Abstract rewriting logic parameterized by a formal system

Marc Aiguier
École Centrale Paris
Laboratoire de Mathématiques Appliquées aux Systèmes (MAS)
Grande Voie des Vignes - F-92295 Châtenay-Malabry
marc.aiguier@ecp.fr

Programme d’Épignomique
523, Place des Terrasses de l’Agora - F-91025 Evry
marc.aiguier@epigenomique.genopole.fr

Diane Bahrami
CEA-LIST Saclay
F-91191 Gif sur Yvette Cedex
diane.bahrami@cea.fr

Delphine Longuet
Centre for Mathematics and Computer Science (CWI),
Dpt. of Software Technology, Amsterdam, Netherlands
delphine.longuet@cwi.nl

Abstract: Since rewriting logic has been introduced, it has proved to be appropriate both as a semantic framework, particularly for concurrent and distributed computation, and as a logical framework, that is, a meta-logic in which other logics

1 Introduction

Since rewriting logic has been introduced [17], it has proved to be appropriate both as a semantic framework, particularly for concurrent and distributed computation, and as a logical framework, that is, a meta-logic in which other logics
can be represented. Indeed, the basic axioms of this logic, which are rewriting rules of the form \( t \rightarrow t' \) where \( t \) and \( t' \) are terms over a given signature, can be read two different ways: either as the local transition of a concurrent system or the inference rule of some logic. For the former, rewriting logic then extends (equational) algebraic specifications to deal with dynamic and concurrent systems. Indeed, algebraic specifications have proven to be suitable for describing complex data structures and the functional aspects of a software system. However, they are insufficient when applied to dynamic and distributed systems. For the latter, rewriting logic is then a “universal” logic within which other formalisms can be translated.

The numerous applications of rewriting logic in the above two areas has shown the importance of increasing its expressive power. The expressive power of the standard rewriting logic can be increased in two ways, by extending either the computational capabilities [7, 6, 14], or the logical capabilities, by considering another logic than the conditional equational logic to parameterize rewriting logic [18, 8].

When we observe all these extensions, at each time, they lead to the four questions:

1. What are the rules of deduction for this extended rewriting logic?
2. Does rewriting coincides with derivability in rewriting logic, i.e. given a rewrite system \( \mathcal{R} \), does \( \overset{*}{\rightarrow}_\mathcal{R} \) coincides with \( \mathcal{R} \vdash \) in rewriting logic?
3. What are the models of a rewrite theory? Are there initial and free models?
4. Is rewriting logic complete with respect to its model theory?

In the future, other applications will certainly lead to extend the standard rewriting logic to other particular aspects. These new extensions will naturally lead to answer the four above questions. However, as this has been observed in [6, 8], these extensions are usually nontrivial generalizations of the original inference rules, model theory, initial and free models, and completeness theorem for rewriting logic over equational logic as developed in [17]. Therefore, in order to facilitate this work, it can be useful to study how to define rewriting logic and how to answer the four above questions at a more abstract level. This is what we propose to do in this paper. This requires first to give an abstract form of logics which parameterize rewriting logic, and then to study rewriting in this abstract framework of logics. In previous papers [1, 2], we proposed such a general framework of rewriting by applying the paradigm “logical-system independent”, that is providing a general framework and conditions (axioms), and adapting and proving the classical definitions and results which underlie rewriting. Such an abstraction allowed one to unify and generalize many different rewriting theories.
The present paper is then devoted to the next step: showing that there exists a valid and useful notion of rewriting logic in this abstract framework. Hence, the present work continues the development of the abstract framework of rewriting developed in [1, 2].

In the abstract rewriting theory developed in [1, 2], abstraction is twofold:

1. Rewritten objects are just elements of a set without any particular structure as to be a set of terms.
2. Rewriting relations are specified by inference rules just defined as relations on formulas of formal systems.

The consequence of both above points is that the work presented here does not aim to generalize the approach developed by Meseguer and many others, that is providing a logical support to a very powerful version of transition systems. Indeed, in this case, rewriting logic is based on a binary relation on terms which is not symmetric (because change is not in general reversible) but transitive.

The present paper goes beyond by only generalizing the transformation that from the equational logic has resulted in the rewriting logic. Hence, the work presented in this paper is different from the one developed in [8] because the generalization defined in [8] rests on the logic which parameterizes rewriting logic: the membership equational logic with frozen operators. On the contrary, in this paper, we only generalize the method which defines a rewriting logic over a given logic (satisfying the conditions of our abstract formalism). Besides, we will show in Section 8 that the rewriting logic over membership equational logic [8] is an instance of our framework.

This paper is organized as follows: In Section 2, we recall standard notations about formal systems, theorem deduction and proof trees. In order to be as self-contained as possible, Section 3 and Section 4 summarize relevant definitions of [1]. Section 5 introduces the notion of rewriting logic at this abstract level. This will answer the first question. In Section 6, we will show that abstract rewriting logic allows to build all elements of the rewriting relation generated by any abstract rewrite system \( R \) under some easy-to-check sufficient conditions. This will answer the second question. Section 7 proposes a model theoretic semantics for abstract rewriting logic. The theorems proving the soundness and completeness of the abstract rewriting logic with respect to this semantics are presented. This will answer the third and fourth questions. Finally, Section 8 exemplifies the abstract framework.

To instantiate our definitions, concepts and results, we will present the formal system of the conditional rewriting logic [17] as a running example and we will present the constrained rewriting logic [13], the rewriting logic parameterized by
the formal system of the conditional membership equational logic \[8\] and the timed rewriting logic \[14\] in Section 8.

Note: The manuscript extends the paper published in the proceedings of FSEN2005 \[3\] with complete proofs of the main results and with expanded definitions, new results (cf. Section 6) and additional examples (cf. Section 8).

2 Preliminaries

A formal system (a so-called calculus) \(S = (F,R)\) over an alphabet \(A\) consists of a set \(F\) of strings over \(A\), called formulas, and a set \(R\) of relations on \(F\), called inference rules. Thus, a rule with arity \(n\) \((n \geq 1)\) is a set of tuples \((\varphi_1, \ldots, \varphi_n)\) of strings of \(F\). Each sequence \((\varphi_1, \ldots, \varphi_n)\) belonging to a rule \(r\) of \(R\) is called an instance of that rule with premises \(\varphi_1, \ldots, \varphi_{n-1}\) and conclusion \(\varphi_n\). It is usually written \(\varphi_1 \ldots \varphi_{n-1} \varphi_n\). If \(n = 1\), the instance is called an axiom and is written \(\varphi\). A deduction in \(S\) from a set of formulas \(\Gamma\) of \(F\) is a finite sequence \((\psi_1, \ldots, \psi_m)\) of formulas such that \(m \geq 1\) and, for all \(i = 1, \ldots, m\), either \(\psi_i\) is an element of \(\Gamma\) or there is an instance \(\varphi_1 \ldots \varphi_{n-1}\) of a rule in \(S\) where \(\varphi_n = \psi_i\) and \(\{\varphi_1, \ldots, \varphi_{n-1}\} \subseteq \{\psi_1, \ldots, \psi_m\}\). A theorem from a set of formulas \(\Gamma\) in \(S\) is a formula \(\varphi\) such that there exists a deduction in \(S\) from \(\Gamma\) with last element \(\varphi\). The existence of such a deduction is usually denoted by the meta-statement \(\Gamma \vdash \varphi\).

Instances of rules can also be composed to build proof trees. Formally, a proof tree \(\pi\) in a formal system \(S\) is a finite tree whose nodes are labelled with formulas of \(F\) in the following way: if a non-leaf node is labelled with \(\varphi_n\) and its predecessor nodes are labelled (from left to right) with \(\varphi_1, \ldots, \varphi_{n-1}\), then \(\varphi_1 \ldots \varphi_{n-1}\) is an instance of a rule of \(S\). Moreover, the leaves in \(\pi\) are either axioms or else rules with no premise and whose conclusion is an element of a given set of hypotheses \(\Gamma\). A proof tree \(\pi\) with root \(\varphi\) is denoted by \(\pi : \varphi\); moreover, we note \(L(\pi)\) the multi-set of its leaves. We write \(\pi = (\pi_1, \ldots, \pi_n, \varphi)_i\), with \(n \in \mathbb{N}\), the proof tree whose last inference rule is \(i = \varphi_1 \ldots \varphi_{n-1}\) and such that, for every \(i \in \{1, \ldots, n\}\), \(\pi_i\) is the subtree of \(\pi\) leading to \(\varphi_i\). Obviously, for any statement of the form \(\Gamma \vdash \varphi\) in a formal system \(S\), there is an associated proof tree \(\pi : \varphi\) whose leaves are axioms or formulas in \(\Gamma\). We also use the notation \(\pi : \varphi\) to indicate a proof tree whose conclusion is the formula \(\varphi\). Using a standard numbering of the tree nodes by natural number strings, we can refer to positions in a proof tree. Thus, given a proof tree \(\pi\), a position of \(\pi\) is a string \(\omega\) on \(\mathbb{N}\) which represents the path from the root of \(\pi\) to the subtree whose conclusion occurs at that position. This subtree \(\pi'\) is denoted by \(\pi|_\omega\). Let us note \(\text{Pos}(\pi)\) the set of positions of \(\pi\). Given a position \(\omega \in \mathbb{N}^*\) in a proof tree \(\pi\), \(\pi[\pi']_\omega\) is the proof tree obtained from \(\pi\) by replacing the subtree \(\pi|_\omega\) by \(\pi'\). The trees \(\pi|_w\) and \(\pi'\) necessarily have the same root. If \(\pi\) and \(\pi'\) : \(\varphi\) are two proof trees and \(w\) is a leaf position of \(\pi\) such that \(\pi|_w = \varphi\), then we use the expression \(\pi \cdot w\ \pi'\) rather than \(\pi[\pi']_w\). This operation
is called \emph{composition} of \( \pi \) and \( \pi' \) on (leaf) position \( w \). If \( R \) is a set of \( n \)-ary rules then \( R^t \) is the set of proof trees inductively constructed from all rule instances in \( R \), and closed under the composition operation.

### 3  Rewriting formal system

Here, we define an abstract framework of logics for which we will show in the two next sections that there exists a notion of rewrite system with an associated notion of rewriting logic.

Rewriting is a method to reason with binary relations (equality [4, 10], inclusion [15] or other non-symmetric relations [5, 19], the ideal membership problem [9], etc.). These binary relations (the set \( E \) in Definition 1) are defined on sets of elements that are homogeneous but that can be different from one rewriting theory to another (simple words, \( \lambda \)-terms, first-order terms, graphs, polygraphs, etc.). Moreover, the behavior of these binary relations is specified by inference rules. For example, in the equational rewriting setting, the behavior of equality is specified by the reflexivity, transitivity and symmetry rules. If we extend to term equations, we add both context and substitution rules. We can then notice that, in all rewriting theories, rewriting relations are specified thanks to a subset of these inference rules (e.g. substitution, context, reflexivity and transitivity) while others are removed of the process (e.g. symmetry). Moreover, preserved inference rules can be split up into two disjoint sets, called \( RS \) and \( De \), specifying rewriting steps and derivations, respectively. Removed inference rules will be put in the set \( Rmv \). In rewriting, rule instances of \( Rmv \) are removed because they generate basic loops in rewriting process, and then lead to obvious nonterminating rewrite relations. In rewriting logic, rule instances in \( Rmv \) are removed because they are not satisfied by the aimed application domain. For instance, as this has already been observed in the introduction, the approach developed by Meseguer and his followers aims to provide a logical support which unifies models of concurrency. Hence, rewriting logic in [17] is viewed as representing irreversible (i.e. non symmetric) transitions between non-identifiable states.

Finally, these binary predicates can be constrained by other predicates (the set \( P \) in Definition 1) such as for instance the definability predicate \( D \) in partial algebras or the membership predicate \( : \) in the membership equational logic. The inference rules defining the behavior of these extra predicates will be put in the set \( Oth \). The name \( Oth \) is used for other (rules). This leads one to characterize a family of formal systems, called \emph{rewriting formal systems}, which will be able to be used to parameterize rewriting logic.

**Definition 1 (Rewriting formal systems).** A formal system \( S = (F, R) \) is a rewriting formal system (rfs) if
- \( F \) is over an alphabet \( A \) equipped with a partition \( E \uplus P \uplus T \uplus \{"("; ","; ")\} \).
- formulas in \( F \) are of the form \( p(u_1, \ldots, u_n) \) such that \( p \in E \cup P \) and each \( u_i \in T \). Moreover, if \( p \in E \) then \( n = 2 \) (i.e. \( p \) is a binary relation name).
- \( R \) is equipped with a partition \( RS \uplus De \uplus Rmv \uplus Oth \) such that for every \( r \in R \setminus Oth \) (resp. \( r \in Oth \)), all the instances of \( r \) have conclusions of the form \( p(u, v) \) with \( p \in E \) (resp. \( p(u_1, \ldots, u_n) \) with \( p \in P \)).

In the following, a rfs will be noted under its extended form \( (T, E, P, RS, De, Rmv, Oth) \).

