INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Termination Proofs using gpo Ordering Constraints

– extended version –

Thomas Genet and Isabelle Gnaedig

N° 3087

January 1997

THÈME 2

Rapport de recherche

ISSN 0249-6399

Termination Proofs using gpo Ordering Constraints

– extended version –

Thomas Genet and Isabelle Gnaedig*

Thème 2 — Génie logiciel
et calcul symbolique
Projet PROTHEO

Rapport de recherche n° 3087 — January 1997 — 37 pages

Abstract: We present here an algorithm for proving termination of term rewriting systems by *gpo* ordering constraint solving. The algorithm gives, as automatically as possible, an appropriate instance of the gpo generic ordering, proving termination of a given system. Constraint solving is done efficiently thanks to a DAG shared term data structure.

Key-words: Term Rewriting, Termination Proof, Ordering Constraint Solving

(Résumé : tsvp)

* Email:{Thomas.Genet,Isabelle.Gnaedig}@loria.fr, http://www.loria.fr/equipe/protheo.html

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

Preuves de terminaison par résolution de contraintes d'ordre gpo

– version étendue –

Résumé : Nous présentons un algorithme de preuve de terminaison des systèmes de réécriture par résolution de contraintes d'ordre *gpo*. L'algorithme fournit, de la façon la plus automatique possible, une instance de l'ordre générique gpo, prouvant la terminaison d'un système de réécriture donné. L'explosion des calculs au cours de la résolution est maîtrisée grâce à une structure de termes partagés.

Mots-clé : Systèmes de réécriture, Preuve de terminaison, Résolution de contraintes d'ordre

Contents

1 Introduction

To prove termination of a Term Rewrite System (TRS for short), the most commonly used method is to define a well-founded ordering between terms and show that each rewrite step is a strictly decreasing step. In general, the proof is made by verification: orderings are tested at random or using human expertise, until an appropriate one is found.

Our goal here, is to reduce human expertise by working in a constructive way: starting from constraints on a generic ordering, we will help the user to build an appropriate specific instance of this ordering by using semi-automatic constraint solving methods.

The generic ordering, we start from, is the general path ordering (*gpo*) designed by Dershowitz and Hoot [3] for expressing in a single notion, a large set of well known orderings: syntactic orderings such as *rpo* [12] or *lpo* [9], as well as semantic orderings like *spo* [9] or polynomial orderings [11]. It is based on a lexicographic combination of *termination functions*. Particular orderings, such as those cited above, are obtained by instantiating termination functions with particular values.

Our idea here is to combine the genericity of *gpo* with the constructive power of the constraint approach, to provide a method as automatic as possible for proving termination of TRS. Starting from inequalities on a general path ordering, we will reduce the set of possibilities for instantiation of termination functions by constraint solving, until we get a particular ordering when it is possible.

The problem tackled here is different from the constraint approach of ordering problems already proposed [2, 15, 16, 14, 8]. All are concerned with the satisfiability problem of ordering constraints. In our approach however, we try to find a *gpo* ordering for validating inequalities between terms.

The algorithm, we propose, works in two steps, using a term sharing data structure for rewrite rules. The first step consists in syntactical solving of *gpo* constraints to give *proof obligations* which are ordering constraints on *gpo* termination functions. The second step consists in proving satisfiability of proof obligations to find an instance of *gpo* satisfying initial *gpo* constraints.

In Section 2, the *gpo* definition and the related termination theorem are recalled. In Section 3, *gpo* constraints, a constraint approach of termination proofs by *gpo* as well as specific data structures for achieving *gpo* constraint solving are presented. Then, in Section 4, deduction rules for *gpo* constraint solving are defined, followed by soundness, completeness and complexity theorems. An example of deduction process is detailed in Section 5. Finally, in Section 6, practical methods for testing satisfiability of proof obligations are given.

2 The starting point: *gpo*

Let F be a set of function symbols, each one with an arity, X a set of variable symbols, $T(F,X)$ the set of terms defined on F and X, and $T(F)$ the set of ground terms. For definitions of multiset, ordering, quasi-ordering, multiset extension, lexicographic extension, well-founded ordering, rewriting and other usual definitions, see [4].

Let us recall the definition of *gpo* ordering from Dershowitz and Hoot [3]. This definition is based on component orderings defined as follows:

A component ordering on $T(F)$ is a pair $\langle\theta_i, \geq_i\rangle$ such that

- θ_i is a homomorphism from $T(F)$ to an algebra A and $\geq_i$ is a well-founded quasi-ordering on A, or
- θ_i is a function (called multiset extraction function in [3]) from terms to multisets of selected immediate subterms, that is $\theta_i(f(s_1, \dots, s_n)) = \{s_{j_1}, \dots, s_{j_m}\}$, such that $j_1, \dots, j_m \in \{1, \dots, n\}$ and $\geq_i$ is the multiset extension of *gpo* itself.

For any quasi-ordering $\geq_i$, we have: $\simeq_i \; = \; \geq_i \cap \leq_i$ and $>_i \; = \; \geq_i \cap \not\leq_i$. The θ_i are called *gpo termination functions.* For any term $s \in T(F)$, we denote by $\Theta_{i,j}(s)$ the tuple $\langle\theta_i(s), \dots, \theta_j(s)\rangle$, where $0 \leq i < j$ and $\theta_i, \dots, \theta_j$ are *gpo* termination functions. Let $>_{lex}$ be the associated ordering, i.e the lexicographic combination of orderings $>_i, \dots, >_j$, and $\simeq_{lex}$ be the associated equivalence, i.e. the lexicographic combination of equivalences $\simeq_i, \dots, \simeq_j$, such that $>_i, \dots, >_j$ and $\simeq_i, \dots, \simeq_j$ are respectively related to the homomorphisms $\theta_i, \dots, \theta_j$. Let $\geq_{lex}$ be the relation $>_{lex} \cup \simeq_{lex}$. We denote $\Theta_{0,k}$ by Θ.

Definition 1 *(General Path Ordering)(Dershowitz & Hoot [3]).*
Let $\langle\theta_i, \geq_i\rangle$ be component orderings. The general path ordering $\geq_{gpo}$ *on $T(F)$ is inductively defined by $\geq_{gpo} \; = \; >_{gpo} \cup \simeq_{gpo}$ where*

- $s = f(s_1, \dots, s_n) >_{gpo} g(t_1, \dots, t_m) = t$

if one of the two following cases holds:

1. *$s_i \geq_{gpo} t$ for some subterm s_i of s, or*
2. *$s >_{gpo} t_1, \dots, s >_{gpo} t_m$ and $\Theta(s) >_{lex} \Theta(t)$.*

- $s = f(s_1, \dots, s_n) \simeq_{gpo} g(t_1, \dots, t_m) = t$

if $s >_{gpo} t_1, \dots, s >_{gpo} t_m$, $t >_{gpo} s_1, \dots, t >_{gpo} s_n$ and $\Theta(s) \simeq_{lex} \Theta(t)$.

Theorem 1 *(Dershowitz & Hoot [3]) Let $\geq_{gpo}$ be a gpo. A rewrite system R terminates on $T(F)$ if*

- *$l\sigma >_{gpo} r\sigma$ for all rules $l \rightarrow r$ of R, all ground substitution σ and,*
- *$\forall s, t \in T(F)$, $s \rightarrow_R t$ and $s \geq_{gpo} t$ implies $f(\dots, s, \dots) \geq_{gpo} f(\dots, t, \dots)$ for any ground context $f(\dots \dots)$.*

Let $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ be a specific instance of $(\Theta, \geq_{lex})$, where $\mathcal{T}_{0,k}$ is the combination of *gpo* termination functions $\tau_0, \dots, \tau_k$, and $\succsim_0, \dots, \succsim_k$ are the related quasi-orderings; $\succ_{lex}$ is the lexicographic combination of $\succ_0, \dots, \succ_k$, $\approx_{lex}$ is the lexicographic combination of

$\approx_0, \ldots, \approx_k$, and $\succsim_{lex} = \succ_{lex} \cup \approx_{lex}$. When choosing a specific $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ for $(\Theta, \geq_{lex})$, we obtain *instances* of *gpo*, such as for example lexicographic path ordering (Kamin & Lévy [9]), multiset path ordering, polynomial path ordering (Lankford [11]). For more details, see Dershowitz and Hoot [3]. An instance of the *gpo* based on a particular $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ will be denoted $\succ^{\Phi}_{gpo}$.

For operationally proving termination of TRSs with *gpo*, the usual approach consists in defining a specific *gpo* $\succ^{\Phi}_{gpo}$ on ground terms and implicitly using $\succ^{\Phi}_{gpo}$ as an ordering on terms with variables by proving that for all rule $l \rightarrow r$ (with $l, r \in T(F, X)$) of the TRS, we have $l \succ^{\Phi}_{gpo} r$ and $\forall s, t \in T(F, X)$, $\forall \sigma$ such that $s\sigma, t\sigma \in T(F)$, we have $s \succ^{\Phi}_{gpo} t \implies s\sigma \succ^{\Phi}_{gpo} t\sigma$. For more clarity, we chose here to explicitly define *gpo* on terms with variables. We first extend the definitions of Θ and $>_{lex}$ on terms with variables as follows.

Definition 2 *Let $s, t \in T(F, X)$.*

- $\Theta(s) >_{lex} \Theta(t)$ *if for all substitution σ s.t. $s\sigma, t\sigma \in T(F)$, we have $\Theta(s\sigma) >_{lex} \Theta(t\sigma)$,*
- $\Theta(s) \simeq_{lex} \Theta(t)$ *if for all substitution σ s.t. $s\sigma, t\sigma \in T(F)$, we have $\Theta(s\sigma) \simeq_{lex} \Theta(t\sigma)$,*
- $\Theta(s) \geq_{lex} \Theta(t)$ *if $\Theta(s) >_{lex} \Theta(t)$ or $\Theta(s) \simeq_{lex} \Theta(t)$.*

Thanks to this extension of the termination functions to $T(F, X)$, the definition of the general path ordering is extended to $T(F, X)$. From now on, we will use the definition of *gpo* on $T(F, X)$.

Theorem 2 *The general path ordering $\geq_{gpo}$ is a quasi-ordering on $T(F, X)$ having the subterm property.*

Proof Straightforward adaptation of proofs of [3] to the non-ground case. See Appendix B.1 for details. In the following proposition, we state that *gpo* is stable by instantiation with any substitution mapping any term to a ground term.

Proposition 1 *(Ground stability) Let $s, t \in T(F, X)$ and $\Phi = (\Theta, \geq_{lex})$. If $s \succ^{\Phi}_{gpo} t$ (resp. $s \approx^{\Phi}_{gpo} t$) then for all substitution σ s.t. $s\sigma, t\sigma \in T(F)$, we have $s\sigma \succ^{\Phi}_{gpo} t\sigma$ (resp $s\sigma \approx^{\Phi}_{gpo} t\sigma$).*

Proof By induction on the height of terms on both sides of the inequality. See Appendix B.2 for details. Thanks to Proposition 1, for proving the first condition of the theorem 1 (termination theorem), it is enough to prove that: $l \succ^{\Phi}_{gpo} r$ for all rules $l \rightarrow r$ of R, and for a ground stable instance $\succ^{\Phi}_{gpo}$ of *gpo*. Restricting to ground stable instances of *gpo* is not critical since the instances of *gpo* used in practice are ground stable.

3 Solving *gpo* constraints using a term sharing data structure

A constraint approach of ordering problems has already been proposed in several works [2, 15, 16, 14, 8]. All are concerned with the satisfiability problem of ordering constraints.

Let s_i and t_i $(i = 1, \ldots, n)$ be terms of $T(F, X)$. Given a precedence $>_F$ and a conjunction of ordering constraints $\bigwedge_{i=1}^{n} s_i > t_i$, Comon [2] and Nieuwenhuis [14] gave algorithms for deciding if there exists a ground substitution σ such that $s_i\sigma >_{lpo} t_i\sigma$ holds for $i = 1, \ldots, n$ ($>_{lpo}$ denotes the *lpo* ordering). Given a conjunction of ordering constraints $\bigwedge_{i=1}^{n} s_i > t_i$, Plaisted [16], and Johann & Socher-Ambrosius [8] gave polynomial time algorithms for deciding if there exists a ground substitution σ and a simplification ordering $\succ$ such that $s_i\sigma \succ t_i\sigma$ holds for $i = 1, \ldots, n$.

In this paper, what we call *gpo ordering constraints* are formulas built on $s > t$ and $s \sim t$ where $s, t \in T(F, X)$. What we call *solving of gpo ordering constraints* $s > t$ (resp. $s \sim t$) is to decide whether there exists a particular instance Φ of *gpo* such that $s \succ_{gpo}^{\Phi} t$ (resp. $s \approx_{gpo}^{\Phi} t$), in constructing a specific Φ. Solving of *gpo* constraints is undecidable in general, but our goal is to build a procedure giving, when it is possible, an appropriate ordering on a semi-automatic way. First, our solving process produces constraints on $(\Theta, \geq_{lex})$ from *gpo* constraints. Second, we try to solve constraints on $(\Theta, \geq_{lex})$ in finding an appropriate instance for Θ and $\geq_{lex}$.

Getting constraints on $(\Theta, \geq_{lex})$ from *gpo* constraints can be achieved by a deduction rule coming directly from *gpo* definition on $T(F, X)$. This rule, called the *decomposition rule*, is based on the same deduction mechanism as the rules presented in [2, 14]:

$$\frac{s > t}{\bigvee_{i=1 \ldots n}(s_i > t) \;\vee\; \bigvee_{i=1 \ldots n}(s_i \sim t) \;\vee\; (\bigwedge_{i=1 \ldots m} s > t_i \;\wedge\; \Theta(s) >_{lex} \Theta(t))}$$

where $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_m)$. If we apply a set of deduction rules, composed by the decomposition rule and trivial constraint simplification rules, to *gpo* ordering constraints, we end with a normal form built on constraints on Θ and $\geq_{lex}$. For the whole set of rules and an example of deduction, see [6]. If we try to implement the decomposition rule as it is, we obtain an explosion of the size of the formula and an explosion of the computation time.

