From Event-B to Lambdapi

Anne Grieu

▶ **To cite this version:**

Anne Grieu. From Event-B to Lambdapi. 10th International Conference on Rigourous State-Based Methods (ABZ 2024), ABZ, Jun 2024, Bergamo, Italy. pp.387-391, 10.1007/978-3-031-63790-2_29 . hal-04691826

HAL Id: hal-04691826

https://ut3-toulouseinp.hal.science/hal-04691826

Submitted on 9 Sep 2024

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

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

From Event-B to Lambdapi *

Anne Grieu[1] [0009−0006−3020−3660]

IRIT, Université de Toulouse, CNRS, INP, UT3, Toulouse, France
anne.grieu@irit.fr

Abstract. B, Event-B and TLA+ are modelling notations based on set theory. Dedukti/Lambdapi is a logical framework based on the $\lambda\Pi$-calculus modulo rewriting in which many theories and logics can be expressed. In the context of ICSPA (ANR project), Lambdapi will be used to exchange models and proofs between the set theory-based formal methods B, Event-B and TLA+. They will rely on the encoding of the respective set theories in Lambdapi. Our current work focuses on translating the mathematical language of Event-B and proof trees obtained with the Rodin platform for Event-B.

1 Context

Deductive formal methods are used to improve confidence in software development, especially for critical systems. The purpose of the ICSPA[1] ANR project is to formally verify proofs performed in B [2], Event-B [3] and TLA+ [13] environments, all based on set theory, and to exchange proofs and models developed with their respective platforms Atelier B [9], Rodin [4] and TLAPS [8]. The framework chosen to express this interoperability is Dedukti/Lambdapi [6] [10], a logical framework making easy the definition of various logics. It is based on Martin-Löf's type theory and supports modulo reasoning through rewrite rules. We will here focus on Event-B and the Rodin Platform, an Eclipse-based IDE and logical framework.

Other works on Atelier B, like the BWare project [11] and its recent development [14] or reconstruction of TLAPS proofs [5], share this objective of interoperability with Dedukti/Lambdapi but have different approaches. For example, in the BWare project, Atelier B proof obligations are translated to Why3 [1] platform which submits it to different SMT provers and then to Dedukti. However the proof effort and the structure of the proof realised in Atelier B is lost. It is expected that the proof can be entirely performed by an SMT tool. Here we propose to translate not only proof obligations but also the user provided proofs so we are able to verify proofs realised with the Rodin toolbox and its connected SMT-solvers. In this first work, our aim is to be able to check in the Lambdapi framework, proof-trees restricted to Rodin basic deduction rules, thus excluding calls to automatic provers.

* Supported by the ICSPA french ANR project.

[1] https://icspa.inria.fr

2 Current work

A Rodin proof contains a proof tree annotated by intermediate sub-goals defined by Event-B formulas, the applied deduction rule and its parameters (expressions, hypotheses, positions in the term, ...). To be able to check proofs made with Rodin in Lambdapi, we have to translate the mathematical language of Event-B and its deduction rules to Lambdapi. The formalism of Event-B is based on first order classical logic with equality extended with arithmetic and set theory. The embedding to Lambdapi relies on Lambdapi standard library of first order logic and equality [2], to which we have added the *classic* axiom $(P \vee \neg P)$. Next we have to express the typed set theory of B/Event-B [2] in Lambdapi and then, when we are able to fully express predicates, we have to perform the translation of the deduction rules of Rodin in Lambdapi. In the following, we present some elements of the Lambdapi syntax and the outline of our work.

2.1 Elements of the Lambdapi syntax

Lambdapi is a proof assistant, an interactive version of the proof system Dedukti based on $\lambda\Pi$-calculus modulo rewriting [7]. Here are examples of its syntax, as found in the file `Prop.lp`. First, we see how to declare the type `Prop` of propositions and the function π which associates to a proposition the type of its proofs. The type-checking of Lambdapi is performed between objects of type `TYPE`. The command `symbol` declares typed identifiers with an optional definition.

```
constant symbol Prop: TYPE;    injective symbol π: Prop → TYPE;
```

We can declare as well the logical connectors, for example, the conjunction, with its signature and its introduction rule:

```
constant symbol ∧: Prop → Prop → Prop;
symbol ∧ᵢ p q: π p → π q → π(p ∧ q);
```

We have also at our disposal the `rule` command to give rewriting rules between terms formed using already declared symbols, separated with the curved arrow $\hookrightarrow$. The two parts of the rewriting rule will be considered as identical, thus allowing reasoning inside the quotient space. It is illustrated by the rule for implication saying that the proof of an implication is a function associating a proof of the conclusion to the proof of the hypothesis:

```
rule π ($p ⇒ $q) ↪ π $p → π $q;
```

2.2 Some design alternatives

An important part of our work is to complete the implementation we began, taking into account the different possibilities to express constructs in Lambdapi.