**Example 1 (Rewriting formal system underlying the conditional equational logic).**

In this example, we define the rfs which parameterizes the conditional rewriting logic associated to the conditional term rewriting modulo a set of equations. Before defining the rfs for this logic, let us recall some definitions and notations useful to this purpose. A signature \( \Sigma \) is a set of function names, each one equipped with an arity in \( \mathbb{N} \). Given a set of variables \( V \), let us note \( T_\Sigma(V) \) the set of terms with variables in \( V \). Given a signature \( \Sigma \), a set of variables \( V \), and a term \( t \in T_\Sigma(V) \), \( \text{Var}(t) \) denotes the set of variables occurring in \( t \).

Atoms are \( \Sigma \)-equations of the form \( t = t' \) where \( t \) and \( t' \) are terms in \( T_\Sigma(V) \). Well-formed formulas are then conditional formulas of the form \( \alpha_1 \land \ldots \land \alpha_n \Rightarrow \alpha_{n+1} \) where for every \( i, 1 \leq i \leq n+1 \), \( \alpha_i \) is a \( \Sigma \)-equation, and theories are any set of formulas. A substitution is a mapping \( \sigma : V \rightarrow T_\Sigma(V) \). It is naturally extended to terms, equations and conditional formulas.

In order to fit conditional formulas into the definition of rfs which only manipulates predicates, any formula of the form \( c \Rightarrow t = t' \) where \( c \) is a finite conjunction of equations, will be noted \( t =_c t' \). Unconditioned equations \( t = t' \) will be noted \( t =_0 t' \). Hence, in the associated rfs, this gives rise to a family of predicates \( =_c \) indexed by finite conjunctions of equations, and inference rules will be relations on such formulas.

Therefore, given a signature \( \Sigma \) and a set of \( \Sigma \)-equations \( Eq \), we define the rfs \( S = (T, E, P, RS, De, Rmv, Oth) \) for the conditional equational logic as follows: Let \( \Gamma \) be a set of formulas \( t =_c t' \)

- \( T = T_\Sigma(V) \),
- \( E = \{=_c \mid c \text{: finite conjunction} \} \) is a set of equalities with for every \( c \) : 
  - \( \text{conjunction}, =_c \overset{\text{def}}{=} T_\Sigma(V) \times T_\Sigma(V) \) (syntactical definition of equations \(^1\)),
- \( P = \{\approx\} \) with \( \approx \overset{\text{def}}{=} T_\Sigma(V) \times T_\Sigma(V) \),
- \( RS \) is the set defined by the following deduction rules:

\(^1\) Any couple of terms \((t, t')\) is a well-formed equation. In any way, this does not mean that it is true.
Reflexivity for each \( t \in T_\Sigma(V) \),
\[
 t =_\emptyset t
\]

Replacement for each \( t = c t' \in \Gamma \) with \( c = \bigwedge_{1 \leq i \leq n} t_i = t'_i \) and every \( \sigma, \sigma' : V \rightarrow T_\Sigma(V) \),
\[
 \forall x \in \text{Var}(t) \cup \text{Var}(t'), \sigma(x) =_\emptyset \sigma'(x) \quad \forall i, 1 \leq i \leq n, \sigma(t_i) =_\emptyset \sigma(t'_i)
\]
\[
 \sigma(t) =_\emptyset \sigma'(t')
\]

Congruence for each \( t(x_1, \ldots, x_n) \),
\[
 \forall i, 1 \leq i \leq n, t_i =_\emptyset t'_i \quad \frac{t(t_1/x_1, \ldots, t_n/x_n) =_\emptyset t'(t'_1/x_1, \ldots, t'_n/x_n)}{
 t =_\emptyset t'}
\]

Equality
\[
 t \approx u \quad u =_\emptyset v \quad v \approx t' \quad \frac{t =_\emptyset t'}{
}
\]

- **De** is the set defined by the following deduction rule:

  **Transitivity**
\[
 t =_\emptyset t' \quad t' =_\emptyset t'' \quad \frac{t =_\emptyset t''}{
}
\]

- **Rmv** is the set defined by the following deduction rule:

  **Symmetry**
\[
 t =_\emptyset t' \quad \frac{t' =_\emptyset t}{
}
\]

- **Oth** is the set defined by all the standard rules of equational reasoning applied on equations of the form \( t \approx t' \) at which we add the following deduction rule:

  **Axiom**
\[
 t = t' \in Eq \quad \frac{t \approx t'}{
 t \in Eq
}
\]

4 Abstract rewriting

In this section, we recapitulate how to define the notion of rewrite systems and derivations in rfs from [1]. In [1], we also gave a meaning, in the abstract framework of rfs, to the usual notions of effluences and proofs by rewriting (abstractions of peaks and valleys, respectively, usual in term rewriting), termination, Church-Rosser property, etc. From these notions, we then gave sufficient conditions to ensure the fundamental results which underlie rewriting used to generate
canonical rewrite systems, such as Newman’s lemma. Then, this has allowed us
to define a generic completion method à la Knuth-Bendix. We refer the interested
reader to our paper [1] for the complete presentation of these notions, results and
extensions.

**Definition 2 (Rewrite systems).** Let $S = (T, E, RS, De, Rmv, Oth)$ be a
rfs. A $S$-rewrite system $R$ is an $E$-sorted set of binary relations $(\rightarrow_p)_{p \in E}$ on $T$
such that: $\forall p \in E, \ \rightarrow_p \subseteq p$ (compatibility with the syntactic definition of $p$ given
in $S$).

**Remark.** When you are interested in the termination of rewrite systems, you
cannot be satisfied with one binary relation $\rightarrow_p$ for every $p \in E$. This is because
the binary relations in $E$ are not necessarily symmetric. Indeed, when we have
to orient a non-symmetric relation $\sqsubseteq$ according to a reduction order $\geq$ (i.e. a
Noetherian order on terms), two rewrite relations are needed: both the intersection of $\sqsubseteq$ and $\geq$, written $\leftrightarrow_{\sqsubseteq}$ and the intersection of $\sqsubseteq$ and $\leq$, written $\leftrightarrow_{\sqsubseteq}$. This is due to the non-symmetry of $\sqsubseteq$: $\leftrightarrow_{\sqsubseteq} \neq (\leftrightarrow_{\sqsubseteq})^{-1}$. Such a pair of rewrite
relations is called a bi-rewrite system. This have been first observed by J. Levy
and J. Agustí who have opened this research field by applying rewriting to all
pre-orders [15]. In [1], because we were interested to generalize rewriting and
then to generalize the associated termination notions, abstract rewrite systems
are defined for every $p \in E$ with two binary relations $\rightarrow_p$ and $\leftrightarrow_p$. Here, termination
of rewrite systems is not considered. Hence, when dealing with a binary relation $p$ which is non-symmetric, we will suppose that $p$ and $p^{-1}$ belong to $E$. Another interest of such an approach is only to consider a rewriting relation $\rightarrow_p$
for each $p \in E$ what facilitates the presentation of rewriting and rewriting logic
by lightening notations.

**Example 2.** In the rfs developed in Example 1, we can consider the following set
of rules from the signature $\Sigma = (true^0, false^0, 0^0, eq^2, \mod^2, gcd^2)$, which
specifies the greatest common divisor:

$$
gcd(n, m) \rightarrow_{eq}(n \mod m, 0) = true \ m
$$

$$
gcd(n, m) \rightarrow_{eq}(n \mod m, 0) = false \ gcd(m, n \mod m)
$$

As another example, dealing with rewriting modulo a set of equations, we can
consider the following rewrite system from the signature $\Sigma = \{0^0, 1^0, +^2, \times^2\}$
which defines Boolean rings:

$$
Eq = \{ \begin{align*}
x + y & \equiv y + x, \\
x \times y & \equiv y \times x, \\
(x + y) + z & \equiv x + (y + z), \\
(x \times y) \times z & \equiv x \times (y \times z) 
\end{align*} \}
$$

$$
\Rightarrow_{Eq} = \{ \begin{align*}
x + x & \rightarrow 0, \\
x \times x & \rightarrow x, \\
0 + x & \rightarrow x, \\
0 \times x & \rightarrow 0, \\
x \times (y + z) & \rightarrow (x \times y) + (x \times z), \\
1 \times x & \rightarrow x, \\
\end{align*} \}
$$
We could be tempted to define rewriting steps and derivations as the closure of each binary relation \( \rightarrow_p \) under \( RS \)'s and \( De \)'s rule instances, respectively, that is orienting the conclusion of \( RS \)'s and \( De \)'s rule instances in the same direction as all their premises (this is how the standard rewriting relation is built in the unconditioned equational rewriting setting). But, there are many deduction rules which do not satisfy such a condition. For instance, this is not observed by the rule Replacement of the rfs of the conditional equational logic presented in Example 1 that parameterizes conditional rewriting and given by: for each \( t =_c t' \) with \( c = \bigwedge_{1 \leq i \leq n} t_i = t_i' \) and every \( \sigma, \sigma' : V \rightarrow T_{\Sigma}(V) \),

\[
\forall x \in \text{Var}(t) \cup \text{Var}(t'), \sigma(x) \Rightarrow \sigma'(x) \quad \forall i, 1 \leq i \leq n, \sigma(t_i) =_a \sigma(t_i')
\]

\[
\sigma(t) =_a \sigma'(t')
\]

Indeed, when dealing with conditional rewriting rules, we have (at least) three potentially interesting definitions of \( \rightarrow^*_R \): given a rewrite system \( R = (\rightarrow^*_e), e \in \text{equation conjunction} \), let us define \( \Theta = \{ t =_c t' \mid t \rightarrow^*_e t' \in R \} \)

1. **Natural conditional rewriting** \( \sigma(t) \rightarrow^*_R \sigma'(t') \) if for every \( x \in \text{Var}(t) \cup \text{Var}(t') \), \( \sigma(x) \rightarrow^*_R \sigma'(x) \) and for every \( i, 1 \leq i \leq n, \Theta \vdash \sigma(t_i) =_a \sigma(t_i') \).

2. **Join conditional rewriting** \( \sigma(t) \rightarrow^*_R \sigma'(t') \) if for every \( x \in \text{Var}(t) \cup \text{Var}(t') \), \( \sigma(x) \rightarrow^*_R \sigma'(x) \) and for every \( i, 1 \leq i \leq n, \sigma(t_i) \downarrow_a \sigma(t_i') \) where \( \downarrow_a \) means there is a term \( t'' \) such that \( \sigma(t_i) \xRightarrow{\sigma} t'' =_a \sigma(t_i') \) or

3. **Normal conditional rewriting** \( \sigma(t) \rightarrow^*_R \sigma'(t') \) if for every \( x \in \text{Var}(t) \cup \text{Var}(t') \), \( \sigma(x) \rightarrow^*_R \sigma'(x) \) and for every \( i, 1 \leq i \leq n, \sigma(t_i) \xRightarrow{\sigma} \sigma(t_i') \)

After seeing this example, it becomes obvious that some premises of rule instances in \( RS \cup De \) have a special status. For any rule instance \( \tau \in RS \cup De \), we gather its “special” premises in the multi-set \( \mathcal{F}(\tau) \subseteq \mathcal{L}(\tau) \) and call them **fixed leaves**. The definition of these fixed leaves is ad-hoc for each rfs. Therefore, given a deduction rule in \( RS \cup De \), the orientation of its conclusion will only be influenced by the orientation of its fixed leaves. In the next definition, we will only define in the abstract framework, normal rewriting. Both natural and join rewriting can also be abstractly defined. In order to simplify the presentation, we do not present here the abstract form of these notions which, however, can be found in our paper [1].

**Definition 3 (Rewriting step and rewriting relations).** Let \( R \) be a \( S \)-rewrite system for which the set of fixed leaves \( \mathcal{F}(\tau) \) of every rule instance \( \tau \in RS \cup De \) has been specified. For every \( p \in E \), \( \rightarrow^*_R \) and \( \xRightarrow{\sigma} \) are two binary relations on \( T \) defined as the least binary relations (according to the set-theoretical inclusion) inductively defined as follows:
1. \( \rightarrow_P \subseteq \rightarrow_P^R \) and \( \rightarrow_P^R \subseteq \rightarrow_P \), and

2. for every \( \iota : p(t, t') \in RS \) (resp. \( \iota : p(t, t') \in De \)) such that:
   - for every leaf \( p'(u, v) \in FL(i), u \rightarrow_P^v \) (resp. \( u \rightarrow_P v \)), and
   - **Normal rewriting:**
     - for every leaf \( p'(u', v') \in L(i) \setminus FL(i) \) with \( p' \in E \), \( u' \rightarrow_P^v v' \), and
     - for every leaf \( p(t_1, \ldots, t_n) \in L(i) \setminus FL(i) \) with \( p \in P \), we have \( \Theta \vdash p(t_1, \ldots, t_n) \) with \( \Theta = \{ p(u, v) | p \in E \land u \rightarrow_P v \} \)

We note \( \rightarrow^R = \bigcup_{p \in E} \rightarrow_P^R \) and \( \rightarrow^R_\sigma = \bigcup_{p \in E} \rightarrow_P^{\sigma} \).

