Reasoning Heuristics for the Theorem-Proving Platform Rodin/Event-B

Jacobus Gideon Ackermann
School of Computing
College of Science, Engineering and Technology
University of South Africa
Florida, South Africa
linear.transformation@gmail.com

John Andrew van der Poll
Graduate School of Business Leadership (SBL)
University of South Africa
Midrand, South Africa
vdpolja@unisa.ac.za
(*contact author*)

***Abstract*—Developments in formal- and mathematical logic; and computing the past couple of decades have paved the way for the automation of deductive reasoning. However, despite theoretical and technological advances in computing, the rapid growth in the search space for complex proofs where the reasoner explores the consequences of irrelevant information, remains problematic. The challenge of a combinatorial explosion of the search space can in many cases be addressed by heuristics. Consequently, in this paper we investigate the extent to which heuristics may usefully be applied in discharging complex set-theoretic proof obligations using the hybrid reasoning environment, Rodin/Event-B. On the strength of our experiments, we develop a set of heuristics to aid the theorem-proving environment in finding proofs for set-theoretic problems which could not be obtained using the default settings. A brief exposition of related work in this area is presented towards the end of the paper.**

***Index Terms*—Automated Reasoning, Event-B/Rodin, Formal specification, Set theory, Theorem proving, Z**

I. Introduction

Many of the specification- and modelling languages that achieved industrial success over the past couple of years are based on mathematical set theory and formal logic. Examples of these are Z [1], the B-method [2] from which the Atelier B development environment stems, and Event-B [3]. The advantage of using a formal specification language in software development is in reasoning formally about the properties of the resultant system at an early stage of development. Discharging proof obligations that arise from these formal specifications is a tedious and error prone activity, in fact as stated by Abrial it is "foolish (and error prone) for the simple reason that it is common to have thousands of such proofs" [3, page xviii].

Set-theoretic constructs are ubiquitous in software specifications as may be observed in a classic Z specification of a telecommunications network by Carroll Morgan [4].

TN

$$reqs, cons : \mathbb{P}\, CON$$
$$avail : \mathbb{P}\, PHONE$$

$$cons \subseteq reqs$$
$$cons \in disjoint$$
$$(\bigcup cons) \subseteq avail$$

where *PHONE* is a basic type $[PHONE]$ and a connection is defined as a set of phones, i.e. $CON == \mathbb{P}\, PHONE$. One of the numerous operations that may be defined on the state *TN*, is:

HangUp

$$\Delta TN$$

$$reqs' = reqs \setminus \{c : cons \mid ph? \in c\}$$

resulting in a proof obligation (PO) that it is possible for *HangUp* to activate a connection (enabling a user that has been waiting for a connection), i.e.

$$\exists\, HangUp \bullet (cons' \setminus cons) \neq \varnothing)$$

Reasoning about the properties of complex set-theoretic expressions for example in the above specification, e.g. power sets, distributed unions ($\bigcup$) and the like is hard. Set theory is inherently hierarchical — a set may be an element of a larger set which may in turn be an element of an even larger set and so forth. These set-theoretic structures are usually expressed in first-order logic, which in itself compounds the problem of discharging proof obligations arising from specifications using an automated reasoner.

Set-theoretic reasoners often derive large sets of consequences; these contain numerous formulae starting from the initial set of axioms. Theorem-provers may spend a lot of time exploring the consequences of relevant information, only to abandon (backtrack) the current branch of a deep search tree and pursue another branch, thereby running into the well-known combinatorial explosion problem of large search spaces [5]. Examples of such nested structures appear in the proofs that follow in this work. Examples of these challenges and attempts to prune a large space to assist an automated reasoner may be observed in [6, page 156], [7] and [8].

Of particular importance is to automate the identification and subsequent discharging of these proof obligations, i.e. employ an automated reasoner instead of an interactive one. The use of well-designed heuristics appears to hold some promise in this regard. The value of heuristics was identified as early on as 1955 during the development of the "Logic Theory

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00332

Machine" [9, page 12], [10]. In 1960 it was predicated by Hao Wang that "strategies in the search for proofs, or what are often called heuristic methods, would gradually become part of the subject matter of inferential analysis" [11, page 220]. Even nowadays the need for intelligent heuristics to appropriately prune a large search space and guide an automated reasoner is is evident in work preceding our research, e.g. [6, page 156] and [7].

The layout of the paper follows: Following the introduction our research methodology is given in Section II. A brief overview of the Rodin/Event-B reasoner used in our work is given in Section III. The proof obligations for which the reasoning environment failed to find a proof using just the default settings, together with appropriate heuristics to remedy these are presented in Section IV. Related work is presented in Section V and conclusions and pointers for further work in this area are given in Section VI.

