Available online at www.sciencedirect.com
SCIENCE DIRECT
Theoretical Computer Science 360 (2006) 386–414
Theoretical Computer Science
www.elsevier.com/locate/tcs

# Semantic foundations for generalized rewrite theories☆

Roberto Bruni[a,*], José Meseguer[b]

[a] *Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti n.2, I-56127 Pisa, Italy*
[b] *CS Department, University of Illinois at Urbana-Champaign, USA*

Received 6 April 2004; received in revised form 6 July 2005; accepted 28 April 2006

Communicated by M. Nielsen

**Abstract**

Rewriting logic (RL) is a logic of actions whose models are concurrent systems. Rewrite theories involve the specification of equational theories of data and state structures together with a set of rewrite rules that model the dynamics of concurrent systems. Since its introduction, more than one decade ago, RL has attracted the interest of both theorists and practitioners, who have contributed in showing its generality as a semantic and logical framework and also as a programming paradigm. The experimentation conducted in these years has suggested that some significant extensions to the original definition of the logic would be very useful in practice. These extensions may develop along several dimensions, like the choice of the underlying equational logic, the kind of side conditions allowed in rewrite rules and operational concerns for the execution of certain rewrites. In particular, the Maude system now supports subsorting and conditional sentences in the equational logic for data, and also frozen arguments to block undesired nested rewrites; moreover, it allows equality and membership assertions in rule conditions. In this paper, we give a detailed presentation of the inference rules, model theory, and completeness of such generalized rewrite theories. Our results provide a mathematical semantics for Maude, and a foundation for formal reasoning about Maude specifications.


*Keywords:* Conditional rewriting logic; Membership equational logic; Contextual rewrites; Semantics

## 0. Introduction

Originally, term rewriting was mainly devised as a technique for equational deduction and simplification to canonical normal form, and its successful spreading can be dated back to the famous Knuth–Bendix completion algorithm in the seventies [26]. More recently, in the nineties, rewriting logic (RL) [34] has emerged as a unified model of concurrency, with rewriting now representing concurrent evolution instead of equational simplification. This perspective has been the basis for several implementation efforts and has stimulated many theoretical developments. In fact, since its original formulation, a substantial body of research has shown that RL has good properties as a *semantic framework*, particularly for concurrent and distributed computation, and also as a *logical framework*, a meta-logic in which other logics can

☆ Research supported by the MIUR Project COFIN 2001013518 COMETA, by the FET-GC Project IST-2001-32747 AGILE, and by ONR Grant N00014-02-1-0715.
* Corresponding author. Tel.: +39 050 2212785; fax: +39 050 2212726.
*E-mail addresses:* bruni@di.unipi.it (R. Bruni), meseguer@cs.uiuc.edu (J. Meseguer).
*URLs:* http://www.di.unipi.it/∼bruni (R. Bruni), http://formal.cs.uiuc.edu/meseguer/ (J. Meseguer).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.04.012

be naturally represented (see the more than 300 references listed in the special TCS issue [30], and the five WRLA Proceedings [35,25,21,23,28]). Indeed, the computational and logical meanings of a rewrite $t \to t'$ are like two sides of the same coin. Computationally, $t \to t'$ means that the state component $t$ can *evolve* to the component $t'$. Logically, $t \to t'$ means that from the formula $t$ one can *deduce* the formula $t'$. RL has also been shown to have good properties as a *declarative programming paradigm*, as demonstrated by the mature and efficient implementations of the ELAN [40], CafeOBJ [19], and Maude [13] languages.

At present, rewriting systems provide advanced technology that encompasses programming languages, bioinformatics, constraint solving, model checking, and theorem proving, with tools whose performance is comparable to that of conventional languages and conventional tool implementations (see, e.g., [30] for a survey and references). The feedback from applications to theory has played an important role in extending the notion of RL with additional features. This paper develops new semantic foundations for a generalized version of RL that has a running implementation (Maude 2.1) and that supports several novel features of RL whose expressiveness has been found very useful in practice.

The close contact with many applications in all the above areas has served as a good stimulus for a *substantial increase in expressive power* of the RL formalism by generalization along several dimensions:

(1) Since a rewrite theory is essentially a triple $\mathcal{R} = (\Sigma, E, R)$, with $(\Sigma, E)$ an equational theory, and $R$ a set of labeled rewrite rules that are applied *modulo* the equations $E$, it follows that RL is *parameterized by the choice of an underlying equational logic*; therefore, generalizations towards more expressive equational logics yield more expressive versions of RL.
(2) Another dimension along which expressiveness can be increased is by allowing *more general conditions* in conditional rewrite rules. These can be either rewrites, like in conditional RL, or provable equalities and memberships in $(\Sigma, E)$, or any mixture of them.
(3) Yet another dimension has to do with the fact that *rewrites should not happen everywhere*, because in many applications suitable evaluation strategies or context-dependent rewrites could considerably improve performance and even avoid non-termination. Correspondingly, rewrite theories can be generalized by *forbidding rewriting under certain operators or operator positions* (frozen operators and arguments). Although this could be regarded as a purely *operational aspect*, the frequent need for it in many applications suggests that it should be supported directly at the semantic level of rewrite theories.

In this paper we generalize rewrite theories along all these three dimensions. Along dimension 1, by choosing *membership equational logic* (MEL) [36,6] as the underlying equational logic. This is a very expressive many-kinded Horn logic whose atomic formulas are equations $t = t'$ and memberships $t : s$, stating that a term $t$ has sort $s$. It provides support for sorts, subsorts, operator overloading and partiality, and it contains as special cases the order-sorted, many-sorted, and unsorted versions of equational logic.

Along dimension 2, we allow very general conditions in conditional rewrite rules which, assuming an underlying membership equational theory $(\Sigma, E)$, can be of the form,

$$(\forall X) r : t \to t' \text{ if } \bigwedge_{i \in I} p_i = q_i \wedge \bigwedge_{j \in J} w_j : s_j \wedge \bigwedge_{l \in L} t_l \to t'_l,$$

where $r$ is the rule label, all terms are $\Sigma$-terms, and the rule can be made conditional to other equations, memberships, and rewrites being satisfied.

Finally, along dimension 3, we allow declaring certain operator arguments as *frozen*, thus blocking rewriting under them. This leads us to define a *generalized rewrite theory* as a four tuple, $\mathcal{R} = (\Sigma, E, \phi, R)$, where $(\Sigma, E)$ is a membership equational theory, $R$ is a set of labeled conditional rewrite rules of the general form above, and $\phi$ is a function assigning to each operator $f : k_1 \dots k_n \to k$ in $\Sigma$ the subset $\phi(f) \subseteq \{1, \dots, n\}$ of its frozen arguments.

As already mentioned, such a notion of generalized rewrite theory has been arrived at through a long and extensive contact with many applications. In fact, practice has gone somewhat ahead of theory: all the above generalizations have already been implemented in Maude 2.1 [14]. The importance of generalizing rewrite theories along dimension 1 has to do with the greater expressiveness allowed by having sorts, subsorts, subsort overloaded operators, and partial functions; all this is further explained in Section 1.3. We shall illustrate the importance of generalizing along dimensions 2 and 3 in Section 2 with an example showing that, in essence, this brings RL and *structural operational semantics* (whose close relationship had already been emphasized in [29,31,32,7]) closer than ever before, even at the syntax level (see Example 2.1). To some extent, frozen arguments are for rewrite theories the analogous of the *strategy annotations* used for equational theories in OBJ, CafeOBJ, and Maude to improve efficiency and/or to guarantee the termination

of computations. In fact, strategy annotations allow for replacing unrestricted equational rewriting by the so-called *context-sensitive rewriting* (see, e.g., [27] and references therein). Thus, in Maude, rewriting with both equations $E$ and rules $R$ can be made context-sensitive.

Given the above notion of generalized rewrite theory, the paper proposes answers to the following questions:

- What are RL's *rules of deduction* for generalized rewrite theories?
- What are the *models* of a generalized rewrite theory? Are there initial and free models?
- Is RL *complete* with respect to its model theory, so that a rewrite is provable from a (generalized) rewrite theory $\mathcal{R}$ if and only if it is satisfied by all models of $\mathcal{R}$?

The answers given (all in the affirmative) are in fact non-trivial *generalizations* of the original inference rules, model theory, initial and free models, and completeness theorem for RL over unsorted equational logic, as developed in [34].

We carry out our constructions exploiting MEL theories to take advantage of its soundness and completeness theorems and the existence of initial/free algebras (these properties are helpful in the proofs of our main theorems). Moreover, MEL is the underlying equational logic of Maude, so that MEL theories form the so-called functional modules of Maude (where rewrite rules are absent) and can be directly used for experimentation and meta-theoretic reasoning based on the Maude tool suite.

In summary, therefore, this paper develops new *semantic foundations* for a generalized version of RL, along several dimensions, that has been found to substantially increase its expressiveness in concrete applications. At the programming language level, this paper does also provide the needed mathematical semantics for Maude 2.1, and a foundation for formal reasoning about Maude specifications. To emphasize this aspect, in most examples (generalized and MEL) rewrite theories will be presented as their corresponding Maude modules. Maude is a declarative language, hence the correspondence at the linguistic level is essentially one-to-one. We shall use Petri nets as the running case study, as they offer a basic model of concurrency that can be used to illustrate all the features of generalized rewrite theories.

### *0.1. Synopsis*

In Section 1.1 we recap from [34] (and slightly rephrase) the original presentation of RL, in Section 1.2 we recall the basics of Petri nets, and in Section 1.3 we review membership equational logic. Sections 2 and 3 present the original contributions of the paper, introducing generalized rewrite theories, their proof theory, their model theory, and the completeness results. Note that the algebras of reachability and decorated sequents are expressed as membership equational theories themselves (a framework not available when [34] was published). In Section 4 we discuss conditional sequents and their provability. Conclusions are drawn in the last section, where we also discuss related and future work. Proofs of main results are collected in Appendix B.

This paper is the full version of [8], extended here with proofs and examples.

## 1. Background

### *1.1. Conditional RL*

Though in the rewriting community it is folklore that rewrite theories are parametric w.r.t. the underlying equational logic of data specification, the details have been fully spelled out only for unsorted equational logic, and rules of the form (1) below. Since only unsorted theories were considered in [34], here (and in Appendix A), but not in the rest of the paper where ordered sorts are used, an (equational) *signature* is a family of sets of *function symbols* (also *operators*) $\Sigma = \{\Sigma_n\}_{n\in\mathbb{N}}$ indexed by arities $n$, and a *theory* is a pair $(\Sigma, E)$ where $E = \{(\forall X_i)\ t_i = t_i'\}_{1\leqslant i\leqslant m}$ is a set of (universally quantified) $\Sigma$-equations, with $t_i, t_i' \in \mathbb{T}_\Sigma(X_i)$ two $\Sigma$-terms with variables in $X_i$. We let $t =_E t'$ denote the congruence modulo $E$ of two terms $t, t'$ and $[t]_E$ or just $[t]$ denote the $E$-equivalence class of $t$ modulo $E$. We shall denote by $t[u_1/x_1, \ldots, u_n/x_n]$ (abbreviated $t[\vec{u}/\vec{x}]$) the term obtained from $t$ by simultaneously replacing the occurrences of $x_i$ by $u_i$ for $1\leqslant i\leqslant n$.

**Definition 1.1** (*Conditional rewrite theory*). A (*labeled*) *conditional rewrite theory* $\mathcal{R}$ is a tuple $\mathcal{R} = (\Sigma, E, R)$, where $(\Sigma, E)$ is an unsorted equational theory and $R$ is a set of (labeled) conditional rewrite rules having the form below,

$$\frac{t \in \mathbb{T}_\Sigma(X)}{(\forall X)\ t \to t}\ \textbf{Reflexivity} \qquad \frac{(\forall X)\ t_1 \to t_2, \quad (\forall X)\ t_2 \to t_3}{(\forall X)\ t_1 \to t_3}\ \textbf{Transitivity}$$

$$\frac{E \vdash (\forall X)\ t = u, \quad (\forall X)\ u \to u', \quad E \vdash (\forall X)\ u' = t'}{(\forall X)\ t \to t'}\ \textbf{Equality}$$

$$\frac{f \in \Sigma_n, \quad (\forall X)\ t_i \to t_i' \text{ for } i \in [1,n]}{(\forall X)\ f(t_1,\ldots,t_n) \to f(t_1',\ldots,t_n')}\ \textbf{Congruence}$$

$$\frac{\begin{array}{c}(\forall X)\ r{:}t \to t' \text{ if } \bigwedge_{1\le i\le \ell} t_i \to t_i' \in R, \quad \theta,\theta'{:}X \to \mathbb{T}_\Sigma(Y) \\ (\forall Y)\ \theta(t_i) \to \theta(t_i') \text{ for } 1 \le i \le \ell, \quad (\forall Y)\ \theta(x) \to \theta'(x) \text{ for } x \in X\end{array}}{(\forall Y)\ \theta(t) \to \theta'(t')}\ \begin{array}{l}\textbf{Nested}\\ \textbf{Replacement}\end{array}$$

Fig. 1. Deduction rules for conditional rewrite theories.

with $t, t', t_i, t_i' \in \mathbb{T}_\Sigma(X)$.

$$(\forall X)\ r{:}t \to t' \text{ if } t_1 \to t_1' \wedge \cdots \wedge t_\ell \to t_\ell'. \tag{1}$$

The theory $(\Sigma, E)$ defines the static data structure for the states of the system (e.g., a free monoid for strings or a free commutative monoid for multisets), while $R$ defines the dynamics (e.g., productions in phrase-structure grammars or transitions in Petri nets).

Given a rewrite theory $\mathcal{R}$, its *RL* is a sequent calculus whose sentences have the form $(\forall X)\ t \to t'$ (with the dual, logical and computational, meaning explained in the Introduction). We say that $\mathcal{R}$ *entails* a sequent $(\forall X)\ t \to t'$, and write $\mathcal{R} \vdash (\forall X)\ t \to t'$, if the sequent $(\forall X)\ t \to t'$ can be obtained by means of the inference rules in Fig. 1. Roughly, (*Reflexivity*) introduces idle computations, (*Transitivity*) expresses the sequential composition of rewrites, (*Equality*) means that rewrites are applied modulo the equational theory $E$, (*Congruence*) says that rewrites can be nested inside larger contexts. The most complex rule is (*Nested Replacement*), stating that given a rewrite rule $r \in R$ and two substitutions $\theta, \theta'$ for its variables such that for each $x \in X$ we have $\theta(x) \to \theta'(x)$, then $r$ can be concurrently applied to the rewrites of its arguments, once that the conditions of $r$ can be satisfied in the initial state defined by $\theta$. Since rewrites are applied modulo $E$, the sequents can be equivalently written $(\forall X)\ [t]_E \to [t']_E$ (or just $(\forall X)\ [t] \to [t']$ when $E$ is understood) and the generic rule (1) can be written as $(\forall X)\ r{:}[t] \to [t']$ if $\bigwedge_{1 \leqslant i \leqslant \ell} [t_i] \to [t_i']$, in which case the (*Equality*) rule is not explicitly needed.

From the model-theoretic viewpoint, the sequents can be decorated with *proof terms* in a suitable algebra that exactly captures concurrent computations. Since this aspect will be generalized in full detail in Section 3.2, the axiomatization for the unsorted case is just sketched below. We remark that each rewrite theory $\mathcal{R}$ has initial and free models and that a completeness theorem reconciles the proof theory and the model theory, stating that a sequent is provable from $\mathcal{R}$ if and only if it is satisfied in all models of $\mathcal{R}$ (called $\mathcal{R}$-*systems*) [34].

The algebra of sequents contains:

- the terms $[t]$ in $\mathbb{T}_{\Sigma,E}$ for idle rewrites, with the operators and equations in $(\Sigma, E)$ lifted to the level of sequents, e.g., if $\alpha_i{:}\,[t_i] \to [t_i']$ for $i \in [1,n]$, then

$$f(\alpha_1,\ldots,\alpha_n){:}\,[f(t_1,\ldots,t_n)] \to [f(t_1',\ldots,t_n')];$$

- one additional operator $r$ with arity $|X| + \ell$ for each rule $r \in R$ of the form (1);
- the concatenation operator $\_\,;\,\_$.

For example, if $\alpha_1{:}\,[t_1] \to [t_2]$ and $\alpha_2{:}\,[t_2] \to [t_3]$, then $\alpha_1;\,\alpha_2{:}\,[t_1] \to [t_3]$ via (*Transitivity*). Similarly, if $\{\beta_i{:}\,[\theta(t_i)] \to [\theta(t_i')]\}_{1 \leqslant i \leqslant \ell}$ and $\{\alpha_x{:}\,[\theta(x)] \to [\theta'(x)]\}_{x \in X}$ are used as premises in (*Nested Replacement*), then the conclusion is decorated by $r(\vec{\alpha}, \vec{\beta})$.