**Example 3.** From Example 1, for any rule instance \( \iota \in RS \cup De \), \( FL(i) \) contains all its premises of the form \( t =_\varnothing t' \) except if \( \iota \) is an instance of the rule **Replacement**. In this last case, we have \( FL(i) = \{ \sigma(x) =_\varnothing \sigma'(x) \mid x \in Var(t) \cup Var(t') \} \). Therefore, rewriting steps are then defined as follows:

- for every \( t \in T_\Sigma(V) \), \( t \rightarrow_{=_\varnothing} \)

- for every \( t \rightarrow \bigwedge_{i \leq n} u_i = v_i \) \( t' \in R \), and every \( \sigma, \sigma' : V \rightarrow T_\Sigma(V) \), if for every \( x \in Var(t) \cup Var(t') \), \( \sigma(x) \rightarrow_{=_\varnothing} \sigma'(x) \) and for every \( i, 1 \leq i \leq n \), \( \sigma(t_i) \rightarrow_{=_\varnothing} \sigma(t'_i) \) then \( \sigma(t) \rightarrow_{=_\varnothing} \sigma(t') \),

- for every \( t(x_1, \ldots, x_n) \in T_\Sigma(V) \), if for every \( i, 1 \leq i \leq n \), \( t_i \rightarrow_{=_\varnothing} t'_i \) then \( t(t_1/x_1, \ldots, t_n/x_n) \rightarrow_{=_\varnothing} t'(t'_1/x_1, \ldots, t'_n/x_n) \) and

- if \( u \rightarrow_{=_\varnothing} v \) and there exists \( s, t \in T_\Sigma(V) \) such that \( Eq \vdash s \approx u \), and \( Eq \vdash v \approx t \), then \( s \rightarrow_{=_\varnothing} t \).

Note both congruence and replacement rules have many premises. This allows to apply rules in parallel to all arguments of an operator (congruence), or in correspondence of all variables of a rule (replacement). Hence, a single rewrite step can apply various rules in parallel.

For every finite conjunction of equations \( c \neq \emptyset \), the rewriting relation \( \rightarrow_{=_\varnothing} = \rightarrow_{=_\varnothing} = \rightarrow_{=_\varnothing} \). Finally, \( \rightarrow^R \) is the transitive closure of \( \rightarrow_{=_\varnothing} \).
5 Abstract rewriting logic

In rewriting logic, basic axioms are rewrite rules of the form \( t \rightarrow t' \) considered as sequents and inference rules are relations on these basic axioms. Inference rules are simply obtained by replacing in every deductive rule of \( De \cup RS \) formulas of the form \( p(t, t') \) by \( t \rightarrow p \), and by erasing rules in \( Rmv \). Formally, we have

**Definition 4.** Let \( S = (T, E, P, RS, De, Rmv, Oth) \) be a rfs. For any \( r \in RS \cup De \), let us note \( \rightarrow_r = \{ \rightarrow_\iota \mid \iota : p(t, t') \in r \land p \in E \} \) where for every \( \iota = \varphi_1 \ldots \varphi_n \) \( p(t, t') \in r \), \( \rightarrow_\iota \) is the instance \( \varphi_1 \ldots \varphi_n t \rightarrow p t' \) where for every \( i, 1 \leq i \leq n \):

- \( \varphi_i = t_i \rightarrow p_i t'_i \) if \( \varphi_i = p_i(t_i, t'_i) \) with \( p_i \in E \), or
- \( \varphi_i = \varphi_i \), otherwise.

**Definition 5 (Abstract rewriting logic (ARL)).** Let \( S = (T, E, P, RS, De, Rmv, Oth) \) be a rfs. Let \( R = (\rightarrow_p)_{p \in E} \) be a SP-rewrite system. We say that \( R \) entails a sequent \( t \rightarrow_p t' \) and write \( R \vdash \text{Ded} t \rightarrow_p t' \) if and only if \( t \rightarrow_p t' \) can be obtained by the following set \( \text{Ded} \) of deduction rules:

\[
\text{Ded} = \{ \rightarrow_r \mid r \in RS \cup De \} \cup Oth
\]

S-rewrite systems are then theories for the underlying abstract rewriting logic.

**Example 4 (The conditional rewriting logic).** The conditional rewriting logic which formalizes the conditional term rewriting modulo a set of equations is defined as follows:

- sentences are sequents of the form \( t \rightarrow_c t' \) where \( c \) is a finite (possibly empty) conjunction of equations,
- a rewriting theory \( R \) is a set of sequents, and
- a rewriting theory \( R \) entails the sequent \( t \rightarrow_c t' \) if it is obtained by the finite application of the following deduction rules:

**Replacement** for each \( t \rightarrow_c t' \in R \) with \( c = \bigwedge_{1 \leq i \leq n} t_i = t_i \) and every \( \sigma, \sigma' : \Sigma \rightarrow T \),

\[
\forall x \in \text{Var}(t) \cup \text{Var}(t'), \sigma(x) \rightarrow_{=a} \sigma'(x) \land t, \sigma(t) \rightarrow_{=a} \sigma'(t')
\]

**Congruence** for each \( t(x_1, \ldots, x_n) \),

\[
\forall i, 1 \leq i \leq n, t_i \rightarrow_{=a} t'_i
\]

\[
\frac{\sigma(t)}{\sigma(t')}
\]

\[
l(t_1/x_1, \ldots, t_n/x_n) \rightarrow_{=a} l(t'_1/x_1, \ldots, t'_n/x_n)
\]
Equality

\[
\begin{align*}
& t \approx u. \quad u \rightarrow_{=_{\emptyset}} v. \quad v \approx t' \\
& \frac{}{t \rightarrow_{=_{\emptyset}} t'}
\end{align*}
\]

Reflexivity for each \( t \in T_{\Sigma}(V) \),

\[
\frac{}{t \rightarrow_{=_{\emptyset}} t}
\]

Transitivity

\[
\frac{t \rightarrow_{=_{\emptyset}} t' \quad t' \rightarrow_{=_{\emptyset}} t''}{t \rightarrow_{=_{\emptyset}} t''}
\]

or all rule instances in \( \text{Oth} \) given in Example 1.

6 Rewriting coincides with derivability

In this section we answer the question: given a \( S \)-rewrite system \( \mathcal{R} \), when is \( \text{ARL} \) sound and complete for \( \rightarrow_{\mathcal{R}} \)?

From Definition 3, we can observe that the definition of \( \rightarrow_{\mathcal{R}} \) defines a proof strategy which restricts the proof search space by selecting, among proof trees generated from the inference rules of \( \text{Ded} \), those equipped with the following structure: for every proof \( \pi \) there does not exist a subproof \( \ldots \frac{p'(u,v) \in \mathcal{F}(t)}{t \rightarrow_{=_{\emptyset}} t} \ldots t' \) such that \( p'(u,v) \in \mathcal{F}(t) \), \( t \in \text{RS} \) and \( t' \in \text{De} \). Here, we will call such proof trees rewrite trees. Soundness of \( \mathcal{R} \vdash_{\text{Ded}} \) with respect to \( \rightarrow_{\mathcal{R}} \) is obvious because \( \rightarrow_{\mathcal{R}} \) defines particular proof trees which can be generated from inference rules of \( \text{Ded} \).

On the contrary, the completeness of \( \mathcal{R} \vdash_{\text{Ded}} \) with respect to \( \rightarrow_{\mathcal{R}} \) is not ensured in general. The reason is that this requires that, for any statement of the form \( \mathcal{R} \vdash_{\text{Ded}} \ldots \), there is a proof tree satisfying the structure given by Definition 3 of \( \rightarrow_{\mathcal{R}} \). Usually, the completeness is the consequence of a stronger result which defines transformation rules of proof trees to rewrite elementary combinations of inference rules (possibly by duplicating a sub-derivation), and showing that the global procedure which is naturally induced by these basic transformations is terminating, and its normal forms are exactly rewrite trees. What is observed in the standard rewriting logic is that these basic proof tree transformations are recognized as the “distribution” of both congruence, replacement and equality over transitivity. For instance, let us suppose the following situation:

**Repl.**

\[
\frac{\pi_1 : \sigma(x_1) \rightarrow_{=_{\emptyset}} \sigma'(x_1) \ldots \pi_k : \sigma(x_k) \rightarrow_{=_{\emptyset}} \sigma'(x_k) \quad \forall i, 1 \leq i \leq n, \pi_i' : \sigma(t_i) \rightarrow_{=_{\emptyset}} \sigma(t_i')}{\sigma(t) \rightarrow_{=_{\emptyset}} \sigma'(t')}
\]

where \( \text{Var}(t) \cup \text{Var}(t') = \{x_1, \ldots, x_k\} \) and there exists \( j, 1 \leq j \leq k \), such that

\[
\pi_j = \frac{\pi_j' : \sigma(x_j) \rightarrow_{=_{\emptyset}} u_j \quad \pi_j'' : u_j \rightarrow_{=_{\emptyset}} \sigma'(x_j)}{\sigma(x_j) \rightarrow_{=_{\emptyset}} \sigma'(x_j)}
\]
This proof can be transformed as follows. Define the substitution $\sigma''$ as follows:

$$\sigma'' : V \rightarrow T_\Sigma(V)$$

$$x_j \mapsto u_j$$

$$x_i \neq x_j \mapsto \sigma'(x_i) \quad 1 \leq i \leq k$$

We then have:

$$\text{Trans.} \quad \pi''_1 : \sigma(t) \rightarrow_{=} \sigma''(t') \quad \pi''_2 : \sigma''(t') \rightarrow_{=} \sigma'(t')$$

$$\sigma(t) \rightarrow_{=} \sigma'(t')$$

where $\pi''_1$ and $\pi''_2$ are respectively:

$$\text{Repl.} \quad \pi_1 \cdot \sigma(x_1) \rightarrow_{=} \sigma''(x_1) \quad \ldots \quad \pi_k \cdot \sigma(x_k) \rightarrow_{=} u_j \quad \ldots \quad \pi_k \cdot \sigma(x_k) \rightarrow_{=} \sigma''(x_k) \quad \forall i, 1 \leq i \leq n, \pi_i \cdot \sigma(t_i) \rightarrow_{=} \sigma(t_i)$$

$$\text{Ref.} \quad \sigma(t) \rightarrow_{=} \sigma'(t')$$

$$\sigma''(t') \rightarrow_{=} \sigma'(t')$$

Other basic proof tree transformations in the standard rewriting logic follow alike permutations (see Example 6 below). By transforming proof trees into proof terms by using the function symbols and the axioms as operators, as well as operators for reflexivity (unary), congruence (binary), replacement (binary) and transitivity (binary), termination can be proven by rewriting techniques. Indeed, a simple application of recursive path orderings (RPO) (see textbooks such as [4, 10, 13] for a complete and comprehensive definition of RPO) for the precedence ordering

$$\text{Repl.} \times \text{Cong.} \sim \text{Equal.} \times \text{Trans.} \times \text{Refl.}, \text{Oth}$$

shows that repeatedly applying the distribution rules terminates in a rewrite tree.

Here, we are going to formalize in our abstract framework, this notion of “distributing”, and to show that this is sufficient to ensure the expected abstract result of completeness. Moreover, we will observe that the abstraction of this notion of “distributing” is satisfied by all logics to which our abstract framework of rewriting applies (see the examples developed in this paper).

**Definition 6 (Rewrite trees).** Let $S = (T, E, P, RS, De, Rmv, Oth)$ be a rfs. A proof tree $\pi$ is called a rewrite tree if and only if there does not exist a position $p \in \text{Pos}(\pi)$ such that $\pi|_p$ is a proof tree of the form $(\pi_1 : \varphi_1, \ldots, \pi_n : \varphi_n, \varphi)_T$ satisfying:

1. $i \in RS$,

2. there exists $i, 1 \leq i \leq n$, such that $\varphi_i \in \mathcal{FL}(i)$ and $\pi_i$ is of the form $(\pi_{i1}, \ldots, \pi_{ik}, \varphi_i)_T$ with $u_i \in De$. 
A basic and powerful idea underlying most of termination methods such as RPO consists in comparing terms of transformation rules by first comparing their root symbols, and recursively comparing the collection of their immediate subterms. This requires first to assume a well-founded ordering $\succeq$ (the precedence ordering) on atomic elements of proof trees (i.e., rule instances). For our purpose, this precedence ordering $\succeq$ has to satisfy the three following conditions: if we note $Axiom = \{ \iota \mid \iota \in RS \land \iota : \text{axiom} \}$ then

1. $(\iota \in RS \setminus Axiom \land \iota' \in De) \implies \iota \succeq \iota'$
2. $(\iota \in (RS \setminus Axiom) \cup De \land \iota' \in Axiom) \implies \iota \succ \iota'$
3. $(\iota \in RS \cup De \land \iota' \in Oth) \implies \iota \succeq \iota'$

**Remark.** The justification of Condition 2. is that the instances in $Axiom$ are proofs which are already in normal form. For Condition 3., its justification is that the instances in $Oth$ will not generate transformations but can be used in the right-hand side of transformation rules.

However, the above three conditions are not sufficient. Indeed, for the above transformation rule between $Replacement$ and $Transitivity$, to ensure termination, $Replacement$ has to be greater (for $\succ$) than $Congruence$. Hence, some rule instances in $RS$ can also be comparable for $\succ$. However, the order between rule instances in $RS$ cannot be abstractly defined and depends of the rewriting logic of interest. In the following of this section, we then assume a precedence ordering $\succeq$ on rule instances in $RS \cup De$ such that the three above conditions are satisfied.

