Constrained rewriting in recognizable theories

Tony Bourdier, Horatiu Cirstea

Constrained rewriting in recognizable theories

Tony Bourdier[1] and Horatiu Cirstea[2]

[1] INRIA Nancy Research Center – Pareo Team
615, rue du Jardin Botanique
54600 Villers-lès-Nancy, France
Tony.Bourdier@inria.fr,
www home page: http://www.tony-bourdier.fr

[2] LORIA & Nancy Unisersité
Campus Scientifique – BP 239
54506 Vandoeuvre-lès-Nancy Cedex, France
Horatiu.Cirstea@loria.fr,
www home page: http://www.loria.fr/ cirstea/

Abstract. Rewriting has long been shown useful for equational reasoning but its expressive power is not always appropriate for certain situations, as for instance for specifying partial functions and exceptions or when dealing with relations over terms. That is why some generalizations of rewriting, such as ordered rewriting, class rewriting, strategic rewriting, conditional or constrained rewriting, have emerged. In particular, constraints over terms are very suitable to define sets of terms thanks to logic formulae. Most of the works on constrained rewriting focus on order, equality, disequality and membership constraints. We extend in this paper the usual notion of constrained rewrite systems by constraining rules not only with classical constraints but with any first order formula and perform rewriting according to an interpretation of functional and predicate symbols. We then show that the specification of such interpretations with tree automata offers us a framework sufficiently powerful to build an algorithm for solving a particular case of constrained matching.

1 Introduction

It is a common claim that rewriting is ubiquitous in computer science and mathematical logic. The rewriting concept appears from the very theoretical settings to the very practical implementations. Rewriting is used in semantics in order to describe the meaning of programming languages but also to perform deduction when describing by inference rules a theorem prover or a constraint solver. It is of course central in systems making the notion of rule an explicit object, like expert systems, programming languages based on equational logic, algebraic specifications, functional programming and transition systems. It is hopeless to try to be exhaustive and the cases we have just mentioned show part of the huge diversity of the rewriting concept.

The central idea of rewriting is to use rewriting to mechanize as much as possible equational reasoning by using oriented axioms called rewrite rules. Several extensions of the basic rewriting has been proposed. For instance, conditional rewrite systems provide a way to handle partial operations and case analysis while class rewriting [Hue80,PS81] can be

used to handle semantic data structures which can be modeled using equational axioms (that cannot be oriented without loosing the termination property of reduction). Rewriting with constraints [KK89] emerged as a unified way to cover the previous extensions but goes beyond this since it is very suitable to define sets of terms thanks to logic formulae. Most of the works on constrained rewriting focus on order, equality, disequality and membership constraints. Anti-pattern rewriting [KKM07] and rewriting in many-sorted algebra [Com92] are two instances of the general constrained rewriting.

Constrained rewrite systems have been used to model systems restricted by security policies [BCJK09] for which the basic rewriting has shown its limitations. Rewriting is performed in this case *w.r.t.* an algebra representing the current state of the system and thus, even if the specification of the policy remains unchanged during the evolution of the system, its semantics (*i.e.* the associated rewriting relation) changes as the system evolves. The formalism defined in [BCJK09] specifies the conditions for solving the underlying constraints in algebras with a finite carrier. We propose here a more general formalism that can deal with infinite structures. The rewriting and consequently the underlying constrained matching are performed *w.r.t.* theories whose formulae are recognizable using tree automata. In this formalism, we characterize the constrained patterns for which a solution can be computed and recognized.

The paper is organized as follows. The next section introduces some basic notions and notations used in the paper. Section 3 gives a definition of constrained rewriting. Section 4 introduces the notion of recognizable theory and defines some operators over recognizable predicates. In Section 5 we define the matching problems intrinsic to constrained rewriting and we identify a class of problems that can be solved with the algorithms we propose. We finally conclude and discuss possible perspectives to this work.

2 Preamble

This section briefly recall basic notions used in this paper; more details can be found in [Coh81,Hin05] for logic considerations, [BN98,KK06] for rewriting considerations and [GS97,C$^+$08,Jac96] for more details on tree automata.

We call *alphabet of symbols* any set E fitted with an application ar from E to $\mathbb{N}$ called *arity*. We say that $u \in E$ is n-ary and we write $u_{/n}$ if $ar(u) = n$. A signature Σ is a pair of two disjoint alphabets of symbols: a set of *function symbols* $\mathcal{F}$ and a set of *predicates* $\mathcal{P}$. $\mathcal{T}_{\mathcal{F},\mathcal{X}}$ is the set of *terms* built from a given alphabet of function symbols $\mathcal{F}$ and a set of variables $\mathcal{X}$. The set of variables occuring in $t \in \mathcal{T}_{\mathcal{F},\mathcal{X}}$ is $\mathcal{V}ar(t)$. If any variable of $\mathcal{V}ar(t)$ occurs only once in t, t is said to be *linear*. If $\mathcal{V}ar(t)$ is empty, t is called a ground term and $\mathcal{T}_{\mathcal{F}}$ denotes the set of all ground terms. A *position* of a term t is a finite sequence of positive integers describing the path from the root of t to the root of the sub-term at that position. The empty sequence representing the root position is denoted by ε. $\mathcal{P}os(t)$ is called the set of positions of t. $t_{|\omega}$, resp. $t(\omega)$, denotes the subterm of t, resp. the symbol of t, at position ω. We denote by $t\,[s]_\omega$ the term t with the subterm at position ω replaced by s. We call *substitution* any mapping from $\mathcal{X}$ to $\mathcal{T}_{\mathcal{F},\mathcal{X}}$ which is the identity except over a finite set of variables $\mathcal{D}om(\sigma)$ called *domain* of σ extended to an endomorphism of $\mathcal{T}_{\mathcal{F},\mathcal{X}}$. σ is often denoted by $\{x \mapsto \sigma(x) \mid x \in \mathcal{D}om(\sigma)\}$. If $\mathcal{C}odom(\sigma) \subseteq \mathcal{T}_{\mathcal{F}}$, σ is said to be ground. For any substitutions σ and σ', $\sigma \circ \sigma'$ denotes the substitution $\{x \mapsto \sigma(\sigma'(x)) \mid x \in \mathcal{D}om(\sigma) \cup \mathcal{D}om(\sigma')\}$. A *rewrite*

rule (over $\mathcal{F}$) is a pair $(lhs, rhs) \in \mathcal{T}_{\mathcal{F},\mathcal{X}} \times \mathcal{T}_{\mathcal{F},\mathcal{X}}$ such that $\mathcal{V}ar(lhs) \subseteq \mathcal{V}ar(rhs)$ and a *rewrite system* is a set of rewrite rules $\mathcal{R}$ inducing a *rewriting relation* over $\mathcal{T}_{\mathcal{F}}$, denoted by $\rightarrow_{\mathcal{R}}$ and such that $t \rightarrow_{\mathcal{R}} t'$ iff there exist $(l, r) \in \mathcal{R}$, $\omega \in \mathcal{P}os(t)$ and a substitution σ such that $t_{|\omega} = \sigma(l)$ and $t' = t\,[\sigma(r)]_{\omega}$.

Given a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, atoms, litterals and formulae over Σ are defined as usual. Free and bound variables of a formula φ are denoted by $\mathcal{FV}ar(\varphi)$ and $\mathcal{BV}ar(\varphi)$. We call Σ-theory any set of Σ-formulae. A *Σ-interpretation* $\Im$ consists in a non-empty set $|\,\Im\,|$ called *carrier* of $\Im$, for each symbol $f_{/n} \in \mathcal{F}$ an application $f_{\Im}$ from $|\,\Im\,|^n$ to $|\,\Im\,|$ called *interpretation* of f in $\Im$ and for each symbol $p_{/n} \in \mathcal{P} \cup \{=\}$ a relation $p_{\Im}$ over $|\,\Im\,|^n$ called *interpretation* of p in $\Im$ such that $=_{\Im}$ is an equivalence relation. The application which associates to each $t \in \mathcal{T}_{\mathcal{F}}$ its interpretation in $\Im$, denoted by $t_{\Im}$, is the unique morphism from $\mathcal{T}_{\mathcal{F}}$ to $\Im$. A Σ-interpretation $\Im$ is said to be *term-generated* iff for any $u \in |\,\Im\,|$, there exists a term $t \in \mathcal{T}_{\mathcal{F}}$ such that $t_{\Im} = u$. Given a Σ-interpretation $\Im$ and a set of variables $\mathcal{X}$, an *$\Im$-valuation* α is a mapping from $\mathcal{X}$ to $|\,\Im\,|$ and induces together with $\Im$ a mapping from $\mathcal{T}_{\mathcal{F},\mathcal{X}}$ to $|\,\Im\,|$. The semantics of a Σ-formula φ in $\Im$ according to the $\Im$-valuation α, denoted by $[\![\varphi]\!]^{\alpha}_{\mathcal{S}}$, is defined as usual in first-order logic. We say that $\Im$ is a *model* of φ or that φ is *valid in* $\Im$ and we write $\Im \models \varphi$ iff $[\![\varphi]\!]^{\alpha}_{\Im}$ is *true* for any valuation α. The set of all models of φ is denoted by $\mathcal{M}od(\varphi)$ and for any theory Γ, $\mathcal{M}od(\Gamma)$ is the set of interpretations which are model of all formulae in Γ. We extend the relation $\models$ to theories as follows: $\Gamma \models \varphi$ iff for any $\Im \in \mathcal{M}od(\Gamma)$, $\Im \models \varphi$.

We call *n-ary (ascending) tree automaton* any quadruple $\mathcal{A} = \langle\, \mathcal{F}, Q, F, \Delta \,\rangle$ such that $\mathcal{F}$ is an alphabet of function symbols, Q is a finite set of *states*, F is a subset of Q whose elements are called *final states* and Δ is a relation over $\mathcal{T}_{\mathcal{F}^n[Q]} \times Q$ whose elements are called *transitions* where Λ is a new symbol and $\mathcal{F}^n\,[Q]$ is the alphabet $(\mathcal{F} \cup \{\Lambda\})^n \setminus \{\langle \Lambda, \ldots, \Lambda \rangle\} \cup Q$ such that $ar : \langle\, f_1, \ldots, f_n \,\rangle \in (\mathcal{F} \cup \{\Lambda\})^n \mapsto \max_{i \in [1,n]} (ar(f_i) \mid f_i \neq \Lambda)$ and $ar : q \in Q \mapsto 0$. We recall that a 1-tuple $\langle\, f \,\rangle$ is simply denoted by f and that there exists only one 0-tuple denoted by $\langle\, \rangle$. An element of $\mathcal{T}_{\mathcal{F}^n[Q]}$ is called a *configuration*. A transition $lhs \rightarrow rhs$ of Δ is *normalized* iff for any $\omega \neq \varepsilon$, $lhs(\omega) \in Q$. An automaton whose transitions are normalized is said normalized. A tree automaton is said *deterministic* iff all its transitions have a different left-hand side. Without loss of generality, we can consider that all automata are normalized and deterministic. The rewriting relation induced by Δ over $\mathcal{T}_{\mathcal{F}^n[Q]}$ is denoted by $\rightarrow_{\mathcal{A}}$ and the language recognized by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{\langle\, t_1, \ldots, t_n \,\rangle \in (\mathcal{T}_{\mathcal{F}})^n \mid \exists q_f \in F, t_1 \otimes \ldots \otimes t_n \xrightarrow{*}_{\mathcal{A}} q_f\}$ where $t = t_1 \otimes \ldots \otimes t_n$ is the configuration such that: $\forall \omega \in \bigcup_{i=1}^{n} \mathcal{P}os(t_i)$, $t(\omega) = \langle\, t_1[\omega), \ldots, t_n[\omega) \,\rangle$ where $u[\omega) = u(\omega)$ if $\omega \in \mathcal{P}os(t)$ and Λ otherwise. The behaviour of tree automata is illustrated in Appendix A. A set E of n-tuples of terms (or equivalently n-ary relation) is said *recognizable* iff there exists an n-ary tree automaton $\mathcal{A}$ such that $E = \mathcal{L}(\mathcal{A})$. The class of recognizable n-ary relations is closed for boolean operations and the corresponding automata are denoted by $\mathcal{A}_1 \cap \mathcal{A}_2$, $\mathcal{A}_1 \cup \mathcal{A}_2$, $\overline{\mathcal{A}_1}$.