II. Research Methodology

Our research methodology follows the research onion of Saunders et al. [12]. As per the 1st layer of the onion, our philosophy is essentially positivist, since the proofs to the proof obligations below follow the rules of first-order logic and mathematical set theory. At layer 2, the research approach is inductive, since proofs of the theorems are established using the heuristics derived. The methodological choice is quantitative, since set-theoretic constructs are derived and execution times are stated precisely. It should be noted that quantitative research is usually associated with Likert scale questionnaires involving surveys, but in out work we view mathematical set theory with associated proof theory as being quantitative. The strategy followed is both experimental (proof attempts) and case studies, being the specific proof obligations attempted, viewed as cases. The time horizon is cross-sectional since the research reported on in this paper was performed over a period of 24 months. The data collection and procedures followed as part of the techniques are conceptual, i.e. no humans other than the researchers were involved.

III. Rodin/Event-B

The B-method stems from the Z specification language and inspired the integration of the Event-B formal method and a development environment, Rodin/Event-B. Like Z, Event-B is a state-based formal method designed by Jean-Raymond Abrial [3] [13]. Its basic mathematical underpinnings are integer arithmetic, first-order logic, and a typed set theory.

While Z and to a large extent Event-B were designed to be formal specification tools (refer Section I), the Rodin software suite [14] embodies an entire development environment beyond the specification phase. It supports the construction of specifications in B, as well as the subsequent verification referred to as Event-B models. Rodin is open source and is based on an Eclipse platform [1], providing "core functionality for syntax analysis and proof-based verification of Event-B models" as per the Rodin user manual [15, page 9]. In essence, Rodin [14] allows for a seamless integration of modelling, reasoning, and ultimate refinement activities.

One of the basic constructs in Rodin is the 'context' and once a formula is included in a Rodin context, the proof-obligation component, called a proof-obligation manager (POM) generates any proof obligations (POs) to be discharged. The POM subsequently tracks the progress of all proof attempts using the available proof rules.

Instead of applying individual proof rules, Rodin tactics are used to construct and manage proofs. A tactic may either be applied automatically (via predefined lists of tactics known as tactic profiles) or interactively in the theorem-proving user interface (UI). Owing to the preference to obtain automated proofs we would attempt to minimise the amount of interaction with the reasoner.

Finally, should the reasoner fail to discharge a PO through the use of simplification and rewrite rules, one or more of Rodin's built-in reasoners are invoked. The standard Rodin installation includes the NewPP prover, but additional plug-in provers can be installed. Three popular plug-ins found to be useful by the researchers are provers from the Atelier B tool suite, namely, ML and PP, the model checker ProB, and the SMT (Satisfiability Modulo Theories) provers CVC3, CVC4, veriT and Z3.

Next we turn to the main part of this paper, namely, developing heuristics for failed proof attempts using just the Rodin default strategies. It should be noted that versions of the proofs reported on in this paper stems from research in [16]. Further details on the proof attempts may be observed in [16].

IV. Reasoning heuristics

A. *Apply the* Default Auto Tactic with SMT

Having installed the SMT solver plug in, two Rodin auto tactics may be selected. These are the *Default Auto Tactic Profile* and the *Default Auto Tactic with SMT*. Upon starting off, the default is the *Default Auto Tactic Profile*. This tactic supports our sentiment of searching for *automated* proofs, using the basic settings first. That said, our first example illustrates a shortcoming in the basic settings, and illustrates the advantage in applying the *Default Auto Tactic with SMT*.

Consider the set *ControlValues* and the function *HasControlValues* defined as:

$$ControlValues \subseteq \mathbb{Z} \tag{1}$$

$$HasControlValues \in \mathbb{P}(\mathbb{P}(\mathbb{Z})) \rightarrow \text{BOOL} \tag{2}$$

$$\forall X \bullet X \subseteq \mathbb{P}(\mathbb{Z}) \Rightarrow (HasControlValues(X) = \text{TRUE} \Leftrightarrow (\exists x \bullet x \in X \wedge ControlValues \subseteq x)) \tag{3}$$

HasControlValues is a Boolean-valued function (defined on a family of sets of integers), evaluating to TRUE iff a member set of the argument contains *ControlValues*.