**Definition 7 (Distributive).** Let $S = (T,E,P,RS,De,\text{Rmv, Oth})$ be a rfs. The set of deductive rules in $Ded$ is said distributive if and only if for every proof tree of the form $(\iota_1, \ldots, \iota_n, t \rightarrow_p t')$ such that:

- each $\iota_i$ is either a sequent $u \rightarrow_{p'} v$ or a formula $p'(u_1, \ldots, u_n)$ with $p' \in P$ or a rule instance of $\bigcup_{\overline{r} \in Ded} \overline{r}$, $\iota \in RS$, and
- there exists $j$, $1 \leq j \leq n$, such that $\iota_j$ is a rule instance $\overline{r'} : u \rightarrow_{p'} v \in Ded$, $\iota' \in De$ and $p'(u,v) \in FL(\iota)$

there exists a rewrite tree $\pi' : t \rightarrow_p t'$ such that for every subtree $(\pi'_1, \ldots, \pi'_m, \varphi)_{\mathcal{E}}$ of $\pi'$: if we note $F = \{ u \rightarrow_{\varphi} v \mid p \in E \land (u,v) \in p \} \cup \{ p(t_1, \ldots, t_n) \mid p \in P \land (t_1, \ldots, t_n) \in p \}$ then

- $\iota \succeq \iota'$
if \(\iota \sim \iota'\) then \((\forall i, 1 \leq i \leq m, \pi'_i \in F \cup Oth \cup Axiom) \wedge L(\iota') \subseteq L(\pi)\)

We note \(\pi \sim \pi'\) such a property between \(\pi\) and \(\pi'\).

Hence, the property to be distributive generates a set of proof tree transformations defined as the closure of \(\sim\) under proof context and composition on leaf position (see the definition in Section 2).

**Example 5.** In the transformation given at the top of this section, the left-hand side is the proof tree

\[\begin{align*}
\text{Repl.} & \quad (\sigma(x_i) \to=\emptyset \sigma'(x_i))_{i \neq j \leq k} \\
\text{Trans.} & \quad \frac{\sigma(x_j) \to=\emptyset u_j \to=\emptyset \sigma'(x_j)}{\sigma(t) \to=\emptyset \sigma'(t')} \\
\text{Repl.} & \quad \sigma(t) \to=\emptyset \sigma'(t')
\end{align*}\]

and the right-hand side is

\[\begin{align*}
\text{Trans.} & \quad \frac{\pi''_1 : \sigma(t) \to=\emptyset \sigma''(t') \pi''_2 : \sigma''(t') \to=\emptyset \sigma'(t')} {\sigma(t) \to=\emptyset \sigma'(t')}
\end{align*}\]

where \(\pi''_1\) and \(\pi''_2\) are respectively:

\[\begin{align*}
\text{Repl.} & \quad \sigma(x_1) \to=\emptyset \sigma''(x_1) \ldots \sigma(x_j) \to=\emptyset u_j \ldots \sigma(x_k) \to=\emptyset \sigma''(x_k) \forall i, 1 \leq i \leq n, \sigma(t_i) \to=\emptyset \sigma(t_i')
\end{align*}\]

\[\begin{align*}
\text{Ref.} & \quad \frac{\sigma''(x_i) \to=\emptyset \sigma'(x_i)} {\sigma''(t') \to=\emptyset \sigma'(t')}
\end{align*}\]

Notice that in both left-hand and right-hand sides, we have removed all the proofs \(\pi_1, \ldots, \pi_k\) and \(\pi'_1, \ldots, \pi'_n\). This is because \(\sim\) defines basic transformation of proof trees.

It is obvious to see that the left-hand side satisfies the three premises of Definition 7. Finally, the right-hand side satisfies both conditions of Definition 7. Indeed, all the occurrences of the rule instances occurring in the right-hand side of the transformation are lower than Replacement for \(\geq\). Moreover, the occurrence of replacement in the right-hand side of the transformation has for direct subtrees leaves whose each of them belongs to the leaves of the left-hand side.

Notice that the property to be distributive is a particular case of RPO. Hence, this property is sufficient to have the expected abstract result of completeness as expressed by the following result:

**Theorem 8.** If \(\text{Ded}\) is distributive then \(\sim\) is terminating and normal forms are rewrite trees.
Proof. Using proof terms for proofs, with the recursive path ordering \( \succ_{\text{rpo}} \) to order proofs induced by the well-founded relation (precedence) \( \succeq \) on rule instances, we show that \( \leadsto \subseteq \succ_{\text{rpo}} \). Let \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \leadsto \pi \) be a transformation rule. By mathematical induction on \( \pi \) let us show that \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \succ_{\text{rpo}} \pi \).

**Basic case.** \( \pi \) is restricted to the sequent \( t \Rightarrow_p t' \). By definition, this means that, \( t \Rightarrow_p t' \) belongs to \( \mathcal{L}(\langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top) \). As simplification orderings have the subterm property, we conclude \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \succ_{\text{rpo}} \pi \), and then by RPO, \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \succ_{\text{rpo}} \pi \).

**General case.** \( \pi \) is of the form \( \langle \pi_1, \ldots, \pi_m, t \Rightarrow_p t' \rangle_\to \). Here two cases have to be considered:

1. \( t' \in \text{De} \). Therefore, by our hypothesis on the precedence ordering \( \succeq \), we know that \( t \supset t' \). But, by induction hypothesis, for every \( i, 1 \leq i \leq n \), \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\to \succ_{\text{rpo}} \pi_i \), and then by RPO, \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \succ_{\text{rpo}} \pi \).

2. \( t' \in \text{RS} \). Here again, two cases have to be considered:

   (a) either \( t \supset t' \). In this case, the proof is related to the one where \( t' \in \text{De} \) just above.

   (b) \( t \sim t' \). By distributivity, we know that each \( \pi_i \in F \cup \text{Oth} \cup \text{Axiom} \). Moreover, we know that \( \mathcal{L}(t') \subseteq \mathcal{L}(\langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top) \). Therefore, we have \( \mathcal{L}(t_1, \ldots, t_n) \supset \pi_i \). By RPO, we then conclude \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \succ_{\text{rpo}} \pi \).

By definition, all critical situations defined by a proof tree \( \langle t_1, \ldots, t_n, t \Rightarrow_p t' \rangle_\top \) satisfying the three premises of Definition 7 are transformed into rewrite trees. Moreover, we have just shown that these transformations allow to transform any proof into a rewrite tree in a finite number of steps. Since rewrite trees are precisely \( \supset \), for every statement \( \mathcal{R} \vdash t \Rightarrow_p t' \), we ensure that there exists a rewrite tree \( \pi : t \Rightarrow_p t' \) with axioms among \( \mathcal{R} \).

**Example 6.** It remains to handle the case of congruence and equality with transitivity. This leads to the two following transformations:

1. Cong. \( \frac{t_1 \Rightarrow_p t_j \land t_1 \neq t_j \land t_j \Rightarrow_p t_j'}{t_1/x_1, \ldots, t_n/x_n \Rightarrow_p t_1/x_1, \ldots, t_j/x_j, \ldots, t_n/x_n} \) \( \pi_1 \) \( \pi_2 \)

2. Cong. \( \frac{t_1 \Rightarrow_p t_j \land t_j \Rightarrow_p u_j}{t_1/x_1, \ldots, t_j/x_j, \ldots, t_n/x_n \Rightarrow_p t_1/x_1, \ldots, u_j/x_j, \ldots, t_n/x_n} \)

where \( \pi_1 \) and \( \pi_2 \) are respectively:
For the precedence ordering $\succeq$ given above in this section, these transformation rules imply that $\text{Ded}$ is distributive. By Theorem 8, we then have the expected result, that is deduction for this rewriting logic coincides with the conditional rewriting as defined in Example 3.

In [17], the author established equivalences between the proof terms of rewriting logic and both step and interleaving sequences (defined by sequential steps - see Section 2.3 in [17]). Hence, the rewriting logic as defined in [17] and recalled as a running example in this paper, deals with “true concurrency”. In [17], this result is expressed as follows:

**Theorem 9 (Meseguer).** For each concurrent $\mathcal{R}$-rewrite $[t] \rightarrow [t']$, either $[t] = [t']$, or there exists an $n \in \mathbb{N}$ and a chain of one-step (concurrent) rewrites $[t] \rightarrow [t_1] \rightarrow \ldots \rightarrow [t_n] \rightarrow [t']$.

This result exactly expresses that for every proof statement $\mathcal{R} \vdash [t] \rightarrow [t']$, there exists a rewrite tree $\pi : [t] \rightarrow [t']$. By Example 6, we have just shown a stronger result which establishes that every proof tree can be transformed into a rewrite tree. Hence, Theorem 9 is a corollary of Theorem 8 applied to the rewriting logic developed in [17] for the precedence ordering and transformation rules given in Examples 6. We can then see Theorem 8 as a generalization of equivalences between the proof terms of rewriting logic.

### 7 Semantics

In this section, we will answer the following question: *what are the models of abstract rewriting logic?*

To achieve this purpose, we follow the approach initiated in [8] to define a model-theoretical presentation of rewrite theories in terms of the models of a suitable theory of the first-order logic. As this was observed in [8], two kinds of models can be defined:

\[ [t] \] is the equivalence class of the term $t$ for the congruence generated by equations in $\text{Eq}$.\footnote{\[ [t] \] is the equivalence class of the term $t$ for the congruence generated by equations in $\text{Eq}$.}
1. Reachability models which focus just on what elements of $T$ can be reached from a certain element $t$ via sequences of rewriting, ignoring how the rewrites can lead to them.

2. Provability models which focus, unlike reachability models, both on what elements of $T$ can be reached from a certain element $t$ via sequences of rewriting and how the rewrites can lead to them.

### 7.1 Reachability models

**Definition 10 (Reachability relation).** Let $\mathcal{R}$ be a $\mathcal{S}$-rewrite system. Let us define $\rightarrow_{\mathcal{R}}$ the $E$-sorted set of binary relations $\rightarrow_{\mathcal{R}}^p$ on $T$ as follows:

$$\forall p \in E, \ t \rightarrow_{\mathcal{R}}^p t' \iff \mathcal{R} \vdash_{\text{Ded}} t \rightarrow_p t'$$

**Remark.** Although the notation is the same, the reachability relation has not to be confused with the rewrite relation $\rightarrow_{\mathcal{R}}$ which has been defined in Definition 3. However, when rewriting coincides with derivability in ARL (see Section 6), both above binary relations denote the same subset of $T \times T$.

The mono-sorted first-order predicate logic is sufficient to define a model-theoretical presentation of the reachability relation associated to a $\mathcal{S}$-rewrite system $\mathcal{R}$.

**Definition 11 (The theory $\text{Reach}(\mathcal{R})$).** Let $\mathcal{S} = (T, E, P, RS, De, Rmv, Oth)$ be a rfs. Let $\mathcal{R}$ be a $\mathcal{S}$-rewrite system. The first-order theory $\text{Reach}(\mathcal{R})$ contains the signature $\Sigma_{\mathcal{R}} = (\mathcal{F}, \mathcal{C}, \mathcal{P})$ and the set $Ax$ of sentences defined respectively, as follows:

- **Signature:**
  
  - $\mathcal{F} = \emptyset$, $\mathcal{C} = T$, and
  
  - $\mathcal{P} = \{\rightarrow_p^2 | p \in E\} \cup P$ ("$\rightarrow_p^2$" denotes the predicate name of arity 2 associated to the binary relation name $p$)

- **Sentences:**
  
  - $\forall p \in E, \forall t, t' \in T, t \rightarrow_p t' \in \mathcal{R} \implies t \rightarrow_p t' \in Ax$, and
  
  - $\forall t = \sum_{i=1}^{n} \varphi_i \in \text{Ded}, \ \bigwedge_{1 \leq i \leq n} \varphi_i \implies \varphi \in Ax$.

Definition 11 calls for some comments:

---

3. $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{P}$ are respectively the set of function, constant and predicate names.
The above theory \( \text{Reach}(\mathcal{R}) \) contains many (usually an infinite number of) sentences in \( Ax \). The reason is that we associate a sentence to each rule instance. Consequently, all sentences in \( Ax \) are ground, that is, all terms which occur in sentences are elements of \( T \). But, as \( T \) is not equipped with any inductive structure for a set of operators and basic elements such as function names in signatures and variables, elements in \( T \) are simple constants, and then the set of operators with arity greater than 1 is empty.

For logics which parameterize existing rewriting logics, a shorter description can be given. Indeed, as this is usual in most logics (anyway all logics used in computing science and mathematics) the underlying inference relation \( \vdash \) is generated from a finite set of deductive rules, that is a single form with infinitely many instantiations. This allows to denote all the instances by a set of generic forms (up to meta-variable renaming). In this case, generic terms which occur in such deductive rules can be replaced by variables in the sentences of \( Ax \).

