Existence Proof Obligations for Constraints, Properties and Invariants in Atelier B

Chapter · May 2020

DOI: 10.1007/978-3-030-48077-6_20

3 authors, including:

David Déharbe
ClearSy System Engineering
114 PUBLICATIONS **940** CITATIONS

SEE PROFILE

Existence Proof Obligations for Constraints, Properties and Invariants in Atelier B

Héctor Ruíz Barradas, Lilian Burdy, and David Déharbe(✉)

CLEARSY Systems Engineering, Aix-en-Provence, France
david.deharbe@clearsy.com

Abstract. Proof obligations of the B method and of Event B use predicates in the Constraints, Sets, Properties and Invariant clauses as hypotheses in proof obligations. A contradiction in these predicates results in trivially valid proof obligations and essentially voids the development. A textbook on the B method [3] presents three "existence proof obligations" to show the satisfiability of the Constraints, Properties and Invariant clauses as soon as they are stated in a component. Together with new existence proof obligations for refinement, this prevents the introduction of such contradictions in the refinement chain. This paper presents a detailed formalization of these existence proof obligations, specifying their implementation in Atelier B.

1 Introduction

The vaunted rigour of formal methods, such as B and Event-B, not only come from the use of a formal notation, but also from the generation and subsequent verification of proof obligations (POs). For instance, in Event-B [2], the model of a system is considered sound only when all POs have been demonstrated. In the B method [1], they guarantee that the refinement-based construction results in implementations faithful to their specification.

Typically, POs are generated at key steps of the design process. Invalid POs reveal errors in the source artefact. By inspecting these proof obligations, the user then identifies, possibly, remaining errors and fixes the source artefact. The process is repeated until all POs are discharged. To conduct the demonstrations, these methods demand that they are conducted with tools. In practice, this is accomplished by a mix of automatic proof and interactive proof. POs are thus the cornerstone of every such formal development.

A PO has the form $H \vdash G$, with H a set of hypotheses, and G the goal. Its validity may stem from a contradiction in H, i.e. have nothing to do with the goal. In the context of B and Event-B, a component with contradictory hypotheses in its POs will be (trivially) correct. In large developments, a contradiction may stay undetected. B addresses this issue with POs associated at the implementation level, i.e. at the very end of the development. At that point, this requires fixing the refinement chain up to the source of the contradiction, which

A. Raschke et al. (Eds.): ABZ 2020, LNCS 12071, pp. 255–259, 2020.
https://doi.org/10.1007/978-3-030-48077-6_20

is costly. Also, components in a B project that do not have an implementation (e.g., foreign interfaces) are not protected. Event-B does not fully address this issue.

Such situations can be easily avoided by adding so called "existence" POs whenever a contradiction may be introduced. An existence PO has the form $\Gamma \Rightarrow \exists V \cdot (\varphi)$, where Γ is the context predicate, φ the predicate that shall not be contradictory, and V a list of identifiers. A textbook on B [3] presents these POs, but without considering component visibility, inclusion and refinement. Existing tools for B and Event-B do not generate these, and we decided to add it to Atelier B. We present the formalization of the POs for the specification (Sect. 2) and the refinement (Sect. 3) levels. We discuss the case of standalone components, and generalize to components with dependencies.

2 Existence Proof Obligations in Specifications

Existence for Parameters. In B, specification components may have sets and scalar parameters. The CONSTRAINTS clause can be used to constrain these parameters. When the machine is instantiated, a PO asks to prove the establishment of the CONSTRAINTS clause, thus guaranteeing the absence of contradictions. If the parametrized machine is not instantiated, the CONSTRAINTS clause can contain undetected contradictions because no PO exists to detect them. Let p denote the parameters, C the predicate in the constraint clause, the existence PO given by [3] for parameters is $\exists p \cdot C$. It has been implemented as such in Atelier B.

Existence for Sets and Constants. The PROPERTIES clause state constraints on sets and constants declared respectively in the SETS and CONSTANTS clauses. Enumerated sets have a single possible valuation, and abstract sets must satisfy the implicit constraint that they are finite non-empty sets of integers. In this way, in order to prove the absence of contradictions in the predicate P of the PROPERTIES clause of a single machine, with no seen or included components, we define the following PO: $e\_sets \Rightarrow \exists(c, s) \cdot (P \wedge a\_sets)$, where e_sets is the conjunction of declarations of enumerated sets in the SETS clause, c is the list of abstract and concrete constants, s is the list of abstract sets, and a_sets is the conjunction of predicates $t \in FIN_1(INTEGER)$ for each variable t in s. Notice that the visibility rules of the language prohibit parameters in the predicate P, so it is useless to have predicate C as an antecedent.

If there are *seen* components in the machine, the predicates in the PROPERTIES clause from the seen components and their included components are in the antecedent of the PO. Moreover, for each abstract set u declared in the seen machine or declared in a machine included by the seen machine, the antecedent of the PO contains a predicate $u \in FIN_1(INTEGER)$. The definition of each enumerated set w declared in these machines is also in the antecedent.

If the machine *includes* components, the definition of their enumerated sets are in the antecedent of the PO, their abstract and concrete constants and the

identifiers of their abstract sets are existentially quantified in the consequent and the predicates of their PROPERTIES clauses, together with the corresponding *a_sets* predicates, are in the body of the existential quantifier.