3 Constrained rewriting

In [KK89] the authors give the foundations for constrained equational reasoning and propose a definition of constrained rewriting in which constraints are equational problems, *i.e.* a

constrained rewriting dealing only with equality. Frameworks for rewriting with membership, equality and inequality constraints have been also proposed [Hoo92,Com92,CD94] and this can be seen as a first step towards an approach integrating more general predicates in equational reasoning. We go a step further and we consider in this paper more general constraints, namely, first order formulae.

Definition 1 (Constrained term rewrite system). *Given a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, we call* constrained rule *over Σ any 3-tuple $(lhs, \varphi, rhs) \in \mathcal{T}_{\mathcal{F},\mathcal{X}} \times \mathcal{F}or_{\Sigma,\mathcal{X}} \times \mathcal{T}_{\mathcal{F},\mathcal{X}}$, denoted by $lhs \xrightarrow{\varphi} rhs$, such that $\mathcal{V}ar(rhs) \subseteq \mathcal{V}ar(lhs) \cup \mathcal{FV}ar(\varphi)$ (φ is omitted when it is $\top$). The terms lhs and rhs are called left-hand side and right-hand side respectively and the formula φ is called the constraint of the rule. We call* constrained term rewrite system *or* CTRS *(over Σ), any set $\mathcal{R}$ of constrained rules over Σ.*

Example 1. Given $\mathcal{F} = \{zero_{/0}, succ_{/1}, f_{/2}\}$ and $\mathcal{P} = \{inf_{/2}\}$, the following sets of rules are constrained rewrite systems over $\Sigma = (\mathcal{F}, \mathcal{P})$:

$$\mathcal{R}_1 = \left\{ f(x,x) \longrightarrow x \;\; ; \;\; f(x,y) \xrightarrow{inf(x,y)} x \;\; ; \;\; f(x,y) \xrightarrow{inf(y,x)} y \right\}$$

$$\mathcal{R}_2 = \left\{ f(zero, v) \longrightarrow v \;\; ; \;\; f(x,v) \xrightarrow{inf(y,x) \,\wedge\, \forall z:\, inf(z,x) \Rightarrow (z=y \vee inf(z,y))} succ(f(y,v)) \right\}$$

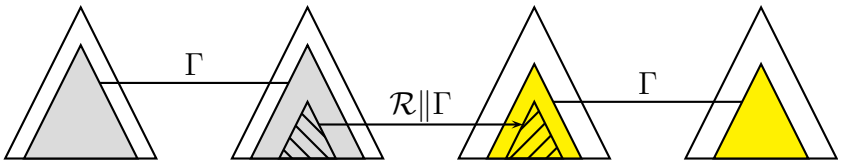
Definition 2 (Constrained rewriting). *Given a signature $\Sigma = (\mathcal{F}, \mathcal{P})$, a Σ-theory Γ and a CTRS $\mathcal{R}$ over Σ, a term $u \in \mathcal{T}_{\mathcal{F}}$* rewrites to *a term $v \in \mathcal{T}_{\mathcal{F}}$ w.r.t. Γ, which is denoted by $u \to_{\mathcal{R}}^{\Gamma} v$ iff there exist (i) two terms t and t' in $\mathcal{T}_{\mathcal{F}}$, (ii) a position $p \in \mathcal{P}os(t)$, (iii) a rewrite rule $lhs \xrightarrow{\varphi} rhs \in \mathcal{R}$, and (iv) a ground substitution σ with $\mathcal{D}om(\sigma) = \mathcal{V}ar(lhs) \cup \mathcal{FV}ar(\varphi)$ such that $t_{|p} = \sigma(lhs)$, $t' = t\,[\sigma(rhs)]_p$ and $\Gamma \models \{u = t \;;\; \sigma(\varphi) \;;\; v = t'\}$.*

The above definition of rewriting *w.r.t.* a Σ-theory naturally extends to term-generated Σ'-interpretations where $\Sigma' \subseteq \Sigma$. Indeed, to any interpretation $\Im$ we can associate at least a theory Γ such that the set of Σ'-interpretations model of Γ whose carrier is $|\,\Im\,|$ is $\{\Im\}$.

Example 2. Given Σ, $\mathcal{R}_1$ and $\mathcal{R}_2$ defined in Example 1, we denote by $\mathcal{N}$ the interpretation whose carrier is $\mathbb{N}$ and such that $zero_{\mathcal{N}} = 0$, $succ_{\mathcal{N}} = n \mapsto n+1$ and inf is interpreted by $<_{\mathbb{N}}$. We denote by $\mathcal{N}^{-1}$ the interpretation which differs from $\mathcal{N}$ only by the interpretation of inf by $>_{\mathbb{N}}$. Then, the term $f(succ(zero), zero)$ rewrites by $\mathcal{R}_1$ into $zero$ *w.r.t.* $\mathcal{N}$ and into $succ(zero)$ *w.r.t.* $\mathcal{N}^{-1}$. We also have $f(succ(zero), f(zero, zero)) \to_{\mathcal{R}_1}^{\mathcal{N}} f(succ(zero), zero) \to_{\mathcal{R}_1}^{\mathcal{N}} zero$. Note that this latter reduction is the only one starting from $f(succ(zero), f(zero, zero))$ since the interpretation of inf in $\mathcal{N}$ is not specified for terms whose head is f. One can see that f behaves as the addition operator *w.r.t.* $\to_{\mathcal{R}_2}^{\mathcal{N}}$. An additional example is given in Appendix B.

One can notice that Definition 2 implicitly introduces a relation $\to_{\mathcal{R}\|\Gamma}$ such that for all t, t' in $\mathcal{T}_{\mathcal{F}}$, $t \to_{\mathcal{R}\|\Gamma} t'$ iff there exist (i) a position $p \in \mathcal{P}os(t)$, (ii) a rewrite rule $lhs \xrightarrow{\varphi} rhs \in \mathcal{R}$, and (iii) a ground substitution σ with $\mathcal{D}om(\sigma) = \mathcal{V}ar(lhs) \cup \mathcal{FV}ar(\varphi)$ such that $t_{|p} = \sigma(lhs)$, $t' = t\,[\sigma(rhs)]_p$ and $\Gamma \models \sigma(\varphi)$.

Using this latter relation, the constrained rewriting relation can be decomposed as follows: $\to_{\mathcal{R}}^{\Gamma} \;=\; =_{\Gamma} \cdot \to_{\mathcal{R}\|\Gamma} \cdot =_{\Gamma}$ and this can be depicted by:



It is not surprising that a similar decomposition exists in rewriting modulo [JK86,PS81,BD89]: given a signature $\Sigma = (\mathcal{F}, \varnothing)$ containing only the equality predicate, a Σ-theory Γ and a constrained rewrite system $\mathcal{R}$ over Σ such that all constraints are the constant $\top$, the relation $\rightarrow_{\mathcal{R}}^{\Gamma}$ is exactly the relation $\rightarrow_{\mathcal{R}/\Gamma}$. When considering only monadic predicates, the formalism instantiates to rewriting with membership constraints [CD94].

4 Recognizable theories

Frühwirth *et al.* showed that recognizable sets play an important role in the computation in Horn clause theories [FSVY91]. Indeed, they proposed a restriction of Horn clauses (called uniform clauses) insuring that the least model consists of recognizable sets of terms and thus can be represented by a tree automaton [NNH02]. Moreover, it has been shown [CD94] that order-sorted signatures could be viewed as regular tree languages and in [GL05] a correspondence between Prolog monadic programs and tree automata was formalized. Starting from theses observations, we propose to describe some theories by tree automata.

4.1 Recognizable presentations

The description of a Σ-theory Γ could be obtained, when it is possible, by characterizing for each predicate p of Σ the set of tuples for which p is completely interpreted, *i.e.* the set $\mathcal{S}(p) \subseteq \mathcal{T}_{\mathcal{F}}^n$ such that for any $\langle\, t_1, \ldots, t_n \,\rangle \in \mathcal{S}(p)$ either $\Gamma \models p(t_1, \ldots, t_n)$ or $\Gamma \models \neg p(t_1, \ldots, t_n)$.

Definition 3 (Recognizable presentation). *Given a signature $\Sigma = (\mathcal{F}, \mathcal{P})$ and a Σ-theory Γ, a* recognizable presentation *$\wp$ of Γ consists of an application $\mathcal{S}_\wp$ which associates to any $p \in \mathcal{P}$ a tree automaton called* support *of p, an application $\mathcal{A}_\wp$ which associates to any $p \in \mathcal{P}$ a tree automaton called* axiomatization *of p and a rewrite system $\mathcal{R}$ over Σ such that: for any $t, t' \in \mathcal{T}_{\mathcal{F}}$, $\Gamma \models t = t'$ iff $t \overset{*}{\longleftrightarrow}_{\mathcal{R}} t'$ and for any $p_{/n} \in \mathcal{P}$, $n > 0$, $\langle\, t_1, \ldots, t_n \,\rangle \in \mathcal{T}_{\mathcal{F}}^n$, $\Gamma \models p(t_1, \ldots, t_n)$ iff $\exists u \in [\![p]\!]_\wp$ s.t. $\langle\, t_1, \ldots, t_n \,\rangle \overset{*}{\longleftrightarrow}_{\mathcal{R}} u$ and $\Gamma \models \neg p(t_1, \ldots, t_n)$ iff $\exists u \in [\![\overline{p}]\!]_\wp$ s.t. $\langle\, t_1, \ldots, t_n \,\rangle \overset{*}{\longleftrightarrow}_{\mathcal{R}} u$ where $[\![p]\!]_\wp$ denotes the set $\mathcal{L}(\mathcal{S}_\wp(p)) \cap \mathcal{L}(\mathcal{A}_\wp(p))$ and $[\![\overline{p}]\!]_\wp$ the set $\mathcal{L}(\mathcal{S}_\wp(p)) \setminus \mathcal{L}(\mathcal{A}_\wp(p))$. The pair $\langle\, \mathcal{S}_\wp(p), \mathcal{A}_\wp(p) \,\rangle$ is called* presentation *of p in $\wp$.*

Definition 4 (Recognizable theory). *We call* recognizable theory *any theory Γ which admits a recognizable presentation. Moreover, if there exists a recognizable presentation of Γ whose rewrite system is empty, Γ is called a* Herbrand recognizable theory.