The axioms express (see Appendix A):

(1) the associativity of $\_\,;\,\_$ with idle rewrites $[t]$ as identities, so that sequents become the arrows of a category whose objects are the equivalence classes of terms $[t]$;

(2) the functoriality of the $(\Sigma, E)$-structure w.r.t. the category above, in the sense that for identities we have $f([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$ and for composition we have $f((\alpha_1; \beta_1), \ldots, (\alpha_n; \beta_n)) = f(\alpha_1, \ldots, \alpha_n); f(\beta_1, \ldots, \beta_n)$ (also the equations in $E$ are lifted to the level of sequents);
(3) the so-called *decomposition* and *exchange* laws, saying that the application of the rule $r$ to $[\theta(t)]$ is concurrent w.r.t. the rewrites of the arguments of $t$, i.e., with $r(\vec{\alpha}, \vec{\beta})$ as above, we have $r(\vec{\alpha}, \vec{\beta}) = r(\theta(\vec{x}), \vec{\beta}); t'[\vec{\alpha}/\vec{x}]$, and for "suitable" $\beta'_i : [\theta'(t_i)] \to [\theta'(t'_i)]$, we have $r(\theta(\vec{x}), \vec{\beta}); t'[\vec{\alpha}/\vec{x}] = t[\vec{\alpha}/\vec{x}]; r(\theta'(\vec{x}), \vec{\beta}')$ (in these equations, the square brackets [_ ] around $t$ and $t'$ are omitted to disambiguate their use around substitutions).

### *1.2. A running case study: Petri nets*

Petri nets offer a simple setting that nicely illustrates the features discussed in the paper. Although the chosen case study may appear complex to non-experts, Petri nets can be more intuitively understood as an elementary form of multiset rewrite systems.

To fix notation, given a set $S$, let $\mathcal{M}(S)$ denote the set of finite multisets over $S$. An element $u \in \mathcal{M}(S)$ can be regarded as a function $u : S \to \mathbb{N}$, and we write $u(a)$ to denote the number of instances of $a \in S$ inside the multiset $u$. We write $a \in \mathcal{M}(S)$ if $u(a) > 0$. Let $N = (S, T, pre, post)$ be a Petri net, where $S$ is the set of places, $T$ is the set of transitions (with $S \cap T = \emptyset$), and $pre, post : T \to \mathcal{M}(S)$ are source and target functions assigning pre- and post-sets to transitions. The operational semantics of $N$ is defined by taking multisets of places $u \in \mathcal{M}(S)$ (called *markings*) as states, with a *step* from state $u$ to state $v$ whenever a multiset of transitions $\mu \in \mathcal{M}(T)$ exists such that, for any $a \in S$:
(1) $u(a) \geqslant \sum_{t \in \mu} \mu(t) \cdot pre(t)(a)$.
(2) $v(a) = u(a) - \sum_{t \in \mu} \mu(t) \cdot pre(t)(a) + \sum_{t \in \mu} \mu(t) \cdot post(t)(a)$.
The rewrite theory associated to $N$ is $\mathcal{R}_N = (\Sigma_N, E_N, R_N)$, where:

- The signature $\Sigma_N$ contains a special constant 0 denoting the empty marking, a constant $a$ for each place $a \in S$ denoting the corresponding singleton marking, and a binary operator $\oplus$ to model multiset union.
- The equations in $E_N$ state that $\oplus$ is associative, commutative and has identity 0 (this means that $E_N$-equivalence classes of ground $\Sigma_N$-terms exactly correspond to multisets over $S$).
- There is an unconditional rewrite rule

$$(\forall \emptyset)\ t : \underline{pre(t)} \to \underline{post(t)}$$

for each transition $t \in T$ (where $\underline{u}$ denotes the obvious $E_N$-equivalence class modeling the multiset $u$).

**Example 1.1.** For example, the net $N_1$ with four places $S_1 = \{a, b, c, d\}$ and three transitions $T_1 = \{t_1, t_2, t_3\}$ with $pre(t_1) = \{a, a\}$, $post(t_1) = \{b\}$, $pre(t_2) = \{b, c\}$, $post(t_2) = \{d\}$, $pre(t_3) = \{a, b\}$, $post(t_3) = \{a\}$ defines the rewrite theory in Fig. 2, written in essentially self-explanatory Maude notation.[1] Note that the rewrite theory is unsorted, although the Maude syntax requires one sort to be defined.

From (*Replacement*) we have e.g., that $\mathcal{R}_{N_1} \vdash (\forall \emptyset)\mathtt{a} \oplus \mathtt{a} \to \mathtt{b}$ and $\mathcal{R}_{N_1} \vdash (\forall \emptyset)\mathtt{b} \oplus \mathtt{c} \to \mathtt{d}$. From (*Equality*) we have that $\mathcal{R}_{N_1} \vdash (\forall \emptyset)\mathtt{c} \oplus \mathtt{b} \to \mathtt{d}$. From (*Congruence*) we have that $\mathcal{R}_{N_1} \vdash (\forall \emptyset)\mathtt{a} \oplus \mathtt{a} \oplus \mathtt{c} \to \mathtt{b} \oplus \mathtt{c}$ and $\mathcal{R}_{N_1} \vdash (\forall \emptyset)\mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c} \to \mathtt{b} \oplus \mathtt{d}$. From (*Transitivity*) we have that $\mathcal{R}_{N_1} \vdash (\forall \emptyset)\mathtt{a} \oplus \mathtt{a} \oplus \mathtt{c} \to \mathtt{d}$.

The corresponding decorated proof terms are, respectively, [t1], [t2], [t2], [t1⊕c], [t1⊕t2] and [(t1⊕c); t2].

By the axiomatization of decorated proof sequents we have e.g., that [t1] = [t1; b], [t2] = [(b⊕c); t2], [t1⊕t2] = [(t1; b)⊕((b⊕c); t2)] = [(t1⊕b⊕c); (b⊕t2)], and, analogously, [t1⊕t2] = [(a⊕a⊕t2); (t1⊕d)]. The last two equalities show a basic fact about concurrency: if two activities can be executed in parallel, then the actual order in which we may describe their execution is immaterial.

---

[1] In all our examples throughout the paper we use Maude syntax, which is so close to the corresponding mathematical notation for defining rewrite theories and MEL theories as to be almost self-explanatory. Rewrite theories are called *system modules* and are introduced with the keyword `mod` and ended with `endm`. Similarly, MEL theories, called *functional modules*, are introduced with the keyword `fmod` and ended with `endfm`. The general point to keep in mind is that each item: a sort, a subsort, an operation, an equation, a rule, etc., is declared with an obvious keyword: `sort`, `subsort`, `op`, `eq` (or `ceq` for conditional equations), `rl` (or `crl` for conditional rules), etc., with each declaration ended by a space and a period. Another important point is the use of "mix-fix" user-definable syntax, with the argument positions specified by underbars; for example, we can have a ternary operator `if _ then _ else _ fi`. A third point to keep in mind is that structural axioms such as associativity, commutativity, and identity do not need to be explicitly given by equations, but are instead declared as properties of the corresponding operator with the keywords: `assoc`, `comm`, and `id:`.

```
mod N1 is
  sort Marking .
   ops a b c d : -> Marking .
    op 0 : -> Marking .
    op _⊕_ : Marking Marking -> Marking [assoc comm id: 0] .
    rl [t1] : a⊕a => b .
    rl [t2] : b⊕c => d .
    rl [t3] : a⊕b => a .
endm
```

Fig. 2. The unsorted rewrite theory $\mathcal{R}_{N_1}$.

### *1.3. Membership equational logic*

In many applications, the unsorted signatures are not expressive enough to reflect in a natural way the features of the system (or language, or logic) to be modeled. The expressiveness can be increased by supporting more sorts (e.g., `Bool`, `Nat`, `Int`) via *many-sorted* signatures and relating them via *order-sorted* signatures (e.g., `NzNat` < `Nat` < `Int`). Equations in $E$ can be made more expressive by allowing *conditions* for their applications. Such conditions can be other equalities or membership assertions $t : s$. Conditional membership assertions can be useful as well. MEL [36,6] possesses all the above features (generalizing order-sorted equational logic) and it is supported by Maude. Since MEL has just equalities and unary predicates (memberships) it is a very simple and standard formalism, namely, it is exactly the monadic subset of many-sorted (although here we call it many-kinded to avoid confusion with sorts, that instead are treated as unary predicates) Horn clause logic with equality [36].

**Definition 1.2** (MEL *signature*). A MEL *signature* is a triple $(K, \Sigma, S)$ (just $\Sigma$ in the following), with $K$ a set of *kinds*, $\Sigma = \{\Sigma_{\delta,k}\}_{(\delta,k)\in K^*\times K}$ a many-kinded signature and $S = \{S_k\}_{k\in K}$ a $K$-kinded family of disjoint sets of sorts.

The kind of a sort $s$ is denoted by $[s]$. A MEL $\Sigma$-algebra $A$ contains a set $A_k$ for each kind $k \in K$, a function $A_f : A_{k_1} \times \cdots \times A_{k_n} \to A_k$ for each operator $f \in \Sigma_{k_1\cdots k_n,k}$ and a subset $A_s \subseteq A_k$ for each sort $s \in S_k$, with the meaning that the elements in sorts are well-defined, while elements without a sort are understood as *errors* or *undefined* elements. We write $\mathbb{T}_{\Sigma,k}$ and $\mathbb{T}_\Sigma(X)_k$ to denote, respectively, the set of closed $\Sigma$-terms with kind $k$ and of $\Sigma$-terms with kind $k$ over variables in $X$, where $X = \{x_1 : k_1, \ldots, x_n : k_n\}$ is a set of kinded variables.

Given a MEL signature $\Sigma$, *atomic formulae* have either the form $(\forall X)\ t = t'$ ($\Sigma$-equation) or $(\forall X)\ t : s$ ($\Sigma$-membership) with $t, t' \in \mathbb{T}_\Sigma(X)_k$ and $s \in S_k$; and $\Sigma$-*sentences* are conditional formulae of the form $(\forall X)A$ if $\bigwedge_i p_i = q_i \wedge \bigwedge_j w_j : s_j$, where $A$ is either a $\Sigma$-equation or a $\Sigma$-membership, and all the variables in $A$, $p_i$, $q_i$, and $w_j$ are in $X$.

**Definition 1.3** (MEL *theory*). A MEL theory is a pair $(\Sigma, E)$ for $\Sigma$ a MEL signature and $E$ a set of $\Sigma$-sentences.

We refer to [36] for the detailed presentation of $(\Sigma, E)$-algebras (i.e., the models of MEL theories), sound and complete deduction rules, initial and free algebras, and theory morphisms.

Order-sorted notation $s_1 < s_2$ (with $s_1, s_2 \in S_k$ for some kind $k$) can be used to abbreviate the conditional membership $(\forall x : k)\ x : s_2$ if $x : s_1$. Similarly, an operator declaration $f : s_1 \times \cdots \times s_n \to s$ abbreviates declaring $f$ at the kind level and giving the membership axiom $(\forall x_1 : k_1, \ldots, x_n : k_n)\ f(x_1, \ldots, x_n) : s$ if $\bigwedge_{1\leqslant i\leqslant n} x_i : s_i$. Therefore, *overloading* of the operator $f$ by adding another declaration $f : s'_1 \times \cdots \times s'_n \to s'$, with $[s_i] = [s'_i]$ and $[s] = [s']$ abbreviates giving yet another conditional membership axiom for $f$. We write $(\forall x_1 : s_1, \ldots, x_n : s_n)\ t = t'$ in place of $(\forall x_1 : k_1, \ldots, x_n : k_n)\ t = t'$ if $\bigwedge_{1\leqslant i\leqslant n} x_i : s_i$. Moreover, for a list of variables of the same sort $s$, we write $(\forall x_1, \ldots, x_n : s)$, and let the sentence $(\forall X)\ t : k$ mean $t \in \mathbb{T}_{(\Sigma,E)}(X)_k$.

We present a variant of our running example of Petri nets to illustrate some nice features of MEL theories.

**Example 1.2.** Regarding Petri nets, the use of more sophisticated equational theories allows us to make the correspondence more precise. For example, we can introduce a sort `Place` just for places, marking the distinction between constants $a \in S$ and the special constant 0 (that has instead sort `Marking`). Moreover, using subsorting, we can declare the sort `Place` as a subsort of `Marking`: in this way a place can still be regarded as a singleton (as in the unsorted

```
mod N1 is including BOOL .
     sorts Place Safe Marking .
  subsorts Place < Safe < Marking .
       ops a b c d : -> Place .
        op 0 : -> Safe .
        op _⊕_ : Marking Marking -> Marking [assoc comm id: 0] .
        op _ in _ : Place Marking -> Bool .
      vars P Q : Place .
       var S : Safe .
       cmb P⊕S : Safe  if (P in S)=false .
        eq P in 0 = false .
       ceq P in Q = true  if P=Q .
       ceq P in Q = false  if P=/=Q .
       ceq (P in Q⊕S) = true  if P=Q .
       ceq (P in Q⊕S) = (P in S)  if P=/=Q .

        rl [t1] : a⊕a => b .
        rl [t2] : b⊕c => d .
        rl [t3] : a⊕b => a .
endm
```

Fig. 3. The MEL rewrite theory $\mathcal{R}_{N_1}$.

representation) without any need for an embedding function like
`op {_} : Place -> Marking`
that would just complicate the notation. Moreover, we can, e.g., introduce a third sort `Safe`, situated between `Place` and `Marking` for modeling *safe* markings, i.e., markings containing at most one instance of each place. Note that all singletons and 0 are safe markings.

The revised Maude specification for the Petri net $N_1$ of our running example is given in Fig. 3. Note that it is a *rewrite theory*, whose underlying equational theory is a MEL theory. We motivate and define in detail in what follow the formal notion of such generalized rewrite theories.

## 2. Generalized rewrite theories and deduction

In this section we present the foundations of generalized rewrite theories over MEL theories and where operators can have frozen arguments.

The main question is whether it always makes sense to rewrite under any operator. The (*Congruence*) rule of RL certainly allows this, but as anticipated in the Introduction, we report below a few examples suggesting that frozen arguments can help the modeling of many systems, such as, for example, process algebras.

**Example 2.1.** Consider for example a reactive process calculus with (among other operators) a non-deterministic choice operator $+$ specified by SOS rules of the form,

$$\frac{P \to P'}{P + Q \to P'}\ \textit{left choice} \qquad \frac{Q \to Q'}{P + Q \to Q'}\ \textit{right choice}.$$

The rewrite theory $\mathcal{R}$ formalizing such a reactive process calculus will then have two corresponding conditional rules, like

$$[\text{left choice}] : P + Q \to P' \ \texttt{if}\ P \to P' \quad [\text{right choice}] : P + Q \to Q' \ \texttt{if}\ Q \to Q'.$$

Furthermore, the $+$ operator should have both arguments *frozen*, i.e., using the notation in Definition 2.1, $\phi(+) = \{1, 2\}$. The only difference w.r.t. the SOS rules is that now the conditions can be satisfied by sequences of rewrites, instead of one-step rewrites (nevertheless, in this particular case, the consistency with the semantics of non-deterministic choice is preserved).

The non-deterministic choice example is based on an operator whose arguments are all frozen, i.e., + is a *frozen operator*. If we add to this process calculus a sequential composition $P; Q$, the fact that $Q$ should not be able to evolve until $P$ has finished its task can be straightforwardly modeled by only declaring the second argument of _ ; _ as frozen, and adding the unconditional rule

$$[\text{serialization}] : \checkmark; Q \to Q$$

(where $\checkmark$ is the "correct termination" process), which throws away the operator _ ; _ , thus unfreezing its second argument. This way, in the process $P; Q$, the component $P$ is free to evolve in the context _ ; $Q$, because the first position of the _ ; _ operator is not frozen.

The usefulness of having frozen operators in rewrite theories has emerged gradually. Stehr et al. first proposed *frozen kinds* in [38]. The generalization of this to a subset $\Omega \subseteq \Sigma$ of operators where all the arguments are frozen emerged in a series of email exchanges between J.-B. Stefani and the second author. The subsequent generalization of freezing operator arguments selectively brings us to the mentioned two levels (for equations and for rules) of context-sensitive rewriting. Note also that independently, Eric Deplagne has introduced *protective* symbols. As explained in detail in [16] (see also [17]) and sketched in [18], this allows one to control the congruence used in deduction modulo [20].

As a further example concerning the use of frozen operators, consider the following Petri net example that poses the marking cardinality problem and discusses how to solve it.

**Example 2.2.** Suppose we want to extend the Petri net specification with a cardinality operator, returning the number of *tokens* (i.e., instances of resources) present in a marking. One possibility is to extend the (signature of the) rewrite theory in Fig. 3 by adding the operator

```
op |_| : Marking -> Nat
```

and the equations

```
eq |0| = 0 .
ceq |P⊕U| = 1 + |U|     if P:Place ∧ U:Marking .
```

But remember that there are also three rewrite rules defined (`t1`, `t2` and `t3`), hence we can have e.g., the sequence of rewrites $\mathtt{a}\oplus\mathtt{a}\oplus\mathtt{c} \to \mathtt{b}\oplus\mathtt{c} \to \mathtt{d}$, which induces, by the (*Congruence*) rule, the rewrites

```
|a⊕a⊕c| => |b⊕c| => |d|
```

Correspondingly we have that $3 \to 2 \to 1$, i.e., the multiset whose cardinality we wish to determine becomes a *moving target*.

One possibility to eliminate weird situations like these in the multiset cardinality example is to distinguish *data kinds* (that should stay frozen) and *state kinds* (that can naturally evolve), as proposed in [38].

However, frozen kinds are too coarse to deal straightforwardly with the non-deterministic choice example (see Example 2.1): sometimes it is convenient to keep frozen also certain states, not just the data. The obvious generalization is thus to consider *frozen operators*.

