Rewriting and Well-Definedness within a Proof System

Issam Maamria
University of Southampton
ECS, University of Southampton
Southampton SO17 1BJ, UK
im06r@ecs.soton.ac.uk
and Michael Butler
University of Southampton
ECS, University of Southampton
Southampton SO17 1BJ, UK
mjb@ecs.soton.ac.uk

Abstract

Term rewriting has a significant presence in various areas, not least in automated theorem proving where it is used as a proof technique. Many theorem provers employ specialised proof tactics for rewriting. This results in an interleaving between deduction and computation (i.e., rewriting) steps. If the logic of reasoning supports partial functions, it is necessary that rewriting copes with potentially ill-defined terms. In this paper, we provide a basis for integrating rewriting with a deductive proof system that deals with well-definedness. The definitions and theorems presented in this paper are the theoretical foundations for an extensible rewriting-based prover that has been implemented for the set theoretical formalism Event-B.

1 Introduction

Term rewriting has an important presence in many areas including abstract data type specifications and automated reasoning. In this regard, many automated theorem provers employ rewriting as a proof technique where it may interleave with deduction. PVS [15] and Isabelle/HOL [14] are higher-order theorem provers that include specialised tactics for rewriting.

The interleaving between rewriting steps and deduction steps poses several difficulties. The termination of rewriting becomes an issue of paramount importance. Many techniques, such as term orderings [5], have been explored to provide good practical solutions to termination problems. We argue that, in the presence of potentially ill-defined terms, rewriting has to be further constrained.

Ill-defined terms arise in the presence of partial functions. They result from the application of functions to terms outside their domain. If ill-definedness is a concern, the adopted reasoning framework has to cope with it. Different approaches exist to reason in the presence of partial functions. Each of these approaches has its own specialised proof calculus. In [12], it is shown that it is possible to reason about partiality without abandoning the well-understood domain of two-valued predicate logic. In that approach, the reasoning is achieved by extending the standard calculus with derived proof rules that preserve well-definedness across proofs. We argue that, in order to integrate rewriting as a proof step in such a calculus, it is necessary that rewriting preserves well-definedness.

In this paper, we present a treatment of term rewriting where term well-definedness is an issue. Our treatment unifies the notions of well-definedness (WD) and rewriting, and provides a basis to integrate rewriting as a proof step within the proof system presented in [12]. Central to our contribution is the concept of WD-preserving rewriting where rewrite rules preserve well-definedness in the same direction in which they are applied. We establish the necessary conditions under which rewriting preserves well-definedness. We, finally, show how a rewrite step can be interleaved with deduction steps in a valid fashion.

1.1 Practical Setting

Event-B [3] is a formalism for discrete systems modelling based on Action Systems [6]. It can be used to model and reason about complex systems such as concurrent and reactive systems. The semantics of a model developed in Event-B is given by means of its proof obligations. These obligations have to be discharged to show consistency of the model with respect to some behavioural semantics.

Modelling in Event-B is conducted by defining contexts and machines. Contexts describe static properties of a model by specifying carrier sets and constants. Machines, as their name suggests, define the dynamics of a model by means of variables (state) and events (transitions). Variables are constrained by invariants. A machine can be refined by another machine, and can see (import) contexts. Proof obligations arise to verify the consistency of a model. For instance, there are proof obligations to establish the refinement relationship between two machines, and to establish invariant preservation by the events (transitions). The logic used in Event-B is typed set theory built on first-order predicate logic, and allows the definition of partial functions. As such, it is necessary that the used proof system handles ill-definedness. Indeed, the proof calculus outlined in [12] is the one used to reason in Event-B. Figure 1 illustrates a simple Event-B model for a door entry system.

The Rodin platform [1] is an open extensible tool for Event-B based on Eclipse[1]. It offers support for specification and proof, and it can be easily extended with other useful tools e.g., there is a plug-in for model checking called Pro-B [10].

1.2 Motivation

The Rodin platform has a proving infrastructure which is extensible with new proof rules. External provers can also be used; Atelier-B [2] provers ML and PP have been incorporated into Rodin. Adding new proof rules requires the use of the Java programming language, knowledge of Eclipse as well as an understanding of the internal architecture of Rodin. A complication of such approach is that newly implemented rules could compromise the soundness of the prover. This work has been carried out as part of an effort to address this limitation of Rodin from the viewpoint of prover extensibility. This paper discusses some theoretical results in the context of rewriting and well-definedness. The ideas presented in this paper have resulted in providing proof support for the set theoretical formalism Event-B [3]. An extensible rewriting-based prover [11] has been implemented and integrated into Rodin.

Outline. In Section 2, we recall some of the preliminary concepts of term rewriting systems. Section 3 describes the necessary conditions under which rewriting preserves well-definedness. Section 4 shows how a WD-preserving rewrite rule can be used in proofs. The application of the previous ideas to Event-B [3] is shown in Section 5. We conclude in Section 6 by stating what we have achieved and its impact on the Event-B toolset [7].

1.3 Related Work

The interleaving between deduction and rewriting steps has gathered much interest given its importance to automated reasoning. In this work, we identify the necessary conditions under which rewriting can interleave with deduction in the proof calculus defined in [12]. In other works, this interleaving is studied from different perspectives.