**Example 7.** As explained in the above comments, we are going to benefit from the fact that inference rules given in Example 4 are free with generators in a set of variables \( V \), to give a shorter description of the theory \( \text{Reach}(\mathcal{R}) \) than the one given in Definition 11. We then have the following description: let \( \mathcal{R} \) be a rewriting theory over a signature \( \Sigma \),

- The signature \( \Sigma_{\mathcal{R}} = (\mathcal{F}, \mathcal{C}, \mathcal{P}) \) is defined by:
  - \( \mathcal{F} = \{ f^n \mid f \in \Sigma, n \geq 1 \}, \mathcal{C} = \{ f^0 \mid f \in \Sigma \}, \) and
  - \( \mathcal{P} = \{ \rightarrow^2 \mid c \text{ finite conjunction} \} \cup \{ \equiv^2 \} \)

- Sentences in \( Ax \) are: to indicate that a term \( t \) has its variables among \( \{x_1, \ldots, x_n\} \), we write \( t(x_1, \ldots, x_n) \), and then \( t(t_1, \ldots, t_n) \) is the term obtained from \( t \) by replacing all variable occurrences \( x_i \) by \( t_i \):
  - sentences in \( \mathcal{R} \) and \( \mathcal{Eq} \),
  - for every \( t \rightarrow^=_a \bigwedge_{i \leq n} t_i = t'_i \in \mathcal{R} \),
    \[
    \bigwedge_{j \leq m} (y_j \rightarrow_a y'_j) \land \bigwedge_{i \leq n} \left( t_i(y_1, \ldots, y_m) \rightarrow_a t'_i(y_1, \ldots, y_m) \right) \Rightarrow t(y_1, \ldots, y_m) \rightarrow_a t'(y_1', \ldots, y_m')
    \]
  - for every \( f^n \in \Sigma \),
    \[
    \bigwedge_{i \leq n} (x_i \rightarrow_a x'_i) \Rightarrow f(x_1, \ldots, x_n) \rightarrow_a f(x'_1, \ldots, x'_n)
    \]
\documentclass[12pt]{article}
\usepackage{amsmath,amssymb}
\begin{document}
\begin{itemize}
\item $x \approx y \land y \rightarrow_{a} z \land z \approx w \Rightarrow x \rightarrow_{a} w$
\item $x \rightarrow_{a} x$
\item $x \rightarrow_{a} y \land y \rightarrow_{a} z \Rightarrow x \rightarrow_{a} z$
\item usual equality axioms for the predicate $\approx$
\end{itemize}

\textbf{Definition 12 (Reachability models).} Let $\mathcal{R}$ be a $\mathcal{SP}$-rewrite system. A reachability model of $\mathcal{R}$ is any first-order structure of $\text{Reach}(\mathcal{R})$, that is the first-order structures over $\Sigma_{\mathcal{R}}$ that satisfy the axioms in $\text{Reach}(\mathcal{R})$.

\textbf{Example 8.} From the theory $\text{Reach}(\mathcal{R})$ developed in Example 7, a model $\mathcal{M}$ is a set $U$ together for any finite conjunction $c$ with a binary relation $\rightarrow^{M}$. For the empty conjunction, $\rightarrow^{M}$ is reflexive and transitive. Therefore, the carrier $U$ of $\mathcal{M}$ can be naturally regarded as a category. This is how the semantics of the standard rewriting logic has been defined in [17]. In this case, all syntactical notions can be interpreted in the language of category theory. Indeed, from the congruence rule, it is obvious to show that the semantics of operator names and then terms with variables, are functors. Therefore, rewritings become natural transformations between functors. Actually, the semantics of rewrite rules in the language of category theory is more complicated because of conditions. Indeed, it is obvious to show from the replacement rule, that the semantics of unconditional rewrite rules of the form $t \rightarrow_{a} t'$ is a natural transformation $\gamma: t^{M} \Rightarrow t'^{M}$.

When conditions occur, rewrite rules define natural transformations between functors resulting of the composition of each functor associated to each term occurring in the conclusion and the subequalizer functor used to solve conditions.\footnote{It is well-known that solutions of substitutions are equalizer between both morphisms associated to terms of equations [11]. Here, terms are semantically denoted by functors. Therefore, subequalizer is the generalization of the notion of equalizer of two functors. We refer the reader to [17] for the complete exposition of this notion.}

The theory $\text{Reach}(\mathcal{R})$ provides a first-order model for the reachability relation. This model is given by:

\textbf{Definition 13 (Herbrand’s model).} Let $\mathcal{R}$ be a $\mathcal{S}$-rewrite system. Let $\mathcal{I}$ be the first-order structure over $\Sigma_{\mathcal{R}}$ defined as follows:

\begin{itemize}
\item the carrier $\mathcal{I}$ is $T$,
\item for every $t \in \mathcal{C}$, $t^{\mathcal{I}} = t$,
\item for every $p \in \mathcal{E}$, $\rightarrow^{\mathcal{I}} = \rightarrow_{\mathcal{R}}$, and
\item for every $p \in \mathcal{P}$, $(t_1, \ldots, t_n) \in p^{\mathcal{I}} \Leftrightarrow \mathcal{R} \vdash_{\text{Def}} p(t_1, \ldots, t_n)$.
\end{itemize}
\end{document}
Theorem 14 (Completeness I). For $\mathcal{R}$ a $S$-rewrite theory,

$$\text{Reach}(\mathcal{R}) \models t \rightarrow_p t' \iff \mathcal{R} \vdash_{\mathsf{Ded}} t \rightarrow_p t'$$

Proof. The only if part. By Definition 13, if $I \models t \rightarrow_p t'$ then $\mathcal{R} \vdash_{\mathsf{Ded}} t \rightarrow_p t'$. It then remains to show that $I$ is a reachability model. By Definition 13, sentences in $\mathcal{R}$ are obviously satisfied. Let $\bigwedge_{1 \leq i \leq n} \varphi_i \Rightarrow \varphi$ be an axiom of $\mathsf{Ax}$. Assume that for every $i$, $1 \leq i \leq n$, $I \models \varphi_i$. By Definition 10 and Definition 13, this means that $\mathcal{R} \vdash_{\mathsf{Ded}} \varphi_i$. Since $\bigwedge_{1 \leq i \leq n} \varphi_i \Rightarrow \varphi \in \mathsf{Ax}$, this means that there exists a deductive rule $\varphi_1 \ldots \varphi_n \varphi$. Therefore, we have $\mathcal{R} \vdash_{\mathsf{Ded}} \varphi$, and then we conclude $I \models \varphi$.

The if part. This is proved by mathematical induction on the structure of proof trees. Assume a reachability model $\mathcal{M}$, and show that $\mathcal{M} \models t \rightarrow_p t'$.

- Basic case. Either $t \rightarrow_p t' \in \mathcal{R}$ or $t \rightarrow_p t' \notin \mathcal{R}$. By Definition 11, we have in both cases that $t \rightarrow_p t' \in \mathsf{Ax}$ and then $\mathcal{M} \models t \rightarrow_p t'$.

- General case. There is a proof $(\pi_1 : \varphi_1, \ldots, \pi_n : \varphi_n, t \rightarrow_p t')_i$. By definition, $i$ is a deductive rule of the form $\varphi_1 \ldots \varphi_n \varphi$. By induction hypothesis, for every $i$, $1 \leq i \leq n$, $\mathcal{M} \models \varphi_i$. By Definition 11, we have $\bigwedge_{1 \leq i \leq n} \varphi_i \Rightarrow t \rightarrow_p t' \in \mathsf{Ax}$. We then conclude $\mathcal{M} \models t \rightarrow_p t'$.

Actually, we have a more general completeness result:

**Theorem 15 (Completeness II).** For $\mathcal{R}$ a $S$-rewrite theory,

$$\text{Reach}(\mathcal{R}) \models \varphi \iff \mathcal{R} \vdash_{\mathsf{Ded}} \varphi$$

$\varphi$ is over Reach($\mathcal{R}$), that is, is either of the form $t \rightarrow_p t'$ or of the form $p(t_1, \ldots, t_n)$ with $p \in P$.

Proof. The proof is similar to the proof of Theorem 14.

**Theorem 16.** $I$ is initial in Reach($\mathcal{R}$).

Proof. Let $\mathcal{M}$ be a reachability model. Note $\nu : I \rightarrow M$ be the interpretation of constants of $T$ in $M$ (the carrier of $\mathcal{M}$). Since $\mathcal{M}$ is a reachability model, by Theorem 14, we have for every $t \rightarrow_p t'$ and every statement $\mathcal{R} \vdash_{\mathsf{Ded}} p(t_1, \ldots, t_n)$ that $\mathcal{M} \models t \rightarrow_p t'$ and $\mathcal{M} \models p(t_1, \ldots, t_n)$. Hence, we have both $(\nu(t), \nu(t')) \in p^{\mathcal{M}}$ and $(\nu(t_1), \ldots, \nu(t_n)) \in p^{\mathcal{M}}$. We then conclude that $\nu$ is an homomorphism. Finally, by Definition 13, $\nu$ is unique.

5 Recall that axioms in $\mathsf{Ax}$ are ground.
7.2 Provability models

As usual, the idea is to attach a proof term to each sequent, so-called decorated sequents. In the rewriting logic developed in [17] and presented as a running example in this paper (this is also true for its extension developed in [8]) proof terms are built from variables, operators in signatures (congruence), labels of rewrite rules in $R$ (replacement), and ";" to compose rewritings (transitivity).

Here, inference rules (proofs) cannot be implicitly taken into account (built) from operators of signatures, variables and other primitive symbols such as ";". The reason is no information is given on both the structure of elements in $T$ and the form of inference rules. Therefore, a symbol operator $f_ι : s_ϕ_1 \times \ldots \times s_ϕ_n \rightarrow s_ϕ_{n+1}$ has to be associated to any inference rule $ι = ϕ_1 \ldots ϕ_n ϕ_{n+1} \in Ded$ where for every $i$, $1 \leq i \leq n + 1$, $s_ϕ_i$ is a sort name which semantically will contain every proof tree $π : ϕ$. For rewriting rules in a $S$-rewrite system $R$, we will index rewriting rules by labels. Therefore, this leads to extend $S$-rewrite systems as follows:

**Definition 17 (Labelled rewrite system).** Let $L$ be a set. A labelled $S$-rewrite system $R$ is an $E$-sorted set of ternary relations $(→_p)_p \in E$ on $L \times T \times T$. For every $(l,t,t')$ in $→_p$, we will use the notation $l : t →_p t'$.

This naturally leads to specify provability models in the many-sorted first-order predicate logic:

**Definition 18 (The theory $Proof(R)$).** Let $S = (T, E, P, RS, De, Rmv, Oth)$ be a rfs. Let us note $S$ the underlying formal system associated to $S$ (see Remark ??). Let $R$ be a labelled $S$-rewrite system. The first-order theory $Proof(R)$ contains the signature $Σ_R = (S,F,P)$ and the set $Ax$ of sentences defined respectively, as follows:

- Signature:
  - $S = \{ s_ϕ \mid ϕ \in S \}$,
  - $F = \bigcup \{ f_ι : s_ϕ_1 \times \ldots \times s_ϕ_n \rightarrow s_ϕ \mid ι = ϕ_1 \ldots ϕ_n ϕ_{n+1} \in Ded \}$
  - $P = \{ Pr_ϕ : s_ϕ \mid ϕ \in S \}$

- Sentences:
  - $∀l : s_{t →_p t'} \in F$, $Pr_{t →_p t'}(l)$, and
  - $∀l = ϕ_1 \ldots ϕ_n \in Ded$, $\bigwedge_{1 \leq i \leq n} Pr_ϕ(x_ϕ_i) \rightarrow Pr_ϕ(f_ι(x_ϕ_1,\ldots,x_ϕ_n)) \in Ax$.

where $x_ϕ_i$ is a variable of sort $s_ϕ_i$. 
Theorem 19. \( \text{Proof}(\mathcal{R}) \) admits an initial model.

Proof. \( \text{Proof}(\mathcal{R}) \) is a universal Horn theory. Theorem 19 is then a corollary of Mal’cev’s theorem (see [16] for an exposition of Mal’cev’s result).

\( \text{Proof}(\mathcal{R}) \) is complete with respect to inference rules of ARL as expressed by the following result:

Theorem 20 (Completeness I). For every rewrite theory \( \mathcal{R} \), we have:

\[
\mathcal{R} \vdash_{\text{Ded}} t \rightarrow_{p} t' \iff \exists \pi \in T_{\Sigma_{\mathcal{R}} \mathcal{R} \mathcal{R} \rightarrow_{p} t'}, \text{Proof}(\mathcal{R}) \models P_{t \rightarrow_{p} t'}(\pi)
\]

Proof. The if part directly results from Definition 18. The only if part is proved by mathematical induction on the structure of proof trees. Actually, we will prove that \( \text{Proof}(\mathcal{R}) \vdash P_{t \rightarrow_{p} t'}(\pi) \) by using the Hilbert calculus for the first-order logic. As the Hilbert calculus for the first-order logic is complete, we will then have that \( \text{Proof}(\mathcal{R}) \models P_{t \rightarrow_{p} t'}(\pi) \).

– Basic case. Two cases have to be considered:

1. \( l : t \rightarrow_{p} t' \in \mathcal{R} \). In this case, \( P_{t \rightarrow_{p} t'}(l) \in \text{Proof}(\mathcal{R}) \).

2. there is a rule \( \iota : t \rightarrow_{p} t' \in \text{Ded} \). In this case, \( P_{t \rightarrow_{p} t'}(f_{\iota}) \in \text{Proof}(\mathcal{R}) \).

