: \Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha)$
$\mathbb{P}_1(-)$	$: \Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha) \to \mathsf{set}(\mathsf{set}(\alpha))$
$- \leftrightarrow -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\alpha_1) \to \mathsf{set}(\alpha_2) \to \mathsf{set}(\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)))$
$-^{-1}$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_2, \alpha_1))$
$\mathsf{dom}(-)$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\alpha_1)$
$\mathsf{ran}(-)$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\alpha_2)$
$- ; -$	$: \Pi\alpha_1, \alpha_2, \alpha_3 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_2, \alpha_3)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_3))$
$- \circ -$	$: \Pi\alpha_1, \alpha_2, \alpha_3 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_2, \alpha_3)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_3))$
$\mathsf{id}(-)$	$: \Pi\alpha : \mathsf{Type}.\mathsf{set}(\alpha) \to \mathsf{set}(\mathsf{tup}(\alpha, \alpha))$
$- \lhd -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\alpha_1) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2))$
$- \rhd -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\alpha_2) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2))$
$- ⩤ -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\alpha_1) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2))$
$- ⩥ -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\alpha_2) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2))$
$-[-]$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\alpha_1) \to \mathsf{set}(\alpha_2)$
$- \mathbin{<\!\!+} -$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2))$
$- \otimes -$	$: \Pi\alpha_1, \alpha_2, \alpha_3 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_3)) \to \mathsf{set}(\mathsf{tup}(\alpha_1, \mathsf{tup}(\alpha_2, \alpha_3)))$
$\mathsf{prj}_1(-)$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{tup}(\mathsf{set}(\alpha_1), \mathsf{set}(\alpha_2)) \to \mathsf{set}(\mathsf{tup}(\mathsf{tup}(\alpha_1, \alpha_2), \alpha_1))$
$\mathsf{prj}_2(-)$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{tup}(\mathsf{set}(\alpha_1), \mathsf{set}(\alpha_2)) \to \mathsf{set}(\mathsf{tup}(\mathsf{tup}(\alpha_1, \alpha_2), \alpha_2))$
$- \parallel -$	$: \Pi\alpha_1, \alpha_2, \alpha_3, \alpha_4 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \mathsf{set}(\mathsf{tup}(\alpha_3, \alpha_4)) \to \mathsf{set}(\mathsf{tup}(\mathsf{tup}(\alpha_1, \alpha_3), \mathsf{tup}(\alpha_2, \alpha_4)))$
$- ⇸ -,\ - \to -,\ - ⤔ -,\ - ↣ -,\ - ⤀ -,\ - ↠ -,\ - ⤗ -,\ - ⤖ - :$	$\Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\alpha_1) \to \mathsf{set}(\alpha_2) \to \mathsf{set}(\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)))$
$-(-)$	$: \Pi\alpha_1, \alpha_2 : \mathsf{Type}.\mathsf{set}(\mathsf{tup}(\alpha_1, \alpha_2)) \to \alpha_1 \to \alpha_2$

Fig. 3. Type constructors and type schemes of the set constructs

References

1. Abrial, J.-R.: The B-Book, Assigning Programs to Meanings. Cambridge University Press, Cambridge (1996). ISBN 0521496195
2. Blanchette, J.C., Paskevich, A.: TFF1: the TPTP typed first-order form with rank-1 polymorphism. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 414–420. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_29
3. Bury, G., Delahaye, D., Doligez, D., Halmagrand, P., Hermant, O.: Automated deduction in the B set theory using typed proof search and deduction modulo. In: Fehnker, A., McIver, A., Sutcliffe, G., Voronkov, A. (eds.) Logic for Programming, Artificial Intelligence and Reasoning (LPAR) - Short Presentations, vol. 35, pp. 42–58. EasyChair, Suva (Fiji), November 2015
4. Delahaye, D., Dubois, C., Marché, C., Mentré, D.: The BWare project: building a proof platform for the automated verification of B proof obligations. In: Ameur, Y.A., Schewe, K.-D. (eds.) ABZ 2014. LNCS, vol. 8477, pp. 126–127. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43652-3_26
5. Dowek, G., Hardin, T., Kirchner, C.: Theorem proving modulo. J. Autom. Reasoning (JAR) **31**(1), 33–72 (2003)