[1] http://www.eclipse.org/

If we, for example, enumerate *ControlValues* $= \{0,1,2,3,4,5\}$, then Rodin's *Default Auto Tactic Profile* fails to find a proof for the theorem

$$\begin{aligned} &\mathit{HasControlValues}(\{\{x \bullet x \in \mathbb{N} \mid x+1\}, \\ &\{x \bullet x \in \mathbb{Z} \mid 3*x+1\}, \{x \bullet 3*x+1=0 \mid x\}\}) \\ &= \text{FALSE} \end{aligned} \tag{4}$$

If we, however, resort to the *Default Auto Tactic with SMT*, then the CVC4 prover is invoked and finds a proof almost immediately.

Based on these findings we define our first heuristic[2]:

> ***Heuristic #1***: Should the default tactic fail to find a proof, then enable Rodin's *Default Auto Tactic with SMT* and adjust the *Timeout for the SMT auto-tactic* to 0.15 seconds.

Note that owing to the above heuristic, Rodin's *Default Auto Tactic with SMT* was adopted for all subsequent proofs in this paper.

B. *Apply model-checking* ProB

Consider the following power set:

$$\begin{aligned} &\mathbb{P}(\{\{0\},\{1\},\{0,2\},\{1,2\}\}) \\ &= \\ &\{\varnothing,\{\{0\}\},\{\{1\}\},\{\{0,2\}\},\{\{1,2\}\},\{\{0\},\{1\}\},\{\{0\}, \\ &\{0,2\}\},\{\{0\},\{1,2\}\},\{\{1\},\{0,2\}\},\{\{1\},\{1,2\}\}, \\ &\{\{0,2\},\{1,2\}\},\{\{0\},\{1\},\{0,2\}\},\{\{0\},\{1\},\{1,2\}\}, \\ &\{\{0\},\{0,2\},\{1,2\}\},\{\{1\},\{0,2\},\{1,2\}\},\{\{0\},\{1\}, \\ &\{0,2\},\{1,2\}\}\} \end{aligned} \tag{5}$$

An attempt at using the default auto and post tactic profiles as recommended by our 1st heuristic to discharge (5), yields no proof. Since the theorem involves set instantiation, we employ ProB, Rodin's model-checking component and a proof is found in less than 2 seconds.

ProB could also discharge corresponding proof attempts of sets containing up to 12 elements in less than 2 seconds. (Note that the power set of a 12-element set contains $2^{12} = 4,096$ elements). None of the other Rodin reasoners were able to prove such theorems. Promising results using ProB in model-checking work was likewise reported by [17].

The above results lead us to the following heuristic:

> ***Heuristic #2***: Apply model-checking through *ProB* when the proof obligation involves the enumeration of set-theoretic elements and all the other Rodin provers fail to find a proof.

C. *Apply equality*

In line with the Zermelo-Fraenkel (ZF) Axiom of Extentionality [18] in (6),

$$(\forall A)(\forall B)((\forall x)(x \in A \leftrightarrow x \in B) \rightarrow (A = B)) \tag{6}$$

Rodin caters for two subset options for rewriting an equality: Rewrite the set-theoretic equality from left-to-right, and rewrite it from right-to-left. Naturally, applying the *apply equality* rule substitutes either side of the equation with the other throughout the proof sequent. If we apply the equality rewrite rule to $\alpha = \beta$, then we have the following options:

- **Apply equality from left to right**: Substitute all occurrences of α with β (i.e. eliminate α from the sequent); and
- **Apply equality from right to left**: Substitute all occurrences of β with α (thereby eliminating β from the sequent).

Consider showing a theorem follows from an axiom as follows (A and B are constants; `axm` denotes an axiom and `thm` indicates a theorem):

$$\begin{aligned} &\texttt{axm1} \quad A \subseteq \mathbb{Z} \land B \subseteq \mathbb{Z} \\ &\texttt{thm2} \quad \mathbb{P}(A \cup B) = \mathbb{P}(A) \cup \mathbb{P}(B) \\ &\qquad\quad \Leftrightarrow\ A \subseteq B \lor B \subseteq A \end{aligned} \tag{7}$$

The reasoner fails to prove (7), neither directly, nor with interactive user assistance. Consequently, we provided Rodin with additional information:

$$\begin{aligned} &\texttt{axm1} \quad A \subseteq \mathbb{Z} \land B \subseteq \mathbb{Z} \\ &\texttt{thm2} \quad \forall X, Y \bullet X \subseteq \mathbb{Z} \land\ \mathbb{P}(X \cup Y) = \mathbb{P}(X) \cup \mathbb{P}(Y) \\ &\qquad\quad \Rightarrow\ X \cup Y \in \mathbb{P}(X) \cup \mathbb{P}(Y) \\ &\texttt{thm3} \quad \mathbb{P}(A \cup B) = \mathbb{P}(A) \cup \mathbb{P}(B) \Leftrightarrow A \subseteq B \lor B \subseteq A \end{aligned} \tag{8}$$