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

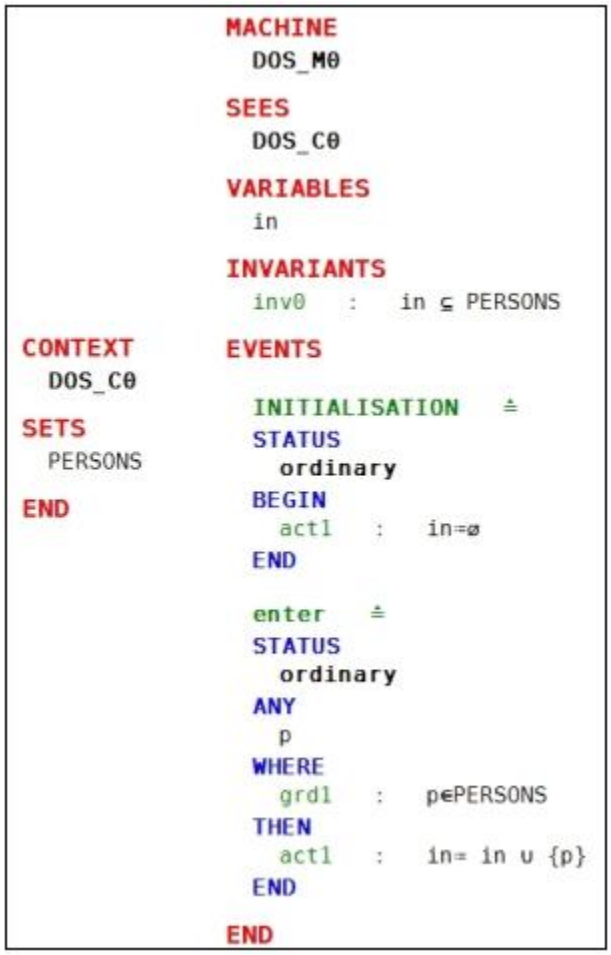

Figure 1: A Simple Model for A Door Entry System

Theorem proving modulo is an approach that removes computational steps from proofs by reasoning modulo a congruence on propositions [9]. The advantage of this technique is that it separates computation steps (i.e., rewriting) from deduction steps in a clean way. In [9], a proof-theoretic account of the combination between computations and deductions is presented in the shape of a sequent calculus modulo. The congruence on propositions, on the other hand, is defined by rewrite rules and equational axioms.

The combination of rewriting and deduction makes properties of rewrite systems of practical interest. Termination and confluence properties of term rewriting systems are important, and have been studied extensively [5, 8]. When rewriting is interleaved with deduction, it is critical that computation steps terminate. Term orderings, in which any term that is syntactically simpler than another is smaller than the other, provides a practical technique to assess the termination of rewrite systems.

In our work, we aim to unify the notions of well-definedness and rewrite systems. Our objective is to characterise the interaction between deduction and rewriting when well-definedness is taken into consideration. This is achieved by identifying the necessary conditions under which

computations can interleave with the deduction steps (i.e., proof rules) in [12].

2 Preliminaries

In this section, we lay the groundwork for the rest of the paper. We briefly introduce the proof calculus defined in [12]. We also shed some light on basic concepts of term rewriting systems. For the rest of this paper, we use the language signature Σ defined by a set V of variable symbols, a set F of function symbols and a set P of predicate symbols. In the next two definitions, we introduce the syntax of the first-order predicate calculus with equality that will be used in the subsequent sections.

Definition 2.1 (Term)**.** *T_Σ, the set of Σ-terms is inductively defined by:*
- *each variable of V is a term;*
- *if $f \in F$, $arity(f) = n$ and each of $e_1, ..., e_n$ is a term, then $f(e_1, ..., e_n)$ is a term.*

Definition 2.2 (Formula)**.** *F_Σ, the set of Σ-formulas is inductively defined by:*
- *$\bot$ is a formula;*
- *$p(t_1, ..., t_n)$ is a formula provided $p \in P$, $arity(p) = n$ and each of $t_1, ..., t_n$ is a term;*
- *$t_1 = t_2$ is a formula provided t_1 and t_2 are terms;*
- *$\varphi \wedge \psi$ is a formula if φ and ψ are formulas;*
- *$\neg\varphi$ is a formula if φ is a formula;*
- *$\forall x.\varphi$ is a formula if $x \in V$ and φ is a formula.*

Note that other logical operators (e.g., $\exists$) can be defined (as in [13]) by means of the operators in the previous definition. For the rest of the paper, we assume a syntactic operator $\mathcal{V}ar : (F_\Sigma \cup T_\Sigma) \to \mathbb{P}(V)$ such that $\mathcal{V}ar(t)$ is the set of variables occurring free in t.

2.1 The Well-Definedness Operator

The well-definedness operator '$\mathcal{D}$' formally encodes what is meant by well-definedness. $\mathcal{D} : (F_\Sigma \cup T_\Sigma) \to F_\Sigma$ is a syntactic operator that maps terms and formulae to their well-definedness predicates. We interpret the formula $\mathcal{D}(F)$ as being valid if and only if F is well-defined. For a detailed treatment of the $\mathcal{D}$ operator, we refer to [4].

The well-definedness (WD) of terms is defined recursively as follows:

$$\mathcal{D}(x) \quad \widehat{=} \quad \top \quad if\ x \in V \tag{2.1}$$

$$\mathcal{D}(f(t_1, ..., t_n)) \quad \widehat{=} \quad \bigwedge_{i=1}^{n} \mathcal{D}(t_i) \ \wedge \ C^f_{t_1,...,t_n} \ . \tag{2.2}$$

where $C^f_{t_1,...,t_n}$ effectively defines the domain of the function f. For this study, we assume that predicate symbols are total. As a result, ill-definedness can only be introduced by terms. Therefore, we have the following:

$$\mathcal{D}(p(t_1, ..., t_n)) \quad \widehat{=} \quad \bigwedge_{i=1}^{n} \mathcal{D}(t_i) \quad if\ p \in P \tag{2.3}$$

$$\mathcal{D}(t_1 = t_2) \quad \widehat{=} \quad \mathcal{D}(t_1) \wedge \mathcal{D}(t_2) \ . \tag{2.4}$$

For the well-definedness of other formulae, we use the following expansions [4]:

$$\mathcal{D}(\bot) \;\widehat{=}\; \top \tag{2.5}$$
$$\mathcal{D}(\neg\varphi) \;\widehat{=}\; \mathcal{D}(\varphi) \tag{2.6}$$
$$\mathcal{D}(\varphi \wedge \psi) \;\widehat{=}\; (\mathcal{D}(\varphi) \wedge \mathcal{D}(\psi)) \vee (\mathcal{D}(\varphi) \wedge \neg\varphi) \vee (\mathcal{D}(\psi) \wedge \neg\psi) \tag{2.7}$$
$$\mathcal{D}(\forall x \cdot \varphi) \;\widehat{=}\; (\forall x \cdot \mathcal{D}(\varphi)) \vee (\exists x \cdot \mathcal{D}(\varphi) \wedge \neg\varphi) \tag{2.8}$$

The well-definedness of formulae built using derived logical operators can be straightforwardly derived, see [13]. An important property of well-definedness conditions is that they are themselves well-defined [12]; i.e.,

$$\mathcal{D}(\mathcal{D}(P)) \;\Leftrightarrow\; \top \;.$$

2.2 The WD-preserving Sequent Calculus

We assume the signature Σ is equipped with a proof theory in the shape of a WD-preserving first-order sequent calculus similar to the one appearing in [12]. A judgement in the aforementioned calculus is called a well-defined sequent, and is of the form $H \vdash_{\mathcal{D}} G$ defined as follows:

$$H \vdash_{\mathcal{D}} G \;\;\widehat{=}\;\; \mathcal{D}(H), \mathcal{D}(G), H \vdash G \;.$$

