Preliminary version. Final version in Theoretical Computer Science (Elsevier), vol. 165, No. 1, pp. 97–131, 1996

A Compositional Semantic Basis for the Analysis of Equational Horn Programs*

María Alpuente [§] Moreno Falaschi [¶] Germán Vidal[§]

Abstract

We introduce a compositional characterization of the operational semantics of equational Horn programs. Then we show that this semantics and the standard operational semantics based on (basic) narrowing coincide. We define an abstract narrower mimicking this semantics, and show how it can be used as a basis for efficient AND-compositional program analysis. As an application of our framework, we show a compositional analysis to detect the unsatisfiability of an equation set with respect to a given equational theory. We also show that our method allows us to perform computations and analysis incrementally in a Constraint Equational setting and that the test of satisfiability in this setting can be done in parallel.

Keywords: Semantic analysis, compositionality, equational logic programming, conditional term rewriting systems.

1 Introduction

Compositionality is a desirable property which has been recognised as fundamental in the semantics of programming languages [38, 40]. Compositionality has to do with a (syntactic) composition operator $\diamond$, and holds when the meaning (semantics) $\mathcal{S}(\mathbf{C}_1 \diamond \mathbf{C}_2)$ of a compound construct is defined by composing the meanings of the constituents $\mathcal{S}(\mathbf{C}_1)$ and $\mathcal{S}(\mathbf{C}_2)$, i.e. for a suitable function $\mathbf{f}_\diamond$, $\mathcal{S}(\mathbf{C}_1 \diamond \mathbf{C}_2) = \mathbf{f}_\diamond(\mathcal{S}(\mathbf{C}_1), \mathcal{S}(\mathbf{C}_2))$. In the case of logic programs [13], the properties of two most important operators have been widely investigated, namely AND-composition (of atoms in a goal or in a clause body) and OR-composition, i.e. composition of (sets of) clauses. In the context of equational logic programming [20, 24], compositionality of the semantics has not been much investigated yet. In this paper, we address the problem of solving equations in equationally defined theories: we want to define compositionally the meaning of the union of $\mathcal{E}$-unification problems. Let a theory $\mathcal{E}$ be fixed and consider two finite equation sets Γ_1, Γ_2. We want to define the meaning of $\Gamma_1 \cup \Gamma_2$ (with respect to $\mathcal{E}$) in terms of the meanings of Γ_1 and Γ_2. Throughout the paper, $\mathcal{E}$ is

*This work has been partially supported by CICYT under grant TIC 95-0433-C03-03 and by HCM project CONSOLE.

[§]Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, Apdo. 22012, 46071 Valencia, Spain. e.mail:{alpuente,gvidal}@dsic.upv.es

[¶]Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze 206, 33100 Udine, Italy. e.mail: falaschi@dimi.uniud.it

assumed to be axiomatized as an equational Horn theory, which is called the 'program' [14, 20, 24]. We do not consider compositionality with respect to the union of programs in this paper. For this topic we refer to [4]. [4] defines a compositional semantics for the direct union of complete theories which correctly models the computational properties related to the use of logical variables.

The semantics of equational Horn programs is usually defined as some variant of *narrowing*, a method for generating complete sets of $\mathcal{E}$-unifiers with respect to a canonical set of clauses. Simple restrictions on narrowing, like narrowing only at basic positions, are still complete for theories which satisfy some additional properties [31]. The use of narrowing as the operational mechanism for computing leads to a powerful extension of ordinary logic programs [20, 35] and the computation model has many opportunities for parallelism. For example, [12] describes a kind of OR-parallelism in which, for each position in the term and each rule in the program, alternative narrowings are explored concurrently according to some heuristic function. Our work concerns an AND-parallel computation model of equational Horn programs, where all subexpressions can be narrowed independently and the computed substitutions obtained so far can then be composed. This mechanism of computation was also mentioned in [35]. We extend the notion of *parallel composition* of substitutions introduced in [22, 33] to the case of equational logic programs. Hence we show how complete sets of $\mathcal{E}$-unifiers for a given goal $\Leftarrow \Gamma_1, \Gamma_2$ can be generated by composing the complete sets of $\mathcal{E}$-unifiers computed by narrowing for the separate subgoals $\Leftarrow \Gamma_1$ and $\Leftarrow \Gamma_2$. This allows us to model the combination of substitutions computed by AND-parallel 'narrowing processes', i.e. by agents which narrow subexpressions in parallel. We show that for unrestricted narrowing a 'semantic' composition operator would be necessary. However, for basic narrowing we achieve a stronger result and show that the substitutions can be composed syntactically.