– General case there is a proof tree \( \pi = (\pi_1 : \varphi_1, \ldots, \pi_n : \varphi_1, t \rightarrow_{p} t') \). By induction hypothesis, for every \( i, 1 \leq i \leq n \), there exists a ground term \( \pi'_i \in T_{\Sigma_{\mathcal{R}} \mathcal{R} \mathcal{R} \rightarrow_{p} t'} \), such that \( \text{Proof}(\mathcal{R}) \vdash P_{t \rightarrow_{p} t'}(\pi'_i) \). Therefore, by assuming that we use the Hilbert calculus for the first-order logic, by instantiation and modus-ponens, we have \( \text{Proof}(\mathcal{R}) \vdash P_{t \rightarrow_{p} t'}(f_{\iota}(\pi'_1, \ldots, \pi'_n)) \).

Observe that in the proof of Theorem 20, we actually recursively define a transformation \( Tr \) from proof trees to proof terms which associates to \( l : t \rightarrow_{p} t' \) the proof term \( l \) and associates to \( (\pi_1, \ldots, \pi_n, \varphi) \), the proof term \( f_{\iota}(Tr(\pi_1), \ldots, Tr(\pi_n)) \).

From the definition of \( \text{Proof}(\mathcal{R}) \) the above completeness result holds for any formula \( \varphi \) of the underlying formal system \( S \), that is:

Theorem 21 (Completeness II). For any rewrite theory \( \mathcal{R} \), we have:

\[
\mathcal{R} \vdash \varphi \iff \exists \pi \in T_{\Sigma_{\mathcal{R}}(X)}_{\varphi}, \text{Proof}(\mathcal{R}) \models P_{\varphi}(\pi)
\]

\( X \) is any set of variables which contains the subset \( \{ x_{\varphi} \mid \varphi \in S \} \).

Proof. The proof is similar to that of Theorem 20.
In general, many proofs concluding that $R \vdash \text{Ded} t \rightarrow pt'$ are possible. In Section 6, we have shown that if Ded is deductive then some proofs can be computationally equivalent with rewrite trees as normal forms. For instance, in [8, 17], proofs belonging to a same equivalence class represent different interleaved sequences (represented by the transitivity rule) for the same concurrent computation (defined by applying an arbitrary number of time both congruence and replacement rules).

The theory $\text{Proof}(R)$ can be defined in order to only consider rewrite tree. This is how Reach$(R)$ and $\text{Proof}(R)$ has been defined in [8].

**Definition 22.** With the notations and hypothesis of definition 18, let us define the theory $\text{Proof}(R) = (\Sigma_R, Ax)$ as follows:

- **Signature**
  - $S$ and $F$ are defined as in Definition 18
  - $P = \{Pr^1_\varphi : s_\varphi \in S\} \cup \{Pr_\varphi : s_\varphi \in S\}$

- **Sentences**
  1. $\forall l : s_{t\rightarrow pt} \in F, Pr^1_{t\rightarrow pt}(l)$
  2. $Pr^1_\varphi(x_\varphi) \Rightarrow Pr_\varphi(x_\varphi)$
  3. $\forall l = \frac{\varphi_1, \ldots, \varphi_n}{\varphi} \in \text{De} \cup \text{Oth}, \bigwedge_{1 \leq i \leq n} Pr_{\varphi_i}(x_{\varphi_i}) \Rightarrow Pr_{\varphi}(f_\rightarrow_\varphi(x_{\varphi_1}, \ldots, x_{\varphi_n}))$
  4. $\forall l = \frac{\varphi_1, \ldots, \varphi_n}{\varphi} \in \text{RS}, \bigwedge_{\varphi_i \in L(i)} Pr^1_{\varphi_i}(x_{\varphi_i}) \land \bigwedge_{\varphi_i \in L(i) \setminus F(L(i))} Pr_{\varphi_i}(x_{\varphi_i}) \Rightarrow Pr^1_{\varphi}(f_\rightarrow_\varphi(x_{\varphi_1}, \ldots, x_{\varphi_n}))$

**Theorem 23 (Completeness).** If Ded is distributive then for every rewrite theory $R$, we have:

$$R \vdash_{\text{Ded}} t \rightarrow pt' \iff \exists \pi \in T_{\Sigma_R, t\rightarrow pt'}, \text{Proof}(R) \vDash Pr_{t\rightarrow pt'}(\pi)$$

**Proof.** The if part. By Definition 22, proof terms $\pi \in T_{\Sigma_R, t\rightarrow pt'}$ for which $Pr_{t\rightarrow pt'}(\pi)$ holds, are rewrite trees. Since Ded is distributive, we have shown in Theorem 8 that there always exists a rewrite tree $\pi$ that concludes $R \vdash_{\text{Ded}} t \rightarrow pt'$.

The only if part. By Theorem 8, there exists a rewrite tree $\pi$ that concludes $R \vdash_{\text{Ded}} t \rightarrow pt'$. For the rest of the proof, we prove by mathematical induction on the structure of rewrite trees that $\text{Proof}(R) \vdash Pr_{t\rightarrow pt'}(\pi)$ where $\vdash$ is the inference relation which has been generated by using the Hilbert calculus.
Basic case. \( \pi \in RS^\# \). By Definition 22, we have \( \text{Proof}(R) \vdash Pr_{t ightarrow p}(\pi) \), and then \( \text{Proof}(R) \vdash Pr_{t ightarrow p}(\pi) \) (instantiation of Axiom 2).

General case. Assume a rewrite tree \( \pi \) of the form \( (\pi_1 : \varphi, \ldots, \pi_n : \varphi_n, t \rightarrow p t') \) where \( t \in \text{Dev \cup Oth} \). By the induction hypothesis, for every \( i, 1 \leq i \leq n \), there exists a ground term \( \pi'_i \in T_{\Sigma, \varphi_i} \) such that \( \text{proof}(R) \vdash Pr_{\varphi_i}(\pi'_i) \). Therefore, by instantiation and modus-ponens, we have:

\[
\text{Proof}(R) \vdash Pr_{\pi_1 \rightarrow p}(\pi'_1, \ldots, \pi'_n)
\]

8 Examples

8.1 Constrained rewriting logic

The rewriting logic defined in this section is parameterized by an equational logic where equations are constrained by a set of first-order formulas (constraints). The followed presentation of this rewriting logic takes inspiration from [13]. Therefore, a constraint is a well-formed formula built over a first-order signature \( \Sigma = (F, P) \) and a set of variables \( V \), where \( F \) and \( P \) are sets of function and predicate names, respectively, each one equipped with an arity in \( \mathbb{N} \).

A constraint language \( L_K[\Sigma, V] \) is given by:

- a set of constraints built over \( \Sigma \) and \( V \). Given a constraint \( c \in L_K[\Sigma, V] \), \( \text{Var}(c) \) denotes the set of free variables of \( c \).
- a \( \Sigma \)-structure \( K \) together with a solution mapping that associates to each constraint \( c \) the set of variable assignment \( \text{Sol}_K(c) = \{ \nu : V \rightarrow K \mid K \models \nu(c) \} \).

In the sequel, we suppose that \( L_K[\Sigma, V] \) always contains the empty constraint \( T \).

A constraint \( c \) is valid in \( L_K[\Sigma, V] \), written \( L_K[\Sigma, V] \models c \), if and only if for any \( \nu : V \rightarrow K \), \( \nu \in \text{Sol}_K(c) \).

When \( K \) is isomorphic to a Herbrand structure \(^6\), that is of particular interest for theorem proving, any solution in \( \text{Sol}_K(c) \) coincides with a substitution \( \sigma \) such that \( K \models \sigma(c) \). Such substitutions are called symbolic solution. The set of all symbolic solutions of a constraint \( c \) is denoted \( SS_K(c) \).

Formulas in the constrained equational logic are sentences of the form \( t = t' \mid c \) where \( t, t' \in T_F(V) \) and \( c \in L_K[\Sigma, V] \) with \( \Sigma = (F, P) \).

Given a first-order signature \( \Sigma = (F, P) \), a set of variables \( V \), a set of \( \Sigma \)-equations \( Eq \) and a \( \Sigma \)-structure \( K \) isomorphic to a Herbrand structure, we define the rfs \( S \) by the tuple \( (T, E, RS, De, Rmv, Oth) \) such that: if \( \Gamma \) be a set of formulas \( t = c t' \) such that \( \text{Var}(c) \subseteq \text{Var}(t) \cup \text{Var}(t') \) then

\(^6\) that is a quotient of the ground term algebra when constraints are equations. In this case, the constraint language is called primal constraint language.
- \( T = T_F(V) \).
- \( E = \{ \approx_c \mid c \in \mathcal{L}_K[\Sigma, V] \} \) s.t. \( \approx_c \triangleq T_F(V) \times T_F(V) \) (syntactic definition of constrained equations).
- \( \mathcal{P} = \{ \approx \} \) s.t. \( \approx \triangleq T_F(V) \times T_F(V) \).
- \( RS \) is the set defined by the following deduction rules:

  **Reflexivity** for each \( t \in T_F(V) \)

  \[
  \begin{array}{c}
  t =_T t
  \end{array}
  \]

  **Replacement** for each \( t =_c t' \in \Gamma \) and every \( \sigma, \sigma' : V \to T_F(V) \)

  \[
  \forall x \in \text{Var}(t) \cup \text{Var}(t'), \; \sigma(x) =_c \sigma'(x) \quad \Rightarrow \\
  \sigma(t) =_c \sigma'(t')
  \]

  where \( c' = \sigma(c) \land \big\lor_{x \in \text{Var}(t) \cup \text{Var}(t')} c_x \) and \( SS_K(c') \neq \emptyset \).

  **Congruence** for each \( t(x_1, \ldots, x_n) \)

  \[
  \forall i, 1 \leq i \leq n, \; t_i =_c t'_i \quad \Rightarrow \\
  t(t_1/x_1, \ldots, t_n/x_n) =_c t(t'_1/x_1, \ldots, t'_n/x_n)
  \]

  where \( c' = c_1 \land \ldots \land c_n \) and \( SS_K(c') \neq \emptyset \).

  **Equality**

  \[
  \begin{array}{c}
  t \approx u \; u =_c v \; v \approx t' \quad \Rightarrow \\
  t =_c t'
  \end{array}
  \]

- \( De \) is the set defined by the following deduction rule:

  **Transitivity**

  \[
  \begin{array}{c}
  t =_c t' \; t' =_c t'' \quad \Rightarrow \\
  t =_c t''
  \end{array}
  \]

  where \( SS_K(c \land c') \neq \emptyset \).

- \( Rmv \) is the set defined by the following deduction rule:

  **Symmetry**

  \[
  \begin{array}{c}
  t =_c t' \quad \Rightarrow \\
  t' =_c t
  \end{array}
  \]

- \( Oth \) is the set defined by all the standard rules of equational reasoning applied to equations of the form \( t \approx t' \) at which we add the following deduction rule:
Axiom \[ \frac{t = t' \in Eq}{t \approx t'} \]

For any rule instance \( t \in RS \cup De, FL(i) \) contains all its premises of the form \( t = c \).

A rewrite system is a \( L_K[\Sigma, V] \)-indexed set of binary relations \( \rightarrow_{w,c} \subseteq T_F(V) \times T_F(V) \).

Rewriting steps are then defined as follows:

- for every \( t \in T_F(V) \), \( t \rightarrow_{\Leftarrow} t \),
- for every \( t \rightarrow_{e} t' \in R \), and every \( \sigma, \sigma' : V \rightarrow T_F(V) \), if for every \( x \in Var(t) \cup Var(t') \), \( \sigma(x) \rightarrow R \sigma'(x) \) and \( SS_K(c') \neq \emptyset \) with \( c' = \sigma(c) \land \bigwedge_{x \in Var(t) \cup Var(t')} c_x \) then \( \sigma(t) \rightarrow_{\Leftarrow} \sigma(t') \),
- for every \( t(x_1, \ldots, x_n) \in T_F(V) \), if for every \( i \), \( 1 \leq i \leq n \), \( t_i \rightarrow_{\Leftarrow} t_i' \), and \( SS_K(c') \neq \emptyset \) with \( c' = c_1 \land \ldots \land c_n \), then \( t(t_1/x_1, \ldots, t_n/x_n) \rightarrow_{\Leftarrow} t(t_1'/x_1, \ldots, t_n'/x_n) \), and
- if \( u \rightarrow_{\Leftarrow} v \) and there exists \( s, t \in T_S(V) \) such that \( Eq \models s \approx u \), and \( Eq \models v \approx t \), then \( s \rightarrow_{\Leftarrow} t \).

\( \rightarrow_{\Leftarrow} \) is the least binary relation that contains \( \rightarrow_{w,c} \) and satisfies:

- if \( t \rightarrow_{\Leftarrow} t', t' \rightarrow_{\Leftarrow} t'' \), \( c = c_1 \land c_2 \) and \( SS_K(c) \neq \emptyset \), then \( t \rightarrow_{\Leftarrow} t'' \).

The associated rewriting logic is then defined by the following inference rules: if \( R \) is a set of sequents of the form \( t \rightarrow_{w,c} t' \) then

**Reflexivity** for each \( t \in T_F(V) \)

\[ \frac{}{t \rightarrow_{\Rightarrow} t} \]

**Replacement** for each \( t \rightarrow_{w,c} t' \in R \) and every \( \sigma, \sigma' : V \rightarrow T_F(V) \)

\[ \frac{\forall x \in Var(t) \cup Var(t'), \sigma(x) \rightarrow_{w,c} \sigma'(x)}{\sigma(t) \rightarrow_{w,c} \sigma(t')} \]

where \( c' = \sigma(c) \land \bigwedge_{x \in Var(t) \cup Var(t')} c_x \), and \( SS_K(c') \neq \emptyset \).
Congruence for each \( t(x_1, \ldots, x_n) \)

\[
\forall i, 1 \leq i \leq n, \ t_i \rightarrow_c t_i' \Rightarrow \ t(t_1/x_1, \ldots, t_n/x_n) \rightarrow_c t(t'_1/x_1, \ldots, t'_n/x_n)
\]

where \( c' = c_1 \land \ldots \land c_n \) and \( SS_K(c') \neq \emptyset \).

