Verification of Event-B proofs throught their translation to Lambdapi

Anne Grieu[1] and Jean-Paul Bodeveix[1]

[1]IRIT, INP, Université de Toulouse, CNRS, Toulouse, France , prenom.nom@irit.fr

Rodin Workshop - June 2025 - Düsseldorf

In order to verify formally the Event-B[1, 2] proofs made by Rodin[3], we are working on an embedding of its mathematical language and proof system in Dedukti/Lambdapi [7, 12]. This logical framework, based on the ΛΠ-calculus modulo, allows implementing various logics and is already used to enable interoperability between proof systems[5, 14]. We use Dedukti as a target-framework to work, in addition, on interoperability of proof systems from various other set-theory, like B and its framework Atelier B[8, 13], TLA+ and its framework TLAPS[4, 10][1].

For this purpose, instead of using a specific library to express the mathematical logic of Event-B[6], we use a library, called `lambdapi-stdlib`[2], of which development is still in progress. It provides basic components to support various logics (propositional and first order logic, HOL, CoC, ...). Using parts of the `lambdapi-stdlib`, we can express the mathematical language of Event-B. On top of this library, we define in Lambdapi the type language of Event-B. Each specific operator of Event-B is defined with its signature and rewriting rules to define their behaviour. Then, we state and prove theorems that will be useful to translate the proofs[11]. To structure the proofs, we introduce theorems, meta-theorems and proof terms generators to encode actions in the proofs made by Rodin (deduction rules, rewriting rules, tactics..).

We present a Rodin plug-in, still in development. It takes as input a Rodin bps file, that gives access to obligation proofs, thanks to Rodin API, and generates a `lambdapi` file to be checked by Lambdapi. The plug-in translates the context, the sets, its formulas, using type information given by Rodin (e.g. when you write $\forall x.x > 0$, the formula that will be translated in Lambdapi is $\forall x \,^{\circ}_{\circ}\, Z$). It translates also the proofs generated by Rodin, following recursively the proof tree and generating a proof term for Lambdapi with the same structure of the Rodin proof tree. Each node matches with one or more rule applied to one or more hypothesis, that discharge a goal or create new goals to be discharged. Some translations are straightforward, but some others require more complex treatment. In the following we illustrate some of these translations. We use these notations:

- x is a variable, $P, P1, P2, P3, G$ are propositions,
- H_n is an identifier created by the plug-in to name the n^{th} hypothesis added in the context,
- `assume`, `apply`, `refine` are Lambdapi tactics[3],
- $\wedge_i$ is a term/theorem from lambdapi-stdlib

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

- `Or2Imp` is a term/theorem of our library:

```
symbol Or2ImpGoal [P Q: Prop] :
    π (((¬ P) ⇒ Q) ⇒ (P ∨ Q));.
```

[1]https://anr.fr/Projet-ANR-21-CE25-0015
[2]https://github.com/Deducteam/lambdapi-stdlib/blob/master/README.md
[3]https://lambdapi.readthedocs.io/en/latest/tactics.html

- Straightforward translation, using tactics:

$\forall$ goal (free x) : $\forall x \cdot P(x)$	$\rightsquigarrow$	`assume x;`	adds x in the context
$\Rightarrow$ goal : P $\Rightarrow G$	$\rightsquigarrow$	`assume` H_n`;`	adds a proof of P in the context
$\top$ goal : $\top$	$\rightsquigarrow$	`refine` $\top_i$`;`	discharges True goal with a proof of true

- Simple translation, applying theorems:

$\vee$ to $\Rightarrow$ in goal : P1 $\vee$ P2	$\rightsquigarrow$	`apply Or2ImpGoal;`	replaces the goal $P1 \vee P2$ by $\neg P1 \Rightarrow P2$
$\wedge$ goal : P1 $\wedge P2$	$\rightsquigarrow$	`refine` $\wedge_i$ `_ _{}{};`	creates 2 new sub-goals P1 and P2

➜ Some behaviours of Rodin needs a special treatment to be expressed in Lambdapi, with the help of the plug-in.

- N-ary operators:

  Some operators are n-ary in Rodin, we can't apply in a straightforward way a Lambdapi-theorem to get the same behaviour with the binary operators defined. For the conjunction of n propositions, for instance, the plug-in generates a composition of introduction rules that will create n new goals.