The reason is that the decomposition rule duplicates constraints. In the example of the *gpo* constraint $f(g(a), g(a)) > g(b)$, decomposition works as follows:

$$\frac{f(g(a), g(a)) > g(b)}{\begin{array}{c} g(a) > g(b) \;\vee\; g(a) > g(b) \;\vee\; g(a) \sim g(b) \;\vee\; g(a) \sim g(b) \;\vee \\ (f(g(a), g(a)) > b \;\wedge\; \Theta(f(g(a), g(a))) >_{lex} \Theta(g(b))) \end{array}}$$

If we implement the decomposition rule as it is, the constraint $g(a) > g(b)$ will be solved twice. This is also the case for constraints $g(a) \sim g(b)$. But duplication is not uniquely due to duplicate occurrences of subterms in terms (like $g(a)$ in $f(g(a), g(a))$). For example, a complete decomposition of the *gpo* constraint $f(f(a)) > f(g(b))$ produces 3 duplications of the constraint $f(a) > g(b)$, 5 duplications of $a > g(b)$, 5 duplications of $f(a) > b$, 4 duplications of $a \sim g(b)$, 7 duplications of $a \sim b$ and 14 duplications of $a > b$. So limitating duplicated computations is *essential* to design an efficient algorithm.

A first solution would be to avoid recomputation of every ordering constraint using a hash table of already solved constraints. This solution avoids repeating the solving of ordering constraints but does not prevent their duplication. In our last example, we would still get 14 duplications of $a > b$, solve the constraint once and refer to the solution 13 times in the hash table to get the solution of other occurrences.

A second solution, we present now, is a method for sharing constraints and their solutions. We chose to use a Directed Acyclic Graph representation for rewrite rules (see [13] for an application of shared term representation of rewrite rules to completion). In this representation, terms are graphs where nodes are labeled by symbols and edges represent the subterm relation. The DAG representation allows sharing of common subterms of distinct terms. On this DAG representation of terms, we additionally define edges representing ordering constraints labeled by logical formulas, we called here *proof obligations*. Proof obligations ($\mathcal{O}$-proofs for short) are defined as follows (recall that Θ denote $\Theta_{0,k} = \langle\theta_0, \ldots, \theta_k\rangle$):

Definition 3 *Let $\mathcal{X}_\mathcal{P}$ be a set of variables called the set of $\mathcal{O}$-proof variables. Let $\top$ be the trivial $\mathcal{O}$-proof, $s, t \in T(F, X)$, $P \in \mathcal{X}_\mathcal{P}$. The set $\mathcal{P}$ of $\mathcal{O}$-proofs is inductively defined by:*

- $\top \in \mathcal{P}$,
- $P \in \mathcal{P}$,
- $\Theta(s) >_{lex} \Theta(t) \in \mathcal{P}$, *and* $\Theta(s) \simeq_{lex} \Theta(t) \in \mathcal{P}$,
- $A \wedge B \in \mathcal{P}$, *if* $A, B \in \mathcal{P}$,
- $A \vee B \in \mathcal{P}$, *if* $A, B \in \mathcal{P}$.

We now define *satisfiability* of $\mathcal{O}$-proofs.

Definition 4 *Let Φ be the pair $(\mathcal{T}_{0,k}, \succsim_{lex})$. Let $P, A, B \in \mathcal{P}$ and $s, t \in T(F, X)$. We say that Φ satisfies P, denoted $\Phi \models P$ iff:*

- $P = \top$, *or*
- $P = A \vee B$ *and* $(\Phi \models A$ *or* $\Phi \models B)$, *or*
- $P = A \wedge B$ *and* $(\Phi \models A$ *and* $\Phi \models B)$, *or*
- $P = \Theta(s) >_{lex} \Theta(t)$ *and* $\mathcal{T}_{0,k}(s) \succ_{lex} \mathcal{T}_{0,k}(t)$, *or*
- $P = \Theta(s) \simeq_{lex} \Theta(t)$ *and* $\mathcal{T}_{0,k}(s) \approx_{lex} \mathcal{T}_{0,k}(t)$.

Let us now define the DAG representation of rewrite rules. We call those graphs *Ordering Constraint Solving Graphs* (OCS graphs for short).

Definition 5 *An OCS graph is a graph $G = (V, E)$ where V is the set of vertices (or nodes) labeled by symbols of F or variables of X, and $E \subseteq V \times V$ is the set of edges labeled by S, R, $>$ or $\sim$ for Subterm, Rewrite, inequality and equivalence edges respectively. The S, R, $>$ edges are directed. The $>$, $\sim$ edges are also labeled by an $\mathcal{O}$-proof. The subterm edges are also labeled by a natural i called the* rank *of the subterm edge. For all node $\mathcal{F} \in V$, labeled by $f \in F$ where arity of f is n, for all $i = 1 \ldots n$, there exists $\mathcal{G}_i \in V$ and a unique subterm edge $(\mathcal{F}, \mathcal{G}_i) \in E$ of rank i.*

The subterm and the rewrite edges in OCS graphs represent the direct subterm relation in the term and the rewrite relation between terms in rules respectively. The edges labeled by $\mathcal{O}$-proofs represent the constraints on $(\Theta, >_{lex})$, we want to obtain from *gpo* constraints in the first step of our solving process. Let us define the function *Term* mapping any node $\mathcal{F}$ of an OCS graph to a term t, such that $\mathcal{F}$ is the top node of the OCS graph representing t.

Definition 6 *Let $G = (V, E)$ be an OCS graph and $\mathcal{F} \in V$. The function $Term$ from V into $T(F, X)$ is inductively defined by:*

1. *if $\mathcal{F}$ is labeled by $x \in X$, then $Term(\mathcal{F}) = x$,*
2. *if $\mathcal{F}$ is labeled by $f \in F$ where arity of f is n, then $Term(\mathcal{F}) = f(Term(\mathcal{T}_1), \ldots, Term(\mathcal{T}_n))$ where for all $i = 1 \ldots n$, $\mathcal{T}_i \in V$, and $(\mathcal{F}, \mathcal{T}_i) \in E$ is a subterm edge of rank i.*

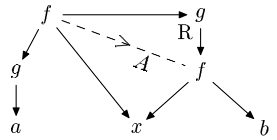
Definition 7 *Let $l, r \in T(F, X)$. An OCS representation of the rewrite rule $l \to r$ is an OCS $G = (V, E)$ such that:*

1. *$\forall \mathcal{F}, \mathcal{F}' \in V$ s.t. $\mathcal{F} \neq \mathcal{F}'$ we have $Term(\mathcal{F}) \neq Term(\mathcal{F}')$, and*
2. *there exist two nodes $\mathcal{F}, \mathcal{G} \in V$ and a unique rewrite edge $(\mathcal{F}, \mathcal{G}) \in E$ such that $Term(\mathcal{F}) = l$ and $Term(\mathcal{G}) = r$.*

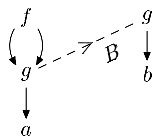
In the previous definition, note that 1. ensures sharing of subterms in the OCS representation of a rewrite rule.

In the following, for any OCS graph $G = (V, E)$ with $\mathcal{F}, \mathcal{G} \in V$, $\mathcal{F} \mathrel{-\!\!>\!\!-} \mathcal{G}$ (resp. $\mathcal{F} \mathrel{-\!\!\sim\!\!-} \mathcal{G}$) will denote an inequality edge (resp. equivalence edge) $(\mathcal{F}, \mathcal{G}) \in E$. We will note $\mathcal{F} \mathrel{\not{-\!\!>\!\!-}} \mathcal{G}$ (resp. $\mathcal{F} \mathrel{\not{-\!\!\sim\!\!-}} \mathcal{G}$) if there is no inequality edge (resp. equivalence edge) $(\mathcal{F}, \mathcal{G}) \in E$. If $Term(\mathcal{F}) = s$ and $Term(\mathcal{G}) = t$, then $\mathcal{F} \mathrel{-\!\!>\!\!-} \mathcal{G}$ (resp. $\mathcal{F} \mathrel{-\!\!\sim\!\!-} \mathcal{G}$) will also be denoted by $s \mathrel{-\!\!>\!\!-} t$ (resp. $s \mathrel{-\!\!\sim\!\!-} t$). Inequality and equivalence edges will be called *ordering edges*. In the following figures, plain arrows will denote subterm edges, plain arrows labeled by R will denote rewriting edges and dashed lines will denote inequality and equivalence edges. Rank labels are omitted but can be deduced from the figures since subterm edges will always be ordered by rank from left to right.

Example 1 *The OCS representation of the rewrite rule $f(g(a), x) \to g(f(x, b))$ with an inequality edge labeled by an $\mathcal{O}$-proof label A is presented in graph 1.1. The OCS graph 1.2 shows how the constraint $g(a) > g(b)$, duplicated in the previous example of decomposition of $f(g(a), g(a)) > g(b)$ can be represented by a unique edge labeled by an $\mathcal{O}$-proof label B.*



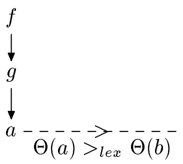
Graph 1.1



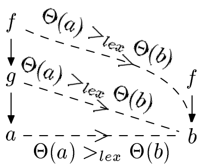
Graph 1.2

An OCS graph allows sharing of terms and sharing of constraints, but it may duplicate $\mathcal{O}$-proofs. In the following, we give an example of duplication as well as a method to avoid it.

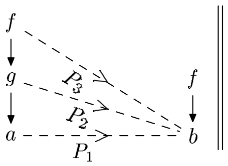
Example 2 *Let $f(g(a))$ and $f(b)$ be terms, where a and b are constants. By definition 4 and definition of gpo, we have $a \succ^{\Phi}_{gpo} b$ if $\Phi \models \Theta(a) >_{lex} \Theta(b)$. We can represent this by OCS Graph 2.1. Thanks to the subterm property, note that $g(a) \succ^{\Phi}_{gpo} b$ if $\Phi \models \Theta(a) >_{lex} \Theta(b)$, because $a \succ^{\Phi}_{gpo} b$ implies $g(a) \succ^{\Phi}_{gpo} b$. Similarly, $g(a) \succ^{\Phi}_{gpo} b$ implies $f(g(a)) \succ^{\Phi}_{gpo} b$. We thus obtain OCS Graph 2.2. The fact that $a \succ^{\Phi}_{gpo} b$* implies *$g(a) \succ^{\Phi}_{gpo} b$ does not appear in Graph 2.2. Then, if we want to prove that there exists Φ such that $a \succ^{\Phi}_{gpo} b$ and $g(a) \succ^{\Phi}_{gpo} b$, we will prove* twice *that there exists Φ such that $\Phi \models \Theta(a) >_{lex} \Theta(b)$. In order to represent implication between $\mathcal{O}$-proofs in the OCS Graph, we can refine the OCS representation by labeling ordering constraint edges by distinct variables (instead of formulas) and factorizing formulas in a substitution mapping each variable to a formula. Thus graph 2.2 can be transformed into graph 2.3, where the substitution is defined on the right side.*



Graph 2.1



Graph 2.2



$P_1 \mapsto \Theta(a) >_{lex} \Theta(b)$
$P_2 \mapsto P_1$
$P_3 \mapsto P_2$

Graph 2.3

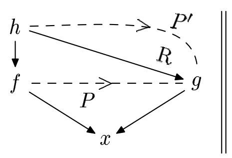
Let us define the structure used in Graph 2.3 more formally.

Definition 8 *A $\mathcal{P}$-substitution σ is an application from $\mathcal{X}_{\mathcal{P}}$ into $\mathcal{P}$, which can be uniquely extended into a homomorphism $\sigma : \mathcal{P} \mapsto \mathcal{P}$.*

Our structure for solving *gpo* ordering constraints is composed of an OCS graph representing a rewrite rule and a $\mathcal{P}$-substitution. Ordering edges of the OCS graph are labeled either by the trivial $\mathcal{O}$-proof $\top$ or by an $\mathcal{O}$-proof variable. The application of the substitution to an inequality (resp: equivalence) edge label of the graph gives an $\mathcal{O}$-proof of the corresponding inequality (resp: equivalence).

Definition 9 *A* Structure for Ordering Constraint Solving *(SOCS for short) of a rule $l \rightarrow r$ is a pair $(G||\sigma)$ where G is an OCS graph representing the rule, and σ is a $\mathcal{P}$-substitution.*

Example 3 *Here is a possible SOCS for the rule $h(f(x)) \rightarrow g(x)$:*



$P \mapsto \Theta(f(x)) >_{lex} \Theta(g(x))$
$P' \mapsto P$

In this SOCS, the inequality edge between nodes labeled by f and g means that we have **at least one** *possible $\mathcal{O}$-proof P for $f(x) > g(x)$. On the right hand side of the SOCS, we find the related $\mathcal{P}$-substitution mapping the variable P to the related $\mathcal{O}$-proof. The mapping $P' \mapsto P$ means that the $\mathcal{O}$-proof P is also an $\mathcal{O}$-proof for edge $h(f(x)) \longrightarrow\!\!\!> g(x)$.*

4 The $\mathcal{C}$-deduction rules

We now define the deduction rules, we use on SOCS to infer constraints on $(\Theta, \geq_{lex})$ from *gpo* constraints. Unlike the top-down process shown with the decomposition rule, the mechanism, we propose now, is bottom-up and is closer to the *gpo* definition itself. Let us first introduce *visibility*, which expresses a notion of sub-formula in $\mathcal{O}$-proofs.

Definition 10 *Let $P, Q \in \mathcal{P}$ be $\mathcal{O}$-proofs. P is* visible *in Q, denoted $P \trianglelefteq Q$ if $(P = Q)$ or $[Q = A \vee B$ and $(P \trianglelefteq A$ or $P \trianglelefteq B)]$.*

For solving *gpo* constraints on a set of rules, we start from a set of *initial SOCS*, one for each rule. Initial SOCSs are SOCSs whose OCS graphs have no ordering edge and whose $\mathcal{P}$-substitutions are empty. The *gpo* constraint solving on SOCS is achieved by a set of deduction rules. These rules transform a SOCS by adding ordering edges to the OCS graph and by constructing the corresponding $\mathcal{P}$-substitution, whose application will provide the corresponding $\mathcal{O}$-proofs. Solving is processed independently for each SOCS corresponding to each rewrite rule, and ends when no deduction rule applies any longer. Let us denote by $\mathcal{C}$ the set of deduction rules, by $\vdash_{\mathcal{C}}$ the deduction relation on SOCS induced by $\mathcal{C}$, and by $\vdash_{\mathcal{C}}^*$ the transitive closure of $\vdash_{\mathcal{C}}$. Let us call $\mathcal{C}$-deduction process the deduction process defined by $\mathcal{C}$. A SOCS is said to be in $\mathcal{C}$-normal form when no deduction rule applies to it. See the set of $\mathcal{C}$-deduction rules in figure 1, where an edge $\xrightarrow{\gtrsim P}$ denotes either $\xrightarrow{> P}$ or $\xrightarrow{\sim P}$.