We have recently defined an equational logic language [1] as an instance of Constraint Logic Programming [23], where the equations to be solved with respect to an equational theory are considered as constraints. For a computational step in this framework, it is essential to be able to (semi-)decide if a set of equations is satisfiable. The computation of complete sets of $\mathcal{E}$-unifiers is less striking in this context, while it is essential a mechanism to evaluate the satisfiability of the constraints incrementally [1]. In [2], we have defined a lazy procedure which does not prove the satisfiability of the equational constraint $\mathbf{c}$ but just checks that $\mathbf{c}$ is not unsatisfiable by means of an approximated narrower. We show that a compositional narrowing can be taken as a basis for a compositional analysis for the problem of unsatisfiability. Then we show that compositionality leads, as a by-product, to an incremental implementation for the analysis. Therefore, while OR-compositionality, i.e. compositionality w.r.t. union of programs, has proven significant for programming with modules [4, 13], we show that AND-compositionality can lead to incremental computations.

This paper is organized as follows. After introducing some preliminary notions in Section 2, Section 3 defines an operator which describes the operational semantics of equational Horn programs in a compositional way. In Section 4, we introduce *compositional conditional narrowing*, an AND-parallel computation model for equational Horn programs. We characterize compositionally the *success* set of a goal, i.e. the set of the computed answer substitutions corresponding to all successful narrowing derivations. That is, we prove that the meaning of a composite goal can be obtained by composing the meanings of its conjuncts, when considering the success set as observable. Then we state the completeness of our semantics. In Section 5 we recall an abstract algorithm for the static analysis of unsatisfiability of equation sets [2, 3] and modify it to perform the analysis compositionally.

The analysis is based on two ideas. First, the construction of a finite graph of terms built from the equational theory, which helps one to know the narrowing derivations which definitely terminate. Secondly, the approximation of the narrowing derivations by means of an abstract calculus whose derivations cover all concrete computations. Our method introduces a generic technique of loop detection to ensure termination. We also present a specific loop-check which does not depend on the equation set to be solved. Section 6 formulates an incremental analyzer for equational constraints and presents some encouraging results from the implementation of the analyzer. Section 7 concludes. The proofs of the theorems are given in Appendix A. A preliminary version of this paper appeared in [5].

2 Preliminaries

We briefly recall some known results about equations, conditional rewrite systems and equational unification. For full definitions refer to [11, 25]. Throughout this paper, $\mathbf{V}$ will denote a countably infinite set of variables and Σ denotes a set of function symbols, each with a fixed associated arity. $\tau(\Sigma \cup \mathbf{V})$ and $\tau(\Sigma)$ denote the sets of terms and ground terms built on $\Sigma \cup \mathbf{V}$ and Σ, respectively. A Σ-equation $\mathbf{s} = \mathbf{t}$ is a pair of terms $\mathbf{s}, \mathbf{t} \in \tau(\Sigma \cup \mathbf{V})$. Terms are viewed as labelled trees in the usual way. Occurrences are represented by sequences, possibly empty, of natural numbers used to address subterms of $\mathbf{t}$, and they are ordered by the prefix ordering $\mathbf{u} \leq \mathbf{v}$ if there exists $\mathbf{w}$ such that $\mathbf{uw} = \mathbf{v}$. We let Λ denote the empty sequence. $\mathbf{O}(\mathbf{t})$ denotes the set of occurrences of a term $\mathbf{t}$. $\bar{\mathbf{O}}(\mathbf{t})$ denotes the set of nonvariable occurrences of a term $\mathbf{t}$. $\mathbf{t}_{|\mathbf{u}}$ is the subterm at the occurrence $\mathbf{u}$ of $\mathbf{t}$. $\mathbf{t}[\mathbf{r}]_{\mathbf{u}}$ is the term $\mathbf{t}$ with the subterm at the occurrence $\mathbf{u}$ replaced with $\mathbf{r}$. $\mathbf{t}[\mathbf{u}]$ denotes the label in $\mathbf{t}$ at occurrence $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{t})$. These notions extend to equations in a natural way. Identity of syntactic objects is denoted by $\equiv$. $\mathbf{Var}(\mathbf{s})$ is the set of distinct variables occurring in the syntactic object $\mathbf{s}$. A *fresh* variable is a variable that appears nowhere else. The symbol $\tilde{}$ denotes a finite sequence of symbols.