Making all arguments frozen is too restrictive. This is not a problem with the cardinality operator |_ |, since there is only one argument, but the more general case of the sequential composition operator given in Example 2.1 shows that the distinction between arguments that are frozen and arguments that are not can be quite useful. Such distinction can be made by specifying the frozen arguments of each operator, which is the most fine-grained solution w.r.t. those mentioned above.

**Definition 2.1** (*Generalized signatures*). A *generalized operator* is a function symbol $f: k_1 \cdots k_n \to k$ together with a set $\phi(f) \subseteq \{1, \ldots, n\}$ of frozen argument positions. We denote by $\nu(f)$ the set $\{1, \ldots, n\} \setminus \phi(f)$ of *unfrozen* arguments, and say that $f$ is *unfrozen* if $\phi(f) = \emptyset$. A *generalized* MEL *signature* $(\Sigma, \phi)$ is a MEL signature $\Sigma$ whose function symbols are generalized operators, i.e., the function $\phi: \Sigma \to \wp_{\mathrm{f}}(\mathbb{N})$ assigns to each operator its set of frozen arguments ($\wp_{\mathrm{f}}(\mathbb{N})$ denotes the set of finite sets of natural numbers and we assume that for any $f: k_1 \cdots k_n \to k$ in $\Sigma$ we have $\phi(f) \subseteq \{1, \ldots, n\}$).

If the $i$th position of $f$ is frozen, then in $f(t_1, \ldots, t_n)$ any subterm of $t_i$ is frozen. This can be made more precise by considering the usual tree-like representation of terms (the same subterm can occur in many distinct positions that

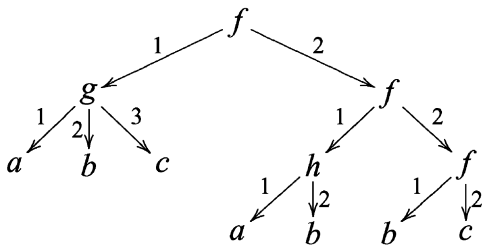


Fig. 4. The tree-like representation of $f(g(a, b, c), f(h(a, b), f(b, c)))$.

are not necessarily all frozen). Positions in a term are denoted by strings of natural numbers, indicating the sequences of branches we must follow from the root to reach that position. For example, the term $t = f(g(a, b, c), f(h(a, b), f(b, c)))$ has two occurrences of the constant $c$ at positions 1.3 and 2.2.2, respectively (see Fig. 4).

We let $t_\pi$ and $t(\pi)$ denote, respectively, the subterm of $t$ occurring at position $\pi$, and the topmost operator in $t_\pi$. For $\varepsilon$ the empty position, we let $t_\varepsilon$ denote the whole term $t$. In the example above, we have $t_{2.1} = h(a, b)$ and $t(2.1) = h$.

**Definition 2.2** (*Frozen occurrences*). The occurrence $t_\pi$ of the subterm of $t$ at position $\pi$ is *frozen* if there exist two positions $\pi_1$, $\pi_2$ and a natural number $n$ such that $\pi = \pi_1.n.\pi_2$ and $n \in \phi(t(\pi_1))$. The occurrence $t_\pi$ is called *unfrozen* if it is not frozen.

In the example above, for $\phi(f) = \phi(g) = \varnothing$ and $\phi(h) = \{1\}$, we have that $t_{2.1.1} = a$ is frozen (because $t(2.1) = h$, whose first argument is frozen), while $t_{1.1} = a$ is unfrozen (because $t(\varepsilon) = f$ and $t(1) = g$, and all the arguments of $f$ and $g$ are unfrozen).

**Definition 2.3** (*Frozen variables*). Given $t \in \mathbb{T}_\Sigma(X)$ we say that the variable $x \in X$ is *frozen* in $t$ if there exists a frozen occurrence of $x$ in $t$, otherwise it is called *unfrozen*.

We let $\phi(t)$ and $v(t)$ denote, respectively, the set of frozen and unfrozen variables of $t$. Analogously, $\phi(t_1, \ldots, t_n)$ (respectively $v(t_1, \ldots, t_n)$) denotes the set of variables for which a frozen occurrence appears in at least one $t_i$ (respectively, that are unfrozen in all $t_i$).

By combining conditional rewrite theories with MEL specifications and allowing frozen arguments, we obtain a rather general notion of rewrite theory.

**Definition 2.4** (*Generalized rewrite theory*). A *generalized rewrite theory* (GRT) is a tuple $\mathcal{R} = (\Sigma, E, \phi, R)$ consisting of: (i) a generalized membership signature $(\Sigma, \phi)$ with, say, kinds $k \in K$, sorts $s \in S$, and with generalized operators $f \in \Sigma$ having frozen arguments according to $\phi$; (ii) a MEL theory $(\Sigma, E)$; (iii) a set $R$ of (universally quantified) labeled conditional rewrite rules $r$ having the general form

$$(\forall X)\ r: t \to t' \text{ if } \bigwedge_{i \in I} p_i = q_i \wedge \bigwedge_{j \in J} w_j : s_j \wedge \bigwedge_{l \in L} t_l \to t'_l \tag{2}$$

where for appropriate kinds $k$ and $k_l$ in $K$, $t, t' \in \mathbb{T}_\Sigma(X)_k$ and $t_l, t'_l \in \mathbb{T}_\Sigma(X)_{k_l}$ for $l \in L$.

The next example illustrates the notion of a GRT.

**Example 2.3.** In the Petri net example discussed so far, the operators with frozen arguments are `|_ |` and `_ in_` (all their arguments must be frozen, i.e., $\phi(|\_\ |) = \{1\}$ and $\phi(\_\ \texttt{in}\_\ ) = \{1, 2\}$). In Maude this is achieved by the declarations:

```
op _ in _ : Place Marking -> Bool [frozen] .
op |_| : Marking -> Nat [frozen] .
```

$$\frac{t \in \mathbb{T}_\Sigma(X)_k}{(\forall X)\, t \to t}\ \textbf{Reflexivity} \qquad \frac{(\forall X)\, t_1 \to t_2, \quad (\forall X)\, t_2 \to t_3}{(\forall X)\, t_1 \to t_3}\ \textbf{Transitivity}$$

$$\frac{E \vdash (\forall X)\, t = u, \quad (\forall X)\, u \to u', \quad E \vdash (\forall X)\, u' = t'}{(\forall X)\, t \to t'}\ \textbf{Equality}$$

$$\frac{\begin{array}{c} f \in \Sigma_{k_1 \cdots k_n, k}, \quad t_i, t_i' \in \mathbb{T}_\Sigma(X)_{k_i} \text{ for } i \in [1,n] \\ t_i' = t_i \text{ for } i \in \phi(f), \quad (\forall X)\, t_j \to t_j' \text{ for } j \in \nu(f) \end{array}}{(\forall X)\, f(t_1, \ldots, t_n) \to f(t_1', \ldots, t_n')}\ \textbf{Congruence}$$

$$\frac{\begin{array}{c} (\forall X)\, r{:}t \to t' \text{ if } \bigwedge_{i \in I} p_i = q_i \wedge \bigwedge_{j \in J} w_j : s_j \wedge \bigwedge_{l \in L} t_l \to t_l' \in R \\ \theta, \theta'{:}X \to \mathbb{T}_\Sigma(Y), \quad \theta(x) = \theta'(x) \text{ for } x \in \phi(t,t') \\ E \vdash (\forall Y)\, \theta(p_i) = \theta(q_i) \text{ for } i \in I, \quad E \vdash (\forall Y)\, \theta(w_j) : s_j \text{ for } j \in J \\ (\forall Y)\, \theta(t_l) \to \theta(t_l') \text{ for } l \in L, \quad (\forall Y)\, \theta(x) \to \theta'(x) \text{ for } x \in \nu(t,t') \end{array}}{(\forall Y)\, \theta(t) \to \theta'(t')}\ \begin{array}{l}\textbf{Nested} \\ \textbf{Replacement}\end{array}$$

Fig. 5. Deduction rules for generalized rewrite theories.

**Remark 2.4.** In general, the keyword `frozen` can be followed by a list of frozen argument positions, grouped within brackets, e.g., for the sequential composition operator discussed in Example 2.1, where only the second argument should be frozen, the corresponding declaration is

```
op _;_ : Process Process -> Process [frozen (2)] .
```

By convention, when no list of argument positions is specified, all argument positions are frozen.

### 2.1. Inference in generalized RL

Given a GRT $\mathcal{R} = (\Sigma, E, \phi, R)$, a *sequent* of $\mathcal{R}$ is a pair of (universally quantified) terms of the same kind $t, t'$, denoted $(\forall X) t \to t'$ with $X = \{x_1 : k_1, \ldots, x_n : k_n\}$ a set of kinded variables and $t, t' \in \mathbb{T}_\Sigma(X)_k$ for some $k$. We say that $\mathcal{R}$ *entails* the sequent $(\forall X)\ t \to t'$, and write $\mathcal{R} \vdash (\forall X)\ t \to t'$, if the sequent $(\forall X)\ t \to t'$ can be obtained by means of the rules in Fig. 5, which are briefly described below.

(*Reflexivity*), (*Transitivity*), and (*Equality*) are the usual rules for idle rewrites, concatenation of (composable) rewrites, and rewriting modulo $E$ (but note that $E$ is here a MEL theory). (*Congruence*) allows rewriting the arguments of a generalized operator, but since this is not desirable for frozen arguments, we add the condition that frozen arguments must stay idle (note that $t_i' = t_i$ is syntactic equality). Any unfrozen argument can still be rewritten, as expressed by the premise $(\forall X)\ t_j \to t_j'$ for $j \in \nu(f)$.

(*Nested Replacement*) is the most complex rule, which takes into account the application of a rewrite rule in its most general form (2). It says that for any rewrite rule $r \in R$ (first premise) and for any (kind-preserving) substitution $\theta$ such that the condition of $r$ is satisfied when $\theta$ is applied to all terms $p_i, q_i, w_j, t_l, t_l'$ involved, then it is possible to apply the rewrite $r$ to $\theta(t)$. Moreover, if $\theta'$ is a second (kind-preserving) substitution for the variables in $X$ such that $\theta$ and $\theta'$ coincide on all frozen variables $x \in \phi(t, t')$ (second line of premises), while the rewrites $(\forall Y)\ \theta(x) \to \theta'(x)$ are provable for the unfrozen variables $x \in \nu(t, t')$ (last premise), then such nested rewrites can be applied concurrently with $r$.

Of course, any unsorted rewrite theory can be regarded as a GRT where: (i) $\Sigma$ has a unique kind and no sorts; (ii) all the operators are total and unfrozen (i.e., $\phi(f) = \varnothing$ for any $f \in \Sigma$); and (iii) conditions in rewrite rules contain neither equalities nor membership predicates. In this case, deduction via rules for conditional rewrite theories (Fig. 1) coincides with deduction via rules for generalized rewrite theories (Fig. 5).

**Theorem 2.1.** *Let $\mathcal{R}$ be a conditional rewrite theory, and let $\hat{\mathcal{R}}$ denote its corresponding* GRT. *Then*:

$$\mathcal{R} \vdash (\forall X)\ t \to t' \quad \Leftrightarrow \quad \hat{\mathcal{R}} \vdash (\forall X)\ t \to t'.$$

**Proof** (*Sketch*). The result can be easily proved by noting that the only differences between the two sets of deduction rules in Figs. 1 and 5 are given by the premises of rules (*Reflexivity*), (*Congruence*) and (*Nested Replacement*). But for the special case of $\hat{\mathcal{R}}$ these differences become immaterial, as detailed below

- (*Reflexivity*): there is a unique kind and all operators are total, hence the two premises are equivalent.
- (*Congruence*): there is a unique kind, all operators are total, and there are no frozen positions (i.e., for all $f$ with arity $n$ we have $\phi(f) = \emptyset$ and $v(f) = \{1, \ldots, n\}$), hence the two premises are equivalent.
- (*Nested Replacement*): all rewrite rules $r$ have neither equations nor memberships in their conditions and there are no frozen positions (i.e., $\phi(t, t') = \emptyset$), hence the two premises are equivalent. □

## 3. Models of generalized rewrite theories

In this section, exploiting MEL, we define the reachability and concurrent model theories of GRTs and prove completeness results.

### 3.1. Reachability models

As usual, reachability models focus just on *what* terms/states can be reached from a certain state $t$ via sequences of rewrites, ignoring *how* the rewrites can lead to them.

**Definition 3.1** (*Reachability relation*). Given a GRT $\mathcal{R} = (\Sigma, E, \phi, R)$, its $\mathcal{R}$-reachability relation $\to_{\mathcal{R}}$ is defined proof-theoretically, for each kind $k$ in $\Sigma$ and each $[t], [t'] \in \mathbb{T}_{\Sigma,E}(X)_k$, by the equivalence:

$$[t] \to_{\mathcal{R}} [t'] \quad \Leftrightarrow \quad \mathcal{R} \vdash (\forall X)\ t \longrightarrow t'.$$

The above definition is sound because we have the following easy lemma.

**Lemma 3.1.** *Let $\mathcal{R} = (\Sigma, E, \phi, R)$ be a* GRT, *and $t \in \mathbb{T}_\Sigma(X)_k$. If $\mathcal{R} \vdash (\forall X)\ t \longrightarrow t'$, then $t' \in \mathbb{T}_\Sigma(X)_k$. Moreover, for any $t, u, u', t' \in \mathbb{T}_\Sigma(X)_k$ such that $u \in [t]_E$, $u' \in [t']_E$ and $\mathcal{R} \vdash (\forall X)\ u \longrightarrow u'$, then $\mathcal{R} \vdash (\forall X)\ t \longrightarrow t'$.*

**Proof.** The first part of the lemma follows by a simple proof induction on the rules in Fig. 5: it is easy to see that the kind of a term is preserved as an invariant by all (provable) rewrites.

The second part of the lemma follows by a direct application of (*Equality*), which states that deduction is independent of the choice of the elements in the equivalence classes $[t]_E$ and $[t']_E$. □

The reachability relation admits a model-theoretic presentation in terms of the free models of a suitable MEL theory. We give the details below as a "warm up" for the model-theoretic concurrent semantics given in Section 3.2. The idea is that $\to_{\mathcal{R}}$ can be defined as the family of relations, indexed by the kinds $k$, given by interpreting the sorts $Ar_k$ in the free model of the following MEL theory $Reach(\mathcal{R})$.

**Definition 3.2** (*The theory $Reach(\mathcal{R})$*). The MEL theory $Reach(\mathcal{R})$ contains the signature and sentences in $(\Sigma, E)$ together with the following extensions:

(1) For each kind $k$ in $\Sigma$ we add:
   (a) a new kind $[Pair_k]$ (for $k$-indexed binary relations on terms of kind $k$) with four sorts $Ar_k^0$, $Ar_k^1$, $Ar_k$ and $Pair_k$ and subsort inclusions: $Ar_k^0\ Ar_k^1 < Ar_k < Pair_k$;
   (b) the operators $(\_ \to \_) : k\ k \longrightarrow Pair_k$ (pair constructor), $\mathtt{s}, \mathtt{t} : Pair_k \longrightarrow k$ (source and target projections), and $(\_\ ;\ \_) : [Pair_k]\ [Pair_k] \longrightarrow [Pair_k]$ (concatenation);

(c) the (conditional) equations and memberships

$$\begin{aligned}
(\forall x, y : k)\ \mathtt{s}(x \to y) &= x\\
(\forall x, y : k)\ \mathtt{t}(x \to y) &= y\\
(\forall z : Pair_k)\ (\mathtt{s}(z) \to \mathtt{t}(z)) &= z\\
(\forall x : k)\ (x \to x) &: Ar_k^0\\
(\forall x, y, z : k)\ (x \to z) &: Ar_k \text{ if } (x \to y) : Ar_k \wedge (y \to z) : Ar_k\\
(\forall x, y, z : k)\ (x \to y); (y \to z) &= (x \to z).
\end{aligned}$$

(2) Each $f : k_1 \dots k_n \longrightarrow k$ in $\Sigma$ with $\nu(f) \neq \emptyset$ is lifted to $f : [Pair_{k_1}] \cdots [Pair_{k_n}] \longrightarrow [Pair_k]$, and for each $i \in \nu(f)$ we declare $f : Ar^0_{k_1} \cdots Ar^1_{k_i} \cdots Ar^0_{k_n} \longrightarrow Ar^1_k$; we then give, for each $i \in \nu(f)$ and letting $X_i = \{x_1 : k_1, \dots, x_n : k_n, y_i : k_i\}$, an equation

$$(\forall X_i)\ f((x_1 \to x_1), \dots, (x_i \to y_i), \dots, (x_n \to x_n)) = f(x_1, \dots, x_n) \to f(x_1, \dots, y_i, \dots, x_n).$$

(3) For each rule in $R$ of the form

$$(\forall X)\ r : t \to t' \text{ if } \bigwedge_{i \in I} p_i = q_i \ \wedge \ \bigwedge_{j \in J} w_j : s_j \ \wedge \ \bigwedge_{l \in L} t_l \to t'_l$$

with, say $t, t'$ of kind $k$, and $t_l, t'_l$ of kind $k_l$, we give the conditional membership,

$$(\forall X)\ (t \to t') : Ar^1_k \text{ if } \bigwedge_{i \in I} p_i = q_i \ \wedge \ \bigwedge_{j \in J} w_j : s_j \ \wedge \ \bigwedge_{l \in L} t_l \to t'_l : Ar_{k_l}.$$

The sorts $Ar^0_k$ and $Ar^1_k$ contain, respectively, idle rewrites and one-step rewrites of $k$-kinded terms, while the sort $Ar_k$ contains rewrites of arbitrary length for such terms. The (*Congruence*) rule is modeled so that exactly one unfrozen argument can be rewritten in one-step (see item 2 in Definition 3.2), and (*Nested Replacement*) is restricted so that no nested rewrites can take place concurrently (item 3). Nevertheless, these two restrictions on how the inference rules are modeled do not alter the reachability relation $Ar_k$, because one-step rewrites can be composed in any admissible interleaved fashion (see the fifth axiom at point 1(c)). Note that the introduction of the concatenation operator $\_ ; \_$ is not really necessary, but its presence is helpful in the proof of Theorem 3.4.

