BARReL: un backend moderne pour Atelier B dans Lean

Démonstration d'outil—AFADL 2026

Ghilain Bergeron et Vincent Trélat
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

18 mai 2026

Résumé

BARReL est une bibliothèque Lean 4 qui fait le lien entre Atelier B, un outil industriel pour la méthode B, et l'assistant à la preuve Lean, en permettant aux utilisateurs de mener leurs développements formels en B—du raffinement des machines jusqu'à l'implémentation—de manière interactive dans Lean, tout en conservant la syntaxe standard de B. Les opérateurs partiels de B sont encodés soigneusement en générant des conditions explicites de bonne définition, en tirant parti des types dépendants de Lean pour imposer, par construction, une discipline de bonne définition. Autrement dit, les obligations de preuve et les étapes de preuve ne peuvent pas reposer implicitement sur des objets mal typés ou mal définis. BARReL propose aussi une automatisation de base pour tenter de décharger automatiquement de telles conditions de bonne définition. L'implémentation utilise les fonctionnalités de métaprogrammation de Lean et est conçue pour être modulaire : étendre le fragment de B pris en charge revient généralement à ajouter de la nouvelle syntaxe et des clauses d'encodage. Nous illustrons l'approche sur une étude de cas représentative et affirmons que BARReL peut constituer une première étape vers une chaîne d'outils fiables pour la méthode B.

Mots-clés : Lean 4, Atelier B, métaprogrammation, preuve interactive, méthode B, obligations de preuve.

1 Introduction

Le support pour l'utilisation d'assistants à la preuve modernes comme backends d'Atelier B [5] reste très limité : les travaux existants ciblent principalement Isabelle [7, 8, 3] pour Event-B [1], ou Rocq [9] avec un accent sur la sémantique formelle de B plutôt que sur un backend intégré [4].

Cet article présente BARReL (B Automated tRanslation for Reasoning in Lean), une bibliothèque Lean 4 [6] qui transforme les obligations de preuve (*proof obliga-*

tions, PO) de B en théorèmes Lean utilisant les primitives ensemblistes de Mathlib [2], en mettant l'accent sur le traitement explicite de la *bonne définition* (*well-definedness*, WD) des opérateurs partiels : chaque utilisation d'un tel opérateur requiert une preuve que ses conditions WD sont satisfaites, empêchant par construction toute instanciation mal définie. BARReL est disponible publiquement sur GitHub.[1]

2 Architecture et fonctionnement

2.1 Chaine de traduction

Le principe de fonctionnement de BARReL (cf. Figure 1) comprend quatre étapes : (1) le fichier B est converti au format BXML puis les PO sont générées via les outils `bxml` et `pog` d'Atelier B ; (2) le fichier XML est analysé et transformé par BARReL en un AST normalisé; (3) les formules B sont traduites en termes Lean en utilisant la métaprogrammation, en prenant soin de traiter correctement les opérateurs partiels; (4) les conditions WD et les PO sont présentées comme des théorèmes Lean, après une passe d'automatisation ciblant les conditions WD de routine.

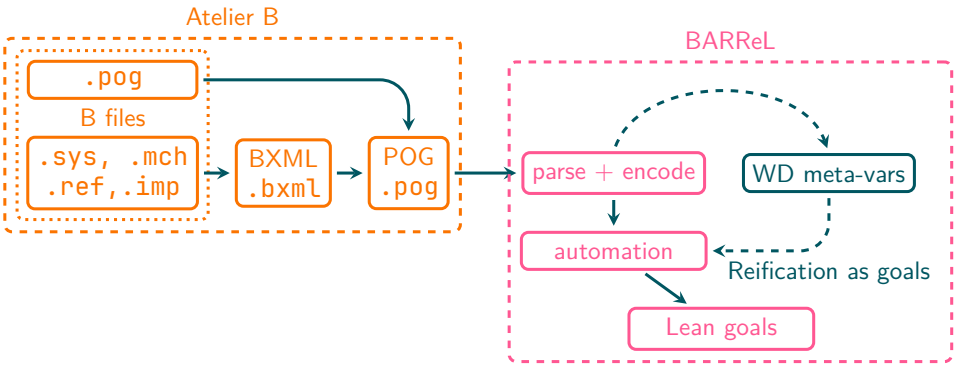

Figure 1: Chaine de fonctionnement de BARReL.

Côté utilisateur, le processus est piloté par deux commandes Lean :

```
import machine MinSearch from "specs/case_study"
prove_obligations_of MinSearch
```

La première commande lance le pipeline complet ; la seconde présente les PO et conditions WD sous forme de buts Lean.

2.2 Encodage de B dans Lean

Les opérateurs standard de B sont définis en Lean en réutilisant les types Set α ($\triangleq \alpha \to \text{Prop}$) et SetRel $\alpha\ \beta$ ($\triangleq \text{Set}\ (\alpha \times \beta)$) de Mathlib. Les notations de B sont

[1] https://github.com/VTrelat/BARReL

reproduites fidèlement afin que les PO restent lisibles par un utilisateur B ; BARReL couvre les principaux opérateurs logiques, ensemblistes, relationnels, fonctionnels et arithmétiques.

2.3 Traitement des opérateurs partiels

De nombreux opérateurs B—min, max, application fonctionnelle, cardinalité—ne sont définis que sous certaines conditions. Atelier B génère des conditions WD en parallèle des PO principales mais sans dépendance explicite : rien ne lie formellement la condition WD au terme dans lequel l'opérateur partiel apparaît. De plus, les conditions WD ne sont pas réémises lors des étapes de preuve interactive, ce qui peut conduire à des incohérences si l'utilisateur instancie des variables quantifiées avec des termes mal définis.