We describe the lattice of equation sets following [7]. We let **Eqn** denote the set of possibly existentially quantified finite sets of equations over terms. Elements of **Eqn** are regarded as (quantified) conjunctions of equations and treated modulo logical equivalence. We let **fail** denote the unsatisfiable equation set, which (logically) implies all other equation sets. Likewise, the empty equation set, denoted by **true**, is implied by all elements of **Eqn**. We write $\mathbf{E} \leq \mathbf{E}'$ if $\mathbf{E}'$ logically implies $\mathbf{E}$. Thus **Eqn** is a lattice ordered by $\leq$ with bottom element **true** and top element **fail**. An equation set is *solved* if it is either **fail** or it has the form $\exists \mathbf{y_1} \ldots \exists \mathbf{y_m}. \{\mathbf{x_1} = \mathbf{t_1}, \ldots, \mathbf{x_n} = \mathbf{t_n}\}$, where each $\mathbf{x_i}$ is a distinct variable not occurring in any of the terms $\mathbf{t_i}$ and each $\mathbf{y_i}$ occurs in some $\mathbf{t_j}$. Any set of equations $\mathbf{E}$ can be transformed into an equivalent one $\mathbf{solve}(\mathbf{E})$ which is solved. We restrict our interest to the set of idempotent substitutions over $\tau(\Sigma \cup \mathbf{V})$, which is denoted by **Sub**. There is a natural isomorphism between substitutions and unquantified equation sets. Given a solved unquantified equation set $\mathbf{E} = \{\mathbf{x_1} = \mathbf{t_1}, \ldots, \mathbf{x_n} = \mathbf{t_n}\}$, we let $\mathbf{sub}(\mathbf{E}) = \{\mathbf{x_1}/\mathbf{t_1}, \ldots, \mathbf{x_n}/\mathbf{t_n}\}$. Also, the equational representation of a substitution $\theta = \{\mathbf{x_1}/\mathbf{t_1}, \ldots, \mathbf{x_n}/\mathbf{t_n}\}$ is the set of equations $\widehat{\theta} = \{\mathbf{x_1} = \mathbf{t_1}, \ldots, \mathbf{x_n} = \mathbf{t_n}\}$. The identity function on $\mathbf{V}$ is called the empty substitution and denoted by ϵ. Given a substitution θ and a set of variables $\mathbf{W} \subseteq \mathbf{V}$, we denote by $\theta_{\restriction \mathbf{W}}$ the substitution obtained from θ by restricting its domain, $\mathbf{Dom}(\theta)$, to $\mathbf{W}$.

We consider the usual preorder on substitutions $\leq$: $\theta \leq \sigma$ iff $\exists \gamma.\, \sigma \equiv \theta\gamma$. Note that $\theta \leq \sigma$ iff $\widehat{\sigma} \Rightarrow \widehat{\theta}$ [33]. A substitution $\{\mathbf{x_1}/\mathbf{t_1}, \ldots, \mathbf{x_n}/\mathbf{t_n}\}$ is a *unifier* of an equation set $\mathbf{E}$ iff $\{\mathbf{x_1} = \mathbf{t_1}, \ldots, \mathbf{x_n} = \mathbf{t_n}\} \Rightarrow \mathbf{E}$. We

denote the set of unifiers of $\mathbf{E}$ by $\mathbf{unif}(\mathbf{E})$ and $\mathbf{mgu}(\mathbf{E})$ denotes the *most general unifier* of the unquantified equation set $\mathbf{E}$. In abuse of notation, we let **fail** denote failure when computing the $\mathbf{mgu}(\mathbf{E})$. While every unquantified equation set has a *most general unifier* [28], this is not generally true for equation sets with existentially quantified variables [7]. We write $\mathbf{s} \stackrel{?}{=} \mathbf{t}$ when we want to point out the fact that two terms $\mathbf{s}$ and $\mathbf{t}$ do unify.