The theory $Reach(\mathcal{R})$ provides an algebraic model for the reachability relation. For ground terms, such a model is given by the interpretation of the sorts $Ar_k$ in the initial model $\mathbb{T}_{Reach(\mathcal{R})}$. For terms with variables in $X$, the reachability model is the free algebra $\mathbb{T}_{Reach(\mathcal{R})}(X)$. This can be summarized by the following theorem, whose proof is given in Appendix B.1.

**Theorem 3.2** (*Completeness of reachability semantics*). *For $\mathcal{R} = (\Sigma, E, \phi, R)$ a* GRT *and $t, t' \in T_\Sigma(X)_k$ we have that all the following statements are equivalent*:
(1) $\mathcal{R} \vdash (\forall X)\ t \to t'$;
(2) $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k$;
(3) $Reach(\mathcal{R}) \models (\forall X)\ (t \to t') : Ar_k$;
(4) $[(t \to t')] \in \mathbb{T}_{Reach(\mathcal{R})}(X)_{Ar_k}$.

**Proof** (*Sketch*). The proof is presented in four parts:
- First, we show that (1) $\Rightarrow$ (2). This implication is proved by rule induction over the proof of $\mathcal{R} \vdash (\forall X)\ t \to t'$, as detailed in Appendix B.1. The proof also requires Lemmas B.1–B.3.
- In the second part we show that (1) $\Leftarrow$ (2). The proof goes by straightforward induction on the proof of $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k$.
- Third, we prove that (2) $\Leftrightarrow$ (3). This result follows by completeness of MEL [36].
- Finally, we show that (3) $\Leftrightarrow$ (4) (by the completeness of MEL [36] and by the existence of initial and free models for MEL theories).

Therefore, by transitivity of logical implication, we conclude that the four statements are all equivalent. □

We conclude this section by showing a few examples of deduction in the theory of reachability models derived from the running example of Petri nets.

**Example 3.1.** Fig. 6 reports (the relevant part of) the MEL theory $Reach(\mathcal{R}_{N_1})$ associated to the generalized rewrite theory $\mathcal{R}_{N_1}$ in Fig. 3 (also extended with the cardinality operator). We recall that both the cardinality and the `_ in_` operators are frozen. For simplicity, the translation of functional modules [2] defining data and operations for the sorts `Nat` and `Bool` is omitted. We write `Ar0` for $Ar^0_{[\texttt{Marking}]}$, and similarly for `Ar1`, `Ar` and `Pair`. We remark that $Reach(\mathcal{R}_{N_1})$ is a Maude functional module, not a system module like $\mathcal{R}_{N_1}$, because $Reach(\mathcal{R}_{N_1})$ contains no rewrite rules. Thus, deduction in $Reach(\mathcal{R}_{N_1})$ is defined by the ordinary inference rules of MEL, namely reflexivity, symmetry, transitivity, congruence, membership, and modus ponens (see [36]). Moreover, Theorem 3.2 guarantees that the marking $v$ is reachable from the initial marking $u$ if and only if the membership sentence $(\forall\emptyset)\ u \to v : Ar$ can be proved in $Reach(\mathcal{R}_{N_1})$.

Note that, the three rewrite rules of $\mathcal{R}_{N_1}$ become membership assertions, populating the sort `Ar1` (of one-step rewrites). Similarly, each marking $u$ generates an idle rewrite (sort `Ar0`), thanks to the membership axiom

```
mb (U --> U) : Ar0 .
```

Subsorting allows us to infer that both idle and one-step rewrites are valid rewrites (sort `Ar`). Since both positions of the multiset union operator are unfrozen, $\_ \oplus \_$ can be applied to rewrites, composing them in *parallel*. Note also that the arguments of frozen operators are no longer moving targets, since only equational simplification can take place for them. The concatenation operator can be used instead to compose any two rewrites such that the target of the first matches the source of the second.

We conclude by listing several deductions that can be straightforwardly proved:

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{b} \to \mathtt{a} : \mathtt{Ar} \quad \text{(firing of } t_3\text{)} \tag{3}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{c} \to \mathtt{a}\oplus\mathtt{c} : \mathtt{Ar} \quad \text{(idle step)} \tag{4}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{c} : \mathtt{Ar} \quad \text{((3), (4) in parallel)} \tag{5}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a} \to \mathtt{b} : \mathtt{Ar} \quad \text{(firing of } t_1\text{)} \tag{6}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{c} \to \mathtt{c} : \mathtt{Ar} \quad \text{(idle step)} \tag{7}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{c} \to \mathtt{b}\oplus\mathtt{c} : \mathtt{Ar} \quad \text{((6), (7) in parallel)} \tag{8}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{b}\oplus\mathtt{c} : \mathtt{Ar} \quad \text{((5), (8) in sequence)} \tag{9}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{b}\oplus\mathtt{c} \to \mathtt{d} : \mathtt{Ar} \quad \text{(firing of } t_2\text{)} \tag{10}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{d} : \mathtt{Ar} \quad \text{((9), (10) in sequence)} \tag{11}$$

Similarly, it can be proved that:

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{d} : \mathtt{Ar}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{b}\oplus\mathtt{b}\oplus\mathtt{c} : \mathtt{Ar}$$

$$Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{b}\oplus\mathtt{d} : \mathtt{Ar}$$

On the other hand, while $Reach(\mathcal{R}_{N_1}) \vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{a} : \mathtt{Pair}$, we have that $Reach(\mathcal{R}_{N_1}) \not\vdash (\forall\emptyset)\ \mathtt{a}\oplus\mathtt{a}\oplus\mathtt{b}\oplus\mathtt{c} \to \mathtt{a} : \mathtt{Ar}$.

---

[2] As already mentioned, Maude has a purely functional sublanguage whose modules are MEL theories, called *functional modules*. They are introduced with the keyword `fmod` and ended with `endfm`. Since the reachability theory is an MEL theory, it is specified in Maude as the functional module `Reach_N1` which here imports the functional submodules `NAT` and `BOOL`.

```
fmod Reach_N1 is including BOOL .     including NAT .
     sorts Place Safe Marking .
  subsorts Place < Safe < Marking .
       ops a b c d : -> Place .
        op 0 : -> Safe .
        op _⊕_ : Marking Marking -> Marking [assoc comm id: 0] .
        op _ in _ : Place Marking -> Bool .
        op |_| : Marking -> Nat .
      vars P Q : Place .
       var S : Safe .
       var U : Marking .
       cmb P⊕S : Safe  if (P in S)=false .
        eq P in 0 = false .
       ceq P in Q = true  if P=Q .
       ceq P in Q = false  if P=/=Q .
       ceq (P in Q⊕S) = true  if P=Q .
       ceq (P in Q⊕S) = (P in S)  if P=/=Q .
        eq |0| = 0 .
        eq |P⊕U| = 1 + |U| .

     sorts Ar0 Ar1 Ar Pair .
  subsorts Ar0 Ar1 < Ar < Pair .
        op _ --> _ : Marking Marking -> Pair .
       ops s t : Pair -> Marking .
        op _ ; _ : [Pair] [Pair] -> [Pair] .
      vars V W : Marking .
       var R : Pair .
        eq s(U --> V) = U .
        eq t(U --> V) = V .
        eq s(R) --> t(R) = R .
        mb (U --> U) : Ar0 .
       cmb (U --> V) : Ar  if (U --> W):Ar ∧ (W --> V):Ar .
        eq (U --> W);(W --> V) = (U --> V) .

        op _⊕_ : Pair Pair -> Pair [assoc comm id: 0] .
        op _⊕_ : Ar1 Ar0 -> Ar1 .
        op _⊕_ : Ar0 Ar1 -> Ar1 .
        eq (U --> V)⊕(W --> W) = (U⊕W) --> (V⊕W) .
        eq (W --> W)⊕(U --> V) = (W⊕U) --> (W⊕V) .

        mb ((a⊕a) --> b) : Ar1 .
        mb ((b⊕c) --> d) : Ar1 .
        mb ((a⊕b) --> a) : Ar1 .
endfm
```

Fig. 6. The MEL theory $Reach(\mathcal{R}_{N_1})$.

### 3.2. Concurrent models

In general, given $t, t' \in \mathbb{T}_\Sigma(X)_k$, many proofs concluding that $\mathcal{R} \vdash (\forall X)t \to t'$ are possible, where we can think of each proof as describing a *concurrent computation* beginning at $t$ and ending at $t'$. Note that: (1) some of the proofs can be *computationally equivalent*, because they represent different interleaved sequences for the same concurrent computation, e.g., the concurrent firing of two transitions in a Petri net, but (2) not all those proofs are necessarily equivalent, as they may, e.g., differ in the underlying set of applied rewrite rules, or in the different causal connections between the applications of the same rules. In this section, we show how to extend the notion of decorated sequents sketched in Section 1.1 to GRTs, so as to define an algebraic model of *true concurrency*[3] for $\mathcal{R}$.

[3] As discussed in detail in Section 4.1 of [30], the algebraic model of true concurrency that we are about to define in full generality includes as special cases many other true concurrency models originally defined in different ways, yet equivalent to an instance of ours. Such models include: residual models for term rewriting, parallel lambda calculus models, process models for Petri nets, proved transition models for CCS, and partial order of events models for Actors.

**Example 3.2.** In $Reach(\mathcal{R}_{N_1})$ there are at least three different ways of proving $Reach(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c} \rightarrow \mathtt{b} \oplus \mathtt{d} : \mathtt{Ar}$ but all of them express the concurrent firing of transitions $t_1$ and $t_2$ and therefore are intuitively equivalent.

But let us consider a more complex case. We have seen that $Reach(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c} \rightarrow \mathtt{d} : \mathtt{Ar}$, and any proof imposes the following order on firings: first $t_3$, then $t_1$ and last $t_2$. By adding an idle token on place $a$, we have of course that $Reach(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c} \rightarrow \mathtt{a} \oplus \mathtt{d} : \mathtt{Ar}$. However, now the same conclusion can be drawn in many different ways, with $t_1$, $t_2$ and $t_3$ applied in different orders. The question is whether all these proofs are equivalent or not. According to the theory of (commutative processes in) Petri nets and given the Curry–Howard-like analogy between logical deduction and computation, the answer to this question should be yes (all the runs leading from $\mathtt{a} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c}$ to $\mathtt{a} \oplus \mathtt{d}$ yield the same commutative process), but which equational theory do we need for proving it? Moreover, how can we filter out undesired rewrites of frozen terms from the model theory?

As usual, decorated sequents are first defined by attaching a *proof term* (i.e., an expression built from variables, operators in $\Sigma$, and rule labels in $R$) to each sequent, and then by quotienting out proof terms modulo suitable functoriality, decomposition, and exchange laws, which capture basic algebraic laws of true concurrency. We can present $\mathcal{R}$'s algebra of sequents as the initial (or free) algebra of a suitable MEL theory $Proof(\mathcal{R})$. With respect to the classical presentation via decorated deduction rules, the MEL specification allows a standard algebraic definition of initial and loose semantics. Moreover, here we can naturally support many-sorted, order-sorted, and MEL data theories instead of just unsorted equational theories as in [34].

The construction of $Proof(\mathcal{R})$ is analogous to that of $Reach(\mathcal{R})$. The kind $[Pair_k]$ of $Reach(\mathcal{R})$ is replaced here by a kind $[Rw_k]$, whose elements include the proofs of concurrent computations. The initial and final states are still defined by means of the source ($\mathtt{s}$) and target ($\mathtt{t}$) operators. Moreover, since the proof of an idle rewrite $[t] \rightarrow [t]$ is $[t]$ itself, we can exploit subsorting to make $k$ a sort of kind $[Rw_k]$, instead of adding a new sort $Ar_k^0$ as done in $Reach(\mathcal{R})$. The sorts $Rw_k^1$ and $Rw_k$ are the analogous of $Ar_k^1$ and $Ar_k$. The sort $Ar_k^1$ was introduced in $Reach(\mathcal{R})$ to deal with the "restricted" form of (*Congruence*) and (*Nested Replacement*). Having decorations at hand, we can restore the full expressiveness of the two inference rules, but the sort $Rw_k^1$ is still useful in axiomatizing proof-decorated sequents: we define the (*Equality*) rule on $Rw_k^1$, lifting the axioms in $E$ to one-step rewrites, and then exploit functoriality and transitivity to extend the equational theory $E$ from $\Sigma$-terms and one-step rewrites to rewrites of arbitrary length in $Rw_k$.

In the next definition (item 3) the notation "$k_1 \cdots Rw_{k_{i_1}} \cdots Rw_{k_{i_m}} \cdots k_n$" denotes the sequence of sorts obtained by replacing in $k_1 \cdots k_n$ each $k_{i_j}$ with $Rw_{k_{i_j}}$ for each $i_j \in \{i_1, \ldots, i_m\}$. Similarly, the notation "$k_1 \cdots Rw^1_{k_{i_j}} \cdots k_n$" denotes the sequence of sorts obtained by replacing in $k_1 \cdots k_n$ the symbol $k_{i_j}$ with $Rw^1_{k_{i_j}}$ for just a particular index $i_j$.

**Definition 3.3** (*The theory Proof* $(\mathcal{R})$). The MEL theory $Proof(\mathcal{R})$ contains the signature and sentences of $(\Sigma, E)$ together with the following extensions:

(1) Each kind $k$ in $\Sigma$ becomes a sort $k$ in $Proof(\mathcal{R})$, with $s < k$ for any $s \in S_k$ in $\Sigma$.
(2) For each kind $k$ in $\Sigma$ we add:
   (a) a new kind $[Rw_k]$ (for $k$-indexed decorated rewrites on $\Sigma$-terms of kind $k$) with sorts all sorts in $k$, $k$ itself, and the (new) sorts $Rw_k^1$ and $Rw_k$, with: $k < Rw_k$ and $Rw_k^1 < Rw_k$;
   (b) the overloaded operators $\_\,;\,\_ : [Rw_k]\ [Rw_k] \longrightarrow [Rw_k]$ and $\mathtt{s}, \mathtt{t} : Rw_k \longrightarrow k$;
   (c) the (conditional) equations and memberships

$$(\forall x : k)\ \mathtt{s}(x) = x$$

$$(\forall x : k)\ \mathtt{t}(x) = x$$

$$(\forall x, y : Rw_k)\ x; y : Rw_k \text{ if } \mathtt{t}(x) = \mathtt{s}(y)$$

$$(\forall x, y : Rw_k)\ \mathtt{s}(x; y) = \mathtt{s}(x) \text{ if } \mathtt{t}(x) = \mathtt{s}(y)$$

$$(\forall x, y : Rw_k)\ \mathtt{t}(x; y) = \mathtt{t}(y) \text{ if } \mathtt{t}(x) = \mathtt{s}(y)$$

$$(\forall x : k, y : Rw_k)\ x; y = y \text{ if } x = \mathtt{s}(y)$$

$$(\forall x : Rw_k, y : k)\ x; y = x \text{ if } \mathtt{t}(x) = y$$

$$(\forall x, y, z : Rw_k)\ x; (y; z) = (x; y); z \text{ if } \mathtt{t}(x) = \mathtt{s}(y) \ \wedge\ \mathtt{t}(y) = \mathtt{s}(z).$$

(3) We lift each operator $f : k_1 \dots k_n \longrightarrow k$ in $\Sigma$ to $f : [Rw_{k_1}] \cdots [Rw_{k_n}] \longrightarrow [Rw_k]$, and for $v(f) = \{i_1, \dots, i_m\}$ we overload $f$ by $f : k_1 \cdots Rw_{k_{i_1}} \cdots Rw_{k_{i_m}} \cdots k_n \longrightarrow Rw_k$ and $f : k_1 \cdots Rw^1_{k_{i_j}} \cdots k_n \longrightarrow Rw^1_k$, for $j = 1, \dots, m$, with equations

$$(\forall X)\ \mathtt{s}(f(x_1, \dots, x_n)) = f(\mathtt{s}(x_1), \dots, \mathtt{s}(x_n))$$

$$(\forall X)\ \mathtt{t}(f(x_1, \dots, x_n)) = f(\mathtt{t}(x_1), \dots, \mathtt{t}(x_n)),$$

where $X = \{x_1 : k_1, \dots, x_{i_1} : Rw_{k_{i_1}}, \dots, x_{i_m} : Rw_{k_{i_m}}, \dots, x_n : k_n\}$ and the equation

$$(\forall Y)\ f(x_1, \dots, (x_{i_1}; y_{i_1}), \dots, (x_{i_m}; y_{i_m}), \dots, x_n) = f(x_1, \dots, x_n); f(x_1, \dots, y_{i_1}, \dots, y_{i_m}, \dots, x_n) \text{ if } \bigwedge_{1 \leqslant j \leqslant m} \mathtt{t}(x_{i_j}) = \mathtt{s}(y_{i_j}),$$

where $Y = \{x_1 : k_1, \dots, x_{i_1}, y_{i_1} : Rw_{k_{i_1}}, \dots, x_{i_m}, y_{i_m} : Rw_{k_{i_m}}, \dots, x_n : k_n\}$.