Let $(\alpha||\nu), (\beta||\delta)$ be SOCS. A deduction rule $\frac{\alpha||\nu}{\beta||\delta}$ of $\mathcal{C}$ *matches* a SOCS $(G||\sigma)$ if α is a pattern of G, if ν matches σ, and if the precondition of $\frac{\alpha||\nu}{\beta||\delta}$ is verified. Then, the *application* of the rule consists in replacing the pattern α of G by the pattern β (which can be identical) and by replacing ν by δ in σ. Each time a new ordering edge is constructed in G, it is supposed to be labeled by a new $\mathcal{O}$-proof variable. Note that nodes $\mathcal{F}$ and $\mathcal{G}$ of the $\mathcal{C}$-deduction rules must always match distinct nodes of G. This prevents from adding cyclic inequality ordering edges (always false w.r.t. *gpo*) and cyclic equivalence edges (always unnecessary for deductions).

Note also that no special strategy is required when rules are applied: either for the choice of the pair of nodes, or for the choice of the rule to apply. As a result, the process can be parallelized as it is. Thanks to the $\mathcal{O}$-proof structure, application of rules can be concurrent on a SOCS: n processes can apply deduction rules concurrently on pairs of nodes of the same SOCS, provided that no processes consider identical or symmetrical (i.e. $(\mathcal{F}, \mathcal{G})$ and $(\mathcal{G}, \mathcal{F})$) pairs of nodes simultaneously.

In case of a sequential implementation, although no strategy is needed, adapted strategies lead to a faster algorithm. For example, in our implementation of $\mathcal{C}$-deduction rules (see Appendix A), we first saturate the graph with trivial $\mathcal{O}$-proofs thanks to rules **SUBTERM Property** and **SUBTERM Trivial** before applying other rules (except **SUBTERM Simplification** that becomes useless since the OCS graph is already saturated with trivial $\mathcal{O}$-proofs).

Let us now give three theorems: soundness, completeness and complexity of $\mathcal{C}$.

Definition 11 *Given a SOCS* $(G||\sigma)$, $s,t \in T(F,X)$, *an edge* $s \xrightarrow{P} t$ *of* G *(resp.* $s \overset{P}{\rightsquigarrow} t$*) is* correct *w.r.t. gpo if for any* $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ *such that* $\Phi \models P$, *we have* $s \succ^{\Phi}_{gpo} t$ *(resp.* $s \approx^{\Phi}_{gpo} t$*). A SOCS* $(G||\sigma)$ *is* correct *w.r.t. gpo if all ordering edges of* G *are correct w.r.t. gpo.*

Theorem 3 *(Soundness) For all initial SOCS* S, *if* $S \vdash^*_{\mathcal{C}} S'$ *then* S' *is correct w.r.t. gpo.*

Theorem 4 *(Completeness) Let* $(G||\sigma)$ *be a SOCS in* $\mathcal{C}$*-normal form, where* $G = (V,E)$. *For all nodes* $\mathcal{F}, \mathcal{G} \in V$, *s.t.* $Term(\mathcal{F}) = s$ *and* $Term(\mathcal{G}) = t$, *where* $s,t \in T(F,X)$*:*

$$\forall \Phi = (\mathcal{T}_{0,k}, \succsim_{lex}) \text{ s.t. } s \succ^{\Phi}_{gpo} t, \text{ there exists an edge } s \xrightarrow{P} t \text{ in } G \text{ s.t. } \Phi \models P\sigma$$

$$\forall \Phi = (\mathcal{T}_{0,k}, \succsim_{lex}) \text{ s.t. } s \approx^{\Phi}_{gpo} t, \text{ there exists an edge } s \overset{P}{\rightsquigarrow} t \text{ in } G \text{ s.t. } \Phi \models P\sigma.$$

Theorem 5 *(Complexity) Let* $l \to r$ *be a rewrite rule,* $(G||\sigma)$ *the initial SOCS of* $l \to r$, N *the number of nodes of* G, *and* M *the non-zero maximal arity of function symbols of the rule. The complexity in time and space of the* $\mathcal{C}$*-deduction process starting from* $(G||\sigma)$ *is polynomial in* N *and* M *in the worst case.*

For detailed proofs of theorems, see Appendices B.3, B.4, B.5.

As explained above, each rule of a rewrite system is treated independently. For constraint solving on the whole set of rules, we have to gather the results relative to rules.

Definition 12 *Let* R *be a rewrite system* $(l_i \to r_i,\ i = 1 \ldots n)$ *whose SOCSs* $(G_i \mid\mid \sigma_i)$, *representing the rules* $l_i \to r_i$ *are in* $\mathcal{C}$*-normal form. Let* P_i *be the* $\mathcal{O}$*-proof label of the edge* $l_i \twoheadrightarrow r_i$ *in* G_i *for any* $i = 1 \ldots n$. *The* global $\mathcal{O}$-proof *of* R *is the* $\mathcal{O}$*-proof:* $P_1\sigma_1 \ \wedge \ \ldots \ \wedge \ P_n\sigma_n$.

Note that if there is a rule $l_i \to r_i$ such that there is no edge $l_i \xrightarrow{P_i} r_i$ in G_i, then there is no possible termination proof with *gpo* for the whole TRS R. Definition 12 shows that $\mathcal{O}$-proofs offer a nice method for dealing with the problem of incrementally adding rules in TRS. This feature can be very useful for completion procedures. Note also that, in a SOCS, we generate inequality edges and equivalence edges for the two possible orientations of the rewrite rules (left to right and right to left). The final orientation of the TRS is defined while constructing its global $\mathcal{O}$-proof. As a result, we can try different orientations for the TRS by choosing different inequality edges and $\mathcal{O}$-proofs in the SOCS while constructing the global $\mathcal{O}$-proof.

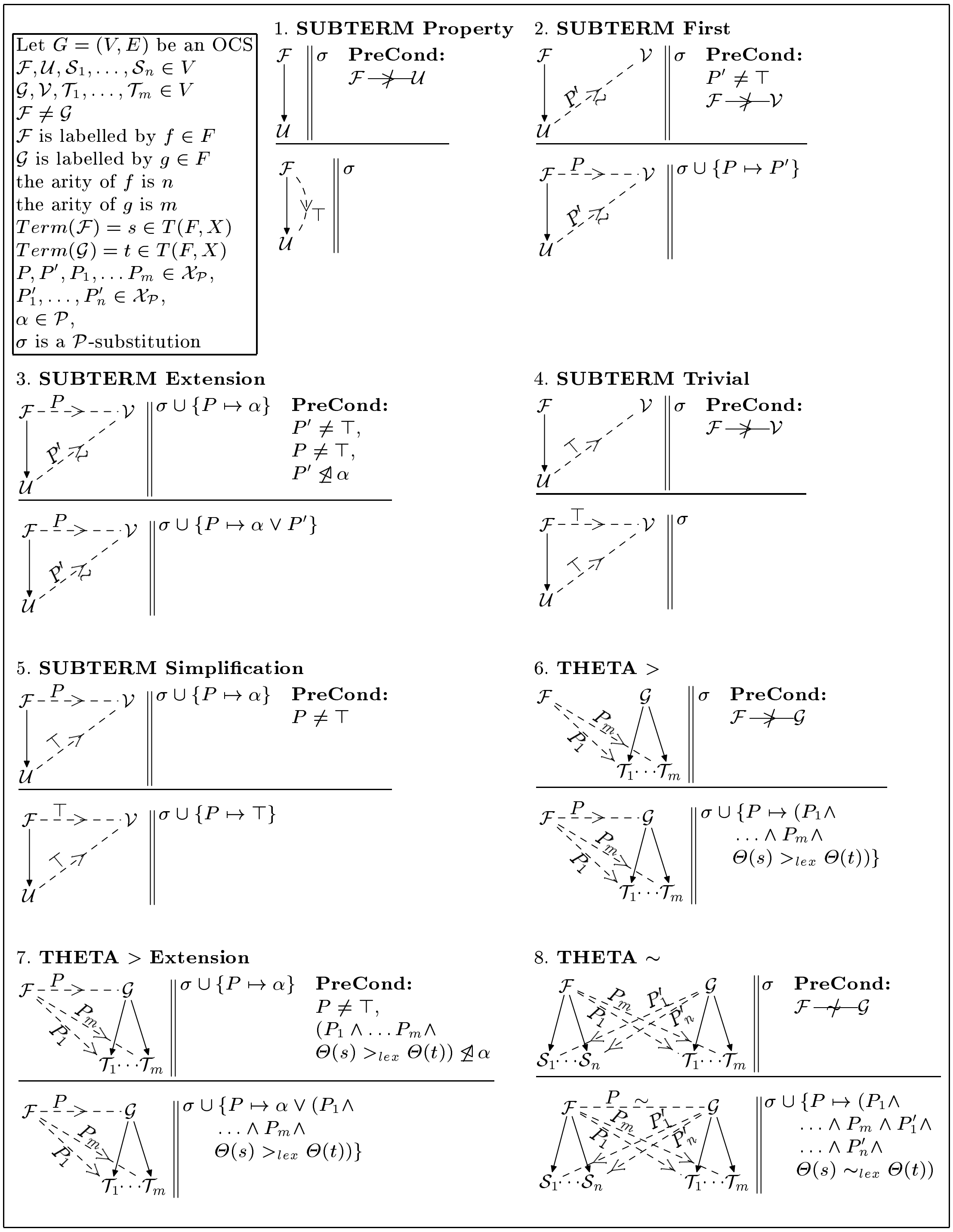


Figure 1: The $\mathcal{C}$-deduction rules

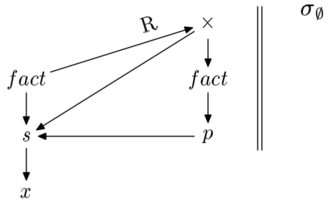
Note also that an other approach would be to consider a unique SOCS for a whole term rewriting system, like in [13] for sets of equations. With such a global approach, our soundness, completeness and complexity results still hold. Moreover, with some specific TRS, we obtain simpler $\mathcal{O}$-proofs with a global approach. However, even if a global approach is theoretically more convincing, our experiments with simple instances of *gpo* show in practice that a local approach is less time consuming.

5 An example of $\mathcal{C}$-deduction process

Consider the following system, borrowed from [3], for computing factorial in unary arithmetic. Let R be:

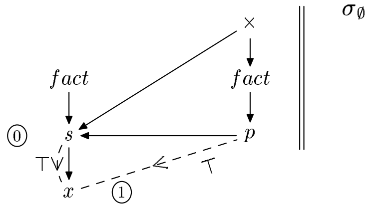
$$
\begin{array}{llll}
p(s(x)) \to x & (1) & s(x) \times y \to (x \times y) + y & (5) \\
fact(0) \to s(0) & (2) & x + 0 \to x & (6) \\
fact(s(x)) \to s(x) \times fact(p(s(x))) & (3) & x + s(y) \to s(x+y) & (7) \\
0 \times y \to 0 & (4) & &
\end{array}
$$

The termination of R cannot be proven with a simplification ordering since rule (3) is self-embedded. However it is possible to prove termination of R with *gpo*. Let $\sigma_\emptyset$ stand for the empty $\mathcal{P}$-substitution. We start with the initial SOCS corresponding to the rule (3) of R, as defined in Section 4:



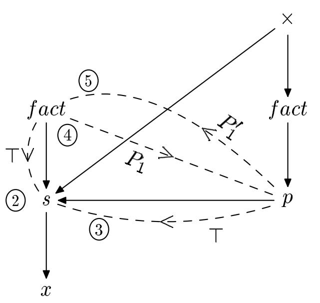
Let us explain *some of the steps* of the resolution process. For better readability, we choose to discard rewrite edges, to represent only significant ordering edges, and to label them by a number.

1. When starting, since there are no ordering edges, the only rule that can be applied is **SUBTERM Property**. Thanks to applications of **SUBTERM Property**, we obtain some trivial orientation edges for direct subterms, as for example ⓪.



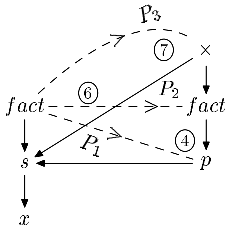
Thanks to edge ⓪, rule **SUBTERM Trivial** can be applied (Nodes $\mathcal{F}$,$\mathcal{U}$ and $\mathcal{V}$ of deduction rule **SUBTERM Trivial** match, respectively, nodes labeled by p, s and x) and edge ① is obtained.

2. Assuming that deduction has raised edge ②, thanks to rule **THETA** $>$, we can deduce edge ④ and introduce the related $\mathcal{P}$-substitution in the SOCS (note that the $\mathcal{O}$-proof of edge ② does not appear in the $\mathcal{P}$-substitution of P_1 since it is trivial). Symmetrically, thanks to edge ③ and another application of **THETA** $>$, we can deduce edge ⑤ and introduce the $\mathcal{P}$-substitution in the SOCS. (Note also that we could add an equivalence edge thanks to edges ②, ③ and rule **THETA** $\sim$).



$P_1 \mapsto \Theta(fact(s(x))) >_{lex} \Theta(p(s(x)))$
$P_1' \mapsto \Theta(p(s(x))) >_{lex} \Theta(fact(s(x)))$

3. Again with rule **THETA** $>$, we can also deduce edge ⑥, thanks to existent edge ④. Still with **THETA** $>$we can add edge ⑦ thanks to new edge ⑥.



$P_1 \mapsto \Theta(fact(s(x))) >_{lex} \Theta(p(s(x)))$
$P_2 \mapsto P_1 \wedge \Theta(fact(s(x))) >_{lex} \Theta(fact(p(s(x))))$
$P_3 \mapsto P_2 \wedge \Theta(fact(s(x))) >_{lex} \Theta(s(x) \times fact(p(s(x))))$

In the following, $P_{(1)}, \ldots, P_{(7)}$ will denote the $\mathcal{O}$-proofs for rules (1) to (7) respectively. We will denote σ as the $\mathcal{P}$-substitution of the final SOCS for rule (3). Since after deduction on the SOCS for rule (3), we obtain $fact(s(x)) \xrightarrow{P_3} s(x) \times fact(p(s(x)))$, the $\mathcal{O}$-proof $P_{(3)}$ for rule (3) is $P_3\sigma$. Note that $\mathcal{O}$-proofs for rules (1), (4) and (6) are trivial ones. Thus, the global $\mathcal{O}$-proof for the complete TRS is $P_{(2)} \wedge P_{(3)} \wedge P_{(5)} \wedge P_{(7)}$.