That is, the well-definedness of H and G is assumed when proving $H \vdash G$. Generally speaking, when proving a sequent $H \vdash G$, the approach suggests proving its validity as well as its well-definedness:

$$\boxed{\text{WD}_{\mathcal{D}} : \quad \vdash_{\mathcal{D}} \mathcal{D}(H \vdash G)} \qquad \boxed{\text{Validity}_{\mathcal{D}} : \quad H \vdash_{\mathcal{D}} G}$$

where $\mathcal{D}(H \vdash G)$ is defined as $\mathcal{D}(\forall \overrightarrow{x} \cdot H \Rightarrow G)$ such that $\overrightarrow{x}$ are the free variables of H and G.

A proof rule is said to preserve well-definedness (WD) iff its consequent and antecedents only contain well-defined sequents (i.e., $\vdash_{\mathcal{D}}$ sequents). Figure 2 introduces the theory $\boldsymbol{FoPCe}_{\mathcal{D}}$ (a collection of WD-preserving inference rules) as developed in [12]. Note that we use $x \backslash H$ to denote the non-freeness condition of x in H. We also use $[x := E]P$ to denote the syntactic replacement of all free occurrences of the variable x in P by the term E. The boxed sequents in Figure 2 correspond to the additional sequents that has to be discharged compared to the classical version of the rule in order to preserve well-definedness.

Proof rules for derived logical operator (i.e., $\Rightarrow$, $\vee$, $\Leftrightarrow$ and $\exists$) can be derived directly from the rules of $\boldsymbol{FoPCe}_{\mathcal{D}}$. The following two proof rules can be derived with a detour through $\vdash$ sequents (classical reasoning):

$$\frac{P, \mathcal{D}(R) \;\vdash_{\mathcal{D}}\; R}{P \;\vdash_{\mathcal{D}}\; R}\; goal_{WD}$$

and

$$\frac{P, \mathcal{D}(P) \;\vdash_{\mathcal{D}}\; R}{P \;\vdash_{\mathcal{D}}\; R}\; hyp_{WD} \;.$$

In Section 3 and 4, we show how rewriting can be interleaved with the inference rules of $\boldsymbol{FoPCe}_{\mathcal{D}}$. For the rest of the paper, we assume that the reader is familiar with the basic notions of rewriting as found, for instance, in [5]. We define the domain and range of a substitution σ (both finite), denoted $\mathcal{D}om(\sigma)$ and $\mathcal{R}an(\sigma)$ respectively, as follows:

$$\mathcal{D}om(\sigma) \;=\; \{x \in V \mid \sigma(x) \neq x\} \;,$$
$$\mathcal{R}an(\sigma) \;=\; \{t \in T_{\Sigma} \mid \exists x \cdot x \in \mathcal{D}om(\sigma) \wedge t = \sigma(x)\} \;.$$

$$\frac{}{H, P \vdash_{\mathcal{D}} P}\ hyp_{\mathcal{D}} \qquad \frac{H \vdash_{\mathcal{D}} Q}{H, P \vdash_{\mathcal{D}} Q}\ mon_{\mathcal{D}} \qquad \frac{H, \neg Q \vdash_{\mathcal{D}} \bot}{H \vdash_{\mathcal{D}} Q}\ contr_{\mathcal{D}}$$

$$\frac{}{H, \bot \vdash_{\mathcal{D}} P}\ \bot hyp_{\mathcal{D}} \qquad \frac{H, P \vdash_{\mathcal{D}} \bot}{H \vdash_{\mathcal{D}} \neg P}\ \neg goal_{\mathcal{D}} \qquad \frac{H \vdash_{\mathcal{D}} P}{H, \neg P \vdash_{\mathcal{D}} Q}\ \neg hyp_{\mathcal{D}}$$

$$\frac{H \vdash_{\mathcal{D}} P \quad H \vdash_{\mathcal{D}} Q}{H \vdash_{\mathcal{D}} P \wedge Q}\ \wedge goal_{\mathcal{D}} \qquad \frac{H, P, Q \vdash_{\mathcal{D}} R}{H, P \wedge Q \vdash_{\mathcal{D}} R}\ \wedge hyp_{\mathcal{D}} \qquad \frac{H \vdash_{\mathcal{D}} P}{H \vdash_{\mathcal{D}} \forall x \cdot P}\ \forall goal_{\mathcal{D}}\ (x \backslash H)$$

$$\frac{}{H \vdash_{\mathcal{D}} E = E}\ = goal_{\mathcal{D}} \qquad \frac{H \vdash_{\mathcal{D}} [x := E]P}{H, E = F \vdash_{\mathcal{D}} [x := F]P}\ = hyp_{\mathcal{D}}$$

$$\frac{\boxed{H \vdash_{\mathcal{D}} \mathcal{D}(P)} \qquad H \vdash_{\mathcal{D}} P \qquad H, P \vdash_{\mathcal{D}} Q}{H \vdash_{\mathcal{D}} Q}\ cut_{\mathcal{D}}$$

$$\frac{\boxed{H \vdash_{\mathcal{D}} \mathcal{D}(E)} \qquad H, [x := E]P \vdash_{\mathcal{D}} Q}{H, \forall x \cdot P \vdash_{\mathcal{D}} Q}\ \forall hyp_{\mathcal{D}}$$

Figure 2: Inference Rules of $\boldsymbol{FoPCe}_{\mathcal{D}}$

Note that the application of a substitution σ to a term l *simultaneously* replaces occurrences of variables by their respective σ-images. For the rest of this work, we restrict substitutions according to the following definition:

Definition 2.3 (Non-conflicting Substitution)**.** *A substitution σ is said to be non-conflicting iff*

$$[\bigcup_{t \in \mathcal{R}an(\sigma)} \mathcal{V}ar(t)] \cap \mathcal{D}om(\sigma) \ = \ \varnothing \ .$$

Intuitively, a non-conflicting substitution can be simulated by a syntactic replacement as follows:

$$\sigma(l) \ \widehat{=} \ [x_1 := \sigma(x_1)]...[x_n := \sigma(x_n)]l \ .$$

such that $x_1, ..., x_n$ are the free variables in l, and $x_i \backslash \sigma(x_j)$ for all i and j where $1 \leq i \leq n$ and $1 \leq j \leq n$. In this case, we have the following important property:

$$\mathcal{D}(\sigma(l)) \ \Leftrightarrow \bigwedge_{e \in \mathcal{R}an(\sigma)} \mathcal{D}(e) \ \wedge \ \sigma(\mathcal{D}(l)) \ ,$$

which can proved by induction on the structure of terms.