Example 3. A presentation $\wp$ of $\mathcal{N}$ (defined in Example 2) is given by $\mathcal{R} = \varnothing$ and:

– $\mathcal{S}_\wp(inf) = ((\mathcal{F} \cup \{\Lambda\})^2, \{q_1, q_2, q_{Nat}\}, \{q_{Nat}\}, \Delta_1)$ where Δ_1 consists of:

$$\begin{cases} \langle\, zero, \Lambda \,\rangle \to q_1 & \langle\, succ, \Lambda \,\rangle\,(q_1) \to q_1 \\ \langle\, succ, zero \,\rangle\,(q_1) \to q_{Nat} & \langle\, \Lambda, zero \,\rangle \to q_2 \\ \langle\, \Lambda, succ \,\rangle\,(q_2) \to q_2 & \langle\, zero, succ \,\rangle\,(q_2) \to q_{Nat} \\ \langle\, zero, zero \,\rangle \to q_{Nat} & \langle\, succ, succ \,\rangle\,(q_{Nat}) \to q_{Nat} \end{cases}$$

– $\mathcal{A}_\wp(inf) = ((\mathcal{F} \cup \{\Lambda\})^2, \{q, q_{inf}\}, \{q_{inf}\}, \Delta_2)$ where Δ_2 consists of:

$$\begin{cases} \langle\, succ, succ \,\rangle\,(q_{inf}) \to q_{inf} & \langle\, zero, succ \,\rangle\,(q) \to q_{inf} \\ \langle\, \Lambda, succ \,\rangle\,(q) \to q & \langle\, \Lambda, zero \,\rangle \to q \end{cases}$$

One can remark that the predicate *inf* is interpreted only for terms built from *succ* and *zero* and thus the support of this predicate denotes this kind of terms. In particular, the theory gives no information about terms whose head symbol is f. Another simple but classical example is the presentation of odd and even integers with a commutative operator f. The corresponding Σ-theory is the following:

$$\left\{ \begin{array}{ccccc} even(zero) & ; & \forall x : odd(x) \Rightarrow even(succ(x)) & ; & \forall x : even(x) \Rightarrow odd(x) \\ \forall x, y : f(x, y) = f(y, x) & ; & \forall x : f(zero, x) = x & ; & \forall x, y : f(succ(x), y) = succ(f(x, y)) \end{array} \right\}$$

A possible recognizable presentation of the theory presented above is given by:

– $\mathcal{S}_\wp(odd) = \mathcal{S}_\wp(even) = (\mathcal{F}, \{q\}, \{q\}, \Delta_3)$ where $\Delta_3 = \{\ zero \to q \ \ ; \ \ succ(q) \to q\ \}$
– $\mathcal{A}_\wp(odd) = (\mathcal{F}, \{q_{odd}, q_{even}\}, \{q_{odd}\}, \Delta_4)$ and $\mathcal{A}_\wp(even) = (\mathcal{F}, \{q_{odd}, q_{even}\}, \{q_{even}\}, \Delta_4)$ where $\Delta_4 = \{\ \ zero \to q_{even} \ \ ; \ \ succ(q_{even}) \to q_{odd} \ \ ; \ \ succ(q_{odd}) \to q_{even} \ \ \}$
– $\mathcal{R} = \{\ f(x, y) \to f(y, x) \ \ ; \ \ f(zero, x) \to x \ \ ; \ \ f(succ(x), y) \to succ(f(x, y))\ \}$

We have $\Gamma \models odd(f(succ(zero), zero)))$ and $f(succ(zero), zero)) \overset{*}{\leftrightarrow}_{\mathcal{R}} succ(zero) \in [\![odd]\!]_\wp$.

4.2 Operations over predicate presentations

Starting from a set of predicates and their automata based presentations one can define new predicates whose presentations are defined using appropriate automata operators. We define in this section such operators that will be intensively used in Section 5 for transforming a formula into a so-called solved form.

Given a signature $\Sigma = (\mathcal{F}, \mathcal{P})$ we denote by $\mathcal{P}^*$ the set of symbols built out of elements of $\mathcal{P}$ and satisfying the following grammar:

$$P, Q \ ::= \ p \mid \top \mid \bot \mid \Box \mid \Omega_\mathcal{F} \mid Id^n_\mathcal{F} \mid \overrightarrow{T} \mid P \sqcap Q \mid P \sqcup Q \mid \widetilde{P} \mid \mathfrak{S}_{i,j}(P) \mid \mathfrak{U}_i(P) \mid \Pi_i(P) \mid \Pi_{t/i}(P)$$

where $p \in \mathcal{P}$, $t \in \mathcal{T}_\mathcal{F}$, $i, j, n \in \mathbb{N}$, $T \in \mathcal{T}_{\mathcal{F}[\mathcal{P}^*]}$ and $\mathcal{T}_{\mathcal{F}[\mathcal{P}^*]}$ is the set of ground terms built from $\mathcal{F}$ and elements of $\mathcal{P}^*$ seen as constant symbols. The meanings of the theses predicates are given in the following definitions.

Definition 5 (Constants). *We define the* 0*-ary predicates* $\top$ *,* $\bot$ *and* $\Box$ *called* constants *such that for any presentation* $\wp$*:* $\mathcal{S}_\wp(\top) = \mathbb{T}$*,* $\mathcal{A}_\wp(\top) = \mathbb{T}$*,* $\mathcal{S}_\wp(\bot) = \mathbb{T}$*,* $\mathcal{A}_\wp(\bot) = \mathbb{F}$ *and* $\mathcal{S}_\wp(\Box) = \mathbb{F}$ *where* $\mathbb{T} = \{\mathcal{F}^0, \{q\}, \{q\}, \{\langle\ \rangle \to q\}\}$ *and* $\mathbb{F} = \{\mathcal{F}^0, \varnothing, \varnothing, \varnothing\}$*. Note that* $\mathcal{L}(\mathbb{T}) = \{\langle\ \rangle\}$ *and* $\mathcal{L}(\mathbb{F}) = \varnothing$*.*

Definition 6 (Universal). *We call* universal *(w.r.t. $\mathcal{F}$) and we denote by $\Omega_{\mathcal{F}}$ the predicate whose support and axiomatization are given by the automaton recognizing all ground terms of $\mathcal{T}_{\mathcal{F}}$ and defined by $\mathbb{U}niv_{\mathcal{F}} = \{\mathcal{F}, \{q\}, \{q\}, \Delta\}$ where $\Delta = \{f(q, \ldots, q) \to q \mid f \in \mathcal{F}\}$.*

Definition 7 (Identity). *We call n-*ary identity *and we denote by $Id^n_{\mathcal{F}}$ the predicate whose support is $\mathbb{U}niv_{\mathcal{F}}$ and whose axiomatization is given by the automaton $\{(\mathcal{F} \cup \{\Lambda\})^n, \{q\}, \{q\}, \Delta\}$ where $\Delta = \{\langle f, \ldots, f \rangle (q, \ldots, q) \to q \mid f \in \mathcal{F}\}$.*

Definition 8 (Recognizer). *For any ground term $t \in \mathcal{T}_{\mathcal{F}[\mathcal{P}^*]}$, we denote by $\mathbb{R}ec(t) = \{\mathcal{F}, Q = \{q_i \mid i \in \mathcal{P}os(t)\} \cup \{q \in Q_A \mid A \in \mathcal{P}^*\}, \{q_\varepsilon\}, \Delta\}$ with $\Delta = \{t(\omega)(q'_1, \ldots, q'_n) \to q_\omega \mid \omega \in \mathcal{P}os(t), t(\omega) \in \mathcal{F}, ar(t(\omega)) = n$ with $q'_i = q_{\omega.i}$ if $t(\omega.i) \notin \mathcal{P}^*$ and $q'_i \in F_{t(\omega.i)}$ if $t(\omega.i) \in \mathcal{P}^*\} \bigcup_{t(\omega) \in \mathcal{P}^*} \Delta_{t(\omega)}$ where $(\mathcal{F}, Q_A, F_A, \Delta_A)$ is the automaton associated to A for any $A \in \mathcal{P}^*$.*
We call recognizer of t *and denote by $\overrightarrow{t}$ the predicate such that $\mathcal{S}_\wp(\overrightarrow{t}) = \mathbb{U}niv_{\mathcal{F}}$ and $\mathcal{A}_\wp(\overrightarrow{t}) = \mathbb{R}ec(t)$ for any presentation $\wp$.*

Definition 9 (Boolean operations). *Given two predicates $p_{/n}$ and $q_{/n}$ with $n \geq 0$, the presentations of the n-ary predicates $p \sqcap q$ (*conjunction*), $p \sqcup q$ (*disjunction*) and $\widetilde{p}$ (*negation*) are defined as follows:*

$\cdot$	$\mathcal{S}_\wp(\cdot)$	$\mathcal{A}_\wp(\cdot)$
$p \sqcap q$	$\left(\mathcal{S}_\wp(p) \cap \overline{\mathcal{A}_\wp(p)}\right) \cup \left(\mathcal{S}_\wp(q) \cap \overline{\mathcal{A}_\wp(q)}\right) \cup (\mathcal{S}_\wp(p) \cap \mathcal{S}_\wp(q))$	$\mathcal{A}_\wp(p) \cap \mathcal{A}_\wp(q)$
$p \sqcup q$	$(\mathcal{S}_\wp(p) \cap \mathcal{A}_\wp(p)) \cup (\mathcal{S}_\wp(q) \cap \mathcal{A}_\wp(q)) \cup (\mathcal{S}_\wp(p) \cap \mathcal{S}_\wp(q))$	$\mathcal{A}_\wp(p) \cup \mathcal{A}_\wp(q)$
$\widetilde{p}$	$\mathcal{S}_\wp(p)$	$\overline{\mathcal{A}_\wp(p)}$

Definition 10 (Swap). *We define the swapping of an automaton $\mathcal{A} = \{(\mathcal{F} \cup \{\Lambda\})^n, Q, F, \Delta\}$ w.r.t. i and $j \in [1, n]$ as the automaton $\mathbb{S}wap_{i,j}(\mathcal{A}) = \{(\mathcal{F} \cup \{\Lambda\})^n, Q, F, \Delta'\}$, whose transitions Δ' are such that $\langle \ldots, f_j, \ldots, f_i, \ldots \rangle (q_1, \ldots, q_m) \to q \in \Delta'$ iff $\langle \ldots, f_i, \ldots, f_j, \ldots \rangle (q_1, \ldots, q_m) \to q \in \Delta$. The operator is extended to any permutation θ of $[1, n]$ as follows: $\langle f_1, \ldots, f_n \rangle (q_1, \ldots, q_m) \to q \in \Delta'$ iff $\langle f_{\theta_1}, \ldots, f_{\theta_n} \rangle (q_1, \ldots, q_m) \to q \in \Delta$. We sometimes denote a permutation θ by the sequence $(\theta_1, \ldots, \theta_n)$ instead of $\{i \mapsto \theta_i \mid i \in [1, n]\}$. Given i and j (resp. θ) the* swap *operator w.r.t. i and j (resp. θ) associates to any predicate p the one denoted by $\mathfrak{S}_{i,j}(p)$ (resp. $\mathfrak{S}_\theta(p)$) such that for any presentation $\wp$, $\mathcal{S}_\wp(\mathfrak{S}_{i,j}(p)) = \mathbb{S}wap_{i,j}(\mathcal{S}_\wp(p))$ (resp. $\mathcal{S}_\wp(\mathfrak{S}_\theta(p)) = \mathbb{S}wap_\theta(\mathcal{S}_\wp(p))$) and $\mathcal{A}_\wp(\mathfrak{S}_{i,j}(p)) = \mathbb{S}wap_{i,j}(\mathcal{A}_\wp(p))$ (resp. $\mathcal{A}_\wp(\mathfrak{S}_\theta(p)) = \mathbb{S}wap_\theta(\mathcal{A}_\wp(p))$). Note that for any theory Γ, predicate p and ground terms $t_1, \ldots, t_n$, we have: $\Gamma \models p(t_1, \ldots, t_i, \ldots, t_j, \ldots, t_n)$ iff $\Gamma \models \mathfrak{S}_{i,j}(p)(t_1, \ldots, t_j, \ldots, t_i, \ldots, t_n)$.*