  $\wedge$ goal : P1 $\wedge P2 \wedge P3$ $\rightsquigarrow$ `refine` $(\wedge_i$ `[P1` $\wedge$ `(P2` $\wedge$ `P3)] _ (` $\wedge_i$ `[P2] [P3] _ _) ){}{}{};`

- Automated actions of Rodin, like elimination of duplicated propositions, are proved once and for all at a meta level in Lambdapi, in a proof-by-reflection approach and the plug-in will instantiate the generic theorem with a mapping function between the position of the propositions and the propositions, in the case of the elimination of duplicated terms and the operator considered by the application of the theorem.

- Call an external prover of Rodin:

  When some sub-goals are proved with the help of a call to an SMT or some other internal prover of Rodin, we handle them by using Zenon modulo[9]. The plug-in will send to Zenon modulo, thanks to a TPTP translation, the lemma to prove, then the obtained lambdapi proof is integrated to the whole proof.

The plug-in we are building is in fact a bridge between Rodin and Lambdapi. We have presented some features of the plug-in, that will be extended to take in account the new needs to include all characteristics of Event-B and Rodin, like a hundred of rules and include the WD. To experiment our embedding of Event-B in Lambdapi, we work on various examples, for instance, several versions of Cantor's theorem.

References

[1] Jean-Raymond Abrial. The B-book - assigning programs to meanings. Cambridge University Press, 1996.

[2] Jean-Raymond Abrial. Modeling in Event-B - System and Software Engineering. Cambridge University Press, 2010.

[3] Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, Thai Son Hoang, Farhad Mehta, and Laurent Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. STTT, 12(6):447–466, 2010.

[4] Coltellacci Alessio. Reconstruction of TLAPS proofs solved by verit in lambdapi. In Uwe Glässer, José Creissac Campos, Dominique Méry, and Philippe A. Palanque, editors, Rigorous State-Based Methods - 9th International Conference, ABZ 2023, Nancy, France, May 30 - June 2, 2023, Proceedings, volume 14010 of Lecture Notes in Computer Science, pages 375–377. Springer, 2023.

[5] Ali Assaf, Guillaume Burel, Raphal Cauderlier, David Delahaye, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Expressing theories in the λΠ-calculus modulo theory and in the Dedukti system. In TYPES: Types for Proofs and Programs, Novi SAd, Serbia, May 2016.

[6] Laurent Voisin Christophe Métayer. The Event-B mathematical language, 2009. `https://webarchive.southampton.ac.uk/deployeprints.ecs.soton.ac.uk/11/4/kernel_lang.pdf`.

[7] Denis Cousineau and Gilles Dowek. Embedding pure type systems in the Lambda-Pi-Calculus Modulo. In Simona Ronchi Della Rocca, editor, Typed Lambda Calculi and Applications, pages 102–117, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[8] David Delahaye, Catherine Dubois, Claude Marché, and David Mentré. 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, ABZ 2014, page 290–293, Berlin, Heidelberg, 2014. Springer-Verlag.

[9] David Déharbe, Pascal Fontaine, Yoann Guyot, and Laurent Voisin. Integrating SMT solvers in Rodin. Science of Computer Programming, 94:130–143, 2014. Abstract State Machines, Alloy, B, VDM, and Z.

[10] Anne Grieu. From Event-B to lambdapi. In Silvia Bonfanti, Angelo Gargantini, Michael Leuschel, Elvinia Riccobene, and Patrizia Scandurra, editors, Rigorous State-Based Methods, pages 387–391, Cham, 2024. Springer Nature Switzerland.

[11] Anne Grieu and Jean-Paul Bodeveix. Encodage du langage mathématique d'Event-B dans lambdapi. In 36es Journées Francophones des Langages Applicatifs (JFLA 2025), Jan 2025, Roiffé, France, 2025.

[12] G. Hondet and F. Blanqui. The New Rewriting Engine of Dedukti. In Proceedings of the 5th International Conference on Formal Structures for Computation and Deduction, Leibniz International Proceedings in Informatics 167, 2020.

[13] Claude Stolze, Olivier Hermant, and Romain Guillaumé. Towards Formalization and Sharing of Atelier B Proofs with Dedukti. working paper or preprint, January 2024.

[14] François Thiré. Sharing a library between proof assistants: Reaching out to the hol family. Electronic Proceedings in Theoretical Computer Science, 274:57–71, July 2018.