6 Proving satisfiability of $\mathcal{O}$-proofs

At this stage of the solving process, we have obtained a set of saturated SOCS (one for each rewrite rule), with non-instantiated $\mathcal{O}$-proofs in the $\mathcal{P}$-substitution part: no assumption is made on Θ nor on $\geq_{lex}$. The next step of our solving process consists of proving the satisfiability of an $\mathcal{O}$-proof by finding particular values $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ of $(\Theta, \geq_{lex})$ in order to satisfy the initial constraints.

Let us show how to proceed in practice for verifying the satisfiability of an $\mathcal{O}$-proof, using the *partial instantiation* process we now define. In the following, what we call *instantiated literals* of an $\mathcal{O}$-proof are either of the form $\tau_i(s) \succ_i \tau_i(t)$ or $\tau_i(s) \approx_i \tau_i(t)$, where τ_i are termination functions and $\succ_i$ are associated orderings.

Definition 13 *Given* $0 \leq i < j$, *a* $\Theta_{i,j}$ *$\mathcal{O}$-proof is an $\mathcal{O}$-proof whose every non-instantiated literal is either of the form* $\Theta_{i,j}(s) >_{lex} \Theta_{i,j}(t)$ *or of the form* $\Theta_{i,j}(s) \simeq_{lex} \Theta_{i,j}(t)$ *where* s, t *are terms of* $T(F, X)$.

Definition 14 *Given* $0 \leq i < j$ *and* P *a* $\Theta_{i,j}$ *$\mathcal{O}$-proof, a* left partial instantiation *(LPI for short) of* P *is obtained by instantiating every* θ_i *in* P *by a particular termination function* τ_i.

Note that if we consider an $\mathcal{O}$-proof $\Theta_{i,j}(s) >_{lex} \Theta_{i,j}(t)$, its LPI is

$$\tau_i(s) \succ_i \tau_i(t) \vee [\tau_i(s) \approx_i \tau_i(t) \wedge \Theta_{i+1,j}(s) >_{lex} \Theta_{i+1,j}(t)].$$

If we consider an $\mathcal{O}$-proof $\Theta_{i,j}(s) \simeq_{lex} \Theta_{i,j}(t)$, its LPI is

$$\tau_i(s) \approx_i \tau_i(t) \wedge \Theta_{i+1,j}(s) \simeq_{lex} \Theta_{i+1,j}(t).$$

A practical method for finding a solution to our constraint problem in a global $\mathcal{O}$-proof thanks to LPI can be based on DAGs. Let $\mathcal{O}$-proof DAGs be and-or DAGs representing $\mathcal{O}$-proofs: a conjunctive $\mathcal{O}$-proof $\alpha \wedge \beta$ is represented by the DAG $\begin{smallmatrix} A \\ | \\ B \end{smallmatrix}$ and a disjunctive $\mathcal{O}$-proof $\alpha \vee \beta$ is represented by the DAG $A \; B$, where A and B are DAGs representing α and β, respectively.

Definition 15 *Given* $0 \leq i < j$ *and* G *an* $\Theta_{i,j}$ *$\mathcal{O}$-proof DAG, an* i-path *of* G *is a pair* (p, A) *where* p *is a path from top to bottom of* G, *and* A *is a tuple of sets* $\langle A_0, \ldots, A_{i-1}, A_i \rangle$, *where* A_u $(0 \leq u \leq i-1)$ *is the set* $\{\alpha | \alpha \in p, \alpha = \tau_u(s) >_u \tau_u(t)$ *or* $\alpha = \tau_u(s) \simeq_u \tau_u(t), s, t \in T(F, X)\}$ *and* $A_i = \{\alpha | \alpha \in p, \alpha = \Theta_{i,j}(s) >_{lex} \Theta_{i,j}(t)$ *or* $\alpha = \Theta_{i,j}(s) \simeq_{lex} \Theta_{i,j}(t), s, t \in T(F, X)\}$.

Definition 16 *Let* S *be a finite set. A set of inequalities and equalities* $A = \{\alpha \succ \beta | \alpha, \beta \in S\} \cup \{\alpha \approx \beta | \alpha, \beta \in S\}$ *is* compatible *iff there exists a quasi-ordering* $\succsim_S$ *on* S *such that* $\alpha \succ \beta \in A \implies \alpha \succ_S \beta$ *and* $\alpha \approx \beta \in A \implies \alpha \approx_S \beta$ *(where* $\succsim_S$ *stands for* $\succ_S \cup \approx_S$*).*

Informally, an $\mathcal{O}$-proof contains a solution if its $\mathcal{O}$-proof DAG contains a path from top to bottom, whose nodes are instantiated and compatible. Let us now introduce the notion of *minimal i-path*, an i-path minimizing the set of constraints on non-instantiated termination functions $\Theta_{i,k}$ and related ordering $>_{lex}$.

Definition 17 *Let (p, A) be an i-path, where $A = \langle A_0, \ldots, A_{i-1}, A_i \rangle$. The i-path (p, A) is minimal if $A_0, \ldots, A_{i-1}$ are compatible sets, and if there exists no i-path (p', B), such that $B = \langle B_0, \ldots, B_{i-1}, B_i \rangle$, where $B_0, \ldots, B_{i-1}$ are compatible sets and $B_i \subset A_i$.*

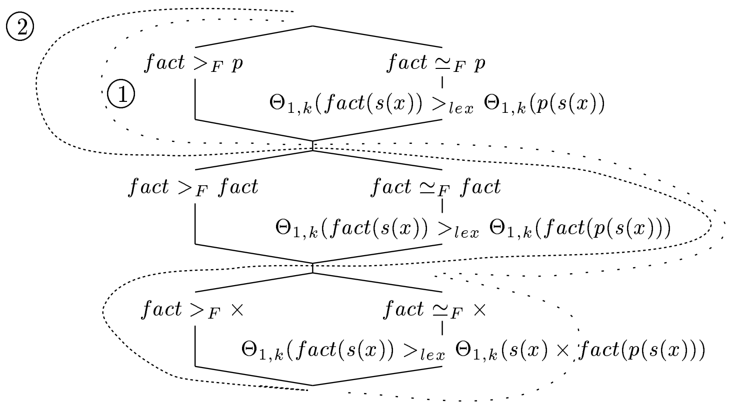
Note that, in general, a minimal i-path is not unique.

Definition 18 *A* satisfiable i-path *is a minimal i-path where $A_i = \emptyset$.*

A satisfiable i-path in an $\mathcal{O}$-proof DAG represents a solution of the related $\mathcal{O}$-proof. We now illustrate those definitions, with an example of satisfiability proof on a specific $\mathcal{O}$-proof. In Section 5, we obtained $P_{(3)} = P_3\sigma$ where σ denote the $\mathcal{P}$-substitution of the final SOCS for rule (3). $P_3\sigma$ can be represented by the $\mathcal{O}$-proof DAG:

$$\begin{array}{c}
\Theta(fact(s(x))) >_{lex} \Theta(p(s(x))) \\
| \\
\Theta(fact(s(x))) >_{lex} \Theta(fact(p(s(x)))) \\
| \\
\Theta(fact(s(x))) >_{lex} \Theta(s(x) \times fact(p(s(x))))
\end{array}$$

Recall that Θ is a simplified notation for $\Theta_{0,k}$. In order to find a specific $\Theta_{0,k}$ and a related $>_{lex}$ satisfying this $\mathcal{O}$-proof, we apply a left partial instantiation on $\Theta_{0,k}$. Let $(\theta_0, \geq_0)$ be a precedence: θ_0 is a function mapping any term to its root symbol, and $\geq_0$ is an ordering on F, still unknown, that we want to infer automatically. Left partial instantiation applied to the previous $\mathcal{O}$-proof DAG leads to the following $\mathcal{O}$-proof DAG:



We now search for a solution, beginning with the simplest one: a satisfiable 1-path in the $\mathcal{O}$-proof DAG. The paths labeled by ① and ②, among others, are 1-paths. Path ① is associated with the tuple $A = \langle A_0, A_1 \rangle$ where

$A_0 = \{fact >_F p, fact \simeq_F fact, fact \simeq_F \times\}$ and
$A_1 = \{\Theta_{1,k}(fact(s(x))) >_{lex} \Theta_{1,k}(fact(p(s(x)))),$

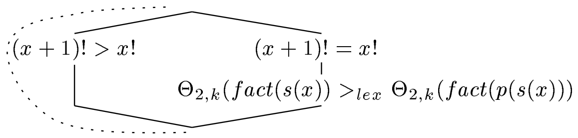
$$\Theta_{1,k}(fact(s(x))) >_{lex} \Theta_{1,k}(s(x) \times fact(p(s(x))))\}$$

Path ② is associated with tuple $B = \langle B_0, B_1 \rangle$ where

$$B_0 = \{fact >_F p, fact \simeq_F fact, fact >_F \times\} \text{ and}$$
$$B_1 = \{\Theta_{1,k}(fact(s(x))) >_{lex} \Theta_{1,k}(fact(p(s(x))))\}$$

Components A_0 and B_0 are both compatible. However, the 1-path ① is not minimal since $B_1 \subset A_1$. In this particular example, there is a unique minimal 1-path that is ②. Note that there is no satisfiable 1-path in this $\mathcal{O}$-proof DAG since ② has a non-empty B_1 component. Since there is no satisfiable 1-path, we now apply another left partial instantiation and search for a satisfiable 2-path. We already have inferred minimal 1-paths, thus it is not necessary to reconsider the whole $\mathcal{O}$-proof DAG, but only the unsolved nodes of the minimal 1-paths. In our example, we have a unique minimal 1-path, hence $\geq_0$ is $\{fact >_F p, fact >_F \times\}$ and we only have to prove satisfiability of the remaining $\mathcal{O}$-proof of B: $\Theta_{1,k}(fact(s(x))) >_{lex} \Theta_{1,k}(fact(p(s(x))))$.

Note that achieving partial instantiation with precedence, testing the compatibility of components A_0 and B_0, and comparing 1-paths with respect to $\subset$, can be automatized. Thus the deduction of minimal 1-paths can be achieved automatically. Finding a minimal 1-path allows us to separate the termination proof in two parts: a first part which can be automatically solved (we deduced a precedence), and a second part requiring human expertise. In our example, the proof requiring human expertise is satisfiability of B_1. To achieve this proof, we now apply left partial instantiation on B_1 and search for a satisfiable 2-path. For θ_1, the user may choose the function interpreting $fact$ as factorial, s as successor, p as predecessor and 0 as zero, and for $\geq_1$, he may choose $\geq_{\mathcal{N}}$: the greater or equal relation on natural numbers, as in [3]. The $\mathcal{O}$-proof becomes:



Then, validity of $(x+1)! >_{\mathcal{N}} x!$ has to be proved by the user. If we choose an interpretation where constants are interpreted as natural numbers, then $(x+1)! >_{\mathcal{N}} x!$ is valid. Thus, we get a satisfiable 2-path associated with tuple:
$\langle\{fact >_F p, fact \simeq_F fact, fact >_F \times\}, \{(x+1)! >_{\mathcal{N}} x!\}, \{\}\rangle$.

If we proceed similarly on the global $\mathcal{O}$-proof DAG for the whole TRS, we can infer automatically a precedence and a (still unique) minimal 1-path associated with a tuple: $C = \langle C_0, C_1 \rangle$ where

$$C_0 = \{fact >_F s, fact >_F p, fact \simeq_F fact, fact >_F \times, \times >_F +, \times \simeq_F \times,$$
$$+ >_F s, + \simeq_F +\} \text{ and}$$

$$C_1 = \{\Theta_{1,k}(fact(s(x))) >_{lex} \Theta_{1,k}(fact(p(s(x)))), \Theta_{1,k}(s(x) \times y) >_{lex} \Theta_{1,k}(x \times y),$$
$$\Theta_{1,k}(x + s(y)) >_{lex} \Theta_{1,k}(x + y)\}.$$

Thus the algorithm extracts the key part of the proof for the whole TRS: the three remaining constraints on $(\Theta_{1,k}, >_{lex})$ of C_1. If we proceed to a left partial instantiation on C_1, with the interpretation θ_1 previously defined completed with interpretation of $\times$ as multiplication and $+$ as addition, then the validity of: $(x+1)! >_{\mathcal{N}} x!$, $(x+1) \times y >_{\mathcal{N}} x \times y$ and $x + y + 1 >_{\mathcal{N}} x + y$ have to be proved by the user. If the tests are fulfilled by the user, then the algorithm ends on a satisfiable 2-path for the global $\mathcal{O}$-proof DAG, that is:

$$\langle\{fact >_F s, fact >_F p, fact \simeq_F fact, fact >_F \times, \times >_F +, \times \simeq_F \times, + >_F s,$$
$$+ \simeq_F +\}, \{(x+1)! >_{\mathcal{N}} x!, (x+1) \times y >_{\mathcal{N}} x \times y, x + y + 1 >_{\mathcal{N}} x + y\}, \{\}\rangle.$$

Note that in particular cases of *gpo*, like *lpo*, where the compatibility testing of every *gpo* termination function is automatic, the whole *gpo* solving process can be automatically achieved. For the *lpo* case, starting from a set of inequalities representing the rules of a TRS, the algorithm provides a precedence proving termination of the initial TRS (if such a precedence exists). Note that this approach is different from [2, 14], where the precedence is given, see Section 3. An implementation of the *lpo* case, providing a decision procedure for existence of a *lpo* for a given TRS has been developed in ECLiPSe[1]. See Appendix A for examples of execution. Let us cite another approach to find a precedence for syntactical orderings like *lpo* or *rpo* [5]. However, this method, unlike ours, is not goal directed since the search for a precedence is not guided by the inequalities to be proved.

Automatic satisfiability provers based on semantic termination functions could also be integrated to the second part of the solving. See [7, 17] for nice examples of what can be done with polynomial interpretations.

7 Conclusion and perspectives

In this paper, we proposed a termination proof algorithm for rewrite rule systems using *gpo* constraint solving on OCS graphs. An OCS graph is a shared term data structure, defined to represent rewrite rules. Its nodes are operators, and its edges represent the rewrite relation, the subterm relation, or the *gpo* ordering relation between terms. The last kind of edge is labeled by a validation proof of the ordering relation, called proof obligation ($\mathcal{O}$-proof).

A set of sound, complete inference rules working in polynomial time was then given to construct $\mathcal{O}$-proofs starting from OCS. These rules work by instantiating $\mathcal{O}$-proof variables while adding ordering edges to the OCS graph of the rewrite rules, till obtaining a normal form.