Definition 11 (Generalisation). *The i^{th} cylindrification of an automaton $\mathcal{A}$, denoted by $\mathbb{C}ylindrify_{/i}(\mathcal{A}_\wp)$, is defined in [C$^+$08] and corresponds to the cartesian product of $\Omega_{\mathcal{F}}$ and $\mathcal{A}$ followed by a swap in 1 and i. We call i^{th}* generalisation *of a predicate p the predicate denoted by $\mathfrak{U}_i(p)$ such that for any presentation $\wp$, $\mathcal{S}_\wp(\mathfrak{U}_i(p)) = \mathbb{C}ylindrify_{/i}(\mathcal{S}_\wp(p))$ and $\mathcal{A}_\wp(\mathfrak{U}_i(p)) = \mathbb{C}ylindrify_{/i}(\mathcal{A}_\wp(p))$. $\mathfrak{U}$ is extended to any finite sequence of integers $I = \{i_1, \ldots, i_m\}$ such that $i_1 < i_2 < \ldots < i_m$ as follows: $\mathfrak{U}_I(p) = \mathfrak{U}_{i_1}(\mathfrak{U}_{i_2}(\ldots(\mathfrak{U}_{i_m}(p))))$. Note that for any theory Γ, predicate p and ground terms $t_1, \ldots, t_n$, we have: $\Gamma \models p(t_1, \ldots, t_n)$ iff $\Gamma \models \forall x : \mathfrak{U}_i(p)(t_1, \ldots, t_{i-1}, x, t_i, \ldots, t_n)$.*

Definition 12 (Projection). *The i^{th} projection of an automaton $\mathcal{A} = \{(\mathcal{F} \cup \{\Lambda\})^n, Q, F, \Delta\}$, $n > 0$, is defined as the automaton $\mathbb{P}roj_i(\mathcal{A}) = \{(\mathcal{F} \cup \{\Lambda\})^{n-1}, Q, F, \Delta'\}$ such that for any $\langle f_1, \ldots, f_{i-1}, f_i, f_{i+1}, \ldots, f_n \rangle (q_1, \ldots, q_m) \to q$ in Δ s.t. $n = 1$ or $\exists j \neq i,\ f_j \neq \Lambda$ the transition $\langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n \rangle (q_1, \ldots, q_k) \to q$ with $k = \max_{j \neq i}(ar(f_j))$ is in Δ'. We call i^{th}* projection *of a predicate p the predicate denoted by $\Pi_i(p)$ such that $\mathcal{S}_\wp(\Pi_i(p)) = \mathbb{P}roj_i(\mathcal{S}_\wp(p))$ and $\mathcal{A}_\wp(\Pi_i(p)) = \mathbb{P}roj_i(\mathcal{A}_\wp(p))$. For any finite sequence of integers $I = \{i_1, \ldots, i_m\}$ such that $i_1 > i_2 > \ldots > i_m$, $\Pi_I(p)$ is a shortcut for $\Pi_{i_1}(\Pi_{i_2}(\ldots(\Pi_{i_m}(p))))$.*

Definition 13 (Projection directed by a ground term). *For any ground term t, the i^{th}* projection of $p_{/n}$ directed by t, *denoted by $\Pi_{t/i}(p)$, equals to $\Pi_i\left(p \sqcap \mathfrak{U}_{1,\ldots,i-1,i+1,\ldots,n}\left(\overrightarrow{t}\right)\right)$.*

We suppose that all considered automata are deterministic and reduced (*i.e.* for any automaton $\mathcal{A}$ and any state q of $\mathcal{A}$, there is at least a ground term t such that $t \to^*_{\mathcal{A}} q$).

5 Constrained matching in recognizable theories

In this section we give a definition of the matching problems intrinsic to constrained rewriting and characterize a large class of problems that can be solved in recognizable theories.

Definition 14 (Matching problem, matching solution). *Given a signature Σ, a* constrained matching problem *or CMP is a formula of the form $p \,\|^{\varphi} \ll^? t$ with $p \in \mathcal{T}_{\mathcal{F},\mathcal{X}}, \varphi \in \mathcal{F}or_{\Sigma,\mathcal{X}}$ and $t \in \mathcal{T}_{\mathcal{F}}$. Given a theory Γ, a ground substitution σ with $\mathcal{D}om(\sigma) = \mathcal{V}ar(p) \cup \mathcal{FV}ar(\varphi)$ is a Γ-*solution *of the constrained matching problem $p \,\|^{\varphi} \ll^? t$ iff $\sigma(p) = t$ and $\Gamma \models \sigma(\varphi)$. If such a substitution exists we say that t Γ-*matches *the* constrained pattern *(p, φ), denoted $p \,\|^{\varphi} \ll_\Gamma t$. The set of Γ-solutions of a problem $p \,\|^{\varphi} \ll^? t$ is denoted by $\mathcal{S}ol_\Gamma(p \,\|^{\varphi} \ll^? t)$.*

We address in this section the problem of computing the set of Γ-solutions of a CMP $p \,\|^{\varphi} \ll^? t$ for a recognizable theory Γ. One can notice that solving a CMP amounts to solving a *matching problem* (MP) of the form $p =^? t$ with $p \in \mathcal{T}_{\mathcal{F},\mathcal{X}}$ and $t \in \mathcal{T}_{\mathcal{F}}$ and a *constraint resolution problem* (CRP) of the form $\Gamma \models^? \psi$ with $\psi \in \mathcal{F}or_{\Sigma,\mathcal{X}}$. An MP $p = t$ can be solved using a classical syntactic matching algorithm [Hue76,KK06] which computes, iff it exists, a substitution σ' with $Dom(\sigma') = \mathcal{V}ar(p)$ such that $\sigma'(p) = t$. We propose in what follows an algorithm that computes the solutions σ'' with $Dom(\sigma'') = \mathcal{FV}ar(\psi)$ of a problem $\Gamma \models^? \psi$ when ψ satisfies certain conditions. The solutions of the initial CMP are obtained then by composing the solutions σ' of the MP with the corresponding solutions of the CRP $\Gamma \models^? \sigma'(\varphi)$.

We focus here on Herbrand recognizable theories and we briefly discuss possible extensions for the proposed approach.

5.1 Transformation of constraints

In order to solve the CRPs we use, as in [CL89] for example, a transformation system that transforms the initial problem into a new equivalent presentation, called *solved form* for which the solution may be trivially extracted.

Definition 15 (Solved form). *A CRP $\Gamma \models^? \varphi$ over $\Sigma = (\mathcal{F}, \mathcal{P})$ is said to be in* solved form *iff φ is an atom $p(x_1, \ldots, x_n)$ where $p \in \mathcal{P}^*$ and $x_1, \ldots, x_n$ are distinct variables. The set of solutions of a CMP in solved form $\Gamma \models^? p(x_1, \ldots, x_n)$ is:*

$$\left\{\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} \mid \langle\, t_1, \ldots, t_n \,\rangle \in [\![p]\!]_{\wp}\right\}$$

where $\wp$ is a presentation of Γ.

Note that when p is $\top$, the set of solutions is a singleton containing the identity substitution (the unique substitution whose domain is empty). If p is $\bot$ or $\Box$ the set of solutions is empty.

As we will see in what follows, the algorithm we propose strongly relies on the semantics of the projection operator given by the following lemma:

Lemma 1 ([Jac96,C+08]). *Let $\Sigma = (\mathcal{F}, \mathcal{P})$ be a signature and $\wp$ be a presentation of a theory Γ. For any $p_{/n+1} \in \mathcal{P}$, $t_1, \ldots, t_n \in \mathcal{T}_{\mathcal{F}}$, $n \geq 0$:*

$$\wp \models \Pi_i(p)(t_1, \ldots, t_n) \text{ iff } \wp \models \exists x : p(t_1, \ldots, t_{i-1}, x, t_i, \ldots, t_n)$$

Starting from this result we build an algorithm which transforms $\Gamma \models^? \varphi$, when it is possible, into an equivalent solved form $\Gamma \models^? \varphi^*$. In order to simplify the description of the transformation system, we will not copy "$\Gamma \models^?$" and will rewrite only the formula. Moreover, since any first order formula has an equivalent prenex normal form [Hin05], we presuppose that any formula of a CRP has been recast in prenex normal form.

We split the presentation of the algorithm according to the intended use of each of the corresponding sets of rules. We show that each of these rules preserves the semantics and thus the correctness and completeness of the algorithm. We suppose that we have at our disposal a presentation $\wp$ of Γ.

The first part of the algorithm consists in isolating existentially quantified subformulae and "solving" them according to Lemma 1. Since existential quantification is distributive over disjunction but not under conjunction, the real scope of an existentially quantified variable can always be expressed with a conjunction of literals. The normalization of a quantified formula *w.r.t.* the sets of rules **N**, **Q** and **U** given below isolates subformulae of the form $\exists y : \bigwedge_{j=1}^{m} a_j$.

Normalization: N

$$(N_1)\ (Q_i x_i)_{i=1}^{q}.\exists y : \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} a_{i,j} \twoheadrightarrow (Q_i x_i)_{i=1}^{q}.\exists y : \bigvee_{(i_1,\ldots,i_n)\in\prod_{k=1}^{n}[1,m_k]} \bigwedge_{j=1}^{n} a_{j,i_j}$$

$$(N_2)\ (Q_i x_i)_{i=1}^{q}.\forall y : \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} a_{i,j} \twoheadrightarrow (Q_i x_i)_{i=1}^{q}.\forall y : \bigwedge_{(i_1,\ldots,i_n)\in\prod_{k=1}^{n}[1,m_k]} \bigvee_{j=1}^{n} a_{j,i_j}$$

Quantifier distribution: Q

$$(Q_1)\ (Q_i x_i)_{i=1}^{q}.\exists y : \bigvee_{i=1}^{n}\bigwedge_{j=1}^{m_i} a_{i,j} \twoheadrightarrow (Q_i x_i)_{i=1}^{q} : \bigvee_{i=1}^{n}\exists y : \bigwedge_{j=1}^{m_i} a_{i,j}$$

$$(Q_2)\ (Q_i x_i)_{i=1}^{q}.\forall y : \bigwedge_{i=1}^{n}\bigvee_{j=1}^{m_i} a_{i,j} \twoheadrightarrow (Q_i x_i)_{i=1}^{q} : \bigwedge_{i=1}^{n}\forall y : \bigvee_{j=1}^{m_i} a_{i,j}$$

Universal elimination: U

$$(U_1)\ \forall y : \bigvee_{j=1}^{m} a_j \twoheadrightarrow \neg\exists y : \bigwedge_{j=1}^{m} \neg a_j \qquad \textit{if } y \textit{ occurs in at least one } a_j$$

$$(U_2)\ \forall y : \bigvee_{j=1}^{m} a_j \twoheadrightarrow \bigvee_{j=1}^{m} a_j \qquad \textit{if } y \textit{ occurs in no } a_j$$

The subformulae of the form $\exists y : \bigwedge_{j=1}^{m} a_j$ obtained after normalisation by the previous rules are transformed into a solved form. A formula $\exists y : \varphi$ can be solved if either φ contains no occurrence of y or if φ is an atom $p(x_1, \ldots, y, \ldots, x_n)$; the corresponding transformations are performed by the sets of rules **E** and **P** given below.

Existential elimination: E

$$(E)\ \exists y : \bigwedge_{j=1}^{m} a_j \twoheadrightarrow \bigwedge_{j=1}^{m} a_j \qquad \textit{if } y \textit{ occurs in no } a_j$$

Projection: P

$$(P)\ \exists y.(\exists y_k)_{k=1}^{m} : p(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) \wedge \psi \twoheadrightarrow (\exists y_k)_{k=1}^{m} : \Pi_i(p)(x_1, \ldots, x_n) \wedge \psi \quad \textit{if } y \notin \{x_1, \ldots, x_n\} \cup \mathcal{FV}ar(\psi)$$

The existentially quantified formulae that do not have this solvable shape can be transformed by the sets of rules **C** and **M** which correspond to the semantics given in Definition 9.