The additional information allowed Rodin to obtain a quick proof of (7). We should note, however, that the system could not prove (8) directly — the proof attempt terminated having generated the following sequent:

$$\mathbb{P}(X \cup Y) = \mathbb{P}(X) \cup \mathbb{P}(Y) \quad \vdash \quad X \cup Y \in \mathbb{P}(X) \cup \mathbb{P}(Y).$$

A trained set-theorist could identify at this point that the following is needed:

$$\begin{aligned} \mathbb{P}(X \cup Y) = \mathbb{P}(X) \cup \mathbb{P}(Y) \quad &\Rightarrow \quad X \cup Y \in \mathbb{P}(X) \cup \mathbb{P}(Y) \\ &\Rightarrow \quad X \cup Y \in \mathbb{P}(X \cup Y) \end{aligned} \tag{9}$$

[2]The time-out value of 0.15 seconds was obtained empirically. If the time-out value is lower, the *Default Auto Tactic with SMT* is unable to prove significantly many more theorems than the *Default Auto Tactic Profile*. When the time-out value is too high, the time required to build a workspace becomes prohibitive.

By interacting with the reasoner we substitute $\mathbb{P}(X) \cup \mathbb{P}(Y)$ with $\mathbb{P}(X \cup Y)$, achieved in Rodin via applying *Equality from right to left*. Having applied the rule, Rodin generated sequent (9), whereafter the ML prover was invoked and produced a proof.

From the above results we derive:

> ***Heuristic #3***: Apply *Equality* to formulae with the set equality operator. Check for useful substitutions or simplifications when trying both *Apply equality from left* to right and *Equality from right to left*.

D. Set equality rewrites

As indicated above, applying the *Set equality rewrites* rule divides a proof obligation containing a set equality into two subset proof obligations.

Consider showing that the distributed intersection of the power set of a non-empty set, say A is equal to the power set of the distributed intersection of A (an exercise in [18, page 33]).

Written in Event-B notation we obtain:

$$\begin{array}{ll} \texttt{axm1} & A \subseteq \mathbb{P}(\mathbb{Z}) \land A \neq \varnothing \\ \texttt{thm2} & \mathbb{P}(\cap(A)) = \cap(\{X \bullet X \in A \mid \mathbb{P}(X)\}) \end{array} \quad (10)$$

Rodin failed to find a proof for (10). Since a set-theoretic equality is involved, we can invoke the interactive set equality rewrite proof rule. Rewriting from left to right yields the PO:

$$\mathbb{P}(\cap(A)) \subseteq \cap(\{X \bullet A \mid \mathbb{P}(X)\}), \quad (11)$$

Theorem (11) was then quickly discharged by the PP prover.

However, the right to left subproof,

$$\cap(\{X \bullet X \in A \mid \mathbb{P}(X)\}) \subseteq \mathbb{P}(\cap(A)), \quad (12)$$

was harder to discharge.

To obtain a proof of (12) we had to invoke a sequence of interactions. These involved applying the *Remove Inclusion*, *Remove Membership*, and the *Quantifier Instantiation* (note *Remove Inclusion* and *Remove Membership* rewrite rules are discussed in Section IV-F, while the *Quantifier Instantiation* rule is presented in Section IV-E) rules to arrive at the following sequent:

$$\frac{\begin{array}{c} X \in A \\ x0 \in x \\ s \in A \\ (\exists X \bullet X \in A \land \mathbb{P}(X) = \mathbb{P}(s)) \quad \Rightarrow \quad x \in \mathbb{P}(s) \end{array}}{x0 \in s}$$

The PP prover subsequently discharged theorem (12), leading to our next heuristic.

> ***Heuristic #4***: For formulae involving set equality, apply the *Set equality rewrites* rule to obtain two subset proofs to discharge separately.

E. Quantifier Instantiation

It is often natural to use function symbols, generally known as *functors* to encode information in terms, for example, when describing an integer as a successor or as a sum. However, terms built up with the aid of functors are more complex, especially those with variables as arguments. This poses challenges to reasoners e.g. [7] owing to difficulties with unification. Consequently, a specifier may decide to replace one or more quantified variables in a formula with correctly typed variables, expressions or constants, in an attempt to avoid nested functors. Both universally- and existentially quantified variables can be instantiated and in the case of an existentially quantified variable appearing in the scope of a universal quantifier, such variable becomes a Skolem function.