Following is an example of the existence PO for the SETS, CONSTANTS and PROPERTIES clauses for a standalone component:

SETS
$S1; S2 = \{UM, DOIS, TRES\};$
$S3 = \{UN, DEUX\}$
CONSTANTS
$c1, e1, e2, e3$
PROPERTIES
$c1 \in NAT \wedge e1 \in INT \wedge$
$e2 \in S2 \wedge e3 \in S1 \wedge$
$(e2 = UM \Rightarrow e1 = 1))$

PO:

$S2 = \{UM, DOIS, TRES\} \wedge$
$S3 = \{UN, DEUX\}$
$\Rightarrow$
$\exists(c1, e1, e2, S1) \cdot ($
$S1 \in FIN(INTEGER) - \{\{\}\} \wedge$
$c1 \in NAT \wedge e1 \in INT \wedge$
$e2 \in S2 \wedge e3 \in S1 \wedge$
$(e2 = UM \Rightarrow e1 = 1))$

Existence for State Variables. The predicate invariant may also contain contradictions. To prevent this, the existence PO of the INVARIANT clause for a standalone machine is $C \wedge P \wedge$ *all_sets* $\Rightarrow \exists(v) \cdot (I)$. The antecedent of this PO contains the predicates C and P from the CONSTRAINTS and PROPERTIES clauses. The predicate *all_sets* is the conjunction of *e_sets* and *a_sets* seen above. The quantifed variable v denotes the list of abstract and concrete variables of the machine.

If there are seen or included components, the antecedent is strengthened with the conjunction of their properties, assertions, invariants and their *all_sets* predicates. In this conjunction, we also consider the clauses of the components possibly included by the seen machines. Moreover, for the included components, the consequent of the PO quantifies over their variables and invariants.

3 Existence Proofs in Refinements

Refinement in B or Event B is used for stepwise development. Refinement POs are designed to be monotonic: If a component S is refined by a component T, these POs guarantee that the invariant of S is also preserved by operations in T. However, existence POs in a refinement are not monotonic in that sense. When an abstract constant or variable is refined by a concrete one, we still need to prove that the properties or invariants specified in the abstraction hold in the refinement.

Existence for Sets and Constants. For a refinement with no seen or included components and no seen or included components in any of its abstractions, the existence PO is intended to avoid contradictions in the predicate P of the PROPERTIES clause of the refinement and all properties of the previous refinements, denoted by the following predicate:

$$e\_sets \wedge abs\_e\_sets \Rightarrow \exists(c, c_a, s, s_a) \cdot (P \wedge a\_sets \wedge abs\_P \wedge abs\_a\_sets)$$

The predicates *e_sets* and *a_sets* are defined as before, *abs_e_sets* denotes the conjunction of declarations of enumerated sets, and *abs_a_sets* denotes the conjunction of $t \in FIN_1(INTEGER)$, for abstract sets t in previous refinements. Predicate *abs_P* is the conjunction of the properties predicates in the previous refinements. The variable lists c and s contain the constants of the refinement and its abstract sets. Finally, the lists c_a and s_a denote all constants and abstract sets in previous refinements. If the refinement or any of its abstractions contains seen or included components, the antecedent and the consequent are strengthened with the clauses of these components as it was done in the corresponding PO of the specification.

Existence for State Variables. The corresponding PO defined for specification components guarantees the absence of contradictions in the invariant. Also, the PO of the establishment of the invariant by the initialization $Init_a$ guarantees the existence of values of the abstract variables v_a satisfying the abstract invariant $I(v_a)$. The PO of the refinement of $Init_a$ by the initialization of a refined component $Init_c$ is not sufficient to guarantee the absence of contradictions in the refined invariant $J(v_c, v_a)$. Therefore, in order to prove the absence of contradictions in the invariant $J(v_c, v_a)$ we need to show that the assignment of some concrete values v to the concrete variables v_c is a refinement of $Init_a$. Formally this refinement is stated by $\exists v \cdot ([v_c := v] \neg [Init_a] \neg J$ which must be proved under the context of the refinement. After simplification, the existence PO for a standalone refinement and only standalone components in its abstractions is defined as follows:

$$C \wedge P \wedge all\_sets \wedge abs\_all\_sets \wedge abs\_P \Rightarrow \exists(v_c) \cdot (\neg [Init_a] \neg J)$$

where *abs_all_sets* is the conjunction of predicates *all_sets* of previous refinements, v_c is the list of abstract and concrete variables of the refinement and J is its invariant.

If there are seen or included components, the antecedent and consequent of the PO are strengthened with the corresponding clauses of these components.

4 Conclusion

This paper presents details of the generation of existence POs for the formal methods B and Event-B. These POs detect inconsistencies that would make trivial, but useless, the correctness of the components, as soon as they are introduced in the development. Their generation has been implemented and will be available in a future release of Atelier B.

References

1. Abrial, J.-R.: The B-Book, Assigning Programs to Meanings. Cambridge University Press, Cambridge (1996)
2. Abrial, J.-R.: Modelling in Event-B, System and Software Engineering. Cambridge University Press, Cambridge (2010)
3. Schneider, S.: The B-Method. Macmillan International, New York (2001)