Complementation: C

$$(C)\ \neg p(x_1, \ldots, x_n) \twoheadrightarrow \widetilde{p}(x_1, \ldots, x_n)$$

Merging: M

$$(M_1)\ p(x_1, \ldots, x_n) \wedge q(x_1, \ldots, x_n) \twoheadrightarrow (p \sqcap q)(x_1, \ldots, x_n)$$

$$(M_2)\ p(x_1, \ldots, x_n) \vee q(x_1, \ldots, x_n) \twoheadrightarrow (p \sqcup q)(x_1, \ldots, x_n)$$

The above transformations are performed on predicates which have the same parameters but this condition is not always satisfied in arbitrary formulae. We use thus the set of rules **O** to rearrange the parameters of atoms so that all variables of the atoms which must be combined respect a certain order, and the set of rules **G** to add parameters to atoms so that their number is the same for all atoms which must be combined.

Orientation: O

$$(O_1)\ p(x_1,\dots,x_{i-1},x_i,x_{i+1},\dots,x_n) \twoheadrightarrow \mathfrak{S}_{i,i+1}(p)(x_1,\dots,x_{i-1},x_{i+1},x_i,\dots,x_n) \quad \textit{if } x_i > x_{i+1}$$

Generalization: G

$$(G_1)\ \begin{array}{l} p(x_1,\dots,x_n) \\ \quad \wedge\ q(y_1,\dots,y_m) \end{array} \twoheadrightarrow \mathfrak{U}_k(p)(x_1,\dots,x_{k-1},y,x_k,\dots,x_n)\ \wedge\ q(y_1,\dots,y_m)$$
$$\textit{if } \{x_1,\dots,x_n\} \cap \{y_1,\dots,y_m\} \neq \varnothing$$
$$\textit{and } y \in \{y_1,\dots,y_m\} \setminus \{x_1,\dots,x_n\} \textit{ and } x_{k-1} < y < x_k$$

$$(G_2)\ \begin{array}{l} p(x_1,\dots,x_n) \wedge \psi \\ \quad \wedge\ q(y_1,\dots,y_m) \end{array} \twoheadrightarrow \mathfrak{U}_k(p)(x_1,\dots,x_{k-1},y,x_k,\dots,x_n)\ \wedge\ q(y_1,\dots,y_m)$$
$$\textit{if } \{x_1,\dots,x_n\} \cap \{y_1,\dots,y_m\} = \varnothing$$
$$\textit{and } y \in \{y_1,\dots,y_m\} \setminus \{x_1,\dots,x_n\} \textit{ and } x_{k-1} < y < x_k$$
$$\textit{and } \psi \textit{ contains no equality}$$

$$(G_3)\ \begin{array}{l} p(x_1,\dots,x_n) \\ \quad \vee\ q(y_1,\dots,y_m) \end{array} \twoheadrightarrow \mathfrak{U}_k(p)(x_1,\dots,x_{k-1},y,x_k,\dots,x_n)\ \vee\ q(y_1,\dots,y_m)$$
$$\textit{if } y \in \{y_1,\dots,y_m\} \setminus \{x_1,\dots,x_n\}$$

When an atom contains several occurrences of the same variable, the rule (P) cannot be applied (since it does not correspond to the cases handled by Lemma 1). This kind of atoms are decomposed using the rule (D_1) and this introduces a new existential quantified variable and an equality over two variables which is eventually transformed using **T** into a recognizable relation.

Syntactic Decomposition: D

$$(D_1)\ p(\dots,y,\dots,y,\dots) \twoheadrightarrow \exists z : p(\dots,y,\dots,z,\dots) \wedge z = y$$

Tree conversion: T

$$(T_1)\ x = y \twoheadrightarrow Id^2_{\mathcal{F}}(x,y) \quad \textit{if } x,y \in \mathcal{X}$$

One can remark that the rules (D_1) and (T_1) could be merged into a single rule but we keep theses rules separated since (T_1) can be applied in other contexts.

We write $\varphi \twoheadrightarrow^* \varphi'$ if φ' is obtained from φ by applying all the rules introduced so far such that at each transformation step the applied rule is chosen from one of the sets $\{\mathbf{P},\mathbf{E},\mathbf{N},\mathbf{Q},\mathbf{U},\mathbf{D},\mathbf{T},\mathbf{O},\mathbf{G},\mathbf{C},\mathbf{M}\}$ in this order.

Proposition 1. *Given a Σ-formula φ and a recognizable theory Γ then, for any ψ such that $\Gamma \models^? \varphi \twoheadrightarrow^* \Gamma \models^? \psi$, we have:*

$$\Gamma \models \sigma(\varphi) \quad \textit{iff} \quad \Gamma \models \sigma(\psi)$$

for any ground substitution σ with $\mathcal{D}om(\sigma) = \mathcal{FV}ar(\varphi)$.

Proof. The proof for the general case stated in Proposition 2 is given in Appendix C.

The transformation $\twoheadrightarrow^*$ not only preserves the semantics of formulae but also eventually leads to a solved form:

Lemma 2. *Any CRP $\Gamma \models^? \varphi$ has a solved form $\Gamma \models^? \varphi^*$ if φ contains no functional symbol.*

Proof. The proof is given in Appendix D.

Proposition 1 together with the Lemma 2 lead to the following theorem:

Theorem 1. *Given a CRP $\Gamma \models^? \varphi$, if Γ is recognizable and if φ contains no functional symbol, then its set of solutions is computable and recognizable.*

We lift now the restriction on the presence of functional symbols and show that we can only obtain a weaker version of Theorem 1. The rule (D_2) describes the extraction of the non-variable arguments which leads to predicates containing only variables and allows thus a direct application of the rules concerned by Theorem 1. (D_3) only rewrites non linear terms into linear ones with an equality constraint. The rules (D_4) and (D_5) correspond to the underlying equivalence relation which in our case is syntactic equality. **ET** eliminates trivial equations and since $\widetilde{\top} = \bot$ and $\widetilde{\bot} = \top$ trivial disequations are also eliminated by the rules (ET) and (C). **R** removes ground terms by including information they contain into some newly built predicates.

Syntactic Decomposition: D

$$
\begin{array}{lll}
(D_2) & p(x_1,\dots,t,\dots,x_n) \twoheadrightarrow & \exists x : x = t \ \wedge \ p(x_1,\dots,x,\dots,x_n) \\
 & & \text{if } t \notin \mathcal{X} \text{ and } t \notin \mathcal{T}_{\mathcal{F}} \\
(D_3) & x = f(\dots,y,\dots,y,\dots) \twoheadrightarrow & \exists z : x = f(\dots,y,\dots,z,\dots) \wedge y = z \\
(D_4) & f(t_1,\dots,t_n) = f(t'_1,\dots,t'_n) \twoheadrightarrow & \bigwedge_{i=1}^{n} t_i = t'_i \quad \text{if } n > 0 \\
(D_5) & f(t_1,\dots,t_n) = g(t'_1,\dots,t'_m) \twoheadrightarrow & \bot
\end{array}
$$

Elimination of trivial equations: ET

$$(ET)\ t = t \twoheadrightarrow \quad \top$$

Reduction: R

$$(R)\ p(t_1,\dots,t_{i-1},t,t_{i+1},\dots,t_n) \twoheadrightarrow \quad \Pi_{i/t}(p)(t_1,\dots,t_{i-1},t_{i+1},\dots,t_n) \quad \text{if } t \in \mathcal{T}_{\mathcal{F}} \text{ and } n \geq 0$$

Orientation: O

$$(O_2)\ t = x \twoheadrightarrow \quad x = t \quad \text{if } x \in \mathcal{X} \text{ and } t \notin \mathcal{X} \text{ or else } x < t$$

The **T** rules given below consist in merging information contained into equality and relational constraints. Unfortunately, such a merging does not always induce a recognizable relation. The **SD** rules are at the heart of the restriction over formulae for which the Theorem 1 can be extended. Indeed, their interest is to make rules **T** applicable when it is possible by checking if an equality constraint together with a recognizable relational constraint induce a recognizable relation. The semantics of (SD_1) is the following: for any $n > 1$, an n-ary relation r is said to be *decomposable* iff there are two relations p and q, m-ary and

k-ary respectively with $k > 0$, $m > 0$, $n = k + m$, and a permutation θ of $[1, n]$ such for any tuple $\langle t_1, \ldots, t_n \rangle$, $\langle t_1, \ldots, t_n \rangle \in r$ iff $\langle t_{\theta_1}, \ldots, t_{\theta_m} \rangle \in p$ and $\langle t_{\theta_{m+1}}, \ldots, t_{\theta_n} \rangle \in q$. This means that the parameters indexed by $\langle \theta_1, \ldots, \theta_m \rangle$ are relationally independent from those indexed by $\langle \theta_{m+1}, \ldots, \theta_n \rangle$. This is analogous to complete bipartite graphs. When (T_4) is not applicable, *i.e.* when the merging of an equality constraint with a relational one does not induce a recognizable relation, either the relational constraint is infinite and the algorithm fails (we cannot transform the global constraint into a recognizable relational one) or it is finite and we decompose the relation into a disjunction of equalities which is the only case which allow to built an equivalent recognizable relation.

Tree conversion and integration: T

$$
\begin{array}{lrll}
(T_2) & x = t & \twoheadrightarrow & \overrightarrow{t}(x) \quad \text{if } t \in \mathcal{T}_{\mathcal{F}[\mathcal{P}^*]} \\
(T_3) & \exists y : x = t \wedge \psi & \twoheadrightarrow & x = \sigma(t) \wedge \psi \\
 & & & \text{if } x \in \mathcal{X}, y \notin \mathcal{FV}ar(\psi),\ \sigma = \{y \mapsto \Omega_{\mathcal{F}}\} \\
 & & & \text{and } y \text{ occurs exactly once in } t \\
(T_4) & \exists y : x = t \wedge p(y) \wedge \psi & \twoheadrightarrow & x = \sigma(t) \wedge \psi \\
 & & & \text{if } x \in \mathcal{X}, y \notin \mathcal{FV}ar(\psi),\ \sigma = \{y \mapsto p\} \\
 & & & \text{and } y \text{ occurs exactly once in } t
\end{array}
$$

$x = t$ *in* (T_2), (T_3) *and* (T_4) *can be replaced with* $\neg(x = t)$

Semantic Decomposition: SD

$$
\begin{array}{lrll}
(SD_1) & p(x_1, \ldots, x_n) & \twoheadrightarrow & \Pi_I(p)(x_i \mid i \in I) \wedge\ \Pi_J(p)(x_j \mid j \in J) \\
 & & & \text{if } p = \mathfrak{S}_{I \cup J}\left(\Pi_I(p) \times \Pi_J(p)\right) \\
 & & & \text{where } I \cup J \text{ is a permutation}^3 \text{ of } [1, n] \\
(SD_2) & \exists y : \begin{array}{l} \psi \wedge x = t \\ \wedge\ p(x_1, \ldots, x_{k-1}, y, \ldots, x_n) \end{array} & \twoheadrightarrow & \displaystyle\bigvee_{u \in [\![\Pi_k(p)]\!]_{\wp}} \left(\sigma_u(\psi \wedge x = t) \wedge \bigwedge_{i=1}^{n} x_i = u_i \right) \\
 & & & \text{if } y \in \mathcal{V}ar(t),\ y \notin \{x_1, \ldots, , x_n\} \cup \mathcal{FV}ar(\psi) \\
 & & & \text{and } [\![\Pi_k(p)]\!]_{\wp} \text{ finite and } u = \langle u_1, \ldots, u_n \rangle \\
 & & & \text{and } \sigma_u : \begin{cases} x_i \mapsto u_i, & i \in [1, n] \\ y \mapsto \Pi_{1/u_1, \ldots, n/u_n}(p) \end{cases}
\end{array}
$$

Propagation: PR

$$(PR)\ p(x_1, \ldots, x_n) \vee \psi \twoheadrightarrow \quad p(x_1, \ldots, x_n) \vee (\psi \wedge \widetilde{p}(x_1, \ldots, x_n))$$

We should point out that the condition of (SD_1) is decidable since the cartesian product of automata is computable and the equality of languages recognized by automata is decidable (since the emptiness is decidable for tree automata). Moreover, since the finiteness of the language recognized by an automaton is decidable [C$^+$08], the condition of (SD_2) is also decidable.