Consider set $\{\{x \bullet x \in \mathbb{Z} \mid 3 * x\}\}$ and suppose we wish to prove the set has property `HasControlValues`, stating a family of sets must contain at least one member set, say S, such that S contains a set of control values.

`HasControlValues` is defined as:

$$\begin{array}{l} \forall X \bullet X \in \mathbb{P}(\mathbb{P}(\mathbb{Z})) \Rightarrow \\ (\mathit{HasControlValues}(X) = \mathit{TRUE} \\ \Leftrightarrow \\ (\exists x \bullet x \in X \land \mathit{ControlValues} \subseteq x)) \end{array} \quad (13)$$

In other words, for $\mathit{ControlValues} = \{x \bullet x \in \mathbb{Z} \mid 6 * x\}$, we want to show that

$$\begin{array}{rl} & \mathit{ControlValues} = \{x \bullet x \in \mathbb{Z} \mid 6 * x\} \\ \Rightarrow & \mathit{HasControlValues}(\{\{x \bullet x \in \mathbb{Z} \mid 3 * x\}\}) = \mathit{TRUE} \end{array}$$

Rodin failed to find a proof, and upon inspecting the proof attempt it was noticed that the input formulae omitted the specification of `HasControlValues`:

$$\frac{\mathit{ControlValues} = \{x \bullet \top \mid 6 * x\}}{\mathit{HasControlValues}(\{\{x \bullet \top \mid 3 * x\}\}) = \mathit{TRUE}}$$

By applying the *lasso* operator (presented in Section IV-G), the required list of input formulae was obtained:

$$\frac{\begin{array}{c} \mathit{ControlValues} = \{x \bullet \top \mid 6 * x\} \\ \mathit{HasControlValues} \in \mathbb{P}(\mathbb{P}(\mathbb{Z})) \rightarrow \mathit{BOOL} \\ \mathit{HasControlValues} \in \mathbb{P}(\mathbb{P}(\mathbb{Z})) \nrightarrow \mathit{BOOL} \\ \{\{x \bullet \top \mid 3 * x\}\} \in \mathit{dom}(\mathit{HasControlValues}) \\ \forall X \bullet \mathit{HasControlValues}(X) = \mathit{TRUE} \\ \Leftrightarrow (\exists x \bullet x \in X \land (\forall x0 \bullet 6 * x0 \in x)) \end{array}}{\mathit{HasControlValues}(\{\{x \bullet \top \mid 3 * x\}\}) = \mathit{TRUE}}$$

Rodin still failed to find a proof with the settings and heuristics presented above.

The Rodin environment, however, embeds a theorem-proving UI to cater for (e.g.) the instantiation of variable X in the final hypothesis of the sequent. Hence, instantiating $[X := \{\{x \bullet \top \mid 3 * x\}\}]$, led to the proof sequent:

$$\frac{\begin{array}{c} \mathit{ControlValues} = \{x \bullet \top \mid 6 * x\} \\ \mathit{HasControlValues} \in \mathbb{P}(\mathbb{P}(\mathbb{Z})) \rightarrow \mathit{BOOL} \\ \mathit{HasControlValues} \in \mathbb{P}(\mathbb{P}(\mathbb{Z})) \nrightarrow \mathit{BOOL} \\ \{\{x \bullet \top \mid 3 * x\}\} \in \mathit{dom}(\mathit{HasControlValues}) \\ \forall x \bullet \neg\, x = \{x \bullet \top \mid 3 * x\} \lor (\exists x0 \bullet \neg\, 6 * x0 \in x) \end{array}}{\mathit{HasControlValues}(\{\{x \bullet \top \mid 3 * x\}\}) = \mathit{TRUE}}$$

Having applied the above instantiation, both the SMT provers, CVC4 and Z3 found a proof. Therefore, we arrive at:

> ***Heuristic #5***: Attempt *quantifier instantiation* to simplify formulas. Quantifiers can often be instantiated with Skolem constant symbols and definitions already present in the sequent.

F. Remove Inclusion *and* Remove Membership

Despite our interest in automated approaches, interactive approaches may often be the only feasible way to find proofs (refer for example the discussion in Section V on Related work). Hence, in this section we consider some term-rewriting approaches for subset- or set-membership proofs.

The following rewrite rules are discussed in this section:

- **Remove inclusion:** The subset relation is replaced by its underlying first-order definition, i.e. we replace $X \subseteq Y$ with $(\forall x)(x \in X \Rightarrow x \in Y)$; and
- **Remove membership:** The definition of elementhood is used to rewrite set-theoretic membership.