We define an equational Horn theory $\mathcal{E}$ to be a finite set of (universally quantified) equational Horn clauses of the form $\mathbf{e} \Leftarrow \mathbf{e_1}, \ldots, \mathbf{e_n}$, $\mathbf{n} \geq 0$, where $\mathbf{e}, \mathbf{e_i}$, $\mathbf{i} = 1, \ldots, \mathbf{n}$, are equations. An equational goal is an equational Horn clause with no head. We let *Goal* denote the set of equational goals. The set of *states* is defined by $\mathbf{State} = \mathbf{Goal} \times \mathbf{Sub}$.

A Conditional Term Rewriting System (CTRS for short) is a pair $(\Sigma, \mathcal{R})$ where $\mathcal{R}$ is a finite set of reduction (or rewrite) rule schemes of the form $(\lambda \rightarrow \rho \Leftarrow \mathbf{C})$, $\lambda, \rho \in \tau(\Sigma \cup \mathbf{V})$, $\lambda \notin \mathbf{V}$ and $\mathbf{Var}(\rho) \subseteq \mathbf{Var}(\lambda)$. The condition $\mathbf{C}$ is a possibly empty conjunction $\mathbf{e_1}, \ldots, \mathbf{e_n}$, $\mathbf{n} \geq 0$, of equations. Variables in $\mathbf{C}$ that do not occur in λ are called extra-variables. If a rewrite rule has no condition we write $\lambda \rightarrow \rho$. When the condition in every rule in $\mathcal{R}$ is empty the system is said to be unconditional. Otherwise, it is said to be conditional. We will often write just $\mathcal{R}$ instead of $(\Sigma, \mathcal{R})$. We let $\mathbf{Ctrs}$ denote the set of conditional term rewriting systems.

A term $\mathbf{s}$ conditionally rewrites to a term $\mathbf{t}$, written $\mathbf{s} \rightarrow_{\mathcal{R}} \mathbf{t}$, if there exists a rule $(\lambda \rightarrow \rho \Leftarrow \mathbf{s_1} = \mathbf{t_1}, \ldots, \mathbf{s_n} = \mathbf{t_n}) \in \mathcal{R}$, an occurrence $\mathbf{u} \in \mathbf{O}(\mathbf{s})$, and a substitution σ such that $\mathbf{s}_{|\mathbf{u}} = \lambda\sigma$, $\mathbf{t} = \mathbf{s}[\rho\sigma]_{\mathbf{u}}$ and, for each $\mathbf{i} = 1, \ldots, \mathbf{n}$, there exists a term $\mathbf{w_i}$ such that $\mathbf{s_i}\sigma \rightarrow^*_{\mathcal{R}} \mathbf{w_i}$ and $\mathbf{t_i}\sigma \rightarrow^*_{\mathcal{R}} \mathbf{w_i}$. When no confusion can arise, we omit the subscript $\mathcal{R}$.

An equational Horn theory $\mathcal{E}$ whose clauses $\lambda = \rho \Leftarrow \mathbf{C}$ satisfy the assumptions $\lambda \notin \mathbf{V}$ and $\mathbf{Var}(\rho) \subseteq \mathbf{Var}(\lambda)$ can be viewed as a CTRS $\mathcal{R}$ where the rules are the heads (implicitely oriented from left to right) and the conditions are the respective bodies. We assume that these assumptions hold for all theories we consider in this paper. The equational theory $\mathcal{E}$ is said to be canonical (complete) if the binary one-step conditional rewriting relation $\rightarrow_{\mathcal{R}}$ defined by $\mathcal{R}$ is noetherian and confluent.