We write now $\varphi \twoheadrightarrow^* \varphi'$ if φ' is obtained from φ by applying all the rules introduced so far such that at each transformation step the applied rule is chosen from one of the sets $\{\mathbf{P}, \mathbf{E}, \mathbf{N}, \mathbf{Q}, \mathbf{U}, \mathbf{D}, \mathbf{ET}, \mathbf{T}, \mathbf{O}, \mathbf{G}, \mathbf{R}, \mathbf{C}, \mathbf{M}, (\mathbf{PR}, \mathbf{SD})_{\mathbf{T}}\}$ in this order and where $(\mathbf{PR}, \mathbf{SD})_{\mathbf{T}}$

[3] $I \cup J$ is the permutation denoted by the sequence $\langle i_1, \ldots, i_m, j_1, \ldots, j_k \rangle$ where $m = \#I$ and $k = \#J$.

denotes the repeated application of **PR** and **SD** conditioned by an immediate subsequent application of **T**.

Proposition 2. *The Proposition 1 holds with the definition of $\twoheadrightarrow^*$ given above.*

Proof. The proof is given in Appendix C.

Example 4. Let φ be the formula $\inf(y,x) \wedge \forall z\ (inf(z,x) \Rightarrow (z = y \vee inf(z,y)))$. Given $\sigma : x \mapsto t \in \mathcal{T}_{\mathcal{F}}$, one can check that (as shown in Appendix E):

$$\Gamma \models^? \sigma(\varphi) \quad \twoheadrightarrow^* \quad \Gamma \models^? \underbrace{\left(\Pi_{2/t}(\inf) \sqcap \widetilde{\Pi}_1 \left(\left(\mathfrak{U}_2\ (\Pi_{2/t}(\inf)) \sqcap \widetilde{Id^2_{\mathcal{F}}} \sqcap \widetilde{\inf}\right)\right)\right)}_{p^*_{[t]}}(y)$$

For $t = succ(succ(succ(zero)))$ and $\wp$ given in Example 3, $\left[\!\left[p^*_{[t]}\right]\!\right]_{\wp} = \{succ(succ(zero))\}$. Thus, $f(succ(succ(succ(zero))), zero) \rightarrow^{\Gamma}_{\mathcal{R}_2} succ(f(succ(succ(zero)), zero))$.

Unfortunately, Lemma 2 cannot be extended to any formula since a system which transforms only recognizable sets cannot compute the solution of any formula.

Proposition 3. *The problem of solving unrestricted CRPs in recognizable theories is undecidable.*

This follows from the undecidability of the emptiness problem for a language denoted by a tree automaton with equality and disequality constraints (AWEDC) [C$^+$08]. Consequently, our algorithm is unable to solve all CRPs and thus to decide the (one-step) application of a CTRS in general. Indeed, in order to be sure that we can decide whether a constrained rule $lhs \xrightarrow{\varphi} rhs$ can be applied or not for any possible argument, we should be able to decide whether (or not) $lhs \,\|^{\varphi} \ll_{\Gamma} t$ for any Γ and t. It is thus necessary to restrict the language of formulae allowed in constraints to ensure that any CRP associated to a CMP can be solved. Since our transformation process can get stuck only due to formulae that cannot be transformed into a recognizable set, we look for a class of CRP which deals only with recognizable sets.

More precisely, we propose an algorithm which decides if a formula can be reduced in solved form by our algorithm if some of its free variables are instantiated by arbitrary terms. The algorithm can then be used to decide for any pair (lhs, φ) if the solution of $\Gamma \models^? \sigma(\varphi)$ will be computable with $\twoheadrightarrow$ for any Γ and σ *s.t.* $\mathcal{D}om(\sigma) = \mathcal{V}ar(lhs)$. For this, we define a new system $\rightsquigarrow$ which transforms formulae built over the initial signature but whose variables are taken either from the initial set of variables or from a new set of symbols called *mutable ground terms*. Intuitively, these symbols denote ground terms whose value is unknown and allow us to explore all possible transformations of a formula independently of the ground instantiations of a certain set of variables (the ones of the *lhs*). We write $\varphi \rightsquigarrow^* \psi$ iff ψ is obtained from φ by applying the sequence of rules $\{\mathbf{P}, \mathbf{E}, \mathbf{N}, \mathbf{Q}, \mathbf{U}, \mathbf{D'}, \mathbf{ET'}, \mathbf{MU}, \mathbf{T'}, \mathbf{O}, \mathbf{G}, \mathbf{R'}, \mathbf{C}, \mathbf{M}, (\mathbf{PR}, \mathbf{SD'})_{\mathbf{T}}\}$ in this order where previously defined rules are completed by the following ones:

Orientation: O

$$\left\|\begin{array}{llll} (O_3)\ t_x = x \rightsquigarrow & x = t_x & \text{if } t_x \text{ is a mutable ground term and } x \in \mathcal{X} \\ (O_4)\ t = t_x \rightsquigarrow & t_x = t & \text{if } t_x \text{ is a mutable ground term and } t \in \mathcal{T}_{\mathcal{F},\mathcal{X}} \end{array}\right.$$

Mutate: MU

$$\left\|\begin{array}{llll} (MU_1) & t_x = t \rightsquigarrow & \{\top\ ;\ \bot\} & \text{if } t \in \mathcal{T}_{\mathcal{F}}, t_x \text{ is a mutable ground term} \\ (MU_2)\ t_x = t \wedge \psi \rightsquigarrow & & \{\psi\ ;\ \bot\} & \text{if } t \in \mathcal{T}_{\mathcal{F},\mathcal{X}}, t_x \text{ is a mutable ground term and } \mathcal{V}ar(t) \cap \mathcal{FV}ar(\psi) = \varnothing \\ (MU_3) & t_x = t \rightsquigarrow & \{\top\ ;\ \bot\ ;\ \Box\} & \text{if } t \in \mathcal{T}_{\mathcal{F}[\mathcal{P}^*]} \text{ and } t_x \text{ is a mutable ground term} \end{array}\right.$$

and where **D'**, **ET'**, **T'**, **R'**, and **SD'** correspond to the non-primed rules changed as follows: (SD'_1) and (SD'_2) include the additional condition: $[\![p]\!]_{\wp}$ does not depend on $\wp$ (which is the case for example if p is built from constants and operators); for (T'_3), (T'_4) and (SD'_2) x is either a variable or a mutable ground term; for (ET') and (R') t is either a ground or mutable ground term; for D'_2 the condition that t is not a mutable ground term is added. Additionally, we suppose that the t_i in all rules stand for terms or mutable ground terms.

Proposition 4. *Given a first order formula φ and a set of variables V, the CRP $\Gamma \models^{?} \sigma(\varphi)$ has a normal form w.r.t. $\twoheadrightarrow$ for any ground substitution σ s.t. $\mathcal{D}om(\sigma) = V$ and recognizable theory Γ iff $\mu(\varphi)$ has a normal form w.r.t. $\rightsquigarrow$ where μ associates to any variable x of V free in φ a mutable ground term t_x. In such a case, we say that φ is* strongly rational *w.r.t. V.*

Thus, given a CTRS, the constrained rewriting can be perform using the transformation system $\twoheadrightarrow$ iff for each rule $lhs \xrightarrow{\varphi} rhs$, φ is strongly rational *w.r.t.* $\mathcal{V}ar(lhs)$.

6 Conclusion

We have proposed in this paper a new definition of constrained rewriting by considering all first order formulae and focused on the corresponding matching problem. In order to deal with this problem, we proposed an original characterization of a large class of theories using tree automata: recognizable theories. We developed next a whole framework to reason about recognizable theories and a transformation system to decide constrained matching in recognizable theories. We have shown that the constrained matching in recognizable theories is undecidable in general but proposed an algorithm to check if the constrained matching problem *w.r.t.* a constrained rewrite system is decidable with the developed transformation system. As we have already said, the algorithms proposed in this paper only covers theories containing no axioms over the equality predicate. However, a simple but useful extension is obtained for theories in which the equality relation can be denoted by a linear term rewriting system such that left and right hand sides of the rules do not share variables [Tis00]. Indeed, such an equality relation is recognizable and $t = t'$ must be translated by $\exists t'' : Eq(t, t') \wedge Eq(t', t'')$ for any terms $t, t' \in \mathcal{T}_{\mathcal{F},\mathcal{X}}$, where Eq is a fresh predicate whose interpretation is the reducibility relation induced by the rewrite system. Although this extension is simple, it is useful in the context of security policies where equality axioms often model simple evaluation functions assigning, for example, constant security levels to

users or objects. General extensions seem difficult to be established since most of the closure properties are lost for general equational tree automata [OT02] and, in particular, they are not closed for boolean operations. Nevertheless, for certain useful cases, recognizable equational tree languages are closed under union and intersection [Ohs01,OS07] and this opens some encouraging perspectives for more general extensions.

References

BCJK09. T. Bourdier, H. Cirstea, M. Jaume, and H. Kirchner. On Formal Specification and Analysis of Security Policies. *http://hal.inria.fr/inria-00429240/en/*, 2009.

BD89. L. Bachmair and Nachum Dershowitz. Completion for rewriting modulo a congruence. *Theoretical Computer Science*, 67(2-3):173–201, 1989.

BN98. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

C$^+$08. H. Comon et al. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2008. release November, 18th 2008.

CD94. H. Comon and C. Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, August 1994.

CL89. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.

Coh81. P. M. Cohn. *Universal algebra*, volume 6 of *Mathematics and its applications*. D. Reidel Publishing Company, 1981. Previous ed.: New York : Harper & Row, 1965.

Com92. H. Comon. Completion of rewrite systems with membership constraints. In W. Kuich, editor, *Int. Coll. on Automata, Languages and Programming, LNCS 623*. Springer Verlag, 1992.

FSVY91. T. Fruhwirth, E. Shapiro, M.Y. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *IEEE Symposium on Logic in Computer Science*, pages 300–309, 1991.

GL05. J. Goubault-Larrecq. The `h1` tool suite documentation. Available on: `http://www.lsv.ens-cachan.fr/~goubault/`, 2005. release December, 6th 2005.

GS97. F. Gćseg and M. Steinby. *Handbook of formal languages*, volume 3: Beyond words, chapter Tree languages. New York, NY, USA, 1997. Springer-Verlag New York, Inc.

Hin05. P.G. Hinman. *Fundamentals of mathematical logic*. A K Peters, Ltd., 2005. ISBN 1-56-881262-0.

Hoo92. C. Hoot. Completion for constrained term rewriting systems. In *Proceedings of the 3rd International Workshop on Conditional Term Rewriting Systems*, LNCS. Springer-Verlag, 1992.