A method was then proposed to prove satisfiability of $\mathcal{O}$-proofs, by instantiating termination functions and by finding a path whose nodes are compatible. For some of these

[1]ECRC Common Logic Programming System

termination functions, compatibility can automatically be verified. For precedence orderings, for instance, it is enough to prove that the relation described by constraints between symbols is a partial ordering on the set of symbols. In addition, for instances of *gpo* where satisfiability of ordering constraints on all the termination functions can automatically be verified, the whole termination proof can be achieved in a fully automatic way. For other cases, the interest of our approach is that the process focuses user's effort to the key parts of the proof by automatically proving simple properties and extracting difficult ones.

Next prospects are the improvement of $\mathcal{O}$-proof satisfiability proofs. Satisfiability procedures could be integrated for more syntactic and semantic termination functions. We are also studying how to combine completion on SOUR Graphs [13] with automatic termination proofs, taking advantage of the similarity between the graph deduction process on SOCS and on SOUR.

Acknowledgments

We would like to thank Hélène Kirchner, Claude Kirchner, Nachum Dershowitz, Christopher Lynch, Polina Strogova and Christophe Ringeissen for comments on this paper.

References

[1] Adel Bouhoula and Michaël Rusinowitch. Implicit induction in conditional theories. *JAR*, 14(2):189–235, 1995.

[2] H. Comon. Solving inequations in term algebras. In *Proc. 5th LICS Symp., Philadelphia (Pa., USA)*, pages 62–69, June 1990.

[3] N. Dershowitz and C. Hoot. Natural termination. *TCS*, 142(2):179–207, May 1995.

[4] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 6, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990.

[5] R. Forgaard and D. Detlefs. An incremental algorithm for proving termination of term rewriting systems. In J.-P. Jouannaud, editor, *Proc. 1st RTA Conf., Dijon (France)*, pages 255–270. Springer-Verlag, 1985.

[6] T. Genet and I. Gnaedig. Solving GPO ordering constraints with shared term data structure. Technical Report 95-R-363, CRIN, 1995.

[7] J. Giesl. Generating polynomial orderings for termination proofs. In Jieh Hsiang, editor, *Proc. 6th RTA Conf., Kaiserslautern (Germany)*, volume 914 of *LNCS*. Springer-Verlag, 1995.

[8] P. Johann and Rolf Socher-Ambrosius. Solving simplification ordering constraints. In J.-P. Jouannaud, editor, *Proc. 1st CCL Conf., Munich (Germany)*, volume 845 of *LNCS*, pages 352–367. Springer-Verlag, 1994.

[9] S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path ordering. Unpublished manuscript, 1980.

[10] M. S. Krishnamoorthy and P. Narendran. A note on recursive path ordering. In *TCS*, volume 40, pages 323–328, 1985.

[11] D. S. Lankford. On proving term rewriting systems are noetherian. Technical report, Louisiana Tech. University, Mathematics Dept., Ruston LA, 1979.

[12] P. Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *JAR*, 6:39–49, 1990.

[13] Christopher Lynch and Polina Strogova. Sour graphs for efficient completion. Technical Report 95-R-343, CRIN, 1995.

[14] R. Nieuwenhuis. Simple lpo constraint solving methods. *IPL*, 47(2), 1993.

[15] R. Nieuwenhuis and A. Rubio. Theorem proving with ordering constrained clauses. In D. Kapur, editor, *Proc. 11th CADE Conf., Saratoga Springs (N.Y., USA)*, volume 607 of *LNCS*, pages 477–491. Springer-Verlag, 1992.

[16] D. Plaisted. Polynomial time termination and constraint satisfaction tests. In C. Kirchner, editor, *Proc. 5th RTA Conf., Montreal (Canada)*, volume 690 of *LNCS*, pages 405–420, Montreal (Québec, Canada), June 1993. Springer-Verlag.

[17] J. Steinbach. Generating polynomial orderings. *IPL*, 49:85–93, 1994.

[18] J. Steinbach and U. Kühler. Check your ordering – termination proofs and open problems. Technical report, Universität Kaiserslautern, 1993.

A Examples of execution

We developed a prototype achieving automatic termination proofs. This prototype generates $\mathcal{O}$-proofs from the TRS, using the $\mathcal{C}$-deduction process on SOCS, described in section 4. Then, it proves satisfiability of $\mathcal{O}$-proofs with an *lpo*-like ordering: satisfiability proofs are achieved with a precedence and a lexicographic extension. Then we search for the first satisfiable path in the global $\mathcal{O}$-proof DAG corresponding to a valid precedence. The output of the prototype is a precedence proving the termination of the TRS with *lpo* ordering. Note that the problem of finding a precedence for an *lpo* on a TRS is NP-complete [10]. The prototype presented here has been developed in ECLiPSe (ECRC Common Logic Programming System). Let us now give some traces of automatic termination proofs obtained by its execution.

Example 1:

```
var. x,y.

plus(zero, x) -> x.
plus(s(x), y) -> s(plus(x, y)).
fib(zero) -> zero.
fib(s(zero)) -> s(zero).
fib(s(s(x))) -> plus(fib(x), fib(s(x))).

------------
Comments: 4 function symbols



Solving is successful
precedence: [prec(fib, >, s), prec(plus, >, s), prec(fib, >, plus)]

Termination proof done in 1.36667 seconds, cpu time
```

Example 2 (example 2.22 of [18]: factorial function).

```
var. x,y.

fac(0) -> 1.
fac(s(x)) -> mult(s(x), fac(x)).
floop(0,y) -> y.
floop(s(x), y) -> floop(x, mult(s(x), y)).
```

```
mult(x, 0) -> 0.
mult(x, s(y)) -> plus(mult(x,y), x).
plus(x,0) -> x.
plus(x, s(y)) -> s(plus(x, y)).
1 -> s(0).

-----------------
Comments: 7 function symbols



Solving is successful
precedence: [prec(fac, >, 0), prec(fac, >, 1), prec(fac, >, plus),
prec(fac, >, s), prec(fac, >, mult), prec(floop, >, plus),
prec(floop, >, s), prec(floop, >, mult), prec(mult, >, s),
prec(mult, >, plus), prec(plus, >, s), prec(1, >, 0), prec(1, >, s)]

Termination proof done in 1.46667 seconds, cpu time
```

Example 3 (example 3.7 of [18]):

```
var. x, y, z.

(x + y) + z -> x + (y +z).
0 + 0 -> 0.
x + (-x) -> 0.
f(0, y, z) -> y.
g(x + y, y) -> f(x + y, x, y).
-(0) -> 0.
-(-(x)) -> x.
-(x + y) -> -(y) + -(x).
x + 0 -> x.
0 + y -> y.
-(x) + x -> 0.
-(x) + (x + y) -> y.
x +(-(x) + y) -> y.
g(x, y) -> f(x, x + -(y), y).

--------
Comments: 5 function symbols
```

```
Solving is successful
precedence: [prec(-, >, 0), prec(-, >, +), prec(g, >, 0), prec(+, >, 0),
prec(g, >, -), prec(g, >, +), prec(g, >, f)]

Termination proof done in 1.23333 seconds, cpu time
```

Example 5 (symbolic differentiation given in [7]):

```
var. alpha, beta.

dx(x) -> one.
dx(a) -> zero.
dx(plus(alpha, beta)) -> plus(dx(alpha), dx(beta)).
dx(times(alpha, beta)) -> plus(times(beta, dx(alpha)),
        times(alpha, dx(beta))).
dx(minus(alpha, beta)) -> minus(dx(alpha), dx(beta)).
dx(neg(alpha)) -> neg(dx(alpha)).
dx(div(alpha, beta)) -> minus(div(dx(alpha), beta),
        times(alpha, div(dx(beta), exp(beta,two)))).
dx(ln(alpha)) -> div(dx(alpha), alpha).
dx(exp(alpha, beta)) -> plus(times(beta, times(exp(alpha, minus(beta, one)),
        dx(alpha))), times(exp(alpha, beta), times(ln(alpha), dx(beta)))).

-----------------
Comments: 13 function symbols.



Solving is successful
precedence: [prec(dx, >, x), prec(one, =, x), prec(x, =, one),
prec(zero, =, a), prec(a, =, zero), prec(dx, >, neg), prec(dx, >, two),
prec(dx, >, div), prec(dx, >, ln), prec(dx, >, one), prec(dx, >, minus),
prec(dx, >, exp), prec(dx, >, times), prec(dx, >, plus)]

Termination proof done in 34.1833 seconds, cpu time
```

Note that our prototype can also prove termination of conditional TRS. For proving termination of conditional TRS ($l_i \rightarrow r_i \Leftarrow s_1^i \simeq t_1^i \wedge \ldots \wedge s_{n_i}^i \simeq t_{n_i}^i, i = 1 \ldots m$), it is enough to solve the *gpo* constraint $\bigwedge_{i=1}^{m} l_i > r_i \wedge l_i > s_1^i \wedge \ldots \wedge l_i > s_{n_i}^i \wedge l_i > t_1^i \wedge \ldots \wedge l_i > t_{n_i}^i$. Termination proofs of conditional TRS are particularly interesting for testing efficiency of our *gpo* ordering constraint solving since there are several *gpo* constraints to solve for each rewrite rule. Let us give a significant trace of execution on a conditional example. The following example is an oriented specification of the *Gilbreath card trick* borrowed from [1].

```
var. x,y,z, x1, x2, x3, y1, y2, y3.

neg(r) ->b if [].
neg(b) ->r if [].
paired(r,b) -> true if [].
paired(b,r) -> true if [].
paired(x,x) -> false if [].
append(null,y) -> y if [].
append(cons(x,y),z) -> cons(x,append(y,z)) if [].
rotate(null) -> null if [].
rotate(cons(x,y)) -> append(y,cons(x,null)) if [].
even(null) -> true if [].
even(cons(x,null)) -> false if[].
even(cons(x1,cons(x2,y))) -> even(y) if[].
opposite(null,y) -> false if [].
opposite(x,null) -> false if [].
opposite(cons(x1,y1),cons(x2,y2)) -> paired(x1,x2) if [].
pairedlist(null) -> true if [].
pairedlist(cons(x,null)) -> true if [].
pairedlist(cons(x1,cons(x2,x))) -> pairedlist(x)
        if [paired(x1,x2) = true].
pairedlist(cons(x1,cons(x2,x))) -> false if[paired(x1,x2)=false].
alter(null) -> true if [].
alter(cons(x1,null)) -> true if [].
alter(cons(x1,cons(x2,x))) -> alter(cons(x2,x))
        if [paired(x1,x2) = true].
alter(cons(x1,cons(x2,x))) -> false if [paired(x1,x2) = false].
shuffle(null,null,null) ->  true if [].
shuffle(x1,cons(x2,x3),null) -> false if [].
shuffle(cons(x1,x2),x3,null) -> false if [].
shuffle(null,null,cons(x1,x2)) -> false if [].
shuffle(cons(x1,y1),null,cons(x3,y3))-> false
        if [paired(x1,x3) = true].
shuffle(cons(x1,y1),null,cons(x3,y3)) -> shuffle(y1,null,y3)
        if[paired(x1,x3) = false].
shuffle(null,cons(x2,y2),cons(x3,y3)) -> false
```

```
        if [paired(x2,x3) = true].
shuffle(null,cons(x2,y2),cons(x3,y3)) -> shuffle(null,y2,y3)
        if [paired(x2,x3) = false].
shuffle(cons(x1,y1),cons(x2,y2),cons(x3,y3)) -> true
        if [paired(x1,x3) = false,
            shuffle(y1,cons(x2,y2),y3)=true].
shuffle(cons(x1,y1),cons(x2,y2),cons(x3,y3)) -> true
        if [paired(x2,x3) = false,
            shuffle(cons(x1,y1),y2,y3)=true].
shuffle(cons(x1,y1),cons(x2,y2),cons(x3,y3)) -> false
        if [paired(x1,x3) = true, paired(x2,x3) = true].
shuffle(cons(x1,y1),cons(x2,y2),cons(x3,y3)) -> false
        if [shuffle(y1,cons(x2,y2),y3)=false,
            shuffle(cons(x1,y1),y2,y3)=false].
shuffle(cons(x1,y1),cons(x2,y2),cons(x3,y3)) -> false
        if [paired(x1,x3) = true, shuffle(cons(x1,y1),y2,y3)=false].
shuffle(cons(x1,y1),cons(x2,y2),cons(x3,y3)) ->false
        if [paired(x2,x3) = true, shuffle(y1,cons(x2,y2),y3)=false].



Solving is successful

precedence: [prec(opposite, >, r), prec(paired,>, r), prec(rotate, >, r),
prec(even, >, r), prec(pairedlist, >, r), prec(alter, >, r),
prec(shuffle, >, r), prec(r, =, null), prec(null, =, r), prec(r, =, false),
prec(false, =, r),  prec(r, =, b), prec(b, =, r), prec(true, =, r),
prec(r, =, true), prec(shuffle, >, b), prec(alter, >, b),
prec(pairedlist, >, b), prec(even, >, b), prec(rotate, >, b),
prec(paired, >, b), prec(opposite, >, b), prec(b, =, null),
prec(null, =, b), prec(b, =, false), prec(false, =, b),
prec(true, =, b), prec(b, =, true), prec(opposite, >, false),
prec(opposite, >, true), prec(opposite, >, null),
prec(paired, >, null),  prec(paired, >, true), prec(paired, >, false),
prec(append, >, cons), prec(rotate, >, false), prec(rotate, >, true),
prec(rotate, >, null), prec(rotate, >, cons), prec(rotate, >, append),
prec(even, >, null), prec(even, >, true), prec(even, >, false),
prec(opposite, >, paired), prec(pairedlist, >, paired),
prec(pairedlist, >, null), prec(pairedlist, >, true),
prec(pairedlist, >, false), prec(alter, >, paired), prec(alter, >, null),
prec(alter, >, true), prec(alter, >, false), prec(true, =, false),
prec(false, =, true), prec(true, =, null), prec(null, =, true),
```

```
prec(shuffle, >, null), prec(false, =, null), prec(null, =, false),
prec(shuffle, >, true), prec(shuffle, >, paired), prec(shuffle, >, false)]

Termination proof done in 56.0333 seconds, cpu time
```

B Proofs

B.1 *gpo* is a quasi-ordering on $T(F,X)$ having the subterm property

All the proofs of the following lemmas on $T(F,X)$ are similar to the proofs of the respective lemmas on $T(F)$ that can be found in [3]. We thus choose here to simply point out and to prove all the additional properties.