Let $\mathcal{R}$ be a CTRS. A function symbol $\mathbf{f} \in \Sigma$ is irreducible iff there is no rule $(\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \in \mathcal{R}$ such that $\mathbf{f}$ occurs as the outermost function symbol in λ, otherwise it is a defined function symbol. In theories where the above distinction is made, the signature Σ is partitioned as $\Sigma = \mathcal{C} \biguplus \mathcal{F}$, where $\mathcal{C}$ is the set of irreducible function symbols and $\mathcal{F}$ is the set of defined function symbols. The elements of $\mathcal{C}$ are often called constructors.

For CTRS $\mathcal{R}$, $\mathbf{r} \ll \mathcal{R}$ denotes that $\mathbf{r}$ is a new variant of a rule in $\mathcal{R}$ such that $\mathbf{r}$ contains no variable previously met during computation (standardised apart). The instantiated left-hand side $\lambda\sigma$ of a reduction rule $(\lambda \rightarrow \rho \Leftarrow \mathbf{C})$ is called a *redex* (*red*ucible *ex*pression) with *contractum* $\rho\sigma$. Given a CTRS $\mathcal{R}$, an equational goal clause $\Leftarrow \mathbf{g}$ conditionally narrows into a goal clause $\Leftarrow \mathbf{g}'$ (in symbols $\Leftarrow \mathbf{g} \stackrel{\theta}{\hookrightarrow} \Leftarrow \mathbf{g}'$) if there exists an equation $\mathbf{e} \in \mathbf{g}$, $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$, a standardised apart variant $(\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}$ and a substitution θ such that $\theta = \mathbf{mgu}(\{\mathbf{e}_{|\mathbf{u}} = \lambda\})$ and $\mathbf{g}' = ((\mathbf{g} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C})\theta$. $\mathbf{s}$ is called a (narrowing) *redex* iff there exists a new variant $(\lambda \rightarrow \rho \Leftarrow \mathbf{C})$ of a reduction rule in $\mathcal{R}$ and a substitution σ such that $\mathbf{s}\sigma \equiv \lambda\sigma$. A *narrowing derivation* is defined by $\Leftarrow \mathbf{g} \stackrel{\theta}{\hookrightarrow}^* \Leftarrow \mathbf{g}'$ iff $\exists \theta_1, \ldots, \theta_{\mathbf{n}}. \Leftarrow \mathbf{g} \stackrel{\theta_1}{\hookrightarrow} \ldots \stackrel{\theta_{\mathbf{n}}}{\hookrightarrow} \Leftarrow \mathbf{g}'$ and $\theta = \theta_1 \ldots \theta_{\mathbf{n}}$. We say that the derivation has length $\mathbf{n}$. If $\mathbf{n} = 0$ then $\theta = \epsilon$. In order to treat syntactical unification as a narrowing step, we add to the CTRS $\mathcal{R}$ the rule $(\mathbf{x} = \mathbf{x} \rightarrow \mathbf{true} \Leftarrow)$, $\mathbf{x} \in \mathbf{V}$. Then $\Leftarrow (\mathbf{t} = \mathbf{s}) \stackrel{\sigma}{\hookrightarrow} \Leftarrow \mathbf{true}$ holds iff $\sigma = \mathbf{mgu}(\{\mathbf{t} = \mathbf{s}\})$. A successful derivation for $\Leftarrow \mathbf{g}$ in $\mathcal{R} \cup \{\mathbf{x} = \mathbf{x} \rightarrow \mathbf{true} \Leftarrow\}$ is a narrowing derivation $\Leftarrow \mathbf{g} \stackrel{\theta}{\hookrightarrow}^* \Leftarrow \mathbf{true}$ and $\theta_{|\mathbf{Var}(\mathbf{g})}$ is called a computed answer substitution for $\Leftarrow \mathbf{g}$ in $\mathcal{R}$.