The proof attempts in Section IV-D revealed that theorem (10) could be discharged once we applied the rewriting rules *Remove Inclusion* and *Remove Membership*. In this section we consider sequences of interactive rules necessary to prove (10), the offending part restated below for ease of reference.

$$\cap(\{X \bullet X \in A \mid \mathbb{P}(X)\}) \subseteq \mathbb{P}\,(\cap(A)) \tag{14}$$

Rodin was unable to discharge (14), but an application of the *Remove inclusion* rule in the UI produces the sequent:

$$\frac{\begin{array}{c} X \in A \\ x \in \cap(X \bullet X \in A \mid \mathbb{P}(X)) \end{array}}{x \in \mathbb{P}(\cap(A))}$$

The reasoner still failed to find a proof, implying the need for additional actions. A sequence of interactive rule applications was called for:

First apply *Remove Membership*

$$\frac{\begin{array}{c} X \in A \\ x \in \cap(X \bullet X \in A \mid \mathbb{P}(X)) \end{array}}{x \subseteq \cap(A)}$$

Next, *Remove Inclusion*:

$$\frac{\begin{array}{c} X \in A \\ x \in \cap(X \bullet X \in A \mid \mathbb{P}(X)) \\ x0 \in x \end{array}}{x0 \in \cap(A)}$$

Third, apply *Remove membership* again:

$$\frac{\begin{array}{c} X \in A \\ x \in \cap(X \bullet X \in A \mid \mathbb{P}(X)) \\ x0 \in x \\ s \in A \end{array}}{x0 \in s}$$

Finally, $x \in \cap(X \bullet X \in A \mid \mathbb{P}(X))$ in the input could be simplified through a further application of *Remove membership*:

$$\frac{\begin{array}{c} X \in A \\ x0 \in x \\ s \in A \\ (\forall s)(((\exists X)(X \in A \land \mathbb{P}(X) = s)) \Rightarrow x \in s) \end{array}}{x0 \in s}$$

Next, the universally quantified variable ($[s := \mathbb{P}(s)]$) can be instantiated:

$$\frac{\begin{array}{c} X \in A \\ x0 \in x \\ s \in A \\ ((\exists X)(X \in A \land \mathbb{P}(X) = \mathbb{P}(s))) \Rightarrow x \in \mathbb{P}(s) \end{array}}{x0 \in s}$$

All the Rodin provers, except for ML could discharge the final proof obligation, leading to:

> ***Heuristic #6***: When appropriate, apply the rules of *Remove Inclusion* and *Remove Membership* to simplify formulae.

G. Using Lasso

In the search for a proof, one can upload all available axioms and theorems as input, or one can load only those deemed necessary for the proof attempt. This may be preferred, since irrelevant information may lead a reasoner astray [7]. In this regard Rodin's *lasso* operation allows for adding additional, backup hypotheses usually having identifiers in common with the goal. The *Hot-list* strategy described by Wos and Pieper [19] in relation to resolution-based reasoning has a similar aim.

To illustrate the use of *lasso*, we constructed two Rodin contexts:

- A base context specifying a function f:

  CONTEXT LassoBaseContext
  CONSTANTS
  f

AXIOMS

axm1: $f \in \mathbb{P}(\mathbb{Z}) \times \mathbb{P}(\mathbb{Z}) \rightarrow \{0,1\}$

axm2:

$\forall X, Y \bullet \{X, Y\} \subseteq \mathbb{P}(\mathbb{Z}) \wedge X \cap Y \neq \varnothing \Leftrightarrow f(X \mapsto Y) = 1$

END

- A lemma *thm5* and a theorem about f as a derived context:

CONTEXT LassoDerivedContext
EXTENDS BaseContext
AXIOMS

thm5: ⟨theorem⟩ $\{2,4,6,8,10\} \cap \{x \bullet x \in \mathbb{Z} \mid x * x\} \neq \varnothing$

Lemma

thm6: ⟨theorem⟩ $f(\{2,4,6,8,10\} \mapsto \{x \bullet x \in \mathbb{Z} \mid x * x\}) = 1$

END

Rodin's POM generated the following PO for thm6:

$$\frac{\neg\ \{2,4,6,8,10\} \cap \{x \bullet \top \mid x * x\} = \varnothing}{f(\{2,4,6,8,10\} \mapsto \{x \bullet \top \mid x * x\}) = 1}$$