Hue76. G. Huet. *Résolution d'equations dans les langages d'ordre 1,2, ...,ω*. PhD thesis, Université de Paris 7 (France), 1976.

Hue80. G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. In *Journal of the ACM*, volume 27, pages 797–821, 1980.

Jac96. F. Jacquemard. *Automates d'arbres et réécriture de termes*. PhD thesis, Université de Paris-Sud, UFR scientifique d'Orsay, 1996.

JK86. J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, 1986.

KK89. C. Kirchner and H. Kirchner. Constrained equational reasoning. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*. ACM, 1989.

KK06. C. Kirchner and H. Kirchner. Rewriting solving proving. Preliminary version of a book, 2006. `http://www.loria.fr/~hkirchne/rsp.pdf`.

KKM07. C. Kirchner, R. Kopetz, and P.-E. Moreau. Anti-pattern matching. In *ESOP 2007: European Symposium on Programming*, Lecture Notes in Computer Science, 2007.

NNH02. F. Nieslon, H.R. Nielson, and Seidl H. Normalizable horn clauses, strongly recognizable relations and spi. In *9th Static Analysis Symposium*, LNCS. Springer-Verlag, 2002.

Ohs01. H. Ohsaki. Beyond regularity: Equational tree automata for associative and commutative theories. In *CSL 2001: Computer Science Logic*, Lecture Notes in Computer Science, pages 539–553, 2001.

OS07. H. Ohsaki and H. Seki. Languages modulo normalization. In *International Symposium on Frontiers of Combining Systems*, Lecture Notes in Computer Science, pages 221–236, 2007.

OT02. H. Ohsaki and T. Takai. Decidability and closure properties of equational tree languages. In *Rewriting Techniques and Applications*, Lecture Notes in Computer Science, pages 114–128, 2002.

PS81. G. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28:233–264, 1981.

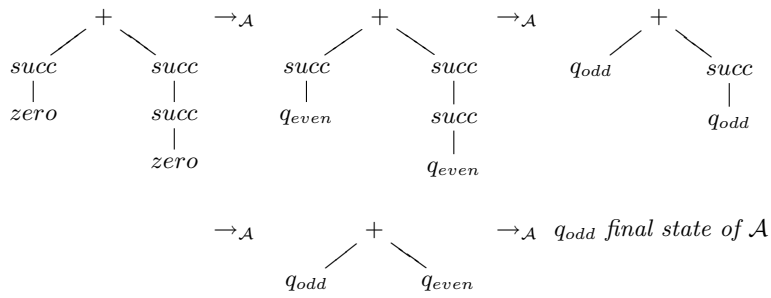
Tis00. S. Tison. Tree automata and term rewrite systems. In *RTA 2000: Rewriting Techniques and Applications*, Lecture Notes in Computer Science, pages 27–30, 2000.

A Examples of tree automata

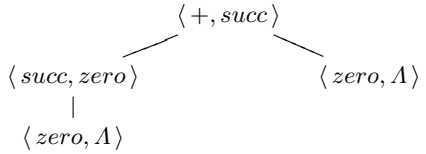
Let be $\mathcal{F} = \{+_{/2}, s_{/1}, zero_{/0}\}$. The following unary automaton $\mathcal{A} = (\mathcal{F}, \{q_{odd}, q_{even}\}, \{q_{odd}\}, \Delta)$ where Δ consists of: $zero \to q_{even}$ and

$$\begin{array}{lll} +(q_{odd}, q_{odd}) \to q_{even} & +(q_{even}, q_{even}) \to q_{even} & succ(q_{even}) \to q_{odd} \\ +(q_{odd}, q_{even}) \to q_{odd} & +(q_{even}, q_{odd}) \to q_{odd} & succ(q_{odd}) \to q_{even} \end{array}$$

recognizes the set of odd numbers in peano representation. The term $succ(zero)+succ(succ(zero))$ is in $\mathcal{L}(\mathcal{A})$:



The behaviour of n-ary automata is the same except that nodes of configurations are n-tuples. For example, to know if the pair $\langle\, succ(zero) + zero, succ(zero)\,\rangle$ is recognized by an automaton, we must build the following initial configuration:



B Example of a CTRS

Example 5. Given Σ, $\mathcal{R}_1$ and $\mathcal{R}_2$ defined in Example 1, We denote by $\mathcal{B}$ the partial Σ-algebra whose carrier is $\{0, 1\}$ and such that $zero_{\mathcal{B}} = 0$, $succ_{\mathcal{B}} = 0 \mapsto 1, 1 \mapsto 0$ and $inf_{\mathcal{B}} = \{(0, 1)\}$, then we have

$$\begin{aligned} f(succ(zero), zero) &\to^{\mathcal{B}}_{\mathcal{R}_2} succ(f(succ(succ(zero)), zero)) \\ &\qquad \text{since } \mathcal{B} \models inf(zero, succ(zero)), \mathcal{B} \models succ(succ(zero)) = zero \\ &\to^{\mathcal{B}}_{\mathcal{R}_2} succ(succ(f(succ(zero), zero))) \\ &\qquad \text{since } \mathcal{B} \models inf(succ(zero), succ(succ(zero))) \\ &\to^{\mathcal{B}}_{\mathcal{R}_2} succ(succ(succ(f(zero, zero)))) \\ &\qquad \text{since } \mathcal{B} \models inf(zero, succ(zero)) \\ &\to^{\mathcal{B}}_{\mathcal{R}_2} succ(zero) \\ &\qquad \text{since } \mathcal{B} \models succ(succ(succ(zero))) = succ(zero) \end{aligned}$$

Obviously, this is not the only possible reduction of $f(succ(zero), zero)$ *w.r.t.* $\to^{\mathcal{B}}_{\mathcal{R}_2}$. Nevertheless, the results of any other reduction can be reduced to $succ(zero)$ and one can see that f behaves as the classical exclusive or $\oplus$ operator *w.r.t.* $\to^{\mathcal{B}}_{\mathcal{R}_2}$.

C Semantics preservation (Propositions 1 and 2)

Let we first recall that given a theory Γ, the semantics of a formula φ (without free variables) in Γ is:

- *true* iff $\forall \Im \in \mathcal{M}od(\Gamma)$, $[\![\varphi]\!]_{\Im} = true$, *i.e.* $\Gamma \models \varphi$,
- *false* iff $\forall \Im \in \mathcal{M}od(\Gamma)$, $[\![\varphi]\!]_{\Im} = false$, *i.e.* $\Gamma \models \neg\varphi$ and
- *undefined* iff $\exists \Im, \Im' \in \mathcal{M}od(\Gamma)$, $[\![\varphi]\!]_{\Im} \neq [\![\varphi]\!]_{\Im'}$, *i.e.* neither $\Gamma \models \varphi$ nor $\Gamma \models \neg\varphi$.

For the following proofs, we will denote by $[\![\varphi]\!]_\Gamma$ the semantics of φ in Γ. When φ and φ' have the same semantics in Γ, we write $\varphi \equiv_\Gamma \varphi'$.

We first prove two lemmas that will be used in the proof of Proposition 2.

Lemma 3. *For any recognizable theory Γ, we have $[\![\top]\!]_\Gamma = true$, $[\![\bot]\!]_\Gamma = false$ and $[\![\Box]\!]_\Gamma = undefined$.*

Proof. If we consider that theses constants are 0-ary predicates, evaluating if a constant $\triangle$ is valid in Γ consists in checking, for any presentation $\wp$ of Γ, if (the only 0-ary tuple) $\langle\,\rangle$ belongs to $\mathcal{L}(\mathcal{S}_\wp(\triangle)) \cap \mathcal{L}(\mathcal{A}_\wp(\triangle))$ (true case), $\mathcal{L}(\mathcal{S}_\wp(\triangle)) \setminus \mathcal{L}(\mathcal{A}_\wp(\triangle))$ (false case) or in $\overline{\mathcal{L}(\mathcal{A}_\wp(\triangle))}$ (undefined case). Only four cases are possible:

$\langle\,\rangle \in \mathcal{L}(\mathcal{S}_\wp(\triangle))$	$\langle\,\rangle \in \mathcal{L}(\mathcal{A}_\wp(\triangle))$	*semantics of* $\triangle$ *in* Γ
✓	✓	*true*
✓		*false*
	✓	*undefined*
		undefined

which corresponds exactly ro Definition 5.

Starting from Definition 9 one can also verify that:

$$\begin{array}{lll} \top \sqcap \triangle \equiv_\Gamma \triangle & \top \sqcup \triangle \equiv_\Gamma \top & \widetilde{\top} \equiv_\Gamma \bot \\ \bot \sqcap \triangle \equiv_\Gamma \bot & \bot \sqcup \triangle \equiv_\Gamma \triangle & \widetilde{\bot} \equiv_\Gamma \top \\ \Box \sqcap \Box \equiv_\Gamma \Box & \Box \sqcup \Box \equiv_\Gamma \Box & \widetilde{\Box} \equiv_\Gamma \Box \end{array}$$

for any theory Γ and constant $\triangle$.

We also need to show that the boolean operations over predicates correspond to the logical operations.

Lemma 4 (Boolean operations). *Let $\Sigma = (\mathcal{F}, \mathcal{P})$ be a signature and a recognizable theory Γ. For any $p_{/n}, q_{/n} \in \mathcal{P}$ and $t_1, \ldots, t_n \in \mathcal{T}_\mathcal{F}$, $n \geq 0$:*

$$\begin{array}{rcl} \Gamma \models (p \sqcap q)(t_1, \ldots, t_n) & \textit{iff} & \Gamma \models p(t_1, \ldots, t_n) \textit{ and } \Gamma \models q(t_1, \ldots, t_n) \\ \Gamma \models (p \sqcup q)(t_1, \ldots, t_n) & \textit{iff} & \Gamma \models p(t_1, \ldots, t_n) \textit{ or } \Gamma \models q(t_1, \ldots, t_n) \\ \Gamma \models \widetilde{p}(t_1, \ldots, t_n) & \textit{iff} & \Gamma \models \neg p(t_1, \ldots, t_n) \end{array}$$

Proof. The semantics of $(p \sqcap q)(t_1, \ldots, t_n)$ in Γ is *true* or $false$ if:

- either the semantics of $p(t_1, \ldots, t_n)$ and the one of $q(t_1, \ldots, t_n)$ are not undefined, *i.e.* $\langle t_1, \ldots, t_n \rangle \in \mathcal{L}(\mathcal{S}_\wp(p) \cap S_\wp(q))$
- or $p(t_1, \ldots, t_n)$ (resp. $q(t_1, \ldots, t_n)$) is false in any model of Γ and then $\langle t_1, \ldots, t_n \rangle \in \mathcal{L}(\mathcal{S}_\wp(p)) \cap \overline{\mathcal{L}(A_\wp(p))}$ (resp. $\langle t_1, \ldots, t_n \rangle \in \mathcal{L}(\mathcal{S}_\wp(q)) \cap \overline{\mathcal{L}(A_\wp(q))}$).

In other cases, the semantics of $(p \sqcap q)(t_1, \ldots, t_n)$ in Γ is undefined. A similar reasoning justifies the support $(\mathcal{S}_\wp(p) \cap \mathcal{A}_\wp(p)) \cup (\mathcal{S}_\wp(q) \cap \mathcal{A}_\wp(q)) \cup (\mathcal{S}_\wp(p) \cap \mathcal{S}_\wp(q))$, for $p \sqcup q$.