(4) For each equation

$$(\forall x_1 : k_1, \dots, x_n : k_n)\ t = t' \text{ if } \bigwedge_{i \in I} p_i = q_i \ \wedge\ \bigwedge_{j \in J} w_j : s_j$$

in $E$, we let $X = \{x_1 : Rw_{k_1}, \dots, x_n : Rw_{k_n}\}$ and add the conditional equation

$$(\forall X)\ t = t' \text{ if } \bigwedge_{i \in I} p_i = q_i \ \wedge\ \bigwedge_{j \in J} \mathtt{s}(w_j) : s_j \ \wedge\ \bigwedge_{j \in J} \mathtt{t}(w_j) : s_j \wedge \bigwedge_{x_h \in \phi(t,t')} x_h : k_h \wedge \bigwedge_{x_h \in v(t,t')} x_h : Rw^1_{k_h}.$$

(5) For each rule

$$(\forall X)\ r : t \to t' \text{ if } \bigwedge_{i \in I} p_i = q_i \ \wedge\ \bigwedge_{j \in J} w_j : s_j \ \wedge\ \bigwedge_{l \in L} t_l \to t'_l$$

in $R$, with, say, $X = \{x_1 : k_1, \dots, x_n : k_n\}$, $t, t'$ of kind $k$, and $t_l, t'_l$ of kind $k'_l$ with $L = \{1, \dots, \ell\}$, we add the operator $r\colon [Rw_{k_1}] \cdots [Rw_{k_n}][Rw_{k'_1}] \cdots [Rw_{k'_\ell}] \to [Rw_k]$ with

(a) the conditional membership for characterizing basic one-step rewrites:

$$(\forall x_1 : k_1, \dots, x_n : k_n, y_1 : Rw_{k'_1}, \dots, y_\ell : Rw_{k'_\ell})\ r(\vec{x}, \vec{y}) : Rw^1_k \text{ if } \Delta$$

where

$$\Delta = \left( \bigwedge_{i \in I} p_i = q_i \ \wedge\ \bigwedge_{j \in J} w_j : s_j \ \wedge\ \bigwedge_{l \in L} \mathtt{s}(y_l) = t_l \ \wedge\ \bigwedge_{l \in L} \mathtt{t}(y_l) = t'_l \right)$$

checks that the conditions for the application of the rule $r$ are satisfied;

(b) the conditional equations and memberships

$$(\forall Y)\ r(\vec{z}, \vec{y}) : Rw_k \text{ if } \Delta \wedge \Psi$$

$$(\forall Y)\ \mathtt{s}(r(\vec{z}, \vec{y})) = t \text{ if } \Delta \wedge \Psi$$

$$(\forall Y)\ \mathtt{t}(r(\vec{z}, \vec{y})) = t'[\mathtt{t}(\vec{z})/\vec{x}] \text{ if } \Delta \wedge \Psi$$

where $Y = \{x_1 : k_1, \dots, x_n : k_n, z_1 : Rw_{k_1}, \dots, z_n : Rw_{k_n}, y_1 : Rw_{k'_1}, \dots, y_\ell : Rw_{k'_\ell}\}$, $\Delta$ is as before, and

$$\Psi = \left( \bigwedge_{x_h \in \phi(t,t')} z_h = x_h \ \wedge\ \bigwedge_{x_h \in v(t,t')} \mathtt{s}(z_h) = x_h \right)$$

relates $\vec{z}$ and $\vec{x}$;

(c) the decomposition law

$$(\forall Z)\ r(\vec{z},\vec{y}) = r(\vec{x},\vec{y}); t'[\vec{z}/\vec{x}] \text{ if } \Delta \wedge \Psi$$

where $Z = \{x_1 : k_1, \ldots, x_n : k_n, z_1 : Rw_{k_1}, \ldots, z_n : Rw_{k_n}, y_1 : Rw_{k'_1}, \ldots, y_\ell : Rw_{k'_\ell}\}$, while $\Delta$ and $\Psi$ are as before;

(d) the exchange law

$$(\forall W)\ r(\vec{x},\vec{y}); t'[\vec{z}/\vec{x}] = t[\vec{z}/\vec{x}]; r(\mathtt{t}(\vec{z}),\vec{y'}) \text{ if } \Delta \wedge \Psi \wedge \Delta' \wedge \Phi$$

where $W = \{x_1 : k_1, \ldots, x_n : k_n, z_1 : Rw^1_{k_1}, \ldots, z_n : Rw^1_{k_n}, y_1 : Rw_{k'_1}, \ldots, y_\ell : Rw_{k'_\ell}, y'_1 : Rw_{k'_1}, \ldots, y'_\ell : Rw_{k'_\ell}\}$, $\Delta$ and $\Psi$ are as before,

$$\Delta' = \left( \bigwedge_{i\in I} p_i[\mathtt{t}(\vec{z})/\vec{x}] = q_i[\mathtt{t}(\vec{z})/\vec{x}] \ \wedge \bigwedge_{j\in J} w_j[\mathtt{t}(\vec{z})/\vec{x}] : s_j \right.$$
$$\left. \wedge \bigwedge_{l\in L} \mathtt{s}(y'_l) = t_l[\mathtt{t}(\vec{z})/\vec{x}] \ \wedge \bigwedge_{l\in L} \mathtt{t}(y'_l) = t'_l[\mathtt{t}(\vec{z})/\vec{x}] \right)$$

checks that the conditions for the application of the rule $r$ are satisfied after applying the rewrites $\vec{z}$ to the arguments of $t$, and the condition

$$\Phi = \left( \bigwedge_{l\in L} y_l; t'_l[\mathtt{t}(\vec{z})/\vec{x}] = t_l[\mathtt{t}(\vec{z})/\vec{x}]; y'_l \right)$$

states the correspondence between the "side" rewrites $\vec{y}$ and $\vec{y'}$ (via $\vec{z}$).

We briefly comment on the definition of $Proof(\mathcal{R})$.

The operators defined in 2(b) are the obvious source/target projections and sequential composition of rewrites, with the axioms stating that, for each $k$, the rewrites in $Rw_k$ are the arrows of a category with objects in $k$.

The operators $f \in \Sigma$ are lifted to functors over rewrites in 3. It is worth noting that when $f \in \Sigma$ is lifted, only unfrozen positions can have rewrites as arguments, and therefore the functoriality is stated w.r.t. unfrozen positions only.

The equations in $E$ are extended to rewrites in 4. Note that the axioms in $E$ are extended to one-step rewrites only (in unfrozen positions), hence, they hold for sequences of rewrites if and only if they can be proved to hold for each rewrite step.

Point 5(a) defines the basic one-step rewrites, i.e., where no rewrite occurs in the arguments $\vec{x}$. Point 5(b) accounts for nested rewrites $\vec{z}$ below $r$, provided that the side-conditions of $r$ are satisfied by the initial state; in particular, note that the expression $r(\vec{z},\vec{y})$ is always equivalent to $r(\vec{x},\vec{y}); t'[\vec{z}/\vec{x}]$ (see decomposition law), where first $r$ is applied at the top of the term and then the arguments are rewritten according to $\vec{z}$ under $t'$.

Finally, the exchange law states that, under suitable hypotheses, the arguments $\vec{x}$ can be equivalently rewritten first, and the rewrite rule $r$ applied later. Note that, as in the equations extending $E$, the exchange law is stated for one-step nested rewrites only. Nevertheless, it can be used in conjunction with the decomposition law to prove the exchange law for arbitrary long sequences of rewrites (provided that it can be applied step-by-step).

An important property for $Proof(\mathcal{R})$ is that it conservatively extends the underlying theory $(\Sigma, E)$ axiomatizing the states. Otherwise, the additional axiom in $Proof(\mathcal{R})$ could collapse state terms that are different in $(\Sigma, E)$. In this regard, the fact of adding the sorts $Rw^1_k$ and $Rw_k$ on top of $k$ is a potential source of term collapses. However, we can prove that, for any GRT $\mathcal{R}$, the theory $Proof(\mathcal{R})$ is a conservative extension of the underlying theory $E$ (the proof is given in Appendix B.2).

**Proposition 3.3.** *Let $\mathcal{R} = (\Sigma, E, \phi, R)$ be a* GRT, *and let $t, t' \in T_\Sigma(X)_k$, and $s \in S_k$ for some kind $k$. Then, for any formula $\varphi$ of the form $t : k$ or $t : s$ or $t = t'$ we have that*

$$E \vdash (\forall X)\ \varphi \quad \Leftrightarrow \quad Proof(\mathcal{R}) \vdash (\forall X)\ \varphi.$$

The main result is that $Proof(\mathcal{R})$ is complete w.r.t. the inference rules in Fig. 5 (the proof is given in Appendix B.3).

**Theorem 3.4** (*Completeness I*). *For any* GRT $\mathcal{R} = (\Sigma, E, \phi, R)$ *and any* $t, t' \in T_\Sigma(X)_k$, *we have*

$$\mathcal{R} \vdash (\forall X)\ t \to t' \quad \Leftrightarrow \quad \exists \alpha.\ \begin{cases} Proof(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \wedge \\ Proof(\mathcal{R}) \vdash (\forall X)\ \mathtt{s}(\alpha) = t \wedge \\ Proof(\mathcal{R}) \vdash (\forall X)\ \mathtt{t}(\alpha) = t'. \end{cases}$$

The relevance of the MEL theory $Proof(\mathcal{R})$ is far beyond the essence of reachability, as it precisely characterizes the true concurrency laws satisfied by the class of computational models of $\mathcal{R}$.

**Definition 3.4** (*Concurrent models of* $\mathcal{R}$). Let $\mathcal{R}$ be a GRT. A *concurrent model* of $\mathcal{R}$ is a $Proof(\mathcal{R})$-algebra.

Since $Proof(\mathcal{R})$ is an ordinary MEL theory, it admits initial and free models [36]. Hence, the completeness result can be further consolidated by stating the equivalence between formulae provable in $Proof(\mathcal{R})$ using MEL deduction rules, formulae holding for all concurrent models of $\mathcal{R}$ (i.e., valid in all $Proof(\mathcal{R})$-algebras) and formulae valid in the initial/free concurrent model.

**Theorem 3.5** (*Completeness II*). *For* $\mathcal{R}$ *a* GRT *and for any* MEL *sentence* $\varphi$ *over* $Proof(\mathcal{R})$ (*and thus, for* $\varphi$ *any of the formulae* $\alpha : Rw_k$, $\mathtt{s}(\alpha) = t$, $\mathtt{t}(\alpha) = t'$), *we have*

$$Proof(\mathcal{R}) \vdash (\forall X)\ \varphi \quad \Leftrightarrow \quad Proof(\mathcal{R}) \models (\forall X)\ \varphi \quad \Leftrightarrow \quad \mathbb{T}_{Proof(\mathcal{R})}(X) \models (\forall X)\ \varphi.$$

Theorem 3.5 can be combined with Theorems 3.4 and 3.2 to state a more general completeness result for $Proof(\mathcal{R})$, showing the equivalence between deduction at the level of generalized rewrite theories, their (initial and free) reachability models, and their (initial and free) concurrent models.

By Theorem 2.1 we also have that the specialized versions of all results for GRT over unsorted (total) equational theories without frozen arguments and without equality and membership conditions in rewrite rules coincide with the classical ones. In particular, for $\mathcal{R}$ an ordinary rewrite theory and $\hat{\mathcal{R}}$ its corresponding GRT, any $\mathcal{R}$-system can be regarded as a concurrent model of $\hat{\mathcal{R}}$, since it is possible to prove the existence of a forgetful functor $\mathbf{M}_\mathcal{R}$ from the category of $Proof(\hat{\mathcal{R}})$-algebras to the category of $\mathcal{R}$-systems. Noticeably, the functor $\mathbf{M}_\mathcal{R}$ preserves initial and free models.

**Example 3.3.** Fig. 7 reports (the relevant part of) the MEL theory $Proof(\mathcal{R}_{N_1})$ associated to the generalized rewrite theory $\mathcal{R}_{N_1}$ of our running example. For simplicity, we have omitted the equational specification $(\Sigma_{N_1}, E_{N_1})$ which is the same as in $Reach(\mathcal{R}_{N_1})$ and $\mathcal{R}_{N_1}$, as well as the translation of the functional modules specifying the sorts `Nat` and `Bool`. We write `Rw1` for $Rw^1_{[\mathtt{Marking}]}$, and similarly for `Rw`. Since all the operations defined on sort `Marking` in $(\Sigma_{N_1}, E_{N_1})$ are total, we avoid adding a new sort for modeling the kind [`Marking`] of $\mathcal{R}_{N_1}$ and simply keep `Marking`. We also leave implicit the declaration of operators at the kind level when they can be inferred from declarations at the sort level.

Note also that $Proof(\mathcal{R}_{N_1})$ is a Maude functional module, not a system module. The rules associated to transitions are now suitable constants, whose sources and targets are defined accordingly to their pre- and post-sets.

To convince the reader that rewrites of frozen arguments are avoided in $Proof(\mathcal{R}_{N_1})$ we can observe e.g., that terms like $|\mathtt{t1} \oplus \mathtt{c}|$ cannot have a sort, because cardinality is just defined for markings, not for rewrites.

To conclude this section we show that all terms $\alpha$ such that $Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ \alpha : \mathtt{Rw}$ with $Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ s(\alpha) = \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c}$ and $Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ t(\alpha) = \mathtt{a} \oplus \mathtt{d}$ can be proved equivalent.

The first thing to observe is that any rewrite $\alpha$ with sort `Rw` can either be decomposed as a sequence of rewrites $\alpha_1; \ldots; \alpha_n$ with $n > 0$ and $Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ \alpha_i : \mathtt{Rw1}$ for any $i \in [1, n]$ or $Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ \alpha : \mathtt{Marking}$ (i.e., $\alpha$ is an idle rewrite).

```
fmod Proof_N1 is
       *** Equational specification from N1 is omitted
     sorts Rw1 Rw .
  subsorts Marking Rw1 < Rw .
       ops s t : Rw -> Marking .
        op _ ; _ : [Rw] [Rw] -> [Rw] .
      var U : Marking .
      vars X Y Z K : Rw .
        eq s(U) = U .
        eq t(U) = U .
       cmb X;Y : Rw  if t(X)=s(Y) .
       ceq s(X;Y) = s(X)  if t(X)=s(Y) .
       ceq t(X;Y) = t(Y)  if t(X)=s(Y) .
       ceq U;X = X  if s(X)=U .
       ceq X;U = X  if t(X)=U .
       ceq X;(Y;Z) = (X;Y);Z  if t(X)=s(Y) ∧ t(Y)=s(Z) .

        op _⊕_ : Rw Rw -> Rw [assoc comm id: 0] .
        op _⊕_ : Rw1 Marking -> Rw1 .
        op _⊕_ : Marking Rw1 -> Rw1 .
        eq s(X⊕U) = s(X)⊕s(U) .
        eq t(X⊕U) = t(X)⊕t(U) .
        eq s(U⊕X) = s(U)⊕s(X) .
        eq t(U⊕X) = t(U)⊕t(X) .
       ceq (X;Y)⊕(Z;K) = (X⊕Z);(Y⊕K)  if t(X)=s(Y) ∧ t(Z)=s(K) .

       ops t1 t2 t3 : -> Rw1 .
        eq s(t1) = a⊕a .
        eq t(t1) = b .
        eq s(t2) = b⊕c .
        eq t(t2) = d .
        eq s(t3) = a⊕b .
        eq t(t3) = a .
endfm
```

Fig. 7. The MEL theory $Proof(\mathcal{R}_{N_1})$.

The second observation is that there are only four proof terms expressing the reachability of $\mathtt{a} \oplus \mathtt{d}$ from $\mathtt{a} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c}$, which are listed below:

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t2} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{a});\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{d});\ (\mathtt{t3} \oplus \mathtt{d}) : \mathtt{Rw}$$

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c});\ (\mathtt{t2} \oplus \mathtt{a} \oplus \mathtt{b});\ (\mathtt{t3} \oplus \mathtt{d}) : \mathtt{Rw}$$

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c});\ (\mathtt{t3} \oplus \mathtt{b} \oplus \mathtt{c});\ (\mathtt{t2} \oplus \mathtt{a}) : \mathtt{Rw}$$

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t3} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{c});\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{c});\ (\mathtt{t2} \oplus \mathtt{a}) : \mathtt{Rw}$$

Third, consider the following equalities, which can be easily proved by functoriality of $\oplus$ (i.e., by the axiom distributing $\oplus$ w.r.t. sequential composition):

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t2} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{a});\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{d}) = (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c});\ (\mathtt{t2} \oplus \mathtt{a} \oplus \mathtt{b})$$

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t2} \oplus \mathtt{a} \oplus \mathtt{b});\ (\mathtt{t3} \oplus \mathtt{d}) = (\mathtt{t3} \oplus \mathtt{b} \oplus \mathtt{c});\ (\mathtt{t2} \oplus \mathtt{a})$$

$$Proof(\mathcal{R}_{N_1}) \vdash (\forall \emptyset)\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c});\ (\mathtt{t3} \oplus \mathtt{b} \oplus \mathtt{c}) = (\mathtt{t3} \oplus \mathtt{a} \oplus \mathtt{a} \oplus \mathtt{c});\ (\mathtt{t1} \oplus \mathtt{a} \oplus \mathtt{c})$$

from which we can derive pairwise equalities among the four proof terms above by exploiting the congruence rule of MEL.