[2] https://github.com/Deducteam/lambdapi-stdlib

For example, we thought about different ways to express the negation in Lambdapi. The connector can be defined with its signature and some rewriting rules, as seen in the left of Fig. 1. Or it could be a definition, with the command `symbol`, using the implication already declared. Then the properties of negation must be proved as theorems as seen in the right of Fig. 1.

```
symbol ¬ :Prop → Prop;
rule ¬ ($P ∧ $Q) ↪ ¬ $P ∨ ¬ $Q
with ¬ ($P ∨ $Q) ↪ ¬ $P ∧ ¬ $Q
with ¬ ($P ⇒ $Q) ↪ $P ∧ ¬ $Q
with ¬ ⊤ ↪ ⊥
with ¬ ⊥ ↪ ⊤
with ¬ (¬ $P) ↪ $P
with ¬ (∃ (λ x, $P.[x]))
          ↪ ∀ (λ x, ¬ ($P.[x]))
with ¬ (∀ (λ x, $P.[x]))
          ↪ ∃ (λ x, ¬ ($P.[x]));
```

```
symbol ¬ p ≔ p ⇒ ⊥;
symbol not_or [P Q: Prop]:
         π (¬ (P ∨ Q) ⇒ ¬ P ∧ ¬ Q) ≔
begin
  assume P Q h;
  apply  ∧ᵢ _ _ _
    {assume p; apply h;
    apply ∨ᵢ₁; apply p}
    {assume q; apply h;
    apply ∨ᵢ₂; apply q}
end;
```

Fig. 1. Different alternatives to define negation.

The choice will have consequences on the upcoming proofs and on the contents of the proof term as rule application is implicit. However rule-based specifications will not be close to the usual formalization of classical logic. For now, we use rules but it makes the kernel logic diverge from that of Rodin. We are modifying our encoding, which makes negation elimination much more complex, but closer to what is performed by the Rodin prover.

Another part is to take into account the peculiarities of Rodin. For instance, some Rodin operators and thus some proof nodes are n-ary, while Lambdapi's operators are of a fixed arity. Currently the Java plug-in generates n-ary proof schemes to match the structure of Rodin proof trees (for instance **And** nodes). An improvement could be to do that processing in Lambdapi, so it could be used by other systems, like TLA+, with n-ary operators.

2.3 Cantor's theorem

To guide the beginning of our work, we made a user guided proof of Cantor's theorem[3] in Rodin, without using internal or SMT provers, and we worked on checking the translation of our proof with Lambdapi. Fig. 2 (left) shows the beginning of the proof tree made with Rodin and the right part shows the four first steps of the translated proof. The gray boxes, with rule details extracted from the Rodin proof tree, are not part of the Lambdapi script. In the comments (lines beginning with //), we can read the nodes exported by Rodin and between the comments and the gray boxes, the Lambdapi tactics.

[3] The Cantor's theorem has been proposed as a case study in ICSPA.

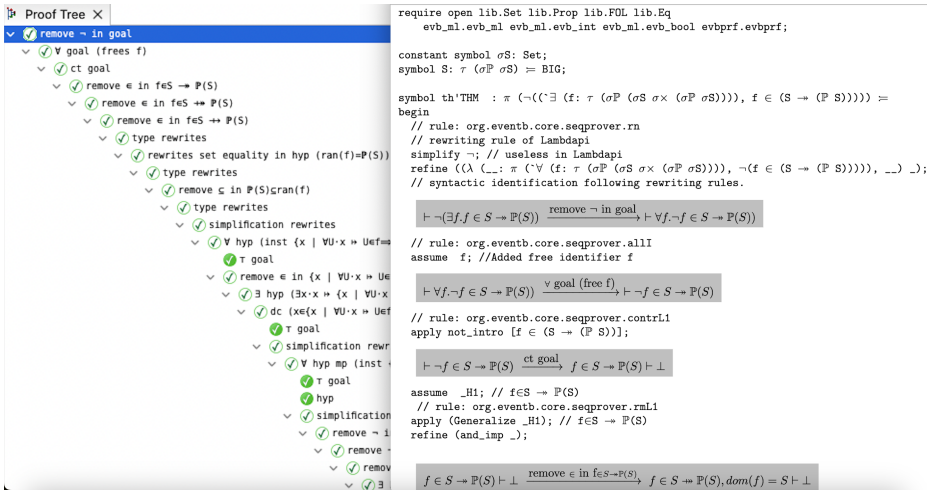

Fig. 2. Rodin proof tree and its Lambdapi translation.

3 Future work

There is still a significant work ahead to perform the complete translation of Event-B proofs to Lambdapi.

The development is still at its beginning and no prototype is available yet. As it was suggested in the previous section, the choices of the representations are not fixed and are meant to move to answer future needs. We have also to complete the translation of all the deduction rules of Rodin.