One of the main concepts of term rewriting is that of positions in terms and formulae where ϵ denotes the root position. Positions within a formula (or a term) describe paths to its subterms and subformulae. When p is a position in a formula F, we write $F|_p$ for the term or formula at position p in formula F. We write $F[s]_p$ for the formula that results from replacing $F|_p$ with s in F.

3 WD-Preserving Rewriting

In this section, we show how rewriting *preserves* equality of terms, validity of formulae and well-definedness of both terms and formulae. The next definitions describe what is meant by a conditional rewrite rule.

Definition 3.1 (Conditional Identity)**.** *A Σ-conditional identity (or simply conditional identity) is a triplet $(l, c, r) \in T_\Sigma \times F_\Sigma \times T_\Sigma$. In this case, l is called the left hand side, r the right hand side, and c the condition of the identity.*

Definition 3.2 (Valid Conditional Identity)**.** *A conditional identity (l, c, r) is valid iff the following sequent is provable*

$$c \vdash_{\mathcal{D}} l = r \ .$$

A conditional identity can be turned into a rewrite rule if it satisfies the syntactic restrictions presented in the following definition:

Definition 3.3 (Conditional Term Rewrite Rule)**.** *A conditional term rewrite rule is a conditional identity (l, c, r) such that:*

1. *l is not a variable,*
2. *$\mathcal{V}ar(c) \subseteq \mathcal{V}ar(l)$,*
3. *$\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$.*

In this case, we use the notation $l \xrightarrow{c} r$ instead of (l, c, r).

In the derivations of Figure 4 and Figure 3, we single out the necessary conditions under which rewriting can be performed. Figure 3 concerns the rewriting of an hypothesis that has an occurrence of a rewrite rule left hand side l. Note the presence of the condition $\sigma(c)$. We assume that the free variables of $\sigma(c)$ also occur free in $\varphi[\sigma(l)]_p$; this ensures that σ denotes the same substitution in both $\sigma(c)$ and $\varphi[\sigma(l)]_p$. We use '$hyp_{WD}; mon_{\mathcal{D}}$' to denote that proof rule $mon_{\mathcal{D}}$ is applied after applying the rule hyp_{WD}. Figure 4, on the other hand, concerns the rewriting of a goal which has an occurrence of a rewrite rule left hand side l.

The boxed sequents correspond to the conditions under which a formula (an hypothesis or the goal) can be rewritten. In summary, a conditional term rewrite rule $l \xrightarrow{c} r$ can be applied to a formula $\varphi[\sigma(l)]_p$ (the goal or one of the hypothesises) iff the following sequents are provable:

$$\sigma(c), \mathcal{D}(\varphi[\sigma(l)]_p) \vdash_{\mathcal{D}} \mathcal{D}(\varphi[\sigma(r)]_p) \tag{3.1}$$

$$\sigma(c) \vdash_{\mathcal{D}} \varphi[\sigma(l)]_p \Leftrightarrow \varphi[\sigma(r)]_p \ . \tag{3.2}$$

In the rest of this section, we examine the sufficient restrictions on conditional term rewrite rules to ensure that sequents 3.2 and 3.1 are provable for a given formula φ, a position p and a substitution σ.

Definition 3.4. *A conditional rewrite rule $l \xrightarrow{c} r$ is said to be WD-preserving iff the following sequent is provable:*

$$\mathcal{D}(l), c \vdash_{\mathcal{D}} \mathcal{D}(r) \ .$$

$$\dfrac{\dfrac{\boxed{\sigma(c), \mathcal{D}(\varphi[\sigma(l)]_p) \ \vdash_{\mathcal{D}} \ \mathcal{D}(\varphi[\sigma(r)]_p)}}{\sigma(c), \varphi[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ \mathcal{D}(\varphi[\sigma(r)]_p)} \ hyp_{WD}; mon_{\mathcal{D}} \quad \boxed{\sigma(c), \varphi[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ \varphi[\sigma(r)]_p} \quad \dfrac{\sigma(c), \varphi[\sigma(r)]_p \ \vdash_{\mathcal{D}} \ R}{\sigma(c), \varphi[\sigma(l)]_p, \varphi[\sigma(r)]_p \ \vdash_{\mathcal{D}} \ R} \ mon_{\mathcal{D}}}{\sigma(c), \varphi[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ R} \ cut_{\mathcal{D}}$$

Figure 3: Hypothesis Rewriting

$$\dfrac{\boxed{\sigma(c), \mathcal{D}(\varphi[\sigma(l)]_p) \ \vdash_{\mathcal{D}} \ \mathcal{D}(\varphi[\sigma(r)]_p)} \quad \dfrac{\sigma(c) \ \vdash_{\mathcal{D}} \ \varphi[\sigma(r)]_p}{\sigma(c), \mathcal{D}(\varphi[\sigma(l)]_p) \ \vdash_{\mathcal{D}} \ \varphi[\sigma(r)]_p} \ mon_{\mathcal{D}} \quad \boxed{\sigma(c), \varphi[\sigma(r)]_p \ \vdash_{\mathcal{D}} \ \varphi[\sigma(l)]_p}}{\dfrac{\sigma(c), \mathcal{D}(\varphi[\sigma(l)]_p) \ \vdash_{\mathcal{D}} \ \varphi[\sigma(l)]_p}{\sigma(c) \ \vdash_{\mathcal{D}} \ \varphi[\sigma(l)]_p} \ goal_{WD}} \ cut_{\mathcal{D}}$$

Figure 4: Goal Rewriting

We turn our attention to rewrite rule application. Consider applying rule $l \xrightarrow{c} r$ to formulae $P[s]_p$ where s is a term as is $P|_p$. The left hand side l is matched against s by finding a substitution σ such that $\sigma(l) = s$ (one-way matching). Provided $\sigma(c)$ holds, $P[s]_p$ can be rewritten to $P[\sigma(r)]_p$.

The following theorem states that the application of a valid and well-definedness preserving conditional term rewrite rule preserves equality (3.3) and well-definedness (3.4) of terms.