Lemma 1 *(Symmetry). If $s \simeq_{gpo} t$ then $t \simeq_{gpo} s$.*

Proof The proof is similar to proof of Lemma 1 in [3] with the additional property we have to prove: $\Theta(s) \simeq_{lex} \Theta(t)$ implies $\Theta(t) \simeq_{lex} \Theta(s)$. From definition of $\Theta(s) \simeq_{lex} \Theta(t)$ we get that $\forall \sigma$ s.t. $s\sigma, t\sigma \in T(F)$, we have $\Theta(s\sigma) \simeq_{lex} \Theta(t\sigma)$. The relation $\simeq_i$ is supposed to be symmetric on ground terms, by definition of *gpo*. Note that, when $\simeq_i$ is the multiset extension of $\simeq_{gpo}$, symmetry of $\simeq_i$ comes from symmetry of *gpo* on ground terms. Thus, we obtain that $\simeq_{lex}$ is symmetric on ground terms. Hence $\forall \sigma$ s.t. $s\sigma, t\sigma \in T(F)$, we have $\Theta(t\sigma) \simeq_{lex} \Theta(s\sigma)$, which is the definition of $\Theta(t) \simeq_{lex} \Theta(s)$ on $T(F,X)$. Let $s, t, u \in T(F,X)$, we denote by $t|_p$ the subterm of t at position p, and by $u[s]$ (or $u[s]_p$) the term u having s as subterm at position p.

Lemma 2 *For the general path ordering, $s \geq_{gpo} t$ implies $s >_{gpo} t|_p$ for each proper subterm $t|_p$ of t.*

Proof The proof does not depend on the definition of Θ and $>_{lex}$ but only on the definition of *gpo*. Since we use the same definition of *gpo*, the proof is similar to the ground case proved in Lemma 2 of [3].

Lemma 3 *(Subterm). The general path ordering satisfies the strict subterm property: $\forall s,\ f(\ldots, s, \ldots) >_{gpo} s$.*

Proof The proof does not depend on the definition of Θ and $>_{lex}$ but only on the definition of *gpo*. Since we use the same definition of *gpo*, the proof is similar to the ground case proved in Lemma 3 of [3].

Lemma 4 *(Reflexivity). The general path ordering $\geq_{gpo}$ is reflexive.*

Proof The proof is similar to proof of Lemma 4 in [3] with the additional property we have to prove: $\Theta(s) \simeq_{lex} \Theta(s)$. The relation $\simeq_i$ is reflexive on ground terms. Therefore, $\simeq_{lex}$ is reflexive on ground terms. Note that, when $\simeq_i$ is the multiset extension of $\simeq_{gpo}$, reflexivity of $\simeq_i$ comes from reflexivity of *gpo* on ground terms. Hence $\forall \sigma$ s.t. $s\sigma \in T(F)$, we have $\Theta(s\sigma) \simeq_{lex} \Theta(s\sigma)$, and by Definition 2, $\Theta(s) \simeq_{lex} \Theta(s)$.

Lemma 5 *For the general path ordering, $s \geq_{gpo} t$ implies $u[s] >_{gpo} t$ for each nonempty context $u[.]$ of s.*

Proof The proof does not depend on the definition of Θ and $>_{lex}$ but only on the definition of *gpo*. Since we use the same definition of *gpo*, the proof is similar to the ground case proved in Lemma 5 of [3]. In the following, $Var(s)$ will denote the set of variables of a term s, and $Dom(\sigma)$ will denote the domain of the substitution σ. Let us now prove a lemma necessary to get transitivity of *gpo* on $T(F, X)$.

Lemma 6 *Let $s, t \in T(F, X)$. Then $s \geq_{gpo} t$ implies $Var(s) \supseteq Var(t)$.*

Proof Assume that we have $s \geq_{gpo} t$ and $Var(s) \not\supseteq Var(t)$. There exists at least a variable x such that $x \in Var(t)$ and $x \notin Var(s)$. Let $C[.]$ be the non-empty context such that $t = C[x]$. Let σ be a substitution such that $Dom(\sigma) = Var(s) \cup Var(t) \setminus \{x\}$, $s\sigma \in T(F)$, and $Var(t\sigma) = Var(C[x]\sigma) = \{x\}$. Let $\sigma' = \sigma \cup \{x \mapsto s\sigma\}$. Then $s\sigma', t\sigma' \in T(F)$, $s\sigma' = s\sigma$ and $t\sigma' = C\sigma'[s\sigma]$. Thanks to the subterm property of *gpo*, we get that $C\sigma'[s\sigma] >_{gpo} s\sigma$. Then, $t\sigma' >_{gpo} s\sigma'$ with $s\sigma', t\sigma' \in T(F)$ which is a contradiction with $s \geq_{gpo} t$ and Proposition 1.

Lemma 7 *(Transitivity) Let $s, t, u \in T(F, X)$.*

1. $s >_{gpo} t \geq_{gpo} u \implies s >_{gpo} u$;
2. $s \simeq_{gpo} t >_{gpo} u \implies s >_{gpo} u$;
3. $s \simeq_{gpo} t \simeq_{gpo} u \implies s \simeq_{gpo} u$.

Proof The proof is similar to proof of Lemma 6 in [3] with three additional properties we have to prove:

1. for $s >_{gpo} t \geq_{gpo} u \implies s >_{gpo} u$, the additional property to prove is $\Theta(s) >_{lex} \Theta(t)$ and $\Theta(t) \geq_{lex} \Theta(u)$ implies that $\Theta(s) >_{lex} \Theta(u)$. From $\Theta(s) >_{lex} \Theta(t)$ and $\Theta(t) \geq_{lex} \Theta(u)$ and Definition 2, we get:

$$\forall \sigma \text{ s.t. } s\sigma, t\sigma \in T(F),\ \Theta(s\sigma) >_{lex} \Theta(t\sigma), \text{ and}$$

$$(\forall \sigma' \text{ s.t. } t\sigma', u\sigma' \in T(F),\ \Theta(t\sigma') >_{lex} \Theta(u\sigma')) \text{ or}$$
$$(\forall \sigma' \text{ s.t. } t\sigma', u\sigma' \in T(F),\ \Theta(t\sigma') \simeq_{lex} \Theta(u\sigma'))\ .$$

   Since $s >_{gpo} t$ and $t \geq_{gpo} u$, thanks to Lemma 6, we get that $Var(s) \supseteq Var(t)$, $Var(t) \supseteq Var(u)$. By transitivity of $\subseteq$, we obtain $Var(s) \supseteq Var(u)$. Hence, $\forall \sigma$ s.t. $s\sigma \in T(F)$, we also have $t\sigma, u\sigma \in T(F)$, and we get:

$$\forall \sigma \text{ s.t. } s\sigma, t\sigma, u\sigma \in T(F),\ \Theta(s\sigma) >_{lex} \Theta(t\sigma) \text{ and } \Theta(t\sigma) >_{lex} \Theta(u\sigma), \text{ or}$$

$$\forall \sigma \text{ s.t. } s\sigma, t\sigma, u\sigma \in T(F),\ \Theta(s\sigma) >_{lex} \Theta(t\sigma) \text{ and } \Theta(t\sigma) \simeq_{lex} \Theta(u\sigma).$$

Thanks to the transitivity of $>_i$ and $\simeq_i$, we can conclude on transitivity of $>_{lex}$ and $\simeq_{lex}$ on ground terms. Note that, when $>_i$ (resp. $\simeq_i$) is the multiset extension of $>_{gpo}$ (resp. $\simeq_{gpo}$), transitivity of $>_i$ (resp. $\simeq_i$) comes from the transitivity of $>_{gpo}$ (resp. $\simeq_{gpo}$) on ground terms. From transitivity of $>_{lex}$ and $\simeq_{lex}$ we get:

$$\forall \sigma \text{ s.t. } s\sigma, u\sigma \in T(F),\ \Theta(s\sigma) >_{lex} \Theta(u\sigma).$$

2. for $s \simeq_{gpo} t >_{gpo} u \implies s >_{gpo} u$, the additional property to prove is $\Theta(s) \simeq_{lex} \Theta(t) >_{lex} \Theta(u)$ implies that $\Theta(s) >_{lex} \Theta(u)$. It can be proven like in the previous case.

3. for $s \simeq_{gpo} t \simeq_{gpo} u \implies s \simeq_{gpo} u$, the additional property to prove is $\Theta(s) \simeq_{lex} \Theta(t) \simeq_{lex} \Theta(u)$ implies that $\Theta(s) \simeq_{lex} \Theta(u)$. It can be proven like in the first case.

Lemma 8 *(Irreflexivity). For any s, $s \not>_{gpo} s$.*

Proof The proof does not depend on the definition of Θ and $>_{lex}$ but only on the definition of *gpo*. Since we use the same definition of *gpo*, the proof is similar to the ground case proved in Lemma 7 of [3].

Lemma 9 *If $s >_{gpo} t$, then $t \not\geq_{gpo} s$.*

Proof Assume that $t \geq_{gpo} s$. Then by transitivity, we get that $s >_{gpo} s$, contradicting Lemma 8.

Lemma 10 *$s \geq_{gpo} t \geq_{gpo} s$ if and only if $s \simeq_{gpo} t$.*

By previous lemmas, we get that $\geq_{gpo}$ is a quasi-ordering with subterm property.

B.2 Ground stability

Proposition 2 *() Let $s, t \in T(F, X)$ and $\Phi = (\Theta, \geq_{lex})$. If $s \succ^{\Phi}_{gpo} t$ (resp. $s \approx^{\Phi}_{gpo} t$) then for all substitution σ s.t. $s\sigma, t\sigma \in T(F)$, we have $s\sigma \succ^{\Phi}_{gpo} t\sigma$ (resp $s\sigma \approx^{\Phi}_{gpo} t\sigma$).*

Proof By induction on the height of terms on both sides of the inequality. If $s = f(s_1, \ldots, s_n) \succ^{\Phi}_{gpo} g(t_1, \ldots, t_m) = t$ then by definition of $>_{gpo}$ on $T(F, X)$, two cases hold:

1. we have $s_i \succsim^{\Phi}_{gpo} t$ for some subterm s_i of s. Then, by induction, we obtain that $\forall \sigma$ s.t. $s_i\sigma, t\sigma \in T(F)$, we have $s_i\sigma \succsim^{\Phi}_{gpo} t\sigma$. Since $\forall \sigma$, $s\sigma, t\sigma \in T(F)$ implies $s_i\sigma, t\sigma \in T(F)$, we obtain that $\forall \sigma$ s.t. $s\sigma, t\sigma \in T(F)$, we have $s_i\sigma \succsim^{\Phi}_{gpo} t\sigma$. Hence by definition of *gpo*, $s\sigma \succ^{\Phi}_{gpo} t\sigma$.

2. we have $s \succ^{\Phi}_{gpo} t_1, \ldots, s \succ^{\Phi}_{gpo} t_m$ and $\Theta(s) >_{lex} \Theta(t)$. Then, by induction, we obtain that $\forall \sigma_1, \ldots, \sigma_m$ s.t. $s\sigma_1, \ldots, s\sigma_m, t_1\sigma_1, \ldots, t_m\sigma_m \in T(F)$, we have $s\sigma_1 \succ^{\Phi}_{gpo} t_1\sigma_1, \ldots, s\sigma_m \succ^{\Phi}_{gpo} t_m\sigma_m$. Then, $\forall \sigma$ s.t. $s\sigma, t_1\sigma, \ldots, t_m\sigma \in T(F)$, we have $s\sigma \succ^{\Phi}_{gpo} t_1\sigma, \ldots, s\sigma \succ^{\Phi}_{gpo} t_m\sigma$. In a similar manner than in the previous case, $\forall \sigma$, $s\sigma, t\sigma \in T(F)$ implies $s\sigma, t_1\sigma, \ldots, t_m\sigma \in T(F)$. Thus $\forall \sigma$ s.t. $s\sigma, t\sigma \in T(F)$, we have $s\sigma \succ^{\Phi}_{gpo} t_1\sigma, \ldots, s\sigma \succ^{\Phi}_{gpo} t_m\sigma$. On an other hand, by definition of Θ on $T(F, X)$, we get from $\Theta(s) >_{lex} \Theta(t)$ that $\forall \sigma$ s.t. $s\sigma, t\sigma \in T(F)$, $\Theta(s\sigma) >_{lex} \Theta(t\sigma)$. Hence $\forall \sigma$ s.t. $s\sigma, t\sigma \in T(F)$, we have $s\sigma \succ^{\Phi}_{gpo} t\sigma$.

The proof is similar for the case of $s \simeq_{gpo} t$.

B.3 Soundness of deduction rules

Let us give three lemmas which are necessary for the proof of soundness.

Lemma 11 *Given $P, Q \in \mathcal{P}$, $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$, we have:*

$$P \trianglelefteq Q \text{ and } \Phi \models P \quad \Longrightarrow \quad \Phi \models Q.$$

Proof By induction on the size of Q. By definition of $P \trianglelefteq Q$, either

- $Q = P$. Then, since $\Phi \models P$, we have $\Phi \models Q$, or
- $Q = A \vee B$ and we have $P \trianglelefteq A$ or $P \trianglelefteq B$. By induction, we obtain $\Phi \models A$ or $\Phi \models B$. Hence, by definition of $\models$, we get $\Phi \models Q$.

Lemma 12 *Given $P, Q \in \mathcal{P}$ and σ a $\mathcal{P}$-substitution,*

$$P \trianglelefteq Q \quad \Longrightarrow \quad P\sigma \trianglelefteq Q\sigma.$$

Proof By induction on the size of Q. By definition of $P \trianglelefteq Q$, either

- $Q = P$. Then $Q\sigma = P\sigma$, hence $P\sigma \trianglelefteq Q\sigma$, or
- $Q = A \vee B$ and we have $P \trianglelefteq A$ or $P \trianglelefteq B$. Then $Q\sigma = A\sigma \vee B\sigma$ and by induction, we obtain that $P\sigma \trianglelefteq A\sigma$ or $P\sigma \trianglelefteq B\sigma$. Hence by definition of $\trianglelefteq$, $P\sigma \trianglelefteq Q\sigma$.

In the following, $s \overset{P}{-\!\diamond\!-} t$ will denote any ordering edge: $s \overset{P}{\longrightarrow\!\!>} t$, $s \overset{P}{-\!\!<} t$ or $s \overset{P}{\sim\!\sim} t$. We will write *correct* as a shorthand for *correct w.r.t. gpo*.