Since the definition of f is absent from the list of hypotheses, Rodin failed to find a proof. Hence we augmented the list of hypotheses through the lasso operator, allowing the reasoner to find a proof:

$$\frac{\begin{array}{c}\neg\ \{2,4,6,8,10\} \cap \{x \bullet \top \mid x * x\} = \varnothing \\ f \in \mathbb{P}(\mathbb{Z}) \times \mathbb{P}(\mathbb{Z}) \rightarrow \{0,1\} \\ f \in \mathbb{P}(\mathbb{Z}) \times \mathbb{P}(\mathbb{Z}) \nrightarrow \mathbb{Z} \\ \{2,4,6,8,10\} \mapsto \{x \bullet \top \mid x * x\} \in dom(f) \\ \forall X, Y \bullet X \cap Y = \varnothing \ \Rightarrow\ f(X \mapsto Y) = 0 \\ \forall X, Y \bullet \neg\ X \cap Y = \varnothing \ \Rightarrow\ f(X \mapsto Y) = 1\end{array}}{f(\{2,4,6,8,10\} \mapsto \{x \bullet \top \mid x * x\}) = 1}$$

The universal quantifier in the last hypothesis was replaced, i.e. it was instantiated

$$\forall X, Y \bullet \neg\ X \cap Y = \varnothing \ \Rightarrow\ f(X \mapsto Y) = 1,$$

with $[X := \{2,4,6,8,10\},\ Y := \{x \bullet \top \mid x * x\}]$ to produce a quick proof, suggesting:

> ***Heuristic #7***: Apply the *lasso* operator when all other heuristics fail.

H. Constants

In this secion we present two logically equivalent Rodin contexts — context SetTheory_Simple and SetTheory. Context SetTheory_Simple contains no constants:

CONTEXT SetTheory_Simple
AXIOMS

thm1: ⟨theorem⟩ $inter(\{\{1,2,3\},\{2,3,4\}\}) = \{2,3\}$

END

Neither the default auto tactic nor the default auto tactic with SMT profiles managed to discharge the proof obligation generated for thm1. But, since the PO involves set enumerations, the ProB model checker found a proof in interactive mode.

Our next context SetTheory defines two constants: $A = \{1,2,3\}$ and $B = \{2,3,4\}$:

CONTEXT SetTheory
CONSTANTS

A

B

AXIOMS

axm1: $A = \{1,2,3\} \wedge B = \{2,3,4\}$

thm2: ⟨theorem⟩ $inter(\{A,B\}) = \{2,3\}$

END

With the above definitions, the PO for thm2 is discharged using the default auto tactic profiles. With the exception of ML, all the provers, namely, PP, CV3, CV4, veriT, and Z3 managed to find a proof, leading to our final heuristic:

> ***Heuristic #8***: Use *constants* for set-theoretic objects to assist the proof attempt and reduce duplication of definitions.

V. Related work

The seminal work by C.A.R. Hoare on axiomatic principles for modern programming [20] and his retrospective analyses some 40 years later [21] deserve special mention. The landmark papers by Alan Robinson [22] [23] in many ways enabled the construction of resolution-based automated reasoners of which the OTTER reasoner [24] is a well-known example, later succeeded by Prover-9 [25]. The use of the Vampire automated reasoner in discharging POs arising from set-theoretic specifications, similar to the work in this paper appears in [26].

Work on establishing reliable tool support by combining the reasoning capabilities of Rodin/Event-B with the strong theoretical foundations of Isabelle/HOL was done by Matthias Schmalz [27]. Research on the automation of Rodin/Event-B proofs using a technique called rippling whereby proofs are automated using meta-level guidance, following by proof correction for an otherwise interactive environment like Rodin/Event-B is presented by Lin et al. [28].

Further afield, significant work on the interactive Pathfinder model checker mainly under the auspices of the NASA Ames Research Centre was done by Lindstrom, Mehlitz and Visser [29], amongst others.

VI. Conclusions and future work

In this paper we presented eight heuristics to aid with discharging set-theoretic POs using the Rodin/Event-B environment. Each heuristic was illustrated with an example for which Rodin failed to find a proof when the *Default Auto Tactic Profile* was in effect. Our 1st heuristic illustrated that the powerful SMT provers have the potential to facilitate the finding of automated proofs. However, we constructed

additional examples that failed to be discharged using the *Default Auto Tactic Profile* or *Default Auto Tactic with SMT* profiles. These proof obligations could only be discharged during Rodin's interactive mode, or by restructuring the context (e.g., introducing constants), thereby adhering to our goal of finding automated proofs.

Future work in this area may include an investigation into identifying heuristics that target the individual inference mechanisms of the different Rodin reasoners, owing to a specifier's preference for a specific inference mechanism. The current suite of provers encompass a wide variety of inference mechanisms, such as resolution (NewPP and PP), term rewriting (ML), model checking (ProB) and Satisfiability Modulo Theories via CVC3, CVC4, veriT and Z3. The Rodin environment holds much promise as a competitive environment for reasoning about the correctness of industrial-sized software specifications. Intelligent 4IR interfaces to assist with the application of heuristics should be developed.