Theorem 3.5. *Let $l \xrightarrow{c} r$ be a conditional term rewrite rule, t be a term, p be a position within t, and σ be a non-conflicting substitution such that*

$$\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(l) \ .$$

If $l \xrightarrow{c} r$ is valid and WD-preserving, then the following two sequents are provable:

$$\sigma(c) \quad \vdash_{\mathcal{D}} \quad t[\sigma(l)]_p = t[\sigma(r)]_p \ , \tag{3.3}$$

$$\mathcal{D}(t[\sigma(l)]_p), \sigma(c) \quad \vdash_{\mathcal{D}} \quad \mathcal{D}(t[\sigma(r)]_p) \ . \tag{3.4}$$

Proof. The following lemma is needed to prove Theorem 3.5:

Lemma 3.6. *Let $l \xrightarrow{c} r$ be a conditional term rewrite rule, and σ be a non-conflicting substitution such that*

$$\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(l) \ .$$

1. *If $l \xrightarrow{c} r$ is valid, then the following sequent is provable:*

$$\sigma(c) \ \vdash_{\mathcal{D}} \ \sigma(l) = \sigma(r) \ .$$

2. *If $l \xrightarrow{c} r$ is WD-preserving, then the following sequents are provable:*

$$\mathcal{D}(\sigma(l)) \wedge \sigma(c) \quad \vdash_{\mathcal{D}} \quad \mathcal{D}(\sigma(r)) \ .$$

Proof. We observe that the sequent

$$\vdash_{\mathcal{D}} \ \forall \overrightarrow{x} \cdot [(\mathcal{D}(l) \wedge \mathcal{D}(c) \wedge \mathcal{D}(r) \wedge c) \Rightarrow l = r] \tag{3.5}$$

($\overrightarrow{x}$ are the free variables of l) is provable if the sequent

$$c \ \vdash_{\mathcal{D}} \ l = r$$

is also provable. We also observe that the sequent

$$\vdash_{\mathcal{D}} \ \mathcal{D}(\forall \overrightarrow{x} \cdot [(\mathcal{D}(l) \wedge \mathcal{D}(c) \wedge \mathcal{D}(r) \wedge c) \Rightarrow l = r]) \tag{3.6}$$

is provable. Since the substitution σ can be simulated as a sequence of syntactic replacements, instantiating $\overrightarrow{x}$ in (3.5) with the appropriate terms in $\mathcal{R}an(\sigma)$ is the main idea of the proof of the first claim. The proof of the second claim follows a similar approach. □

1. *Proof of sequent* (3.3): We proceed by induction on the structure of the term t.

(a) **Base Case:** t is a variable, $t = x$. In this case (3.3) becomes

$$\sigma(c) \ \vdash_{\mathcal{D}} \ x[\sigma(l)]_\epsilon = x[\sigma(r)]_\epsilon \ ,$$

since variables have only one position (ϵ the root position). This simplifies to

$$\sigma(c) \ \vdash_{\mathcal{D}} \ \sigma(l) = \sigma(r) \ ,$$

which is a provable sequent according to Lemma 3.6.

(b) **Inductive Case:** t is a function, $t = f(t_1, ..., t_n)$. We distinguish the cases $p = \epsilon$ and $p = iq$ for $1 \leq i \leq n$ and some position q.

i. Case $p = \epsilon$: this case is similar to the base case.

ii. Case $p = iq$: we assume the following inductive hypothesis (in this case a provable sequent)

$$\sigma(c) \ \vdash_{\mathcal{D}} \ t_i[\sigma(l)]_q = t_i[\sigma(r)]_q \ ,$$

and we show that

$$\sigma(c) \ \vdash_{\mathcal{D}} \ f(t_1, ..., t_i[\sigma(l)]_q, ..., t_n) = f(t_1, ..., t_i[\sigma(r)]_q, ..., t_n) \ ,$$

is a provable sequent where $iq = p$.

□

2. *Proof of sequent* (3.4): We proceed by induction on the structure of the term t.

(a) **Base Case:** t is a variable, $t = x$. In this case (3.4) becomes

$$\mathcal{D}(x[\sigma(l)]_\epsilon), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}(x[\sigma(r)]_\epsilon) \ ,$$

since variables only have the root position ϵ. This simplifies to

$$\mathcal{D}(\sigma(l)), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(r)) \ ,$$

which is a provable sequent according to Lemma 3.6.

(b) **Inductive Case:** t is a function, $t = f(t_1, ..., t_n)$. We distinguish the cases $p = \epsilon$ and $p = iq$ for $1 \leq i \leq n$ and some position q.

i. Case $p = \epsilon$: this case is similar to the base case.

ii. Case $p = iq$: We assume the following inductive hypothesis

$$\mathcal{D}(t_i[\sigma(l)]_q), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}(t_i[\sigma(r)]_q) \ ,$$

and we show that

$$\mathcal{D}(f(t_1, ..., t_i[\sigma(l)]_q, ..., t_n)), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}(f(t_1, ..., t_i[\sigma(r)]_q, ..., t_n)) \ ,$$

is a provable sequent where $iq = p$.

□

□

The following theorem asserts that Definition 3.2 and 3.4 are adequate for a conditional term rewrite rule to preserve validity and well-definedness when applied to a formula.

Theorem 3.7. *Let $l \xrightarrow{c} r$ be a conditional term rewrite rule, f be a formula, p be a position within f such that $f|_P$ is a term, and σ be a non-conflicting substitution such that*

$$\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(l) \ .$$

If $l \xrightarrow{c} r$ is valid and WD-preserving, then the following two sequents are provable:

$$\sigma(c) \quad \vdash_{\mathcal{D}} \quad f[\sigma(l)]_p \Leftrightarrow f[\sigma(r)]_p \ , \tag{3.7}$$

$$\mathcal{D}(f[\sigma(l)]_p), \sigma(c) \quad \vdash_{\mathcal{D}} \quad \mathcal{D}(f[\sigma(r)]_p) \ . \tag{3.8}$$

Proof.

1. *Proof of sequent* (3.7): We proceed by induction on the structure of the formula f. We show a sketch of the proof, and only cover three interesting cases.