Furthermore, the automated provers are a strength of Rodin. They are provided by integrated (deduction or rewriting-based) tools or external SMT solvers fig. 3 for which Rodin removes set constructs. After treatment, the SMT solver gives its result to Rodin. Then Lambdapi terms or tactics should be built from internal provers and SMT proof traces and incorporated to the current proof. Then, we will have to express the full formalism of Event-B, to define machines, events and the mechanisms of proof obligations, some of the major features of the formal method to be able to export Event-B developments.

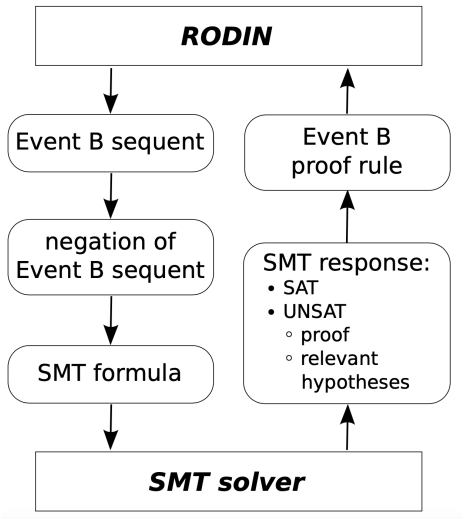

Fig. 3. Rodin and SMT solvers[12].

References

1. Why3, a tool for deductive program verification, GNU LGPL 2.1, https://www.why3.org/
2. Abrial, J.R.: The B-Book - Assigning programs to meanings. Cambridge University Press (1996)
3. Abrial, J.: Modeling in Event-B - System and Software Engineering. Cambridge University Press (2010). https://doi.org/10.1017/CBO9781139195881, https://doi.org/10.1017/CBO9781139195881
4. Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. STTT **12**(6), 447–466 (2010)
5. Alessio, C.: Reconstruction of TLAPS proofs solved by VeriT in lambdapi. In: Glässer, U., Campos, J.C., Méry, D., Palanque, P.A. (eds.) Rigorous State-Based Methods - 9th International Conference, ABZ 2023, Nancy, France, May 30 - June 2, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14010, pp. 375–377. Springer (2023). https://doi.org/10.1007/978-3-031-33163-3_29, https://doi.org/10.1007/978-3-031-33163-3_29
6. Assaf, A., Burel, G., Cauderlier, R., Delahaye, D., Dowek, G., Dubois, C., Gilbert, F., Halmagrand, P., Hermant, O., Saillard, R.: Expressing theories in the $\lambda\Pi$-calculus modulo theory and in the Dedukti system. In: TYPES: Types for Proofs and Programs. Novi SAd, Serbia (May 2016), https://minesparis-psl.hal.science/hal-01441751
7. Boespflug, M., Carbonneaux, Q., Hermant, O.: The $\lambda\prod$-calculus modulo as a universal proof language. In: Pichardie, D., Weber, T. (eds.) Proceedings of the Second International Workshop on Proof Exchange for Theorem Proving, PxTP 2012, Manchester, UK, June 30, 2012. CEUR Workshop Proceedings, vol. 878, pp. 28–43. CEUR-WS.org (2012), http://ceur-ws.org/Vol-878/paper2.pdf
8. Chaudhuri, K., Doligez, D., Lamport, L., Merz, S.: Verifying safety properties with the TLA + proof system (11 2010). https://doi.org/10.1007/978-3-642-14203-1_12
9. CLEARSY: Atelier B Tool (2024), https://www.atelierb.eu/en/
10. Cousineau, D., Dowek, G.: Embedding pure type systems in the Lambda-Pi-calculus modulo (2023)
11. Delahaye, D., Dubois, C., Marché, C., Mentré, D.: The BWare project: Building a proof platform for the automated verification of b proof obligations. In: Proceedings of the 4th International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z - Volume 8477. p. 290–293. ABZ 2014, Springer-Verlag, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43652-3_26, https://doi.org/10.1007/978-3-662-43652-3_26
12. Déharbe, D., Fontaine, P., Guyot, Y., Voisin, L.: Integrating SMT solvers in Rodin. Science of Computer Programming **94**, 130–143 (2014). https://doi.org/https://doi.org/10.1016/j.scico.2014.04.012, https://www.sciencedirect.com/science/article/pii/S016764231400183X, Abstract State Machines, Alloy, B, VDM, and Z
13. Lamport, L.: Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley (2002), http://research.microsoft.com/users/lamport/tla/book.html
14. Stolze, C., Hermant, O., Guillaumé, R.: Towards Formalization and Sharing of Atelier B Proofs with Dedukti (Jan 2024), https://hal.science/hal-04398119, working paper or preprint