Proposition 2 Given a Σ-formula φ and a recognizable theory Γ then, for any ψ such that $\Gamma \models^? \varphi \twoheadrightarrow^* \Gamma \models^? \psi$, we have:

$$\Gamma \models \sigma(\varphi) \quad \textit{iff} \quad \Gamma \models \sigma(\psi)$$

for any ground substitution σ with $\mathcal{D}om(\sigma) = \mathcal{FV}ar(\varphi)$.

Proof. The rules (N_1) and (N_2) perform the generalized distributivity law and thus clearly preserve the semantics. (Q_1) and (Q_2) preserve the semantics since the existential quantifier distributes over disjunctions and the universal quantifier distributes over conjunctions. (U_1) applies the De Morgan law $\neg(a \wedge b) \equiv (\neg a) \vee (\neg b)$ and the fact that $\forall x : \varphi$ is equivalent to $\neg\exists x : \neg\varphi$. The correctness of the rules (U_2) and (E) come from the fact that for any quantifier Q, $Qx : \varphi$ is equivalent to φ if $x \notin \mathcal{FV}ar(\varphi)$. The correctness of rule (P) follows from Lemma 1. The preservation of the semantics of rules **C** and **M** is proved in Lemma 4. Rules O_1 and T_1 as well as the **G** rules are preserving semantics by construction (*i.e.* definition of the corresponding operators). For D_1 the property obviously holds.

The remaining **D** rules preserve the semantics if the equality over terms is interpreted as the syntactic equality, which is the case here. The correctness of rule ET follows from Lemma 3. The property obviously holds for rule O_2. The correctness for the rule (R) follows from the semantics of the projection directed by a ground term given by Lemma 1 and Definition 13. Note that the semantics in Γ of the projection of a unary predicate p directed by t corresponds to the evaluation of $\Gamma \models p(t)$.

As far as it concerns the remaining **T** rules, one can notice that the rules T_3 and T_4 are the only ones that can lead to a formula containing terms from $\mathcal{T}_{\mathcal{F}[\mathcal{P}^*]}$. Moreover, if the original formula contains no such terms (which is the case for our constraints) then any $p \in \mathcal{P}^*$ occuring in a $t \in \mathcal{T}_{\mathcal{F}[\mathcal{P}^*]}$ from the resulted formula is necessary unary. According to Definition 8, for any theory Γ presented by $\wp$, $\overrightarrow{t}$ denotes in Γ the set of ground terms built from t by replacing any $p \in \mathcal{P}^*$ by a ground term t' which belongs to $[\![p]\!]_\wp$, *i.e.* such that $\Gamma \models p(t')$. Rules (T_3) and (T_4) build terms of $\mathcal{T}_{\mathcal{F}[\mathcal{P}^*]}$ for the case where their variables are constrained and respectively not in related formulae while rule (T_2) just transforms equalities involving a variable and such a term into the corresponding predicate. Notice the linearity conditions that guarantee that there exist no implicit equality constraints that might be forgotten by the transformation.

The correctness of rule (SD_1) follows from the fact that given $p \in \mathcal{P}^*$ such that $p = \mathfrak{S}_{I \cup J}(\Pi_I(p) \times \Pi_J(p))$, $\Gamma \models p(t_1, \ldots, t_n)$ iff $\Gamma \models \mathfrak{S}_{I \cup J}(\Pi_I(p) \times \Pi_J(p))(t_1, \ldots, t_n)$ iff $\Gamma \models \Pi_I(p)(t_i \mid i \in I) \wedge \Pi_J(p)(t_j \mid j \in J)$ for any $t_i \in \mathcal{T}_\mathcal{F}$.

Finally, to see that (SD_2) preserves the semantics of the problem it transforms, it is sufficient to remark that:

- for any formula ψ, variable x and ground term t, $x = t \wedge \psi$ is equivalent to $x = t \wedge \sigma(\psi)$ where $\sigma : x \mapsto t$,
- for any predicate p, $\exists p(y) \wedge \psi$ is valid in Γ iff there exists a $t \in [\![p]\!]_\wp$ such that $\sigma(\psi)$ is valid in Γ where $\wp$ is a presentation of Γ and $\sigma : y \mapsto t$; $\Gamma \models \exists p(y) \wedge \psi$ is then equivalent to $\Gamma \models \bigvee_{t \in [\![p]\!]_\wp} \sigma_t(\psi)$ with $\sigma_t : y \mapsto t$.

Once again, the linearity condition guarantees that no implicit constraints are lost.

Finally, rule PR obviously preserves the semantics.

D Solved forms (Lemma 2)

Lemma 2 Any CRP $\Gamma \models^? \varphi$ has a solved form $\Gamma \models^? \varphi^*$ if φ contains no functional symbol.

Proof. Any formula in prenex form φ is of the general shape : $(Q_i x_i)_{i=1}^m : \psi$ where $(Q_i x_i)_{i=1}^m$ is seen as a "waiting line" in which any quantifier will be successively "resolved" (*i.e.* removed). Whatever the last quantifier Q_m is, the formula can be rewritten by the rules **N**, **Q** and **U** into a formula of the form:

$$(Q_i x_i)_{i=1}^{m-1} : \bigvee \begin{cases} \exists x_m : \bigwedge_j a_j^1 & \longleftarrow \psi_1 \\ \dots & \\ \exists x_m : \bigwedge_j a_j^q & \longleftarrow \psi_q \end{cases}$$

or

$$(Q_i x_i)_{i=1}^{m-1} : \bigwedge \begin{cases} \neg\exists x_m : \bigwedge_j \neg a_j^1 & \longleftarrow \psi_1 \\ \dots & \\ \neg\exists x_m : \bigwedge_j \neg a_j^q & \longleftarrow \psi_q \end{cases}$$

where a_j^k are literals. Each ψ_k can be resolved independently of the others as follows:

1. any non-linear literal (*i.e.* in which a variable occurs several times) can be transformed into a linear one (containing no equalities) using the rules **D** and **T**;
2. any literal can be transformed into an atom using **C** and we can obtain thus a conjunction containing only linear atoms;
3. this kind of conjunctions can be transformed by the rules **O** and **G** into a conjunction whose atoms have all the same parameters and in the same order:

$$\exists x_m : \bigwedge_j p_j^k(y_1, \dots, y_n)$$

4. this conjunction can be transformed by **M** into an atom:

$$\exists x_m : p^k(y_1, \dots, y_n)$$

5. the rules in **P** or **E** can then be repeatedly applied to obtain an atom without quantifier:

$$p_k^*(z_1, \ldots, z_{n-1})$$

These transformations are applied over each ψ_k and then we obtain:

$$(Q_i x_i)_{i=1}^{m-1} : \bigvee \begin{cases} p_1^*(z_1^1, \ldots, z_{n_1}^1) \\ \ldots \\ p_q^*(z_1^q, \ldots, z_{n_q}^q) \end{cases}$$

and respectively

$$(Q_i x_i)_{i=1}^{m-1} : \bigwedge \begin{cases} \neg p_1^*(z_1^1, \ldots, z_{n_1}^1) \\ \ldots \\ \neg p_q^*(z_1^q, \ldots, z_{n_q}^q) \end{cases}$$

If $m \leq 1$, we can obtain the solved form of φ by applying **M** (if $m = 0$, we skip the last step. Alternatively, we can extract a next quantifier from the waiting line and repeat the sequence of transformations described above. Thus, we eventually reach a quantifier free formula whose solved form can be easily obtained.

E Detail of the Example 4

Example 4: Let φ be the formula $\inf(y, x) \wedge \forall z\ (inf(z, x) \Rightarrow (z = y \vee inf(z, y)))$. Given $\sigma : x \mapsto t \in \mathcal{T}_{\mathcal{F}}$, one can check that:

$$\Gamma \models^? \sigma(\varphi) \quad \twoheadrightarrow^* \quad \Gamma \models^? \underbrace{\left(\Pi_{2/t}(\inf) \sqcap \widetilde{\Pi}_1 \left(\left(\mathfrak{U}_2\ (\Pi_{2/t}(\inf)) \sqcap \widetilde{Id_{\mathcal{F}}^2} \sqcap \widetilde{\inf}\right)\right)\right)}_{p^*_{[t]}}(y)$$

Indeed: $\forall z : \bigwedge \begin{cases} inf(y,t) \\ \neg inf(z,t) \vee z = y \vee inf(z,y) \end{cases}$

$$\twoheadrightarrow_{(Q_2)} \bigwedge \begin{cases} \forall z : inf(y,t) \\ \forall z : \neg inf(z,t) \vee z = y \vee inf(z,y) \end{cases}$$

$$\twoheadrightarrow_{(U_2)} \bigwedge \begin{cases} inf(y,t) \\ \forall z : \neg inf(z,t) \vee z = y \vee inf(z,y) \end{cases}$$

$$\twoheadrightarrow_{(R_1)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \forall z : \neg \Pi_{2/t}(inf)(z) \vee z = y \vee inf(z,y) \end{cases}$$

$$\twoheadrightarrow_{(U_1)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \exists z : \Pi_{2/t}(inf)(z) \wedge \neg(z = y) \wedge \neg inf(z,y) \end{cases}$$

$$\twoheadrightarrow_{(C)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \exists z : \Pi_{2/t}(inf)(z) \wedge \neg(z = y) \wedge \widetilde{inf}(z,y) \end{cases}$$

$$\twoheadrightarrow_{(T_1)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \exists z : \Pi_{2/t}(inf)(z) \wedge \neg Id_{\mathcal{F}}^2(z,y) \wedge \widetilde{inf}(z,y) \end{cases}$$

$$\twoheadrightarrow_{(C)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \exists z : \Pi_{2/t}(inf)(z) \wedge \widetilde{Id_{\mathcal{F}}^2}(z,y) \wedge \widetilde{inf}(z,y) \end{cases}$$

$$\twoheadrightarrow_{(G)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \exists z : \mathfrak{U}_2\ (\Pi_{2/t}(inf))(z,y) \wedge \widetilde{Id_{\mathcal{F}}^2}(z,y) \wedge \widetilde{inf}(z,y) \end{cases}$$

$$
\begin{aligned}
&\twoheadrightarrow_{(M_1)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \exists z : \left(\mathfrak{U}_2\ (\Pi_{2/t}(inf)) \sqcap \widetilde{Id^2_{\mathcal{F}}} \sqcap \widetilde{inf} \right)(z, y) \end{cases} \\
&\twoheadrightarrow_{(P)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \neg \Pi_1 \left(\left(\mathfrak{U}_2\ (\Pi_{2/t}(inf)) \sqcap \widetilde{Id^2_{\mathcal{F}}} \sqcap \widetilde{inf} \right) \right)(y) \end{cases} \\
&\twoheadrightarrow_{(C)} \bigwedge \begin{cases} \Pi_{2/t}(inf)(y) \\ \widetilde{\Pi_1} \left(\left(\mathfrak{U}_2\ (\Pi_{2/t}(inf)) \sqcap \widetilde{Id^2_{\mathcal{F}}} \sqcap \widetilde{inf} \right) \right)(y) \end{cases} \\
&\twoheadrightarrow_{(M_1)} \left(\Pi_{2/t}(inf) \sqcap \widetilde{\Pi}_1 \left(\left(\mathfrak{U}_2\ (\Pi_{2/t}(inf)) \sqcap \widetilde{Id^2_{\mathcal{F}}} \sqcap \widetilde{inf} \right) \right) \right)(y)
\end{aligned}
$$

For $t = succ(succ(succ(zero)))$ and $\wp$ given in Example 3, $\left[\!\left[p^*_{[t]} \right]\!\right]_{\wp} = \{succ(succ(zero))\}$. Thus, $f(succ(succ(succ(zero))), zero) \rightarrow^{\Gamma}_{\mathcal{R}_2} succ(f(succ(succ(zero)), zero))$.