   (a) **Base Case:** f is of the shape $r(t_1, ..., t_n)$ such that $r \in P$ and $t_1, ..., t_n$ are terms. In this case, position p can only be of the form iq for some position q and $1 \leq i \leq n$ since the root position is of a formula. Therefore, (3.7) becomes

   $$\sigma(c) \ \vdash_{\mathcal{D}} \ r(t_1, ..., t_n)[\sigma(l)]_p \Leftrightarrow r(t_1, ..., t_n)[\sigma(r)]_p \ ,$$

   where $p = iq$ for some position q and $1 \leq i \leq n$. This can be rewritten to

   $$\sigma(c) \ \vdash_{\mathcal{D}} \ r(t_1, ..., t_i[\sigma(l)]_q, ..., t_n) \Leftrightarrow r(t_1, ..., t_i[\sigma(r)]_q, ..., t_n) \ .$$

   This amounts to proving the following two sequents:

   $$\sigma(c), r(t_1, ..., t_i[\sigma(l)]_q, ..., t_n) \ \vdash_{\mathcal{D}} \ r(t_1, ..., t_i[\sigma(r)]_q, ..., t_n) \ ,$$
   $$\sigma(c), r(t_1, ..., t_i[\sigma(r)]_q, ..., t_n) \ \vdash_{\mathcal{D}} \ r(t_1, ..., t_i[\sigma(l)]_q, ..., t_n) \ .$$

   Using Theorem 3.5, both sequents can be shown to be provable.

   (b) **Inductive Case:** f is of the shape $\varphi \wedge \psi$ such that φ and ψ are formulae. In this case, (3.7) becomes

   $$\sigma(c) \ \vdash_{\mathcal{D}} \ (\varphi \wedge \psi)[\sigma(l)]_p \Leftrightarrow (\varphi \wedge \psi)[\sigma(r)]_p \ . \tag{3.9}$$

   Position p can only be of the form $p = 1q$ or $p = 2q$ for some position q. We distinguish the two cases:

   i. $p = 1q$: In this case, sequent (3.9) becomes

   $$\sigma(c) \ \vdash_{\mathcal{D}} \ (\varphi[\sigma(l)]_q \wedge \psi) \Leftrightarrow (\varphi[\sigma(r)]_q \wedge \psi) \ . \tag{3.10}$$

   To proceed, we assume the following inductive hypothesis

   $$\sigma(c) \ \vdash_{\mathcal{D}} \ (\varphi[\sigma(l)]_q) \Leftrightarrow (\varphi[\sigma(r)]_q) \ , \tag{3.11}$$

   and we show that sequent (3.10) is provable.

   ii. $p = 2q$: analogous to the previous case.

(c) **Inductive Case:** f is of the shape $\forall x \cdot \varphi$ such that φ is a formula. In this case, (3.7) becomes

$$\sigma(c) \ \vdash_{\mathcal{D}} \ (\forall x \cdot \varphi)[\sigma(l)]_p \Leftrightarrow (\forall x \cdot \varphi)[\sigma(r)]_p \ . \tag{3.12}$$

Position p can only be of the form $p = 1q$ for some position q. Sequent (3.12) simplifies to

$$\sigma(c) \ \vdash_{\mathcal{D}} \ (\forall x \cdot \varphi[\sigma(l)]_q) \Leftrightarrow (\forall x \cdot \varphi[\sigma(r)]_q) \ . \tag{3.13}$$

To proceed, we assume that the following sequent is provable:

$$\sigma(c) \ \vdash_{\mathcal{D}} \ (\varphi[\sigma(l)]_q) \Leftrightarrow (\varphi[\sigma(r)]_q) \ , \tag{3.14}$$

and we show that sequent (3.13) is provable. □

2. *Proof of sequent* (3.8): is similar to the proof of sequent (3.7). We only show one inductive case.

(a) **Inductive Case:** f is of the shape $\varphi \wedge \psi$ such that φ and ψ are formulae. In this case, (3.8) becomes

$$\mathcal{D}((\varphi \wedge \psi)[\sigma(l)]_p), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}((\varphi \wedge \psi)[\sigma(r)]_p) \ . \tag{3.15}$$

Position p can only be of the form $p = 1q$ or $p = 2q$ for some position q. We distinguish the two cases:

i. $p = 1q$: In this case, sequent (3.15) becomes

$$\mathcal{D}((\varphi[\sigma(l)]_q \wedge \psi)), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}((\varphi[\sigma(r)]_q \wedge \psi)) \ . \tag{3.16}$$

To proceed, we assume that the following sequent is provable:

$$\mathcal{D}((\varphi[\sigma(l)]_q)), \sigma(c) \ \vdash_{\mathcal{D}} \ \mathcal{D}((\varphi[\sigma(r)]_q)) \ , \tag{3.17}$$

and we show that sequent (3.16) is provable.

ii. $p = 2q$: analogous to the previous case.

□

□

Summary. In this section, we have defined the criteria for the validity and well-definedness preservation of term rewrite rules when rewriting interleaves with the rule of the proof system developed in [12]. In the next section, we show how rewriting can be systematically used as a proof step.

4 Rewriting as a Proof Step

Rewriting can be used in proofs alongside the WD-preserving sequent calculus. Conditional term rewrite rules which have the same left hand side are grouped together. For this purpose, we use a more convenient notation. Given a valid and WD-preserving (grouped) conditional term rewrite rule

$$\begin{aligned} l \rightarrow \quad & c_1 : r_1 \\ & \ldots \\ & c_n : r_n \end{aligned}$$

we can add the following proof step to our calculus

$$\frac{\left\{\begin{array}{l} H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(c_1) \vee ... \vee \sigma(c_n)) \\ H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ \sigma(c_1) \vee ... \vee \sigma(c_n) \\ H, \sigma(c_1), P[\sigma(r_1)]_p \ \vdash_{\mathcal{D}} \ R \ \ ... \ \ H, \sigma(c_n), P[\sigma(r_n)]_p \ \vdash_{\mathcal{D}} \ R \end{array}\right.}{H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ R} \ \rightarrow hyp_{\mathcal{D}} \tag{4.1}$$

under the proviso that all free variables of $\sigma(r)$ (for all i such that $1 \leq i \leq n$) occur free in $P[\sigma(l)]_p$. This proof step allows the hypothesis $P[\sigma(l)]_p$ to be rewritten to several cases according to the rewrite rule. Under the proviso that all free variables of $\sigma(r_i)$ (for all i such that $1 \leq i \leq n$) occur free in $R[\sigma(l)]_p$, the following proof step can be added for goal rewriting

$$\frac{\left\{\begin{array}{l} H \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(c_1) \vee ... \vee \sigma(c_n)) \\ H \ \vdash_{\mathcal{D}} \ \sigma(c_1) \vee ... \vee \sigma(c_n) \\ H, \sigma(c_1) \ \vdash_{\mathcal{D}} \ R[\sigma(r_1)]_p \ \ ... \ \ H, \sigma(c_n) \ \vdash_{\mathcal{D}} \ R[\sigma(r_n)]_p \end{array}\right.}{H \ \vdash_{\mathcal{D}} \ R[\sigma(l)]_p} \ \rightarrow goal_{\mathcal{D}} \ . \tag{4.2}$$

Proof steps (4.1) and (4.2) can be derived using the cut rule, followed by a disjunction elimination (i.e., case split) after which rewriting can be applied. We now examine some special cases that can be used to facilitate proofs.