Finally, by the transitivity rule of MEL we can conclude that all the proof terms are identified in $Proof(\mathcal{R}_{N_1})$.

## 4. Provability of conditional rules

In the above, we have directed our attention to the case of deduction of unconditional sequents $\mathcal{R} \vdash (\forall X)\ t \to t'$. Here, we extend the RL inference system of Section 2.1 to prove conditional sequents, that is, sequents of the form:

$$(\forall X)\ t \to t' \quad \text{if} \quad \mathbb{C},$$

where

$$\mathbb{C} = \left( \bigwedge_{i \in I} p_i = q_i \ \wedge \ \bigwedge_{j \in J} w_j : s_j \ \wedge \ \bigwedge_{l \in L} t_l \to t'_l \right).$$

Let us introduce the notation $\mathcal{R} \vdash (\forall X)\ \mathbb{C}$ in place of the conjunction

$$\bigwedge_{i \in I} (E \vdash (\forall X)\ p_i = q_i) \ \wedge \ \bigwedge_{j \in J} \left(E \vdash (\forall X)\ w_j : s_j\right) \ \wedge \ \bigwedge_{l \in L} \left(\mathcal{R} \vdash (\forall X)\ t_l \to t'_l\right).$$

We give now a sound and complete inference system to prove conditional sequents. To this aim, we define a notion of *provability of a conditional sequent* by a rewrite theory:

$$\mathcal{R} \vdash (\forall X)\ t \to t' \ \text{ if } \ \mathbb{C}.$$

Exploiting a suitable "theorem of constants" we can reduce proving a conditional sequent to proving an unconditional sequent after adding the assumptions $\mathbb{C}$ to $\mathcal{R}$.

**Definition 4.1** (*Implication introduction*). We say that the conditional rule $(\forall X)\ t \to t' \ \text{ if } \ \mathbb{C}$ is *provable* in $\mathcal{R}$ iff $\mathcal{R}^X_{\mathbb{C}} \vdash (\forall \emptyset)\ t \to t'$, where $\mathcal{R}^X_{\mathbb{C}}$ is the generalized rewrite theory obtained from $\mathcal{R}$ by turning all the variables in $X$ into fresh constants (thus making ground the conditions $\mathbb{C}$ and the rewrite $t \to t'$) and by adding the (now grounded) conditions $\mathbb{C}$ as a new set of axioms.

We write

$$\mathcal{R} \vdash (\forall X)\ t \to t' \quad \text{if} \quad \mathbb{C}$$

to say that $(\forall X)\ t \to t' \ \text{ if } \ \mathbb{C}$ is provable in $\mathcal{R}$.

**Theorem 4.1.** *The inference system of generalized rewrite theories enriched with* "*implication introduction*" *is sound and complete for proving conditional sentences.*

**Proof.** The proof is based on the analogous result for MEL in [36].

We first prove that $\mathcal{R} \vdash (\forall X)\ t \to t' \ \text{ if } \ \mathbb{C}$ if and only if $\mathit{Reach}(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k \ \text{ if } \ \tilde{\mathbb{C}}$, where $k$ is the kind of $t$ and $t'$, and $\tilde{\mathbb{C}}$ is the obvious translation of $\mathbb{C}$, where for each $l \in L$ the rewrite $t_l \to t'_l$ is translated to the membership $(t_l \to t'_l) : Ar_{k_l}$. This follows by a straightforward application of rules for implication introduction in MEL and in generalized rewrite theories, as in fact $\mathit{Reach}(\mathcal{R}^X_{\mathbb{C}})$ and $\mathit{Reach}(\mathcal{R})^X_{\tilde{\mathbb{C}}}$ essentially coincide.

The thesis then follows by soundness and completeness of MEL. □

## 5. Concluding remarks, related work, and future work

We have defined *generalized rewrite theories* to substantially extend the expressiveness of RL in many applications. We have given rules of deduction for these theories, defined their models as MEL algebras, and shown that initial and free models exist (for both reachability and true concurrency models). We have also shown that this generalized RL is complete with respect to its model theory, and that our results generalize the original results for unsorted rewrite theories in [34].

When evaluating the trade-offs between the complexity of the presentation and the expressiveness of the proposed rewrite theories, we have preferred to give the precise foundational semantics for the most general form of rewrite

theories used in practice. Our results show that MEL is expressive enough to embed GRTs as MEL theories plus some syntactic sugar. This is very useful both to provide a simple model-theoretic semantics for GRTs and also for meta-theoretic reasoning. For example, we can prove some properties of a concurrent system specified by $\mathcal{R}$ by reasoning inductively on the theory $Reach(\mathcal{R})$ (or $Proof(\mathcal{R})$) with a tool such as Maude's ITP [15,12]. However, what is in essence a map of logics giving equationally defined *models* to GRTs should not be used for *execution* purposes, where the intrinsic separation of concerns in GRTs (i.e., equational vs. operational reasoning) is fundamental in most applications.

The theory $Proof(\mathcal{R})$ has an obvious reading as the GRT counterpart of the classic Curry–Howard isomorphism (proofs as terms). Along this line of research there is a flourishing literature that focuses on the full integration of type theory with RL. We can mention, for example, Stehr's work integrating RL and a generalized calculus of constructions in his Open Calculus of Constructions [41] (an equational extension of the Calculus of Constructions), the joint work of Stehr with the second author about the formalization of Pure Type Systems in RL [39] (both papers are parts of a more ambitious project concerning the implementation in Maude of a proof-assistant for the Open Calculus of Constructions), the work by Cirstea et al. on the $\rho$-calculus [10], where pattern-matching facilities are added to $\lambda$-abstraction rephrasing Barendregt's $\lambda$-cube categorization. In the $\rho$-calculus, abstraction expresses rewrite rules and rewriting is performed through application. Other papers addressing the integration of $\lambda$-calculus and equational term rewriting systems are, e.g., [5,24,2]. A second direction investigated by Barthe et al. concerns the extension with pattern-matching facilities of Pure Type Systems [3]. However, with the exception of [41] (and of [9,1], where matching constraints and unification are dealt with explicitly even at the syntax level), the focus of all the above approaches tends to disregard the treatment of the underlying algebraic theory, which is somewhat externalized and hidden in the matching phase of (rewrite) applications.

In our presentation, the use of MEL offers a solution to this matter, relying on the (sound and complete) inference system of the logic, which is simple yet rather sophisticated and expressive. A second difference is the treatment of frozen arguments, which is not considered in the above-mentioned work. Although at first sight frozen arguments could be regarded as just an operational tool for improving the performance of computation/proof-searching, the analogy with process calculi and the frequent distinction between data and states suggest that the model theory is strongly influenced by frozen arguments. For example, many process calculi comes with a notion of reactive contexts, which are the only contexts where reductions can be nested and thus they form a relevant part of the specification. We have shown that frozen arguments can be dealt with uniformly in MEL, which can be useful for theorem proving purposes when reasoning about such theories. A third difference is that the aforementioned papers tend to exploit higher-order frameworks, whereas for meta-reasoning we prefer to rely on (logical) reflection [11,4].

We conjecture that an operational view of frozen operators can be conveniently presented in *tile logic* [22], an extension of RL where rewrites can be easily synchronized and made conditional to the behavior of subterms, but the details have still to be worked out.

The use of equational theories for presenting the proof theory and model theory of GRTs has been preferred to other more category-theory-oriented frameworks like monads and sketches, because we have tried to make the results accessible with the most economic means possible, without assuming acquaintance with category theory. Future work will make more explicit the 2-categorical nature of our model theory, and will develop the semantics of *generalized rewrite theory morphisms*, extending the ideas in [33,37]. Another topic worth investigating is the semantics of parameterized GRTs.

## Acknowledgments

We thank Jean-Bernard Stefani, Mark-Oliver Stehr, and Peter Ölveczky, for helping us along the way to frozen operators, and all the members of the Maude team for invaluable insights towards more general notions of rewrite theory. We especially thank Miguel Palomino and Narciso Martí-Oliet for many helpful comments. We also thank the anonymous referees for their careful reading and insightful remarks.

## Appendix A. Axiomization of decorated sequents in conditional rewriting logic

To keep the paper self-contained, we recall here from [34] the axioms quotienting out decorated sequents of (unsorted) conditional RL, that were informally discussed in Section 1.1. To keep the presentation short, whenever sequential

composition is involved in the expressions appearing in the axioms, we assume that the corresponding domain and codomain match.

(1) *Category structure*:
   (a) *Associativity*: for all decorated sequents $\alpha$, $\beta$, $\gamma$,

$$(\alpha; \beta); \gamma = \alpha; (\beta; \gamma).$$

   (b) *Identities*: for each $\alpha: [t] \to [t']$,

$$\alpha; [t'] = \alpha = [t]; \alpha.$$

(2) *Functoriality of the* $(\Sigma, E)$*-structure*: for each $f \in \Sigma_n$, $n \in \mathbb{N}$,
   (a) *Preservation of composition*: for all decorated sequents $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$,

$$f(\alpha_1; \beta_1, \ldots, \alpha_n; \beta_n) = f(\alpha_1, \ldots, \alpha_n); f(\beta_1, \ldots, \beta_n).$$

   (b) *Preservation of identities*: for all $t_1, \ldots, t_n$,

$$f([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)].$$

   (c) *Axioms in* $E$: for each axiom $(\forall x_1, \ldots, x_n)\ t = t'$ in $E$, and for all decorated sequents $\alpha_1, \ldots, \alpha_n$,

$$t[\alpha_1/x_1, \ldots, \alpha_n/x_n] = t'[\alpha_1/x_1, \ldots, \alpha_n/x_n].$$

(3) *Concurrency*: for each rewrite rule

$$(\forall x_1, \ldots, x_n)\ r: [t] \to [t'] \text{ if } \bigwedge_{1 \leqslant i \leqslant \ell} [t_i] \to [t'_i]$$

in $R$,
   (a) *Decomposition*: for all decorated sequents

$$\alpha_1: [w_1] \to [w'_1], \ldots, \alpha_n: [w_n] \to [w'_n],$$

$$\beta_1: [t_1[\vec{w}/\vec{x}]] \to [t'_1[\vec{w}/\vec{x}]], \ldots, \beta_\ell: [t_\ell[\vec{w}/\vec{x}]] \to [t'_\ell[\vec{w}/\vec{x}]],$$

   we let

$$r(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_\ell) = r([w_1], \ldots, [w_n], \beta_1, \ldots, \beta_\ell); t'[\vec{\alpha}/\vec{x}].$$

   (b) *Exchange*: for all decorated sequents

$$\alpha_1: [w_1] \to [w'_1], \ldots, \alpha_n: [w_n] \to [w'_n]$$

$$\beta_1: [t_1[\vec{w}/\vec{x}]] \to [t'_1[\vec{w}/\vec{x}]], \ldots, \beta_\ell: [t_\ell[\vec{w}/\vec{x}]] \to [t'_\ell[\vec{w}/\vec{x}]]$$

$$\beta'_1: [t_1[\vec{w'}/\vec{x}]] \to [t'_1[\vec{w'}/\vec{x}]], \ldots, \beta'_\ell: [t_\ell[\vec{w'}/\vec{x}]] \to [t'_\ell[\vec{w'}/\vec{x}]],$$

   such that

$$\beta_1; t'_1[\vec{\alpha}/\vec{x}] = t_1[\vec{\alpha}/\vec{x}]; \beta'_1, \ldots, \beta_\ell; t'_\ell[\vec{\alpha}/\vec{x}] = t_\ell[\vec{\alpha}/\vec{x}]; \beta'_\ell,$$

   we let

$$r([w_1], \ldots, [w_n], \beta_1, \ldots, \beta_\ell); t'[\vec{\alpha}/\vec{x}] = t[\vec{\alpha}/\vec{x}]; r([w'_1], \ldots, [w'_n], \beta'_1, \ldots, \beta'_\ell).$$

## Appendix B. Proofs of main results

### *B.1. Proof of Theorem 3.2*

**Proof.** As explained in Section 3.1, while the equivalence of the first two statements is obtained by proving separately the two corresponding implications, the other equivalences can be proved directly. Thus, the proof of Theorem 3.2 is divided in four parts. Here, we detail only the first part, i.e., the implication (1) ⇒ (2), which requires the three Lemmas B.1–B.3 inserted immediately after this proof. The remaining four parts are described in the main text.

The implication (1) ⇒ (2) is proved by rule induction over the proof of $\mathcal{R} \vdash (\forall X)\ t \to t'$.

For (*Reflexivity*) the result is proved by applying the modus ponens rule of MEL with substitution $[t/x]$ to the fourth sentence in 1(c) of Definition 3.2 to obtain $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0$, then by subsorting $Ar_k^0 < Ar_k$ we get the result.

For (*Equality*), we have by hypothesis that $Reach(\mathcal{R}) \vdash (\forall X)\ (u \to u') : Ar_k$, and by conservativity of $E$ in $Reach(\mathcal{R})$ we also have $Reach(\mathcal{R}) \vdash (\forall X)\ t = u$ and $Reach(\mathcal{R}) \vdash (\forall X)\ u' = t'$. Therefore, we can apply the congruence rule of MEL to conclude the result.

For (*Congruence*) we have by hypothesis that $Reach(\mathcal{R}) \vdash (\forall X)\ (t_j \to t'_j) : Ar_k$ for $j \in v(f)$. But then, by Lemma B.2, for each $j \in v(f)$ we have that either $Reach(\mathcal{R}) \vdash (\forall X)\ (t_j \to t'_j) : Ar_k^0$ (and therefore, by Lemma B.1, that $E \vdash (\forall X)\ t_j = t'_j$) or that there exist $t_{j,1}, \ldots, t_{j,n_j} \in \mathbb{T}_\Sigma(X)_k$, with $t_{j,1} = t_j$ and $t_{j,n_j} = t'_j$ such that $Reach(\mathcal{R}) \vdash (\forall X)\ (t_{j,i} \to t_{j,i+1}) : Ar_k^1$ for all $i = 1, \ldots, n_j - 1$, in which case, by 2 in Definition 3.2 we have

$$\begin{aligned} &Reach(\mathcal{R}) \vdash (\forall X)\, f(t'_1, \ldots, t'_{j-1}, (t_{j,i} \to t_{j,i+1}), t_{j+1}, \ldots, t_n) \\ &\quad = f(t'_1, \ldots, t'_{j-1}, t_{j,i}, t_{j+1}, \ldots, t_n) \to f(t'_1, \ldots, t'_{j-1}, t_{j,i+1}, t_{j+1}, \ldots, t_n). \end{aligned}$$

Moreover, by the membership rule of MEL we have

$$Reach(\mathcal{R}) \vdash (\forall X)\ f(t'_1, \ldots, t'_{j-1}, t_{j,i}, t_{j+1}, \ldots, t_n) \to f(t'_1, \ldots, t'_{j-1}, t_{j,i+1}, t_{j+1}, \ldots, t_n) : Ar_k^1$$

and therefore, by subsorting, that

$$Reach(\mathcal{R}) \vdash (\forall X)\ f(t'_1, \ldots, t'_{j-1}, t_{j,i}, t_{j+1}, \ldots, t_n) \to f(t'_1, \ldots, t'_{j-1}, t_{j,i+1}, t_{j+1}, \ldots, t_n) : Ar_k.$$

Finally, by repeated application of the fifth rule in 1(c) of Definition 3.2 we get the result.

For (*Nested Replacement*), by conservativity of $E$ in $Reach(\mathcal{R})$, we can assume that $Reach(\mathcal{R}) \vdash (\forall Y)\ \theta(p_i) = \theta(q_i)$ for all $i \in I$, and that $Reach(\mathcal{R}) \vdash (\forall Y)\ \theta(w_j) : s_j$ for all $j \in J$. Moreover, by the induction hypothesis, we also have $Reach(\mathcal{R}) \vdash (\forall Y)\ (\theta(t_i) \to \theta(t'_i)) : Ar_k$ for all $i \in L$. Therefore, we can apply the modus ponens rule of MEL with substitution $\theta$ to the conditional membership in 3 of Definition 3.2 to infer that $Reach(\mathcal{R}) \vdash (\forall Y)\ (\theta(t) \to \theta(t')) : Ar_k^1$, and, by subsorting $Ar_k^1 < Ar_k$, that

$$Reach(\mathcal{R}) \vdash (\forall Y)\ (\theta(t) \to \theta(t')) : Ar_k. \tag{B.1}$$

By induction hypotheses, we also have that $Reach(\mathcal{R}) \vdash (\forall Y)\ (\theta(x) \to \theta'(x)) : Ar_k$ for all $x \in v(t, t')$, while $\theta(x) = \theta'(x)$ for all $x \in \phi(t, t')$. Thus, we can apply Lemma B.3 to prove that

$$Reach(\mathcal{R}) \vdash (\forall Y)\ (\theta(t') \to \theta'(t')) : Ar_k. \tag{B.2}$$

But then, we can apply the fifth rule in 1(c) of Definition 3.2 using as premises the facts (B.1) and (B.2) above to get the result.