An important kind of equational Horn theories are those which are level-confluent [15, 31]. Let $\rightarrow_{\mathcal{R}}$ be the conditional rewrite relation corresponding to a Horn equational theory $\mathcal{E}$. Then $\rightarrow_{\mathcal{R}}$ is equivalent to $\rightarrow_{\mathcal{R}} = \cup_{\mathbf{i}\geq 0}\{\rightarrow_{\mathcal{R}_\mathbf{i}}\}$, where:

(1) $\rightarrow_{\mathcal{R}_0} = \emptyset$; and

(2) $\mathbf{t} \rightarrow_{\mathcal{R}_{\mathbf{n}+1}} \mathbf{t}'$ holds iff there is a rule $(\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \in \mathcal{R}$, a non-variable occurrence $\mathbf{u}$ of $\mathbf{t}$ and a substitution σ such that $\mathbf{t}_{|\mathbf{u}} = \lambda\sigma$, $\mathbf{t}' = \mathbf{t}[\rho\sigma]_\mathbf{u}$ and $\mathbf{s}\sigma \downarrow_{\mathcal{R}_\mathbf{n}} \mathbf{s}'\sigma$ for all $(\mathbf{s} = \mathbf{s}') \in \mathbf{C}$. $\rightarrow_{\mathcal{R}}$ is called level-confluent [15] iff, for each $\mathbf{n} \geq 0$, the relation $\rightarrow_{\mathcal{R}_\mathbf{n}}$ is confluent. Level confluence can still be ensured by syntactic conditions [15]. Of course, level confluence implies confluence. We call $\mathcal{R}$ level-canonical if $\rightarrow_{\mathcal{R}}$ is level-confluent and noetherian. This is equivalent to say that each $\rightarrow_{\mathcal{R}_\mathbf{n}}$ is canonical. The notion of level-canonicity of a conditional term rewriting system extends in the obvious way to equational Horn theories.

Each equational Horn theory $\mathcal{E}$ generates a smallest congruence relation $=_\mathcal{E}$ called $\mathcal{E}$-**equality** on the set of terms $\tau(\Sigma \cup \mathbf{V})$ (the least equational theory which contains all logic consequences of $\mathcal{E}$ under the entailment relation $\models$ obeying the axioms of equality for $\mathcal{E}$). $\mathcal{E}$ is a presentation or axiomatization of $=_\mathcal{E}$. In abuse of notation, we sometimes speak of the equational theory $\mathcal{E}$ to denote the theory axiomatized by $\mathcal{E}$. $\mathcal{E}$-equality is extended to substitutions by $\sigma =_\mathcal{E} \theta$ iff $\forall \mathbf{x} \in \mathbf{V}.\ \mathbf{x}\sigma =_\mathcal{E} \mathbf{x}\theta$.

A finite set of equations $\Gamma = \{\mathbf{s}_1 = \mathbf{t}_1, \ldots, \mathbf{s}_\mathbf{n} = \mathbf{t}_\mathbf{n}\}$ together with an equational theory $\mathcal{E}$ is called an $\mathcal{E}$-**unification** problem. A substitution σ is an $\mathcal{E}$-**unifier** of the equation set Γ iff $\mathcal{E} \models (\widehat{\sigma} \Rightarrow \Gamma)$ [30]. By abuse of notation, σ is often called **solution**. The set $\mathcal{U}_\mathcal{E}(\Gamma)$ of all $\mathcal{E}$-**unifiers** of Γ is recursively enumerable [14, 20, 39].

For $\mathcal{E}$-**unification** problems, the notion of most general unifier generalizes to complete sets of minimal (incomparable) $\mathcal{E}$-**unifiers**. A set $\mathbf{S}$ of $\mathcal{E}$-unifiers of the equation set Γ is complete iff every $\mathcal{E}$-unifier σ of Γ factors into $\sigma =_\mathcal{E} \theta\gamma$ for some substitutions $\theta \in \mathbf{S}$ and γ. A complete set of $\mathcal{E}$-unifiers of a system of equations may be infinite. Minimal complete sets $\mu\mathcal{U}_\mathcal{E}(\Gamma)$ of $\mathcal{E}$-**unifiers** of Γ do not always exist. An $\mathcal{E}$-**unification** procedure is complete if it generates a complete set of $\mathcal{E}$-**unifiers** for all input equation system. Conditional narrowing has been shown to be a complete $\mathcal{E}$-unification algorithm for canonical theories satisfying different restrictions [20, 31]. Since unrestricted narrowing has quite a large search space, several strategies to control the selection of redexes have been devised to improve the efficiency of narrowing by getting rid of some useless derivations (see [16] for a recent survey). Narrowing at only *basic* positions [15, 20, 21, 31, 32] has been proven to be a complete $\mathcal{E}$-unification algorithm for level-canonical Horn equational theories.