Equality

\[
\frac{t \approx u \Leftrightarrow v \approx v'}{t \rightarrow_c t'}
\]

Transitivity

\[
\frac{t \rightarrow_c t' \quad t' \rightarrow_c t''}{t \rightarrow_c c' \quad t''}
\]

where \( SS_K(c \land c') \neq \emptyset \).

We can easily adapt the transformation rules given in Example 6, and then show that rewriting coincides with derivability in this rewriting logic.

In full generality, it is not possible to directly specify \( \text{Reach}(\mathcal{R}) \) and \( \text{Proof}(\mathcal{R}) \) in a first-order logic. The problem is to deal with satisfiability conditions of constraints in the structure \( \mathcal{K} \) that occur in inference rules. However, because we have assumed that \( \mathcal{K} \) is isomorphic to a Herbrand structure, a constrained rewrite rule \( t \rightarrow_c t' \) schematizes the following set of rewrite rules:

\[ S(t \rightarrow_c t') \{ \sigma(t) \rightarrow \sigma(t') \mid \sigma \in SS_K(c) \} \]

By structural induction on proof trees, we have:

\[ \mathcal{R} \vdash t \rightarrow_c t' \iff \forall \sigma \in SS_K(c), \ S(\mathcal{R}) \vdash^\beta \sigma(t) \rightarrow_c \sigma(t') \]

where \( \vdash^\beta \) is the inference relation generated from the inference rules of the rewriting logic given in Example 4.

Hence, the class of models of \( \text{Reach}(\mathcal{R}) \) and \( \text{Proof}(\mathcal{R}) \) are respectively isomorphic to the class of models of \( \text{Reach}(S(\mathcal{R})) \) and \( \text{Proof}(S(\mathcal{R})) \). By Theorems 14, 16, 19, 20, and 23, both theories admit initial models and the constraint rewriting logic describes in this section is complete w.r.t. its model theories.

8.2 Membership rewriting logic with frozen operators

The rewriting logic defined in this section is parameterized by a generalization of the conditional Membership Equational Logic (MEL), called \( \text{MEL with frozen operators} \) [8].

The conditional membership equational logic (MEL) belongs to the family of algebraic specification formalisms that have been defined to extend basic algebraic specifications in order to support subsorts and partially of function symbols. Before presenting the rfs for conditional membership equational logic with frozen operators, let us recall the basic notions and notations of this logic.

A MEL signature with frozen operators (called generalized MEL signature in [8]) is a triple \((K, \Sigma, S)\) (just \( \Sigma \) in the following) where:
- $K$ is a set of kinds,
- $\Sigma = (K, F)$ is a standard many-kinded signatures where each function name $f : k_1 \times \ldots \times k_n \to k$ is together with a set $\Phi(f) \subseteq \{1, \ldots, n\}$ of frozen arguments positions, and
- $S$ is a $K$-indexed family of sets $S_k$ (so called $K$-set).

Given a $K$-set $V$ of variables, for every $k \in K$, $T_\Sigma(V)_k$ is the standard set of terms of kind $k$, free with generators in $V$, and $T_\Sigma(V)$ is the $K$-indexed family $(T_\Sigma(V)_k)_{k \in K}$. Let us define $\Phi$ and $\nu$ the two binary relations on $T_\Sigma(V)$ as follows:

$$\Phi(t, t') \iff \exists p \in \mathbb{N}, \exists 1 \leq i \leq p, \exists \alpha = \alpha_1 \cdot \alpha_2 \in Pos(t), \begin{cases} t' = t_{i_1} \\ t_{i_1} = f(t_1, \ldots, t_p) \wedge i \in \Phi(f) \end{cases}$$

$$\nu(t, t') \iff \exists \alpha \in Pos(t), t' = t_{i_1} \wedge -\Phi(t, t')$$

Let us define $\Phi(t) = \{x \mid \Phi(t, x)\}$ and $\nu(t) = \{x \mid \nu(t, x)\}$.

Atoms are either equations $t = t'$ where $t$ and $t'$ are terms of the same kind, or membership formula $t : s$ where $t$ is a term of kind $k$ and $s \in S_k$. In [8], conditions of rewrite rules are increased to allow equations, memberships and rewritings. This leads naturally to consider in the underlying rfs, three kinds of $K$-indexed family of equality predicates:

1. $\approx_k$ to make rewritings modulo a set of equations $Eq$,
2. $\equiv_k$ to increase conditions in order to allow equations, and
3. $=_k$ to denote equations which will be transformed into rewritings.

Conditional formulas are then any sentence $\alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \alpha$ where each $\alpha_i$ ($1 \leq i \leq n$) is either of the form $t_i =_k t'_i$, or $t_i \equiv_k t'_i$ or $t_i :_k s_i$, and $\alpha$ is of the form $t =_k t'$. A substitution is a $K$-indexed family of applications $\sigma_k : V_k \to T_\Sigma(V)_k$. It is naturally extended to terms and formulas.

Given a MEL signature $\Sigma$ and a set of equations $Eq$, we define the rfs $S$ by the tuple $(T, E, RS, De, Rmv, Oth)$ such that: if $\Gamma$ be a theory in MEL with frozen operators then

$$- T = T_\Sigma(V) \cup (\bigcup_{k \in K} S_k),$$
$$- E = \{=_k,c \mid k \in K, c : \text{finite conjunction} \} \text{ s.t. } =_k,c \overset{df}{=} T_\Sigma(V)_k \times T_\Sigma(V)_k$$

(syntactic definition of equations),
- \( P = \{ \cdot, \equiv_k, \approx_k \mid k \in K \} \) s.t. \( \cdot \overset{df}{=} T_\Sigma(V)_k \times S_k \) (syntactic definition of memberships), and \( \equiv_k, \approx_k \overset{df}{=} T_\Sigma(V)_k \times T_\Sigma(V)_k \).

- \( RS \) is the set defined by the following deduction rules:

**Reflexivity** for each \( k \in K \) and each \( t \in T_\Sigma(V)_k \),

\[
\frac{}{t = k, \emptyset t}
\]

**Replacement** for each \( t =_{k,c} t' \) with \( c = \bigwedge_{i \in I} t_i \equiv_k t_i' \land \bigwedge_{j \in J} j \equiv_k s_j \land \bigwedge_{l \in L} t_l =_{k,\emptyset} t_l' \) and all substitutions \( \sigma, \sigma' \),

\[
\forall i \in I, \sigma(t_i) \equiv_k \sigma(t_i') \quad \forall j \in J, \sigma(t_j) \equiv_k s_j \quad \forall l \in L, \sigma(t_l) =_{k,\emptyset} \sigma(t_l')
\]

\[
\forall x \in \Phi(t) \cup \Phi(t'), \sigma(x) = \sigma'(x) \quad \forall x \in \nu(t) \cap \nu(t'), \sigma(x) =_{\emptyset} \sigma'(x)
\]

\[
\sigma(t) =_{k,\emptyset} \sigma'(t')
\]

**Congruence** for each \( t(x_1, \ldots, x_n) \) with \( x_i \in V_{k_i} \), if we note \( I \subseteq \{1, \ldots, n\} \) and \( J = \{1, \ldots, n\} \setminus I \) such that \( \Phi(t) = \{x_i \mid i \in I\} \) and \( \nu(t) = \{x_j \mid j \in J\} \), then

\[
\forall i \in I, t_i \equiv_k t_i' \quad \forall j \in J, t_j =_{k,\emptyset} t_j'
\]

\[
\frac{}{t(t_1/x_1, \ldots, t_n/x_n) =_{k,\emptyset} t'(t_1'/x_1, \ldots, t'_n/x_n)}
\]

**Equality1**

\[
\frac{t \approx_k u, u =_{k,\emptyset} v, v \approx_k t'}{t =_{k,\emptyset} t'}
\]

- \( De \) is the set defined by the following deduction rule:

**Transitivity**

\[
\frac{t =_{k,\emptyset} t', t' =_{k,\emptyset} t''}{t =_{k,\emptyset} t''}
\]

- \( Rmv \) is the set defined by the following deduction rule:

**Symmetry**

\[
\frac{t =_{k,\emptyset} t'}{t' =_{k,\emptyset} t}
\]

- \( Oth \) is the set defined by all the standard rules of equational reasoning for each of the predicates \( \equiv_k \) and \( \approx_k \) at which we add the two following deduction rules:

**Axiom**

\[
\frac{t = t' \in Eq, t, t' \in T_\Sigma(V)_k}{t =_{k} t'}
\]
Equality

\[
t \cong_k u \quad u \equiv_k v \quad v \cong_k t'
\]

For any rule instance \( \iota \in RS \cup De, \mathcal{FL}(\iota) \) contains all its premises of the form \( t \equiv_{k,0} t' \) except if \( \iota \) is an instance of the rule Replacement. In this last case, \( \mathcal{FL}(\iota) = \{ (s(x) \equiv \sigma'(x) \mid x \in \nu(t) \cap \nu(t')) \} \). Therefore, if we note \( CnJ \) the set of all finite conjunctions of atoms, then a rewrite system \( \mathcal{R} \) is a \( K \times CnJ \)-indexed set of binary relations \( \rightarrow_{=_{k,c}} \subseteq T_{\Sigma(V)} \times T_{\Sigma(V)} \). Rewriting steps are then defined as follows:

- for every \( t \in T_{\Sigma(V)} \), \( (t, t) \in \rightarrow_{=_{k,\#}} \),
- \( \rightarrow_{=_{k,\#}} \subseteq \rightarrow_{=_{k,0}} \),
- for every \( t \rightarrow_{k,c} t' \in \mathcal{R} \) with \( c = \bigwedge_{i \in I} t_i \equiv_{k_i} t'_i \land \bigwedge_{j \in J} t_j : s_j \land \bigwedge_{l \in L} t_l =_{k_l,0} t'_l \) and all substitutions \( \sigma, \sigma' \), if for every \( x \in \nu(t) \cap \nu(t') \), \( \sigma(x) \rightarrow_{=_{k,\#}} \sigma'(x) \) and:
  - \( \forall i \in I, \Theta \vdash \sigma(t_i) \equiv_{k_i} \sigma(t'_i) \),
  - \( \forall j \in J, \Theta \vdash \sigma(t_j):s_j \),
  - \( \forall x \in \Phi(t) \cup \Phi(t'), \Theta \vdash \sigma(x) \equiv_{=_{k,\#}} \sigma'(x) \), and
  - Normal rewriting, \( \forall l \in L, \sigma(t_l) \rightarrow_{=_{k,\#}} \sigma(t'_l) \)
then \( \sigma(t) \rightarrow_{=_{k,\#}} \sigma(t') \),
- for every \( f : k_1 \times \ldots \times k_n \rightarrow k \in \Sigma \), if for every \( 1 \leq i \leq n \), \( t_i \rightarrow_{=_{k,\#}} t'_i \) then \( f(t_1, \ldots, t_n) \rightarrow_{=_{k,\#}} f(t'_1, \ldots, t'_n) \), and
- if \( u \rightarrow_{=_{k,\#}} v \) and there exists \( s, t \in T_{\Sigma(V)} \) such that \( E\varphi \vdash s \cong_k u \), and \( E\varphi \vdash v \cong_k t \), then \( s \rightarrow_{=_{k,\#}} t \).

The associated rewriting logic is then defined by the following inference rules:

- **Reflexivity** for each \( k \in K \) and each \( t \in T_{\Sigma(V)} \),

\[
t \rightarrow_{=_{k,\#}} t
\]

- **Replacement** for each \( t \rightarrow_{=_{k,c}} t' \in \mathcal{R} \) with \( c = \bigwedge_{i \in I} t_i \equiv_{k_i} t'_i \land \bigwedge_{j \in J} t_j : s_j \land \bigwedge_{l \in L} t_l =_{k_l,0} t'_l \) and all substitutions \( \sigma, \sigma' \),

\[
\forall i \in I, \sigma(t_i) \equiv_{k_i} \sigma(t'_i) \quad \forall j \in J, \sigma(t_j) : s_j \quad \forall l \in L, \sigma(t_l) \rightarrow_{=_{k,\#}} \sigma(t'_l)
\]

\[
\forall x \in \Phi(t) \cup \Phi(t'), \sigma(x) \equiv_{k,c} \sigma'(x) \quad \forall x \in \nu(t) \cap \nu(t'), \sigma(x) \rightarrow_{=_{k,\#}} \sigma'(x)
\]

\[
\sigma(t) \rightarrow_{=_{k,\#}} \sigma(t')
\]
$k_x$ is the kind of the variable $x$ (i.e. $x \in V_{k_x}$)

**Congruence** for each $t(x_1, \ldots, x_n)$ with $x_i \in V_{k_i}$, if we note $I \subseteq \{1, \ldots, n\}$ and $J = \{1, \ldots, n\} \setminus I$ such that $\Phi(t) = \{x_i \mid i \in I\}$ and $\nu(t) = \{x_j \mid j \in J\}$, then

$$\forall i \in I, t_i \equiv_{k_i} t'_i \quad \forall j \in J, t_j \rightarrow_{=_{k_j}} t'_j$$

$t(t_1/x_1, \ldots, t_n/x_n) \rightarrow_{=_{k}} t(t'_1/x_1, \ldots, t'_n/x_n)$

**Equality1**

$$t \approx_{k} u \quad u \rightarrow_{=_{k}} v \quad v \approx_{k} t'$$

$$t \rightarrow_{=_{k}} t'$$

**Transitivity**

$$t \rightarrow_{=_{k}} t' \quad t' \rightarrow_{=_{k}} t''$$

$$t \rightarrow_{=_{k}} t''$$

Both rules of **Replacement** and **Congruence** given in this section are simply extension of both same rules given in Example 4. Hence, the transformation rules given in Example 6 can be easily adapted to this rewriting logic. For the preceding ordering

**Repl. $\searrow$ Cong. $\sim$ Equality1 $\searrow$ Trans. $\searrow$ Refl.**

these basic transformations are distributive and then terminating by Theorem 8. Hence, for the rewriting logic developed in this section, rewriting coincides with derivability.