BARReL adopte une approche différente : chaque opérateur partiel est encodé comme une fonction *totale* prenant en paramètre une *preuve* que ses conditions WD sont satisfaites, via `Classical.choose`. Ainsi, il est impossible de construire un terme utilisant un opérateur partiel sans fournir cette preuve. Par exemple, $\mathsf{min}(S)$ et $F(x)$ sont encodés comme suit :

```
noncomputable
abbrev min (S : Set a) (wd : min.WD S) : a :=
  Classical.choose wd.isBoundedBelow

noncomputable
abbrev app (f : SetRel a b) (x : a) (wd : app.WD f x) : b :=
  Classical.choose wd.isInDomain
```

Lors de la transcription d'une formule B contenant un opérateur partiel, BARReL insère une *métavariable* Lean à l'emplacement de la preuve WD. Cette métavariable est ensuite réifiée en un théorème auxiliaire à décharger, garantissant une dépendance explicite entre la condition WD et le terme englobant.

2.4 Automatisation

Avant de présenter les buts à l'utilisateur, BARReL exécute une passe d'automatisation ciblant les conditions WD de routine. Cette couche applique, avec retour arrière, des lemmes auxiliaires étiquetés par catégorie (`wd_min`, `wd_max`, `wd_app`, `wd_card`) afin de ramener les conditions WD à des faits élémentaires d'appartenance, d'inclusion de domaine, de finitude ou de bornes. Les obligations restantes—préservation de l'invariant, simulation de raffinement—sont laissées à l'utilisateur, qui dispose de l'écosystème Lean et Mathlib pour les décharger.

3 Évaluation

BARReL a été évalué sur une chaîne de raffinement à trois niveaux calculant le minimum d'un ensemble fini non vide d'entiers : une spécification abstraite, un premier raffinement réduisant le non-déterminisme et un second introduisant un parcours explicite de tableau. Cette étude de cas est disponible dans le dépôt du projet.[2]

Le tableau 1 compare les statistiques entre Atelier B et BARReL. Côté BARReL, les 156 conditions WD sont toutes automatiquement déchargées ; les 34 PO restantes sont prouvées par des scripts Lean courts. Le nombre de PO diffère au premier raffinement car Atelier B éclate les conjonctions de l'invariant en obligations séparées. Inversement, BARReL génère davantage de conditions WD, car il en produit une par occurrence d'opérateur partiel dans chaque PO, tandis qu'Atelier B les factorise au niveau de la machine ; toutes sont néanmoins déchargées automatiquement dans les deux cas.

Table 1: Statistiques de la chaîne de raffinement MinSearch. «Auto» désigne les obligations déchargées sans intervention (Force 1 pour Atelier B ; automatisation par défaut pour BARReL).

	PO		**WD (Atelier B)**		**WD (BARReL)**	
Fichier	AB	BARReL	Total	Auto	Total	Auto
MinSearch	4	4	4	4	4	4
MinSearch_r1	19	9	2	2	52	52
MinSearch_r2	21	21	9	9	100	100

Au-delà de ces différences quantitatives, BARReL offre une garantie structurelle absente du prouveur interactif d'Atelier B. Certaines commandes d'instanciation de ce dernier (par ex. `se`) n'imposent pas la bonne définition des termes fournis par l'utilisateur, ce qui peut conduire à dériver une contradiction. Sous l'encodage de BARReL, les types dépendants de Lean garantissent que tout opérateur partiel est accompagné de sa preuve WD : de telles instanciations mal définies sont rejetées par construction.

4 Conclusion et perspectives

BARReL est un backend léger et modulaire pour Atelier B dans Lean 4. L'implémentation, réalisée entièrement par métaprogrammation Lean, consiste en environ 1 300 lignes de code pour l'outillage principal et 1 300 lignes pour la bibliothèque de lemmes et l'automatisation. Les directions futures incluent l'extension du fragment de B couvert, la réduction de la redondance des conditions WD, et l'enrichis-

[2] https://github.com/VTrelat/BARReL/tree/master/examples

sement de l'automatisation. À terme, BARReL pourrait également être équipé de son propre générateur de PO, vérifié et intégré directement dans Lean.

Références

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

[2] Anne Baanen, Matthew Robert Ballard, Johan Commelin, Bryan Gin-ge Chen, Michael Rothgang, and Damiano Testa. Growing Mathlib: maintenance of a large scale mathematical library. In Valeria de Paiva and Peter Koepke, editors, *Intelligent Computer Mathematics*, pages 51–70, Cham, 2026. Springer Nature Switzerland.

[3] Benoît Ballenghien and Burkhart Wolff. Event-B as DSL in Isabelle and HOL experiences from a prototype. In Silvia Bonfanti, Angelo Gargantini, Michael Leuschel, Elvinia Riccobene, and Patrizia Scandurra, editors, *Rigorous State-Based Methods - 10th International Conference, ABZ 2024, Bergamo, Italy, June 25-28, 2024, Proceedings*, volume 14759 of *Lecture Notes in Computer Science*, pages 241–247. Springer, 2024.

[4] Jean-Paul Bodeveix and Mamoun Filali. Type synthesis in B and the translation of B to PVS. In Didier Bert, Jonathan P. Bowen, Martin C. Henson, and Ken Robinson, editors, *ZB 2002:Formal Specification and Development in Z and B*, pages 350–369, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[5] Clearsy. Atelier B. https://www.atelierb.eu.

[6] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, page 625–635, Berlin, Heidelberg, 2021. Springer-Verlag.

[7] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.

[8] Matthias Schmalz. *Formalizing the logic of Event-B: Partial functions, definitional extensions, and automated theorem proving*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2012.

[9] The Rocq Development Team. The Rocq prover, September 2025.