For (*Transitivity*) we directly apply the fifth rule in 1(c) of Definition 3.2 to the hypothesis to get the result. □

**Lemma B.1.** *If* $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0$, *then* $E \vdash (\forall X)\ t = t'$.

**Proof.** It is immediate from the fact that in $Reach(\mathcal{R})$ the only equations imposed on terms of kinds $k \in K$ are those in $E$ and from the fact that $Reach(\mathcal{R})$ contains a unique axiom for assigning sort $Ar_k^0$ to a pair $(t \to t')$, which requires $t$ and $t'$ to be indistinguishable under $Reach(\mathcal{R})$ (and thus under $E$). □

**Lemma B.2.** *If* $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k$, *then either* $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0$ (*and therefore* $E \vdash (\forall X)\ t = t'$) *or there exist* $t_1, \ldots, t_n \in \mathbb{T}_\Sigma(X)_k$ *such that* $t = t_1$, $t' = t_n$ *and* $Reach(\mathcal{R}) \vdash (\forall X)\ (t_i \to t_{i+1}) : Ar_k^1$ *for all* $i = 1, \ldots, n-1$.

**Proof.** We proceed by induction on the proof of $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k$. Due to the ordering of sorts in $Reach(\mathcal{R})$, there are three possibilities under which $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k$:

- $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0$, which is in fact the first option in the statement.
- $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^1$, which is just an instance of the second option.
- There exists $t''$ with $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t'') : Ar_k$ and $Reach(\mathcal{R}) \vdash (\forall X)\ (t'' \to t') : Ar_k$, but then, by the induction hypothesis, four cases can arise:
  (1) $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t'') : Ar_k^0$ and $Reach(\mathcal{R}) \vdash (\forall X)\ (t'' \to t') : Ar_k^0$. Then, by Lemma B.1, $E \vdash (\forall X)\ t = t''$ and $E \vdash (\forall X)\ t'' = t'$, from which we obtain $E \vdash (\forall X)\ t = t'$ by the transitivity rule of MEL. Thus, we can apply the fourth axiom in 1(c) of Definition 3.2 to conclude that $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0$ (first option in the statement).
  (2) $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t'') : Ar_k^0$ and there exist $t_1, \ldots, t_n \in \mathbb{T}_\Sigma(X)_k$ with $t_1 = t''$ and $t_n = t'$, such that we have $Reach(\mathcal{R}) \vdash (\forall X)\ (t_i \to t_{i+1}) : Ar_k^1$ for all $i = 1, \ldots, n-1$. Then, by Lemma B.1, $E \vdash (\forall X)\ t = t''$ and therefore $E \vdash (\forall X)\ t = t_1$, leading to the second option in the statement.
  (3) $Reach(\mathcal{R}) \vdash (\forall X)\ (t'' \to t') : Ar_k^0$ and there exist $t_1, \ldots, t_n \in \mathbb{T}_\Sigma(X)_k$ with $t_1 = t$ and $t_n = t''$, such that we have $Reach(\mathcal{R}) \vdash (\forall X)\ (t_i \to t_{i+1}) : Ar_k^1$ for all $i = 1, \ldots, n-1$. Then, by Lemma B.1, $E \vdash (\forall X)\ t'' = t'$, and therefore $E \vdash (\forall X)\ t_n = t'$, leading to the second option in the statement.
  (4) There exist $t'_1, \ldots, t'_{n'} \in \mathbb{T}_\Sigma(X)_k$ and $t''_1, \ldots, t''_{n''} \in \mathbb{T}_\Sigma(X)_k$ with $t'_1 = t$, $t'_{n'} = t''_1 = t''$, $t''_{n''} = t'$, such that we have $Reach(\mathcal{R}) \vdash (\forall X)\ (t'_i \to t'_{i+1}) : Ar_k^1$ for all $i = 1, \ldots, n'-1$ and $Reach(\mathcal{R}) \vdash (\forall X)\ (t''_i \to t''_{i+1}) : Ar_k^1$ for all $i = 1, \ldots, n''-1$. Then, composing the two sequences (i.e., letting $n = n' + n'' - 1$ and taking $t_1, \ldots, t_n \in \mathbb{T}_\Sigma(X)_k$ defined as $t_i = t'_i$ for all $i = 1, \ldots, n'$ and as $t_{i+n'} = t''_{i+1}$ for all $i = 1, \ldots, n''-1$) we satisfy the second option in the statement. □

**Lemma B.3.** *Let* $t \in \mathbb{T}_\Sigma(X)_k$ *and let* $\theta, \theta' : X \to \mathbb{T}_\Sigma(Y)$ *be two substitutions such that*

- $\theta(x) = \theta'(x)$, *for all* $x \in \phi(t)$;
- $Reach(\mathcal{R}) \vdash (\forall X)\ (\theta(x) \to \theta'(x)) : Ar_k$, *for all* $x \in v(t)$.

*Then,* $Reach(\mathcal{R}) \vdash (\forall X)\ (\theta(t) \to \theta'(t)) : Ar_k$.

**Proof.** The proof goes by induction on the structure of $t$:

- If $t = x$ for some $x \in X$, then the result trivially holds.
- If $t = f(t_1, \ldots, t_n)$, then by the inductive hypothesis we can assume that $Reach(\mathcal{R}) \vdash (\forall X)\ (\theta(t_i) \to \theta'(t_i)) : Ar_k$ for all $i = 1, \ldots, n$. Moreover, for all $i \in \phi(f)$ we have $\theta(t_i) = \theta'(t_i)$, because all the variables appearing in $t_i$ are included in $\phi(t)$. Furthermore, for all $i \in v(f)$ we have, by Lemma B.2, that either $Reach(\mathcal{R}) \vdash (\forall X)\ (\theta(t_i) \to \theta'(t_i)) : Ar_k^0$ or that there exist $t_{i,1}, \ldots, t_{i,n_i} \in \mathbb{T}_\Sigma(Y)$ with $t_{i,1} = \theta(t_i)$ and $t_{i,n_i} = \theta'(t_i)$ such that $Reach(\mathcal{R}) \vdash (\forall X)\ (t_{i,j} \to t_{i,j+1}) : Ar_k^1$ for all $j = 1, \ldots, n_i$. It follows by 2 in Definition 3.2 that we have

$$Reach(\mathcal{R}) \vdash (\forall Y)\ \big(f(\theta'(t_1), \ldots, \theta'(t_{i-1}), t_{i,j}, \theta(t_{i+1}), \ldots, \theta(t_n)) \to f(\theta'(t_1), \ldots, \theta'(t_{i-1}), t_{i,j+1}, \theta(t_{i+1}), \ldots, \theta(t_n))\big) : Ar_k^1$$

for all $i \in v(f)$ and for all $j = 1, \ldots, n_i$. By subsorting $Ar_k^1 < Ar_k$ we also have

$$Reach(\mathcal{R}) \vdash (\forall Y)\ \big(f(\theta'(t_1), \ldots, \theta'(t_{i-1}), t_{i,j}, \theta(t_{i+1}), \ldots, \theta(t_n)) \to f(\theta'(t_1), \ldots, \theta'(t_{i-1}), t_{i,j+1}, \theta(t_{i+1}), \ldots, \theta(t_n))\big) : Ar_k.$$

Thus, noticing again that $\theta(t_i) = \theta'(t_i)$ whenever $i \in \phi(f)$, by repeated application of the fifth rule in 1(c) of Definition 3.2 we can conclude that

$$Reach(\mathcal{R}) \vdash (\forall Y)\ \big(f(\theta(t_1), \ldots, \theta(t_n)) \to f(\theta'(t_1), \ldots, \theta'(t_n))\big) : Ar_k.$$

Then the thesis follows from observing that $\theta(t) = f(\theta(t_1), \ldots, \theta(t_n))$ and $\theta'(t) = f(\theta'(t_1), \ldots, \theta'(t_n))$ by definition of substitution. □

*B.2. Proof of Proposition 3.3*

**Proof.** That deduction in $E$ implies deduction in $\mathit{Proof}(\mathcal{R})$ is obvious by the fact that $\mathit{Proof}(\mathcal{R})$ contains all axioms in $E$.

The only way in which distinct terms of kind $k$ in $\mathbb{T}_{\Sigma,E}(X)$ can be collapsed at the level of the sort $k$ (or below $k$) in $\mathit{Proof}(\mathcal{R})$ is through source/target projections, because all the additional operations return terms of sort $Rw_k^1$ and $Rw_k$. Therefore, we just need to prove that $\mathit{Proof}(\mathcal{R}) \vdash (\forall X)\ t = t'$ implies that $E \vdash (\forall X)\ \mathsf{s}(t) = \mathsf{s}(t')$ (and $E \vdash (\forall X)\ \mathsf{t}(t) = \mathsf{t}(t')$). To prove this we first select a few axioms from $\mathit{Proof}(\mathcal{R})$ that inductively define source/target projections, and then we show that all the other axioms respect source and target projections (up-to $E$).

We let $ST$ be the set of axioms below, taken from $\mathit{Proof}(\mathcal{R})$.

$$
\begin{aligned}
(\forall x : k)\ \mathsf{s}(x) &= x \\
(\forall x : k)\ \mathsf{t}(x) &= x \\
(\forall x, y : Rw_k)\ \mathsf{s}(x; y) &= \mathsf{s}(x) \\
&\text{if } \mathsf{t}(x) = \mathsf{s}(y) \\
(\forall x, y : Rw_k)\ \mathsf{t}(x; y) &= \mathsf{t}(y) \\
&\text{if } \mathsf{t}(x) = \mathsf{s}(y) \\
(\forall x_1 : k_1, \ldots, x_{i_1} : Rw_{k_{i_1}}, \ldots, x_{i_m} : Rw_{k_{i_m}}, \ldots, x_n : k_n)\ \mathsf{s}(f(x_1, \ldots, x_n)) &= f(\mathsf{s}(x_1), \ldots, \mathsf{s}(x_n)) \\
(\forall x_1 : k_1, \ldots, x_{i_1} : Rw_{k_{i_1}}, \ldots, x_{i_m} : Rw_{k_{i_m}}, \ldots, x_n : k_n)\ \mathsf{t}(f(x_1, \ldots, x_n)) &= f(\mathsf{t}(x_1), \ldots, \mathsf{t}(x_n)) \\
(\forall x_i : k_i, z_i : Rw_{k_i}, y_j : Rw_{k'_j})\ \mathsf{s}(r(\vec{z}, \vec{y})) &= t \\
&\text{if } \Psi \wedge \Delta \\
(\forall x_i : k_i, z_i : Rw_{k_i}, y_j : Rw_{k'_j})\ \mathsf{t}(r(\vec{z}, \vec{y})) &= t'[\mathsf{t}(\vec{z})/\vec{x}] \\
&\text{if } \Psi \wedge \Delta
\end{aligned}
$$

for each $f : k_1 \ldots k_n \longrightarrow k \in \Sigma$, with $\nu(f) = \{i_1, \ldots, i_m\}$, and for each generalized rewrite rule in $R$,

$$(\forall X)\ r : t \to t' \text{ if } \bigwedge_{i \in I} p_i = q_i \ \wedge\ \bigwedge_{j \in J} w_j : s_j \ \wedge\ \bigwedge_{l \in L} t_l \to t'_l$$

with, say, $X = \{x_1 : k_1, \ldots, x_n : k_n\}$, $t, t'$ of kind $k$, and $t_l, t'_l$ of kind $k'_l$ with $L = \{1, \ldots, \ell\}$, so that

$$\Delta = \left( \bigwedge_{i \in I} p_i = q_i \ \wedge\ \bigwedge_{j \in J} w_j : s_j \ \wedge\ \bigwedge_{l \in L} \mathsf{s}(y_l) = t_l \ \wedge\ \bigwedge_{l \in L} \mathsf{t}(y_l) = t'_l \right)$$

$$\Psi = \left( \bigwedge_{x_h \in \phi(t,t')} z_h = x_h \ \wedge\ \bigwedge_{x_h \in \nu(t,t')} \mathsf{s}(z_h) = x_h \right).$$

Note that the axioms in $ST$ inductively strip away from the expression $\alpha(\vec{x})$ all the operators not in $\Sigma$, so that the canonical forms of $\mathsf{s}(\alpha(\vec{x}))$ (and of $\mathsf{t}(\alpha(\vec{x}))$ as well) are of the form $t[\mathsf{s}(\vec{x})/\vec{y}]$ for $t \in \mathbb{T}_\Sigma(\vec{y})$ and $\vec{y}$ variables typed over the kinds in $K$. Then, the implication $\mathit{Proof}(\mathcal{R}) \vdash (\forall X)\ t : k \ \Rightarrow\ E \vdash (\forall X)\ t : k$ is straightforward, because $\mathit{Proof}(\mathcal{R})$ does not contain any membership sentence

$$(\forall X)\ \alpha : k \text{ if } \bigwedge_{i \in I} \alpha_i = \beta_i \ \wedge\ \bigwedge_{j \in J} \gamma_j : s_j$$

except for those already in $E$. Similarly when the membership concerns a sort $s < k$. Thus we are just left with proving that, for all the other (conditional) equations in $\mathit{Proof}(\mathcal{R})$ of the general form

$$(\forall X)\ \alpha = \beta \text{ if } \bigwedge_{i \in I} \alpha_i = \beta_i \ \wedge\ \bigwedge_{j \in J} \gamma_j : s_j$$

we have that $\mathsf{s}(\alpha)$ and $\mathsf{s}(\beta)$ (respectively, $\mathsf{t}(\alpha)$ and $\mathsf{t}(\beta)$), as inductively defined by axioms in $ST$, are equivalent up-to $E$. This is shown below by taking the axioms in the same order as they are presented in Definition 3.3:

(1) Under the hypotheses $x : k$ and $y : Rw_k$ and $x = \mathsf{s}(y)$, we have that $\mathsf{s}(x; y) = \mathsf{s}(x) = x$ by inductive definition of $\mathsf{s}$, and therefore $\mathsf{s}(x; y) = \mathsf{s}(y)$ by transitivity. Similarly, under the same hypotheses, $\mathsf{t}(x; y) = \mathsf{t}(y)$.

(2) Under the hypotheses $x : Rw_k$ and $y : k$ and $\mathsf{t}(x) = y$, we have that $\mathsf{s}(x; y) = \mathsf{s}(x)$ by inductive definition of $\mathsf{s}$. Similarly, under the same hypotheses, $\mathsf{t}(x; y) = \mathsf{t}(y) = y$ by inductive definition of $\mathsf{t}$ and therefore $\mathsf{t}(x; y) = \mathsf{t}(x)$ by transitivity.

(3) Under the hypotheses $x, y, z : Rw_k$, $\mathsf{t}(x) = \mathsf{s}(y)$ and $\mathsf{t}(y) = \mathsf{s}(z)$, we have that $\mathsf{s}(x; (y; z)) = \mathsf{s}(x)$ by inductive definition of $\mathsf{s}$ and that $\mathsf{s}((x; y); z) = \mathsf{s}(x; y) = \mathsf{s}(x)$ for the same reason.

(4) Under the hypotheses $x, y, z : Rw_k$, $\mathsf{t}(x) = \mathsf{s}(y)$ and $\mathsf{t}(y) = \mathsf{s}(z)$, we have that $\mathsf{t}(x; (y; z) = \mathsf{t}(y; z)) = \mathsf{t}(z)$ by inductive definition of $\mathsf{t}$ and that $\mathsf{t}((x; y); z) = \mathsf{t}(z)$ for the same reason.

(5) Let $f : k_1 \dots k_n \longrightarrow k$ an operator in $\Sigma$, with $v(f) = \{i_1, \dots, i_m\}$, then under the hypotheses $x_1 : k_1, \dots, x_{i_1}, y_{i_1} : Rw_{k_{i_1}} \dots, x_{i_m}, y_{i_m} : Rw_{k_{i_m}}, \dots, x_n : k_n$ and $\mathsf{t}(x_{i_j}) = \mathsf{s}(y_{i_j})$ for all $j = 1, \dots, m$, we have

$$\begin{aligned}\mathsf{s}(f(x_1, \dots, (x_{i_1}; y_{i_1}), \dots, (x_{i_m}; y_{i_m}), \dots, x_n)) &= f(\mathsf{s}(x_1), \dots, \mathsf{s}(x_{i_1}; y_{i_1}), \dots, \mathsf{s}(x_{i_m}; y_{i_m}), \dots, \mathsf{s}(x_n))\\ &= f(x_1, \dots, \mathsf{s}(x_{i_1}), \dots, \mathsf{s}(x_{i_m}), \dots, x_n)\end{aligned}$$

and

$$\mathsf{s}(f(x_1, \dots, x_n); f(x_1, \dots, y_{i_1}, \dots, y_{i_m}, \dots, x_n)) = \mathsf{s}(f(x_1, \dots, x_n)) = f(\mathsf{s}(x_1), \dots, \mathsf{s}(x_n)).$$

(6) For any conditional equation

$$t = t' \text{ if } \bigwedge_{i\in I} p_i = q_i \ \wedge \ \bigwedge_{j\in J} \mathsf{s}(w_j) : s_j \ \wedge \ \bigwedge_{j\in J} \mathsf{t}(w_j) : s_j \ \wedge \ \bigwedge_{x_h\in\phi(t,t')} x_h : k_h \ \wedge \ \bigwedge_{x_h\in v(t,t')} x_h : Rw^1_{k_h}$$

that extends a conditional equation in $E$, we simply observe that $t$ and $t'$ only involve operators in $\Sigma$ and therefore $\mathsf{s}(t) = t[\mathsf{s}(x_1)/x_1, \dots, \mathsf{s}(x_n)/x_n]$ and analogously for $\mathsf{s}(t')$, $\mathsf{s}(w_j)$, so that the result follows as an instance of the corresponding (original) equation in $E$. The same applies in the case of target projections.