4.1 Unconditional Term Rewrite Rules

A term rewrite rule $l \xrightarrow{c} r$ is called *unconditional* iff $c \mathrel{\widehat{=}} \top$. In this case, steps (4.1) and (4.2) can be simplified as follows:

$$\frac{H, P[\sigma(r)]_p \ \vdash_{\mathcal{D}} \ R}{H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ R} \ \rightarrow uhyp_{\mathcal{D}} \tag{4.3}$$

$$\frac{H \ \vdash_{\mathcal{D}} \ R[\sigma(r)]_p}{H \ \vdash_{\mathcal{D}} \ R[\sigma(l)]_p} \ \rightarrow ugoal_{\mathcal{D}} \ . \tag{4.4}$$

4.2 Case-complete Grouped Term Rewrite Rules

A grouped term rewrite rule

$$\begin{array}{rl} l \rightarrow & c_1 : r_1 \\ & ... \\ & c_n : r_n \end{array}$$

is called *case-complete* iff the following sequent is provable:

$$\vdash_{\mathcal{D}} \ c_1 \vee ... \vee c_n \ .$$

In this case, steps (4.1) and (4.2) can be simplified as follows:

$$\frac{\left\{\begin{array}{l} H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(c_1) \vee ... \vee \sigma(c_n)) \\ H, \sigma(c_1), P[\sigma(r_1)]_p \ \vdash_{\mathcal{D}} \ R \ \ ... \ \ H, \sigma(c_n), P[\sigma(r_n)]_p \ \vdash_{\mathcal{D}} \ R \end{array}\right.}{H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ R} \ \rightarrow chyp_{\mathcal{D}} \tag{4.5}$$

$$\frac{\left\{\begin{array}{l} H \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(c_1) \vee ... \vee \sigma(c_n)) \\ H, \sigma(c_1) \ \vdash_{\mathcal{D}} \ R[\sigma(r_1)]_p \ \ ... \ \ H, \sigma(c_n) \ \vdash_{\mathcal{D}} \ R[\sigma(r_n)]_p \end{array}\right.}{H \ \vdash_{\mathcal{D}} \ R[\sigma(l)]_p} \ \rightarrow cgoal_{\mathcal{D}} \ . \tag{4.6}$$

4.3 Top-level Occurrence

Definition 4.1 (Top-Level Occurrence)**.** *Let t be a term, f be a formula, p be a position within f. We say that t has a top-level occurrence in f if f is either of the form*

1. *$q(t_1, ..., t_n)[t]_p$ where $q \in P$ and t_1,..., t_n are terms, or;*
2. *$(t_1 = t_2)[t]_p$ where t_1 and t_2 are terms.*

If t has a top-level occurrence in f, then it also has a top-level occurrence in $\neg f$.

We have the following interesting property:

Proposition 4.2. *If the term t has a top-level occurrence in formula f, then the following holds*

$$\vdash_{\mathcal{D}} \ \mathcal{D}(f) \Rightarrow \mathcal{D}(t) \ .$$

If we further constrain grouped conditional term rewrite rules such that we have

$$\vdash_{\mathcal{D}} \ \mathcal{D}(l) \Rightarrow \bigwedge_{i=1}^{n} \mathcal{D}(c_i) \ ,$$

Proposition 4.2 can be used to simplify proofs. Let $P[\sigma(l)]_p$ be a formula such that $\sigma(l)$ occurs at the top-level. Since the grouped term rewrite rule is valid and WD-preserving, and using the previous proposition, we have the following

$$\vdash_{\mathcal{D}} \ \mathcal{D}(P[\sigma(l)]_p) \Rightarrow \mathcal{D}(\sigma(l))$$

and, consequently:

$$\vdash_{\mathcal{D}} \ \mathcal{D}(P[\sigma(l)]_p) \Rightarrow \bigwedge_{i=1}^{n} \mathcal{D}(\sigma(c_i))$$

under the proviso that all free variables of $\sigma(c_i)$ (for all i such that $1 \leq i \leq n$) occur free in $P[\sigma(l)]_p$. In this particular case, the sequents

$$H, P[\sigma(l)]_p \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(c_1) \vee ... \vee \sigma(c_n)) \ ,$$
$$H, P \ \vdash_{\mathcal{D}} \ \mathcal{D}(\sigma(c_1) \vee ... \vee \sigma(c_n))$$

in (4.1) and (4.2) respectively, are guaranteed to be provable. As such, they could be removed from the list of sub-goals that the modeller sees.

5 Applications to Event-B

As mentioned in 1.1, Event-B modelling is carried out using two constructs: contexts and machines. A third construct, called *theory*, has been implemented to bring a degree of meta-reasoning to the Rodin platform [7]. The theory construct has the following shape:

1. *Sets.* A theory can define a number of given sets which define the types on which the theory is parametrised.

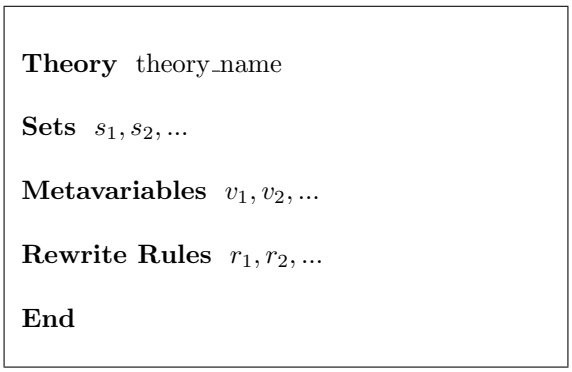

Figure 5: The Theory Construct

2. *Metavariables.* A theory can define a number of metavariables that can be used to specify rewrite rules. Each metavariable is associated with a type; this can be constructed using the given sets of the theory as well as the built-in types (e.g., $\mathbb{Z}$) using type constructors. For example, if a given set S is defined within a theory, then $\mathbb{P}(\mathbb{Z}) \times S$ can be used as a type for a metavariable.