REFERENCES

[1] D. Lightfoot, *Formal specification using Z*. Macmillan Press, 1991.

[2] J.-R. Abrial, *The B Book: Assigning Programs to Meanings*. Cambridge, England: Cambridge University Press, 1996.

[3] ——, *Modeling in Event-B: System and software engineering*. Cambridge University Press, 2010.

[4] C. C. Morgan, "Specification of a Communication System," in *Distributed Computing Systems: Synchronisation, Control, and Communication*, Y. Paker and J.-P. Verjus, Eds. Academic Press, 1983, pp. 93 – 108.

[5] E. L. Lusk, "Controlling redundancy in large search spaces: Argonne-style theorem proving through the years," in *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. Springer, 1992, pp. 96–106.

[6] A. Bundy, "A survey of automated deduction," in *Artificial intelligence today*. Springer, 1999, pp. 153–174.

[7] P. S. Steyn, "Validating reasoning heuristics using next generation theorem provers," MSc dissertation, University of South Africa, 2009.

[8] J. A. van der Poll, "Formal methods in software development: A road less travelled," *South African Computer Journal*, no. 45, pp. 40–52, Jul. 2010.

[9] A. Newell and H. Simon, "The logic theory machine–a complex information processing system," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 61–79, 1956.

[10] D. Mackenzie, "The automation of proof: A historical and sociological exploration," *IEEE Annals of the History of Computing*, vol. 17, no. 3, pp. 7–29, 1995.

[11] H. Wang, "Proving theorems by pattern recognition I," *Communications of the ACM*, vol. 3, no. 4, pp. 220–234, 1960.

[12] M. Saunders, P. Lewis, and A. Thornhill, *Research methods for business students*, 8th ed. Harlow: Pearson Education, 2019.

[13] J.-R. Abrial, "The Event-B modelling notation," *wiki.event-b.org*, 2007.

[14] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "Rodin: An open toolset for modelling and reasoning in Event-B," *STTT*, vol. 12, no. 6, pp. 447–466, 2010.

[15] *Rodin User's Handbook*, Deploy Project, 2012.

[16] J. G. Ackermann, "Evaluating Reasoning Heuristics for a Hybrid Theorem-Proving Platform," MSc dissertation, School of Computing, College of Science, Engineering and Technology, University of South Africa, 2018.

[17] T. Hörne and J. A. van der Poll, "Planning as model checking: The performance of proB vs NuSMV," in *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. ACM, 2008, pp. 114–123.

[18] H. B. Enderton, *Elements of set theory*. Academic Press, 1977.

[19] L. Wos and G. W. Pieper, "The Hot List Strategy," *Journal of Automated Reasoning*, no. 22, pp. 1 – 44, 1999. [Online]. Available: https://doi.org/10.1023/A:1005909914693

[20] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[21] ——, "Retrospective: An Axiomatic Basis for Computer Programming," *Communications of the ACM*, vol. 52, no. 10, pp. 30 – 32, October 2009.

[22] J. A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," *Journal of the Association for Computing Machinery*, vol. 12, no. 1, pp. 23 – 41, January 1965.

[23] ——, "Automatic Deduction with Hyper-Resolution," *International Journal of Computer Mathematics*, vol. 1, no. 3, pp. 227 – 234, 1965.

[24] W. W. McCune, *OTTER 3.0 Reference Manual and Guide*, Argonne National Laboratory, Argonne, Illinois, August 1995, aNL-94/6.

[25] ——, *Prover9 and Mace4, Current version: 2009-11A*. [Online]. Available: https://www.cs.unm.edu/ mccune/prover9/manual/2009-02A/

[26] P. Steyn and J. A. van der Poll, "Validating reasoning heuristics using next-generation theorem-provers," in *Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS-2007), In conjunction with ICEIS 2007, Funchal, Madeira, Portugal*, June 2007, pp. 43–52.

[27] M. Schmalz, "Formalizing the logic of Event-B," Doctor of Sciences, ETH ZURICH, 2012.

[28] Y. Lin, A. Bundy, G. Grov, and E. Maclean, "Automating Event-B invariant proofs by rippling and proof patching," *Formal Aspects of Computing*, vol. 31, no. 1, pp. 95 – 129, 2019.

[29] G. Lindstrom, P. Mehlitz, and W. Visser, "Model checking real time Java using Java PathFinder," 10 2005, pp. 444–456.