In [8], the membership equational logic has been used to specify both theories $\text{Reach}(\mathcal{R})$ and $\text{Proof}(\mathcal{R})$. As the membership equational logic does not deal with predicates except equality and membership, to specify $\text{Reach}(\mathcal{R})$ (resp. $\text{Proof}(\mathcal{R})$) in [8], it has been added for any kind $k \in K$ of the MEL signature $\Sigma$ which underlies the rfs, a new kind $[\text{Pair}_k]$ (resp. $[\text{Rw}_k]$) with three sorts $\text{Ar}_0^k$, $\text{Ar}_1^k$, and $\text{Ar}_k$ (resp. two sorts $\text{Rw}_1^k$ and $\text{Rw}_k$), and two operators $\rightarrow: \text{Ar}_k \times \text{Ar}_k \rightarrow [\text{Pair}_k]$ and $\rightarrow: [\text{Pair}_k] \times [\text{Pair}_k] \rightarrow \text{Pair}_k$ (resp. $\rightarrow: \text{Ar}_k \times \text{Ar}_k \rightarrow [\text{Rw}_k]$ and $\rightarrow: [\text{Rw}_k] \times [\text{Rw}_k] \rightarrow \text{Rw}_k$). The kind $[\text{Pair}_k]$ (resp. $[\text{Rw}_k]$) contains all rewritings (resp. the proofs of concurrent computations), and $\text{Ar}_0^k$, $\text{Ar}_1^k$ and $\text{Ar}_k$ (resp. $\text{Rw}_1^k$ and $\text{Rw}_k$) denote respectively, idle rewrites, one-step rewrites and rewrites of arbitrary length. Finally, $\rightarrow$ and $\rightarrow$ denote respectively, rewritings and composition of rewritings.

In [8], both $\text{Reach}(\mathcal{R})$ and $\text{Proof}(\mathcal{R})$ are specified by considering rewrite trees (i.e. concurrent computations defined by proof trees where **Replacement**, **Congruence** and **Equality1** never occur under **Transitivity**). Hence, the theory

7 Actually, in [8], to specify the theory $\text{Proof}(\mathcal{R})$, the operator $\rightarrow$ is specified by means of two operators $s, t : [\text{Rw}_k] \rightarrow k$ defining respectively, the source and the target of rewritings.
Proof($\mathcal{R}$) in [8] follows Definition 22 where $Rw_k^1$ and $Rw_k$ are the analogous of $Pr^1_{\rightarrow_{k=\emptyset}}$ and $Pr_{\rightarrow_{k=\emptyset}}$. Hence, if we note $MELProof(\mathcal{R})$ and $Proof(\mathcal{R})$ the MEL proof theory defined in [8] and the first-order theory defined by following Definition 22, we have:

$$MELProof(\mathcal{R}) \models \pi : Rw_k \land s(\pi) = t \land t(\pi) = t' \iff Proof(\mathcal{R}) \models Pr^1_{\rightarrow_{k=\emptyset}} t'(\pi)$$

### 8.3 Timed rewriting logic

Time rewriting logic extends the standard unconditional rewriting logic in order to deal with time-sensitive systems. As usual, parallelism is handled by allowing to apply rules in parallel to all arguments of an operator (congruence), or in correspondence of all variables of a rule (replacement). The basic difference is that the standard rewriting logic is an asynchronous logic whereas time rewriting logic is a formalism with a strong synchronization mechanism which will be provided by both synchronous replacement and synchronous congruence rules. The consequence is that rewriting logic proof terms and interleaving sequences are not equivalent (see below), that is derivability does not coincide with rewritings.

Time in this logic is abstractly modeled by Archimedean monoids, i.e., any set $R$ together with a distinguished element $0$, an internal law $+$ and a partial ordering $\geq$ such that:

- $0$ is neutral for $+$,
- $+$ is associative, and
- **Archimedean property** for every non zero element $r_1 \in R$ and every element $r_2 \in R$, there exists $n \in \mathbb{N}$ such that $r_1 + \ldots + r_1 > r_2$.

In [14], it is supposed that there exists an equational axiomatization $SP_{Time} = (\Sigma_{Time}, Eq_{Time})$ of any monoid $(R, 0, +, \geq)$ used in time rewriting logic such that $\Sigma_{Time}$ is the sorted signature $\{\{\text{Time}, \text{Bool}\}, \{0, +, \geq\}\}$, and $R$ is isomorphic to $T_{\Sigma_{Time}/\equiv}$, where $\equiv$ is the least congruence generated by the equations of

$$Eq_{Time} = \{t = t'|t, t' \in T_{\Sigma_{Time}\cup\{\mathcal{E} \rightarrow_{\mathcal{E} Time}|r \in R\}} \land R \models t = t'\}$$

Therefore, given a sorted signature $\Sigma$ containing $\Sigma_{Time}$, an Archimedean monoid $(R, +, 0, \geq)$ equationally axiomatized by $(\Sigma_{Time}, Eq_{Time})$, a set of equations $Eq$ over $\Sigma$ and a set of variables $V$, we define the rfs $\mathcal{S} = (T, E, P, RS, Dc, Rmv, Oth)$ as follows:

---

8 We refer the interesting reader to [8] for the complete presentation of both theories.

9 This property is required in order to solve Zeno’s paradox.
- $T = T_\Sigma(V)$.
- $E = T_{\Sigma + \Sigma_{Time}}$ with for every $r \in T_{\Sigma + \Sigma_{Time}}$, $r \overset{def}{=} T_\Sigma(V) \times T_\Sigma(V)$. $^{10} t \ r \ t'$ means that $t$ evolves to $t'$ in time $r$.
- $P = \{ \approx \}$ with $\approx \overset{def}{=} T_\Sigma(V) \times T_\Sigma(V)$ where $\Sigma' = \Sigma \cup \Sigma_{Time}$.
- $RS$ is the set defined by the following deductive rules:
  
  **Synchronous replacement** for each $t \ r \ t'$ and every substitutions $\sigma, \sigma'$,
  
  \[
  \forall x \in \text{Var}(t) \cup \text{Var}(t'), \sigma(x) \ r \ \sigma'(x) \quad \Rightarrow \quad \sigma(t) \ r \ \sigma'(t')
  \]

  **Synchronous congruence** for each $t(x_1, \ldots, x_n)$,
  
  \[
  \forall i, 1 \leq i \leq n, \ t_i \ r \ t_i' \quad \Rightarrow \quad t(t_1/x_1, \ldots, t_n/x_n) \ r \ t(t_1'/x_1, \ldots, t_n'/x_n)
  \]

  **Equality**
  
  \[
  t \approx u \ r \ \approx \ r' \ u \ r \ v \approx t' \quad \Rightarrow \quad t \ r \ t'
  \]

  **Transitivity**
  
  \[
  t \ r \ t', \ t' \ r' \ t'' \quad \Rightarrow \quad t \ (r + r') \ t''
  \]

  **Rmv** is empty.

  **Oth** is the set defined by all the standard rules of equational reasoning applied on equations of the form $t \approx t'$ at which we add the following deduction rule:

  **Axiom**
  
  \[
  t = t' \in Eq \cup Eq_{Time} \quad \Rightarrow \quad t \approx t'
  \]

  Both reflexivity and symmetry rules are dropped in order to model the necessity of change in time, that is component of precesses cannot stand idle and $t \ r \ t'$ does not necessarily mean that $t' \ r \ t$.

  From all what has already been done for the standard rewriting logic in this paper, defining time rewriting logic and rewriting (i.e. rewriting steps and rewriting relation) from the above rfs is pure formality and then is left to the reader.

$^{10}$ Here, we are overloading the notations because we use time terms in $T_{\Sigma + \Sigma_{Time}}$ as binary relation names.
As observed above, rewriting does not coincide with derivability in time rewriting logic. This is because both reflexivity is dropped and all rewritings in synchronous replacement and congruence take the same time $r$. A way to re-obtain such a result would be, either to move 11 Synchronous replacement and Synchronous congruence in $Dc$, or to add in the above rfs the following deductive rule:

**Reflexivity**

\[
\frac{}{t =_0 t}
\]

and to relax both rules Synchronous replacement and Synchronous congruence as follows:

**Replacement** for each $t =_{r_0} t'$ and every substitutions $\sigma, \sigma'$, if $\text{Var}(t) \cup \text{Var}(t') = \{x_1, \ldots, x_k\}$ then

\[
\forall 1 \leq i \leq k, \sigma(x_i) =_{r_i} \sigma'(x_i) \quad r' = \max\{r_j \mid 0 \leq j \leq k\}
\]

\[
\sigma(t) =_{r'} \sigma(t')
\]

**Congruence** for each $t(x_1, \ldots, x_n)$,

\[
\forall 1 \leq i \leq n, t_i =_{r_i} t_i' \quad r' = \max\{r_j \mid 1 \leq j \leq n\}
\]

\[
t(t_1/x_1, \ldots, t_n/x_n) =_{r} t(t_1'/x_1, \ldots, t_n'/x_n)
\]

In [14], semantics is defined by the category of functional dynamic models that satisfy the labelled rewriting rules. A functional dynamic model is defined by a couple $(A, \Gamma_0)$ where $A$ is a $\Sigma$-algebra such that $A_{\mathbb{N}_{\text{Time}}} = R$ and $\Gamma_0 = \{\gamma_{\pi} \subseteq A \times R \times A \mid \pi \in T_{\Sigma_R}(X)_\nu\}$, 12 A functional dynamic model $(A, \Gamma_0)$ satisfies $Pr_{t \rightarrow t'}(\pi)$ if and only if $(t^A, r, t'^A) \in \gamma_{\pi}$ where $t^A : (V \rightarrow A) \rightarrow A$ is the mapping which from a valuation $\nu : V \rightarrow A$ is inductively defined on the structure of terms as follows:

- $x^A(\nu) = \nu(x)$ for every $x \in V$, and

- $f(t_1, \ldots, t_n)^A(\nu) = f^A(t_1^A(\nu), \ldots, t_n^A(\nu))$

We can easily show that if we note $FDAlg(R)$ the class of functional dynamic algebras satisfying the labelled rewriting rules in $R$, then we have the following result:

**Proof** ($R$) $\models Pr_{t \rightarrow t'}(\pi) \iff \forall (A, \Gamma_0) \in FDAlg(R), (t^A, r, t'^A) \in \gamma_{\pi}$

Hence, all the results of initiality and completeness given in [14] are corollaries of Theorems 19 and 20.

11 This is the approach followed in [14]. Hence, a one-step sequential rewrite is every conclusion of any proof tree only composed of the Equality rule (called “timed compatibility” in [14]).

12 See the definition of the first-order signature $\Sigma_R$ in Section 7.
9 Conclusion

In this paper, we have shown the existence of a notion of rewriting logic for any rewriting theory satisfying the conditions of a general framework of rewriting. This has given rise to an abstract form of rewriting logic for which we have studied the model theoretical semantics, and given an initiality theorem and two theorems proving respectively the soundness and completeness of the abstract rewriting logic with respect to this semantics. We have also shown that derivability coincides with rewriting for distributive rewriting logics. This last result generalizes an important result of rewriting logic devoted for concurrent and distributed computations [8, 17] which states that concurrency can be reduced to interleaving.

Four examples illustrate both the good choices of abstraction and the scope of our generic framework. Among other, we have shown that the present work went beyond the generalization of rewriting logic defined in [8] by instantiating our generic framework with the rewriting logic over the membership equational logic with frozen operators. This has been obtained by only generalizing the transformation that from the equational logic has resulted in the rewriting logic.

In order to validate our approach, we are continuing to check that we can indeed cover other already known extensions of the rewriting logic. One of these extensions is more particularly interesting: the rewriting logic with probabilities [6]. Indeed, the authors in [6] defined the rewriting logic with probabilities by only considering the operational semantics. Instantiating the rewriting logic with probabilities in our generic framework would then be able to allow to study what is the notion of model of a given probabilistic rewrite theory, and among the four questions presented in this paper, the ones which are satisfied in this context.

References