Definition 19 *Let $(G \mid\mid \sigma)$ be a SOCS. Let $s \overset{P}{-\!\diamond\!-} t$ and $s' \overset{P'}{-\!\diamond\!-} t'$ be ordering edges of G.*

- *$s' \overset{P'}{-\!\diamond\!-} t'$ is* above *$s \overset{P}{-\!\diamond\!-} t$ if s and t are subterms (possibly not strict) of s' and t' respectively,*
- *$s' \overset{P'}{-\!\diamond\!-} t'$ is* below *$s \overset{P}{-\!\diamond\!-} t$ if it is not above $s \overset{P}{-\!\diamond\!-} t$.*

Lemma 13 *Let S be an initial SOCS, and $S \vdash_{\mathcal{C}}^{*} S'$ where $S' = (G \mid\mid \sigma \cup \{P \mapsto \alpha\})$. Let $s \overset{P}{\multimap} t$ be an ordering edge of G where $P \in \mathcal{X}_{\mathcal{P}}$.*

1. *If $s' \overset{P'}{\multimap} t'$ is an edge of G below $s \overset{P}{\multimap} t$ then $P'(\sigma \cup \{P \mapsto \alpha\}) = P'\sigma$, and*
2. *$\alpha(\sigma \cup \{P \mapsto \alpha\}) = \alpha\sigma$.*

Proof When an edge $s \overset{P}{\multimap} t$ is constructed in G, P is necessarily a new variable. Then, P may occur in $\mathcal{O}$-proofs of other edges if the new edge $s \overset{P}{\multimap} t$ is used as a precondition for other $\mathcal{C}$-deductions. In that case, new edges added (whose $\mathcal{O}$-proof contains P) are necessarily edges *above* $s \overset{P}{\multimap} t$ because construction of edges by $\mathcal{C}$-deduction rules is performed bottom-up. Thus, the variable P cannot appear in α nor in $\mathcal{O}$-proofs of edges below $s \overset{P}{\multimap} t$. Hence

1. for any edge $s' \overset{P'}{\multimap} t'$ below $s \overset{P}{\multimap} t$, we have $P'(\sigma \cup \{P \mapsto \alpha\}) = P'\sigma$, and
2. $\alpha(\sigma \cup \{P \mapsto \alpha\}) = \alpha\sigma$.

Let us recall the soundness theorem of the deduction rules.

Theorem 3. *For any initial SOCS S, if $S \vdash_{\mathcal{C}}^{*} S'$ then S' is correct w.r.t. gpo.*
Proof In the following, $S = (G \mid\mid \rho)$, $S' = (G' \mid\mid \varphi)$; $\mathcal{F}, \mathcal{U}, \mathcal{G}, \mathcal{T}_1, \ldots, \mathcal{T}_m$ and $\mathcal{V}$ are nodes of G (hence of G') s.t. $Term(\mathcal{F}) = s$, $Term(\mathcal{G}) = t$, $Term(\mathcal{U}) = u$, $Term(\mathcal{V}) = v$, $Term(\mathcal{T}_1) = t_1, \ldots, Term(\mathcal{T}_m) = t_m$, $\mathcal{F}$ is labeled by f and $\mathcal{G}$ is labeled by g. We first prove that the application of any deduction rule of $\mathcal{C}$ on a correct SOCS S leads to another correct SOCS S'. By case on the deduction rules which can be applied:

SUBTERM Property This rule applies if we have a subterm edge between $\mathcal{F}$ and $\mathcal{U}$. In that case, S' is obtained by the addition of an edge $\mathcal{F} \overset{\top}{\twoheadrightarrow} \mathcal{U}$ to G. Since every edge of G is already correct, all we need to prove is that this new edge is correct. This is trivially true since $Term(\mathcal{U}) = u$, $Term(\mathcal{F}) = f(\ldots, u, \ldots)$ and $\forall\Phi\ f(\ldots, u, \ldots) \succ_{gpo}^{\Phi} u$, by subterm property of *gpo*.

SUBTERM First Like in the case of **SUBTERM Property**, we have $s = f(\ldots, u, \ldots)$, and G' differs from G only by a new inequality edge. The extension of the $\mathcal{P}$-substitution ρ into $\varphi = \rho \cup \{P \mapsto P'\}$ does not concern preexistent edges of G, since P is a new variable. Then, we only need to prove that the new edge is correct. We already know that there is an edge $\mathcal{U} \overset{P'}{\twoheadrightarrow} \mathcal{V}$ or $\mathcal{U} \overset{P'}{\rightsquigarrow} \mathcal{V}$ in G. Since S is correct, $\mathcal{U} \overset{P'}{\twoheadrightarrow} \mathcal{V}$ (or $\mathcal{U} \overset{P'}{\rightsquigarrow} \mathcal{V}$) is correct, which implies that $\forall\Phi$ s.t. $\Phi \models P'\rho$, we have $u \succsim_{gpo}^{\Phi} v$. The new edge is labeled by P. The new $\mathcal{P}$-substitution is $\varphi = \rho \cup \{P \mapsto P'\}$. Hence, $P\varphi = P'\varphi$. Since P is a new variable, P does not appear in P' nor in ρ hence $P'\varphi = P'\rho$. Then, by transitivity of $=$, $P\varphi = P'\rho$, hence $\forall\Phi$ s.t. $\Phi \models P\varphi$ we have $u \succsim_{gpo}^{\Phi} v$. Moreover, $s = f(\ldots, u, \ldots)$ and $\forall\Phi$ we have $s \succ_{gpo}^{\Phi} u$ by subterm property of *gpo*. Then, by transitivity of *gpo* we obtain that $\forall\Phi$ s.t. $\Phi \models P\varphi$ we have $s \succ_{gpo}^{\Phi} v$.

SUBTERM Extension We have $s = f(\ldots, u, \ldots)$. The only transformation of S processed by the rule is done on the substitution of P. We achieve the proof of correctness in three steps: we first prove that after the transformation of the substitution, edges below $s \xrightarrow{P} v$ remain correct. Second, we prove that edge $s \xrightarrow{P} v$ itself remains correct, and third that edges above $s \xrightarrow{P} v$ are also correct.

We first prove that any edge below $s \xrightarrow{P} v$ remains correct. Let $k \xrightarrow{Q} l$ be an edge of $(G||\rho)$ below $s \xrightarrow{P} v$. The edge $(k \xrightarrow{Q} l||\rho)$ is correct since $(G||\rho)$ is correct. Let $\rho = \sigma \cup \{P \mapsto \alpha\}$ and $\varphi = \sigma \cup \{P \mapsto \alpha \vee P'\}$ the $\mathcal{P}$-substitution obtained after the application of the rule. Since $k \xrightarrow{Q} l$ is correct, we have: $\forall \Phi$ such that $\Phi \models Q\rho$ we have $k \succ_{gpo}^{\Phi} l$. By Lemma 13, we obtain that $Q\varphi = Q\sigma = Q\rho$. Thus, $\forall \Phi$ such that $\Phi \models Q\varphi$, we have $\Phi \models Q\rho$, hence $k \succ_{gpo}^{\Phi} l$. The proof is similar for an edge $k \stackrel{Q}{\leadsto} l$.

Now we have to prove that the edge $s \xrightarrow{P} v$ is correct. Let $\rho = \sigma \cup \{P \mapsto \alpha\}$ and $\varphi = \sigma \cup \{P \mapsto \alpha \vee P'\}$ the $\mathcal{P}$-substitution obtained after the application of the rule. We have $P\varphi = \alpha\varphi \vee P'\varphi$. By definition of $\models$, $\Phi \models P\varphi$ if $\Phi \models \alpha\varphi$ or $\Phi \models P'\varphi$. Since P does not appear in $\alpha\sigma$ nor in $P'\sigma$, we get that $\alpha\varphi = \alpha\sigma$ and $P'\varphi = P'\sigma$. Hence $\Phi \models P\varphi$ if $\Phi \models \alpha\sigma$ or $\Phi \models P'\sigma$. Now,

- whether $\Phi \models \alpha\sigma$; then, by hypothesis, the SOCS S is correct, hence $\forall \Phi$ such that $\Phi \models \alpha\rho$, we have $s \succ_{gpo}^{\Phi} v$. By Lemma 13, we get that $\alpha\rho = \alpha\sigma$. Hence $\forall \Phi$ such that $\phi \models \alpha\sigma$, we have $s \succ_{gpo}^{\Phi} v$.
- or $\Phi \models P'\sigma$; then by Lemma 13 we obtain that $P'\rho = P'\sigma$. Since, by hypothesis, the SOCS S is correct, we get that $\forall \Phi$ such that $\Phi \models P'\rho$, we have $u \succsim_{gpo}^{\Phi} v$, that induces $s \succ_{gpo}^{\Phi} v$. Since $P'\rho = P'\sigma$ we obtain that: $\forall \Phi$ such that $\Phi \models P'\sigma$ we have $s \succ_{gpo}^{\Phi} v$.

We still have to prove that edges above $s \xrightarrow{P} v$ remain correct in spite of the transformation of ρ into φ. Indeed, in edges above $s \xrightarrow{P} v$, we may encounter ordering edges whose $\mathcal{O}$-proofs are including P and we map P to a different $\mathcal{O}$-proof by the $\mathcal{P}$-substitution φ. The correction of edges above $s \xrightarrow{P} t$ lies on the fact that $(s \xrightarrow{P} v \mid\mid \rho)$ is correct. The edge $(s \xrightarrow{P} v \mid\mid \varphi)$ is still correct, which implies correctness of edges above.

SUBTERM Trivial Like in the case of **SUBTERM First**, we have $s = f(\ldots, u, \ldots)$, and G' differs from G only by a new inequality edge. The application of the rule has not extended the $\mathcal{P}$-substitution: $\rho = \varphi = \sigma$. Hence, every edge of G remains correct in G'. We only have to prove the correction of the new edge. By assumption, $u \xrightarrow{\top} v$ is correct, hence $\forall \Phi$ s.t. $\Phi \models \top$ we have $u \succ_{gpo}^{\Phi} v$. By subterm property, we obtain that $\forall \Phi$ s.t. $\Phi \models \top$ we have $s \succ_{gpo}^{\Phi} v$.

SUBTERM Simplification There are two transformations: P is replaced by the trivial $\mathcal{O}$-proof $\top$ and P is mapped to $\top$ in the $\mathcal{P}$-substitution. Like in the proof for **SUBTERM Extension**, every edge below $s \xrightarrow{\top} v$ remains correct in S', thanks to Lemma 13. The proof that edge $s \xrightarrow{\top} v$ is correct is akin to the proof **SUBTERM First**, provided that $\mathcal{O}$-proof are trivial ones. The case of edges above $s \xrightarrow{\top} v$ can be treated as for **SUBTERM Extension**: i.e. edges above $s \xrightarrow{\top} v$ are correct provided that $s \xrightarrow{\top} v$ remains correct.

THETA $>$ Assume that $Term(\mathcal{F}) = s$, $Term(\mathcal{G}) = t = g(t_1, \ldots, t_m)$, $Term(\mathcal{T}_1) = t_1, \ldots,$ and $Term(\mathcal{T}_m) = t_m$. Recall that, before the application of the rule, the OCS graph is G and the $\mathcal{P}$-substitution is $\rho = \sigma$, and after the application the graph becomes G' and the $\mathcal{P}$-substitution becomes $\varphi = \sigma \cup \{P \mapsto P_1 \wedge \ldots \wedge P_m \wedge \Theta(s) >_{lex} \Theta(t)\}$. Every edge of G is still in G'. For any edge $k \overset{Q}{\leftrightarrow} l$ of G, since P is a new variable, we have $Q\varphi = Q\sigma$. Since $(G||\rho)$ is correct, $(k \overset{Q}{\leftrightarrow} l||\rho)$ is correct, and since $Q\varphi = Q\sigma$, we obtain that $(k \overset{Q}{\leftrightarrow} l||\varphi)$ is correct. Thus, every edge of G remains correct in G', and we only have to prove the correctness of the new edge itself. Since edges $(\mathcal{F} \xrightarrow{P_1} \mathcal{T}_1||\varphi), \ldots, (\mathcal{F} \xrightarrow{P_m} \mathcal{T}_m||\varphi)$ are correct, we get that:

- $\forall \Phi_1$ such that $\Phi_1 \models P_1\varphi$ we have $s \succ^{\Phi_1}_{gpo} t_1$,
- ...
- $\forall \Phi_m$ such that $\Phi_m \models P_m\varphi$ we have $s \succ^{\Phi_m}_{gpo} t_m$.

As a result, $\forall \Phi$ such that $\Phi \models P_1\varphi$ and ...and $\Phi \models P_m\varphi$, we have $s \succ^{\Phi}_{gpo} t_1, \ldots, s \succ^{\Phi}_{gpo} t_m$. Since $\varphi = \sigma \cup \{P \mapsto P_1 \wedge \ldots \wedge P_m \wedge \Theta(s) >_{lex} \Theta(t)\}$, we have $P\varphi = P_1\varphi \wedge \ldots P_m\varphi \wedge \Theta(s) >_{lex} \Theta(t)$. By definition of $\models$, we have $\forall \Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ s.t. $\Phi \models \Theta(s) >_{lex} \Theta(t)$ then $\mathcal{T}_{0,k}(s) >_{lex} \mathcal{T}_{0,k}(t)$. Hence, $\forall \Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$ s.t. $\Phi \models P\varphi$, we have $s \succ^{\Phi}_{gpo} t_1, \ldots, s \succ^{\Phi}_{gpo} t_m$ and $\mathcal{T}_{0,k}(s) >_{lex} \mathcal{T}_{0,k}(t)$ and finally, by definition of *gpo*, $\forall \Phi$ such that $\Phi \models P\varphi$ we have $s \succ^{\Phi}_{gpo} t$.

THETA $>$ **Extension** The proof for the extension case is akin to the **THETA** $>$ case and edges above are treated like in the **SUBTERM Extension** case.

THETA $\sim$ The proof is similar to the **THETA** $>$ case.

Thus, we proved that if S is a correct SOCS and $S \vdash_{\mathcal{C}} S'$, then S' is correct. By induction on the number of steps of $\vdash_{\mathcal{C}}$ applied, we get that if S is correct and $S \vdash^*_{\mathcal{C}} S''$ then S'' is also correct. Moreover, an initial SOCS is always correct w.r.t. *gpo*. As a result, any SOCS generated by deduction rules starting from an initial SOCS is also correct w.r.t. *gpo*.