3. *Rewrite Rules.* Rewrite rules are one-directional equations that can be used to rewrite formulae to equivalent forms. As part of specifying a rewrite rule, the *theory developer* decides whether the rule can be applied automatically without user intervention or interactively following a user request.

The theory construct can be extended to enable the specification of inference rules. In brief, it facilitates the following:

- specification of proof rules within the same platform providing a degree of meta-reasoning within Rodin,
- validation of specified proof rules to ensure that the soundness of the prover is not compromised.

The validation of rewrite rules is achieved by means of proof obligations. Definition 3.2 and 3.4 defined the criteria for validity and WD-preservation of rewrite rules.

The theory construct has been developed as part of a *rule-based prover* [11] which, in brief, offers the following capabilities:

1. Users can develop theories in the same way as contexts and machines. At the moment, theory development includes specification of rewrite rules including definition of sets and metavariables. Metavariables must be defined with their types which can be constructed from the theory sets and any built-in types (e.g., $\mathbb{Z}$) using type constructors (e.g., $\mathbb{P}$).

2. Users can validate rewrite rules through generated proof obligations. The proof obligations generated for rules are to establish soundness, well-definedness preservation as well as case-completeness.

3. Users can deploy theories to a specific directory where they become available to the interactive and automatic provers of Rodin. Theory deployment adds soundness information to all deployed rules.

4. Users can use rewrite rules defined within the deployed theories as a part of the proving activity. A pattern matching mechanism is implemented to calculate applicable rewrite rules to any given sequent.

Examples. The following two rules are valid and WD-preserving:

$$card(i..j) \xrightarrow{i \leq j} j - i + 1 \tag{5.1}$$

$$card(i..j) \xrightarrow{i > j} 0\ , \tag{5.2}$$

where i and j are integers, $i..j$ denotes an integer range, and $card$ denotes the cardinality operator. The following rules are not WD-preserving:

$$a \xrightarrow{\top} \frac{a}{a} \tag{5.3}$$

$$(f \lhd \{z \mapsto y\})(x) \xrightarrow{x \neq z} f(x), \tag{5.4}$$

where a is an integer, f a relation, x, y, and z are of arbitrary types. Moreover, '$\lhd$' denotes relational override. Rule 5.4 is not WD-preserving since there could be a case where $f \lhd \{z \mapsto y\}$ is a function but f is not. For instance, consider $f = \{1 \mapsto 2, 1 \mapsto 3, 2 \mapsto 4\}$, then $f \lhd \{1 \mapsto 5\} = \{1 \mapsto 5, 2 \mapsto 4\}$. In this case, $(f \lhd \{1 \mapsto 5\})(1)$ is well-defined, but $f(1)$ is not.

6 Future Work & Conclusions

In this paper, we provided a treatment of well-definedness and rewriting. We singled out the necessary conditions under which rewriting preserves well-definedness. These conditions are necessary for the valid interleaving between rewriting steps and deduction in the WD-preserving proof calculus presented in [12]. In our study, we used the language signature Σ whereby terms are only defined using other terms. In general, however, terms can also be constructed using formulae e.g., set comprehension $\{x \cdot P\}$. This changes the well-definedness conditions of terms, and it is interesting to establish whether the conditions outlined in Definition 3.2 and 3.4 are indeed sufficient.

We have presented a study unifying the notions of term rewriting and well-definedness in the context of the interleaving between deduction and rewriting. The results of this paper provided the theoretical foundations of an extensible rewriting-based prover (also called rule-based prover) that has been implemented for Event-B.

References

[1] J.-R. Abrial, M. Butler, S. Hallerstede, and L. Voisin. An open extensible tool environment for Event-B. In *International Conference on Formal Engineering Methods (ICFEM)*, LNCS. SV, 2006.

[2] Jean-Raymond Abrial and Dominique Cansell. Click'n prove: Interactive proofs within set theory. In David A. Basin and Burkhart Wolff, editors, *TPHOLs*, volume 2758 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2003.

[3] Jean-Raymond Abrial and Stefan Hallerstede. Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B. *Fundam. Inf.*, 77(1-2):1–28, 2007.

[4] Jean-Raymond Abrial and Louis Mussat. On using conditional definitions in formal theories. In *ZB '02: Proceedings of the 2nd International Conference of B and Z Users on Formal Specification and Development in Z and B*, pages 242–269, London, UK, 2002. Springer-Verlag.

[5] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[6] R. J. R. Back. Refinement calculus, part ii: parallel and reactive programs. In *REX workshop: Proceedings on Stepwise refinement of distributed systems: models, formalisms, correctness*, pages 67–93, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[7] Michael Butler and Stefan Hallerstede. The Rodin Formal Modelling Tool. *BCS-FACS Christmas 2007 Meeting - Formal Methods In Industry, London.*, December 2007.

[8] Nachum Dershowitz. Term Rewriting Systems by (Marc Bezem, Jan Willem Klop, and Roel de Vrijer, eds.), Cambridge University Press, Cambridge Tracts in Theoretical Computer Science 55, 2003, hard cover: ISBN 0-521-39115-6, xxii+884 pages. *Theory and Practice of Logic Programming*, 5(03):395–399, 2005.

[9] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. *J. Autom. Reason.*, 31(1):33–72, 2003.

[10] Michael Leuschel and Michael Butler. ProB: A model checker for B. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *FME 2003: Formal Methods*, LNCS 2805, pages 855–874. Springer-Verlag, 2003.

[11] Issam Maamria, Michael Butler, Andrew Edmunds, and Abdolbaghi Rezazadeh. On an Extensible Rule-based Prover for Event-B. In *ABZ '10: Proceedings of the 2nd international conference on Abstract State Machines, B and Z*, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] Farhad Mehta. A practical approach to partiality — a proof based approach. In *ICFEM '08: Proceedings of the 10th International Conference on Formal Methods and Software Engineering*, pages 238–257, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] Farhad Mehta. *Proofs for the Working Engineer*. PhD Thesis, ETH Zurich, 2008.

[14] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[15] S. Owre, J. M. Rushby, , and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, jun 1992. Springer-Verlag.