(7) For each generalized rewrite rule in $R$,

$$(\forall X)\ r : t \to t' \text{ if } \bigwedge_{i\in I} p_i = q_i \ \wedge \ \bigwedge_{j\in J} w_j : s_j \ \wedge \ \bigwedge_{l\in L} t_l \to t'_l$$

with, say, $X = \{x_1 : k_1, \dots, x_n : k_n\}$, $t, t'$ of kind $k$, and $t_l, t'_l$ of kind $k'_l$ with $L = \{1, \dots, \ell\}$, and under the hypotheses

- $x_h : k_h$ and $z_h : Rw_{k_h}$, for $h = 1, \dots, n$; while $y_l : Rw_{k'_l}$, for $l = 1, \dots, \ell$;
- $p_i = q_i$, for $i \in I$ and $w_j : s_j$, for $j \in J$;
- $\mathsf{s}(y_l) = t_l$ and $\mathsf{t}(y_l) = t'_l$, for $l \in L$;
- $z_h = x_h$, for $x_h \in \phi(t, t')$, while $\mathsf{s}(z_h) = x_h$, for $x_h \in v(t, t')$;

then we have $\mathsf{s}(r(\vec{z}, \vec{y})) = t$ and $\mathsf{s}(r(\vec{x}, \vec{y}); t'[\vec{z}/\vec{x}]) = \mathsf{s}(r(\vec{x}, \vec{y})) = t$, and analogously, $\mathsf{t}(r(\vec{z}, \vec{y})) = t'[\mathsf{t}(\vec{z})/\vec{x}]$ with also $\mathsf{t}(r(\vec{x}, \vec{y}); t'[\vec{z}/\vec{x}]) = \mathsf{t}(t'[\vec{z}/\vec{x}]) = t'[\mathsf{t}(\vec{z})/\vec{x}]$. □

*B.3. Proof of Theorem 3.4*

**Proof.** Let $Flat(\mathcal{R})$ be the theory obtained by adding the axiom

$$(\forall x : Rw_k, y : Rw_k)\ x = y \text{ if } \mathsf{s}(x) = \mathsf{s}(y) \ \wedge \ \mathsf{t}(x) = \mathsf{t}(y)$$

to the theory $Proof(\mathcal{R})$.

Then it is obvious that

$$\exists\alpha.\ Proof(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \ \wedge \ Proof(\mathcal{R}) \vdash (\forall X)\ \mathsf{s}(\alpha) = t \ \wedge \ Proof(\mathcal{R}) \vdash (\forall X)\ \mathsf{t}(\alpha) = t'$$

if and only if

$$\exists\alpha.\ Flat(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \ \wedge \ Flat(\mathcal{R}) \vdash (\forall X)\ \mathsf{s}(\alpha) = t \ \wedge \ Flat(\mathcal{R}) \vdash (\forall X)\ \mathsf{t}(\alpha) = t'.$$

The advantage of using $Flat(\mathcal{R})$ instead of $Proof(\mathcal{R})$ is that now all the equations in $Proof(\mathcal{R})$ are irrelevant, except for those defining source and target projections (the set $ST$ defined in the proof of Proposition 3.3) and those in $E$.

In particular, we can just prove the equivalence

$$Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k \ \Leftrightarrow\ \exists \alpha.\ \begin{cases} Flat(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{s}(\alpha) = t \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{t}(\alpha) = t' \end{cases}$$

and then conclude the proof by applying the result of Theorem 3.2 to infer the thesis as the consequence of a chain of equivalences.

In proving the equivalence between deduction in $Flat(\mathcal{R})$ and in $Reach(\mathcal{R})$ the four Lemmas B.4–B.7 are helpful (they are inserted immediately after this proof).

Then, we can prove the implication

$$Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k \ \Leftarrow\ \exists \alpha.\ \begin{cases} Flat(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{s}(\alpha) = t \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{t}(\alpha) = t' \end{cases}$$

by the following three steps:

(1) first, we exploit Lemma B.5 to decompose $\alpha$ into suitable $n$ sequents $\alpha_1, \ldots, \alpha_n$;
(2) then, we apply ($n$ times) Lemma B.7 to each $\alpha_i$ to obtain a sequence of composable steps in $Reach(\mathcal{R})$ with source and targets defined accordingly to the $\alpha_i$;
(3) finally we compose the sequence of undecorated steps using the operator $\_\,;\,\_$ of $Reach(\mathcal{R})$ to conclude that $Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k$.

The reverse implication

$$Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k \ \Rightarrow\ \exists \alpha.\ \begin{cases} Flat(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{s}(\alpha) = t \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{t}(\alpha) = t' \end{cases}$$

is proved by the following three steps:

(1) first we exploit Lemma B.2 to decompose $(t \to t')$ into suitable $n$ (sequentially composable) sequents $(t_i \to t_{i+1})$ for $i = 1, \ldots, n-1$ with $t = t_1$ and $t' = t_n$;
(2) then, we apply ($n$ times) Lemma B.7 to each $(t_i \to t_{i+1})$ to obtain a sequence of composable steps $\alpha_i$ in $Flat(\mathcal{R})$ with $Flat(\mathcal{R}) \vdash \mathtt{s}(\alpha_i) = t_i$ and $Flat(\mathcal{R}) \vdash \mathtt{t}(\alpha_i) = t_{i+1}$ for $i = 1, \ldots, n-1$;
(3) finally, we let $\alpha = \alpha_1; \cdots; \alpha_n$. □

**Lemma B.4.** *Let $t, t' \in T_\Sigma(X)_k$ and $s \in S_k$ for some kind $k$. Then, for any formula $\varphi$ of the form $t : k$ or $t : s$ or $t = t'$ we have that: $E \vdash (\forall X)\ \varphi \quad \Leftrightarrow \quad Flat(\mathcal{R}) \vdash (\forall X)\ \varphi$.*

**Lemma B.5.** *If $\exists \alpha.\ Flat(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k \ \wedge\ \mathtt{s}(\alpha) = t \ \wedge\ \mathtt{t}(\alpha) = t'$, then there exist $n$ sequents $\alpha_1, \ldots, \alpha_n$ such that $Flat(\mathcal{R}) \vdash (\forall X)\ \alpha_i : Rw_k^1$ for $i = 1, \ldots, n$ and $Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{t}(\alpha_i) = \mathtt{s}(\alpha_{i+1})$ for $i = 1, \ldots, n-1$ such that $Flat(\mathcal{R}) \vdash (\forall X)\ \alpha = \alpha_1; \cdots; \alpha_n$.*

**Lemma B.6.** *For $t, t' \in T_\Sigma(X)$ we have the equivalences*:

$$Flat(\mathcal{R}) \vdash (\forall X)\ t = t' \quad \Leftrightarrow \quad Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0$$

$$Flat(\mathcal{R}) \vdash (\forall X)\ t : k \quad \Leftrightarrow \quad Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t) : Ar_k^0$$

**Lemma B.7.** *For $t, t' \in T_\Sigma(X)$ we have the equivalence*:

$$\left.\begin{array}{l} Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^0 \vee \\ Reach(\mathcal{R}) \vdash (\forall X)\ (t \to t') : Ar_k^1. \end{array}\right\} \ \Leftrightarrow\ \exists \alpha.\ \begin{cases} Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{s}(\alpha) = t \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \mathtt{t}(\alpha) = t' \wedge \\ Flat(\mathcal{R}) \vdash (\forall X)\ \alpha : Rw_k^1 \end{cases}$$

## References


[1] P. Baldan, C. Bertolissi, H. Cirstea, C. Kirchner, A rewriting calculus for cyclic higher-order term graphs, in: M. Fernández (Ed.), Proc. of TERMGRAPH 2004, Second Internat. Workshop on Term Graph Rewriting, Electronic Notes in Theoretical Computer Science, Vol. 127.5, Elsevier Science, Amsterdam, 2005.

[2] F. Barbanera, M. Fernández, H. Geuvers, Modularity of strong normalization in the algebraic-lambda-cube, J. Funct. Programming 7 (6) (1997) 613–660.

[3] G. Barthe, H. Cirstea, C. Kirchner, L. Liquori, Pure pattern type systems, in: Proc. of POPL 2003, 30th SIGPLAN–SIGACT Symp. on Principles of Programming Languages, ACM SIGPLAN Notices, Vol. 38(1), Association for Computing Machinery, 2003, pp. 250–261.

[4] D. Basin, M. Clavel, J. Meseguer, Reflective metalogical frameworks, ACM Trans. Comput. Log. 5 (3) (2004) 528–576.

[5] F. Blanqui, J.-P. Jouannaud, M. Okada, Inductive-data-type systems, Theoret. Comput. Sci. 272 (1–2) (2002) 41–68.

[6] A. Bouhoula, J.-P. Jouannaud, J. Meseguer, Specification and proof in membership equational logic, Theoret. Comput. Sci. 236 (2000) 35–132.

[7] C. Braga, J. Meseguer, Modular rewriting semantics in practice, in: N. Martí-Oliet (Ed.), Proc. of WRLA 2004, Fifth Internat. Workshop on Rewriting Logic and its Applications, Vol. 117, Electronic Notes in Theoretical Computer Science, Elsevier Science, Amsterdam, 2005, 393–416.

[8] R. Bruni, J. Meseguer, Generalized rewrite theories, in: J.C.M. Baeten, J.K. Lenstra, J. Parrow, G.J. Woeginger (Eds.), Proc. of ICALP 2003, 30th Internat. Colloq. on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 2719, Springer, Berlin, 2003, pp. 252–266.

[9] H. Cirstea, G. Faure, C. Kirchner, A rho-calculus of explicit constraint application, in: N. Martí-Oliet (Ed.), Proc. of WRLA 2004, Fifth Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 117, Elsevier Science, Amsterdam, 2005.

[10] H. Cirstea, C. Kirchner, L. Liquori, The rho cube, in: F. Honsell, M. Miculan (Eds.), Proc. of FOSSACS 2001, Fourth Internat. Conf. on Foundations of Software Science and Computation Structures, Lecture Notes in Computer Science, Vol. 2030, Springer, Berlin, 2001, pp. 168–183.

[11] M. Clavel, Reflection in Rewriting Logic: Metalogical Foundations and Metaprogramming Applications, CSLI Publications, 2000.

[12] M. Clavel, The ITP Tool's home page, 2005, ⟨http://maude.sip.ucm.es/itp/⟩.

[13] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, J. Quesada, Maude: specification and programming in rewriting logic, Theoret. Comput. Sci. 285 (2002) 187–243.

[14] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, Maude 2.1 manual, 2005, ⟨http://maude.cs.uiuc.edu⟩.

[15] M. Clavel, F. Durán, S. Eker, J. Meseguer, Building equational proving tools by reflection in rewriting logic, in: Proc. of the CafeOBJ Symposium '98, Numazu, Japan. CafeOBJ Project, 1998, ⟨http://maude.cs.uiuc.edu⟩.

[16] E. Deplagne, Système de preuve modulo récurrence, Thèse de doctorat, Université Nancy 1, 2002.

[17] E. Deplagne, C. Kirchner, Induction as deduction modulo. Rapport de recherche, 2004.

[18] E. Deplagne, C. Kirchner, H. Kirchner, Q.-H. Nguyen, Proof search and proof check for equational and inductive theorems, in: F. Baader (Ed.), Proc. of CADE 2003, 19th Internat. Conf. on Automated Deduction, Lecture Notes in Computer Science, Vol. 2741, Springer, Berlin, 2003, pp. 297–316.

[19] R. Diaconescu, K. Futatsugi, CafeOBJ Report: the language, proof techniques, and methodologies for object-oriented algebraic specification, AMAST Series in Computing, Vol. 6, World Scientific, Singapore, 1998.

[20] G. Dowek, T. Hardin, C. Kirchner, Theorem proving modulo, J. Automat. Reasoning 31 (1) (2003) 33–72.

[21] K. Futatsugi (Ed.), Proc. of WRLA 2000, Third Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 36, Elsevier Science, Amsterdam, 2000.

[22] F. Gadducci, U. Montanari, The tile model, in: G. Plotkin, C. Stirling, M. Tofte (Eds.), Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, Cambridge, MA, 2000, pp. 133–166. Also Technical Report TR-27/96, Dipartimento di Informatica, Università di Pisa, 1996.

[23] F. Gadducci, U. Montanari (Eds.). Proc. of WRLA 2002, Fourth Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 71, Elsevier Science, Amsterdam, 2002.

[24] J.-P. Jouannaud, M. Okada, Abstract data type systems, Theoret. Comput. Sci. 173 (2) (1997) 349–391.

[25] C. Kirchner, H. Kirchner (Eds.), Proc. of WRLA'98, Second Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 15, Elsevier Science, Amsterdam, 1998.

[26] D.E. Knuth, P.B. Bendix, Simple word problems in universal algebra, in: J. Leech (Ed.), Computational Problems in Abstract Algebras, Pergamon Press, Oxford, 1970, pp. 263–297.

[27] S. Lucas, Termination of rewriting with strategy annotations, in: Proc. of LPAR 2001, Eighth Internat. Conf. on Logic for Programming, Artificial Intelligence and Reasoning, Lecture Notes in Artificial Intelligence, Vol. 2250, Springer, Berlin, 2001, pp. 669–684.

[28] N. Martí-Oliet (Ed.), Proc. of WRLA 2004, Fifth Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 117, Elsevier Science, Amsterdam, 2005.

[29] N. Martí-Oliet, J. Meseguer, Rewriting logic as a logical and semantic framework, in: D.M. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic, second ed., Vol. 9, Kluwer Academic Publishers, Dordrecht, 2002, pp. 1–87.

[30] N. Martí-Oliet, J. Meseguer, Rewriting logic: roadmap and bibliography, Theoret. Comput. Sci. 285 (2) (2002) 121–154.

[31] N. Martí-Oliet, K. Sen, P. Thati, An executable specification of asynchronous pi-calculus semantics and may testing in Maude 2.0, in: F. Gadducci, U. Montanari (Eds.), Proc. of WRLA 2002, Fourth Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 71, 2002.

[32] N. Martí-Oliet, A. Verdejo, Executable structural operational semantics in Maude, Technical Report 134-03, Dpto. Sistemas Informáticos y Programación, Universidad Complutense de Madrid, 2003.

[33] J. Meseguer, Rewriting as a unified model of concurrency, Technical Report SRI-CSL-90-02R, SRI International, Computer Science Laboratory, 1990.
[34] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, Theoret. Comput. Sci. 96 (1992) 73–155.
[35] J. Meseguer (Ed.), Proc. of WRLA'96, First Internat. Workshop on Rewriting Logic and its Applications, Electronic Notes in Theoretical Computer Science, Vol. 4, Elsevier Science, Amsterdam, 1996.
[36] J. Meseguer, Membership algebra as a logical framework for equational specification, in: F. Parisi-Presicce (Ed.), Proc. of WADT'97, 12th Workshop on Recent Trends in Algebraic Development Techniques, Lecture Notes in Computer Science, Vol. 1376, Springer, Berlin, 1998, pp. 18–61.
[37] J. Meseguer, Functorial semantics of rewrite theories, in: H.-J. Kreowski, U. Montanari, F. Orejas, G. Rozenberg, G. Taentzer (Eds.), Formal Methods in Software and Systems Modeling: Essays Dedicated to Hartmut Ehrig, on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, Vol. 3393, Springer, Berlin, 2005, pp. 220–235.
[38] J. Meseguer, P.C. Ölveczky, M.-O. Stehr, Rewriting logic as a unifying framework for Petri nets, in: H. Ehrig, G. Juhás, J. Padberg, G. Rozenberg (Eds.), Advances in Petri Nets: Unifying Petri Nets, Lecture Notes in Computer Science, Vol. 2128, Springer, Berlin, 2001, pp. 250–303.
[39] J. Meseguer, M.-O. Stehr, Pure type systems in rewriting logic, in: From Object-Orientation to Formal Methods: Dedicated to The Memory of Ole-Johan Dahl, Lecture Notes in Computer Science, Vol. 2128, Springer, Berlin, 2004, pp. 250–303.
[40] Protheo Team, The ELAN home page, 2005, ⟨http://elan.loria.fr⟩.
[41] M.-O. Stehr, Programming, specification, and interactive theorem proving—towards a unified language based on equational logic, rewriting logic, and type theory, Ph.D. Thesis, Universität Hamburg, Fachbereich Informatik, Germany, 2002, ⟨http://www.sub.uni-hamburg.de/disse/810/⟩.