B.4 Completeness of deduction rules

Let us recall the theorem of completeness.

Theorem 4. *Let $(G||\sigma)$ be a SOCS in $\mathcal{C}$-normal form, where $G = (V, E)$. For all nodes $\mathcal{F}, \mathcal{G} \in V$, s.t. $Term(\mathcal{F}) = s$ and $Term(\mathcal{G}) = t$, where $s, t \in T(F, X)$:*

$$\forall \Phi = (\mathcal{T}_{0,k}, \succsim_{lex}) \text{ s.t. } s \succ^{\Phi}_{gpo} t, \text{ there exists an edge } s \xrightarrow{P} t \text{ in } G \text{ s.t. } \Phi \models P\sigma$$

$$\forall \Phi = (\mathcal{T}_{0,k}, \succsim_{lex}) \text{ s.t. } s \approx^{\Phi}_{gpo} t, \text{ there exists an edge } s \overset{P}{\rightsquigarrow} t \text{ in } G \text{ s.t. } \Phi \models P\sigma.$$

Proof We first prove that if $s \succ^{\Phi}_{gpo} t$ (resp. $s \approx^{\Phi}_{gpo} t$) then there must be an edge $s \xrightarrow{P} t$ (resp. $s \overset{P}{\rightsquigarrow} t$). Assuming that $s \succ^{\Phi}_{gpo} t$ (resp. $s \approx^{\Phi}_{gpo} t$) and that the process had stopped without creating any edge $s \xrightarrow{P} t$ (resp. $s \overset{P}{\rightsquigarrow} t$) leads to a contradiction. We give a proof of this fact by induction on the size of terms on both sides of the inequality (or the equality).

Let us assume that $s = f(s_1, \ldots, s_n) \succ^{\Phi}_{gpo} g(t_1, \ldots, t_m) = t$ and that there is no edge $s \xrightarrow{P} t$. Since $s \succ^{\Phi}_{gpo} t$, by definition of *gpo*, either:

- $\exists 1 \geq i \geq n$ such that $s_i \succsim^{\Phi}_{gpo} t$. Applying induction, we obtain that there is an edge $s_i \xrightarrow{P_i} t$ or $s_i \overset{P_i}{\rightsquigarrow} t$. Since there is still no edge between s and t, deduction rule **SUBTERM First** or **SUBTERM Trivial** can be applied. This is a contradiction with the assumption that the SOCS S was in $\mathcal{C}$-normal form.
- $s \succ^{\Phi}_{gpo} t_1, \ldots, t_m$ and $\Theta(s) >_{lex} \Theta(t)$. Applying induction on the m inequalities, we obtain that there are m edges $s \xrightarrow{P_i} t_i$. Since there is still no edge between s and t, deduction rule **THETA** $>$ can be applied, which contradicts the assumption that the SOCS S is in $\mathcal{C}$-normal form.

Let us assume that $s = f(s_1, \ldots, s_n) \approx^{\Phi}_{gpo} g(t_1, \ldots, t_m) = t$ and that there is no edge $s \overset{P}{\rightsquigarrow} t$. Since $s \approx^{\Phi}_{gpo} t$, by definition of *gpo*: $s \succ^{\Phi}_{gpo} t_1, \ldots, t_m$, $t \succ^{\Phi}_{gpo} s_1, \ldots, s_m$ and $\Theta(s) \sim \Theta(t)$. By induction, we obtain that there are edges $s \xrightarrow{P_i} t_i$ $(i = 1 \ldots m)$ and $t \xrightarrow{P'_j} s_j$ $(j = 1 \ldots n)$. Then rule **THETA** $\sim$ can be applied. This is a contradiction with the assumption that S was in $\mathcal{C}$-normal form.

Now, we prove that for all Φ such that $s \succ^{\Phi}_{gpo} t$ (resp. $s \approx^{\Phi}_{gpo} t$), if the SOCS $S = (G \mid\mid \sigma)$ is in $\mathcal{C}$-normal form and G has an edge $s \xrightarrow{P} t$ (resp. $s \overset{P}{\rightsquigarrow} t$), then $\Phi \models P\sigma$. If $P = \top$, then by definition of $\models$, we trivially get that $\Phi \models P$. For the case where $P \neq \top$, we proceed by induction on the size of terms on both sides of the inequality (or the equality). Let us assume that $s = f(s_1, \ldots, s_n) \succ^{\Phi}_{gpo} g(t_1, \ldots, t_m) = t$. Let $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$. By definition of *gpo*, whether:

1. $\exists 1 \geq i \geq n$ such that $s_i \succsim^{\Phi}_{gpo} t$. Thanks to the first part of the proof, we know that there is an edge $s \xrightarrow{P} t$ and there is an edge $s_i \xrightarrow{P_i} t$ or an edge $s_i \overset{P_i}{\rightsquigarrow} t$. If $P_i = \top$ then, since $P \neq \top$, we can apply rule **SUBTERM Simplification**, which contradicts the fact that S is in $\mathcal{C}$-normal form. Hence, $P_i \neq \top$. Applying induction on $s_i \succsim^{\Phi}_{gpo} t$, we obtain that $\Phi \models P_i\sigma$. Provided that $\{P \mapsto \alpha\} \in \sigma$, if $P_i \not\trianglelefteq \alpha$, then we

can apply **SUBTERM Extension** which contradicts the fact that S is in $\mathcal{C}$-normal form. Thus $P_i \trianglelefteq \alpha$, and by Lemma 12, $P_i\sigma \trianglelefteq \alpha\sigma$. By Lemma 11, we get $\Phi \models \alpha\sigma$. Since $\{P \mapsto \alpha\} \in \sigma$, $P\sigma = \alpha\sigma$, and we finally get that $\Phi \models P\sigma$.

2. $s \succ_{gpo}^{\Phi} t_1, \ldots, t_m$ and $\mathcal{T}_{0,k}(s) >_{lex} \mathcal{T}_{0,k}(t)$. Thanks to the first part of the proof, we know that there are edges $s \xrightarrow{P_1} t_1, \ldots, s \xrightarrow{P_m} t_m$ and $s \xrightarrow{P} t$. Applying induction on the m inequalities, we obtain that $\Phi \models P_1\sigma, \ldots, \Phi \models P_m\sigma$. Since $\mathcal{T}_{0,k}(s) >_{lex} \mathcal{T}_{0,k}(t)$ and $\Phi = (\mathcal{T}_{0,k}, \succsim_{lex})$, by definition of $\models$, we obtain that $\Phi \models \Theta(s) >_{lex} \Theta(t)$. Like in Case 1, provided that $\{P \mapsto \alpha\} \in \sigma$ and assuming that the $\mathcal{O}$-proof containing $P_1, \ldots, P_m$ and $\Theta(s) >_{lex} \Theta(t)$ is not visible in α, we get a contradiction, since we can apply rule **THETA > Extension**. Thus the $\mathcal{O}$-proof containing $P_1, \ldots, P_m$ and $\Theta(s) >_{lex} \Theta(t)$ is visible in α. Finally, like in Case 1 and thanks to Lemma 12 and Lemma 11, we obtain $\Phi \models P\sigma$.

For the case $s = f(s_1, \ldots, s_n) \approx_{gpo}^{\Phi} g(t_1, \ldots, t_m) = t$, the proof is simpler: $s \approx_{gpo}^{\Phi} t$ implies that $s \succ_{gpo}^{\Phi} t_1, \ldots, t_m$ and $t \succ_{gpo}^{\Phi} s_1, \ldots, s_n$ and $\Theta(s) \simeq_{lex} \Theta(t)$. In a similar manner than in previous cases, there are edges $s \xrightarrow{P_1} t_1, \ldots, s \xrightarrow{P_m} t_m$ and $t \xrightarrow{P'_1} s_1, \ldots, t \xrightarrow{P'_n} s_n$ and $s \xrightarrow{P} t$. Applying induction on the $m+n$ inequalities, we obtain that $\Phi \models P_1, \ldots, P_m$ and $\Phi \models P'_1, \ldots, P'_n$. By hypothesis, we have $s \approx_{gpo}^{\Phi} t$, hence by definition of *gpo*: $\mathcal{T}_{0,k}(s) \simeq_{lex} \mathcal{T}_{0,k}(t)$. Thus, by definition of $\models$, $\Phi \models \Theta(s) \simeq_{lex} \Theta(t)$. Finally, $\Phi \models P\sigma$.

B.5 Complexity of deduction rules

We give an upper bound for the number of possible deduction (or applied rules) on a single SOCS. Let N be the maximal number of nodes in the OCS Graph of the SOCS, and M be the maximal arity of function symbols of F. The number of distinct pairs of nodes is $N \times (N-1)$ (provided that we have to consider separately symmetrical pairs of nodes: $(\mathcal{F}, \mathcal{G})$ and $(\mathcal{G}, \mathcal{F})$).

We first give an upper bound for the number of applications of each rule to any pair of nodes. Let $\mathcal{F}$ (labeled by f) and $\mathcal{G}$ (labeled by g) be two nodes of an OCS graph G s.t. $Term(\mathcal{F}) = s = f(s_1, \ldots, s_n)$ and $Term(\mathcal{G}) = t = g(t_1, \ldots, t_m)$. By case on the rule applied: rules **SUBTERM Property**, **SUBTERM First**, **SUBTERM Trivial**, **THETA >** and **THETA ∼** can be applied only if there is no existing edge between $\mathcal{F}$ and $\mathcal{G}$. When applied, they generate an edge between $\mathcal{F}$ and $\mathcal{G}$. Since no rule removes edges, those rules can only be applied once for each pair of vertices.

Rules **SUBTERM Extension**, **SUBTERM Simplification** and **THETA > Extension** may apply even if there is already an edge between $\mathcal{F}$ and $\mathcal{G}$. We now prove that the number of application of these rules is bounded.

Remark first that in a SOCS $S = (G||\rho)$ with $\rho = \sigma \cup \{P \mapsto \alpha\}$ such that there exists an $\mathcal{O}$-proof β and $\beta \trianglelefteq \alpha$, then whatever the deduction rule applied to the SOCS S, either

- P is replaced by $\top$, or

- the SOCS becomes $S' = (G'||\rho')$ where $\rho' = \sigma' \cup \{P \mapsto \alpha'\})$ and $\beta \trianglelefteq \alpha'$.

The precondition of **THETA > Extension** verifies that $(P_1 \wedge \ldots \wedge P_m \wedge \Theta(s) >_{lex} \Theta(t)) \not\trianglelefteq \alpha$. After the application of this rule, we have $P_1 \wedge \ldots \wedge P_m \wedge \Theta(s) >_{lex} \Theta(t) \trianglelefteq \alpha'$ where α' is the new $\mathcal{P}$-substitution of the SOCS. Hence, this rule can only be applied once for any pair of distinct nodes $\mathcal{F}$ and $\mathcal{G}$.

The precondition of **SUBTERM Simplification** is similar: since P has to be different from $\top$, this rule can only be applied once.

The precondition $P' \not\trianglelefteq \alpha$ of **SUBTERM Extension** prevents it to add a specific $\mathcal{O}$-proof P' to P twice. As a result, we just have to prove that the number of possible added $\mathcal{O}$-proofs by the rule **SUBTERM Extension** is bounded. Since the maximal arity of symbols of F is M, one can only add M $\mathcal{O}$-proofs coming from edges $s_i \xrightarrow{P_i} t$ and M $\mathcal{O}$-proofs coming from edges $s_i \overset{P_i'}{\rightsquigarrow} t$. So the number of added $\mathcal{O}$-proofs is bounded by $2M$.

We can sum up the number of added $\mathcal{O}$-proofs by case on the kind of edge added to the OCS Graph:

- for edges $s \rightarrow t$, there are M possible $\mathcal{O}$-proofs constructed by **SUBTERM** rules thanks to $s_i \rightarrow t$ edges. There are also M possible $\mathcal{O}$-proofs constructed thanks to $s_i \rightsquigarrow t$ edges. Finally, there is also a possible $\mathcal{O}$-proof constructed by a **THETA** rule. We then construct a maximum of $2M + 1$ $\mathcal{O}$-proofs.
- for the symmetrical case $t \rightarrow s$, the bound is strictly the same: $2M + 1$.
- for $s \rightsquigarrow t$, one can construct a unique $\mathcal{O}$-proof thanks to the **THETA** $\sim$ rule.

For any pair of vertices, the maximal number of applied rules is $4M + 3$. As a result, for the complete OCS Graph, the upper bound for the number of deductions is:

$$N \times (N - 1) \times (4M + 3)$$

Since any construction of $\mathcal{O}$-proof is linear in space (for **THETA** $>$, **THETA** $>$ **Extension**, **THETA** $\sim$, linear on M) or constant (for all the other rules), global complexity *in space* is polynomial on M and N: $O(N^2M^2)$.

We now investigate time complexity. First, an important point is complexity of testing whether a rule may apply or not. The rule **THETA** $\sim$ has the heaviest testing cost: trying to apply it consists in verifying that there are edges between the top symbol f of the term s and every subterms $t_1, \ldots, t_m$ of the term t, and vice versa with g the top symbol of the term t and subterms $s_1, \ldots, s_n$ of the term s. If we assume that the graph is stored in an array, testing if an edge exists can be done in constant time, thus testing if the rule **THETA** $\sim$ applies is linear on M. The application of the rule consists in adding a new edge and/or modifying or adding an $\mathcal{O}$-proof. The rule **THETA** $\sim$ has the heaviest application cost: applying rule **THETA** $\sim$ consists in adding a new edge labeled by a new $\mathcal{O}$-proof variable and in constructing a new $\mathcal{O}$-proof in the $\mathcal{P}$-substitution which is a conjunction of variables $P_1, \ldots, P_m$ and $P_1', \ldots, P_n'$. Constructing the edge in a graph stored in an array can be achieved in constant time and constructing the conjunction of at most $2M$ variables is linear in M.

As a result, for any couple of vertices, testing if a rule can be applied and applying it (i.e constructing the $\mathcal{O}$-proof) can be done in linear time: $O(M)$. The complexity of a sequential algorithm applying each deduction rule once to any pair of nodes is $O(N^2M)$ (since there is a constant number of deduction rules). In the worst case, the process described above tries to apply every rule on any pair of nodes and applies only *one* rule. We already showed that at most $N \times (N-1) \times (4M+3)$ rules are applied. Thus, the global complexity of the $\mathcal{C}$-deduction process in the worst case is: $O(N^4M^2)$.

Note that the last result is an upper bound on a naive sequential algorithm. It is easy to get better results in practice in using specific strategies.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399