3 Equational Parallel Composition

In the following, we recall the notion of *parallel composition* of substitutions, denoted by $\Uparrow$. Roughly speaking, parallel composition is the operation of unification generalized to substitutions.

Parallel composition corresponds to one of the basic operations performed by the AND-parallel execution model of logic programs [18, 22, 33]. Namely, when two subgoals (of the same goal) are run in parallel, the answer substitutions (computed independently) have to be combined to get the final result. This 'combination' can be done as follows [18, 22, 33]. Given two idempotent substitutions θ_1 and θ_2, we let $\theta_1 \Uparrow \theta_2 = \mathbf{mgu}(\widehat{\theta_1} \cup \widehat{\theta_2})$. Parallel composition is idempotent, commutative, associative and has a null

element **fail** and an identical element ϵ. $\Uparrow$ is lifted to sets of substitutions by

$$\Theta_1 \Uparrow \Theta_2 = \begin{cases} \bigcup_{\theta_1 \in \Theta_1, \theta_2 \in \Theta_2} \{\theta_1 \Uparrow \theta_2\} & \text{if it is different from } \{\mathbf{fail}\} \\ \emptyset & \text{otherwise.} \end{cases}$$

Parallel composition was proposed in [33] as a basis for a compositional characterization of the semantics of Horn Clause Logic. We are able to generalize the notion of parallel composition to the case when unification in equational theories is considered. In the following definition we formalize the notion of *equational parallel composition*, denoted by $\Uparrow_{\mathcal{E}}$.

Definition 3.1 *Let $\theta_1, \theta_2 \in \mathbf{Sub}$. We define the operator $\Uparrow_{\mathcal{E}} : \mathbf{Sub} \times \mathbf{Sub} \rightarrow \wp(\mathbf{Sub})$ as follows:*

$$\theta_1 \Uparrow_{\mathcal{E}} \theta_2 = \mathcal{U}_{\mathcal{E}}(\widehat{\theta_1} \cup \widehat{\theta_2}).$$

Example 1 *Let $\mathcal{E} = \{\mathbf{f}(0) = 0,\ \mathbf{f}(\mathbf{g}(\mathbf{X})) = \mathbf{g}(\mathbf{X}),\ \mathbf{g}(0) = \mathbf{c}(0),\ \mathbf{g}(\mathbf{c}(\mathbf{X})) = \mathbf{g}(\mathbf{X})\}$.*

1. *Let $\theta_1 = \{\mathbf{X}/\mathbf{g}(\mathbf{Z})\}$ and $\theta_2 = \{\mathbf{X}/\mathbf{c}(\mathbf{Z})\}$. Then $\{\mathbf{X}/\mathbf{c}(0), \mathbf{Z}/0\} \in \theta_1 \Uparrow_{\mathcal{E}} \theta_2$.*
2. *Let $\theta_1 = \{\mathbf{X}/\mathbf{f}(0)\}$ and $\theta_2 = \{\mathbf{X}/\mathbf{g}(\mathbf{Z})\}$. Then $\theta_1 \Uparrow_{\mathcal{E}} \theta_2 = \emptyset$.*

The operator $\Uparrow_{\mathcal{E}}$ can be lifted to sets of substitutions as follows.

Definition 3.2 *Given $\Theta_1, \Theta_2 \in \wp(\mathbf{Sub})$, let:*

$$\Theta_1 \Uparrow_{\mathcal{E}} \Theta_2 = \bigcup_{\theta_1 \in \Theta_1, \theta_2 \in \Theta_2} \theta_1 \Uparrow_{\mathcal{E}} \theta_2.$$

It is straightforward to show that $\Uparrow_{\mathcal{E}}$ is commutative and associative and has null element $\emptyset$.

Given an $\mathcal{E}$-unification problem $\Gamma = \Gamma_1 \cup \Gamma_2$, a compositional characterization of the set of all $\mathcal{E}$-unifiers of Γ is given in the following proposition.

Proposition 3.3 *Let $\Gamma = \Gamma_1 \cup \Gamma_2$, $\Theta_1 = \mathcal{U}_{\mathcal{E}}(\Gamma_1)$ and $\Theta_2 = \mathcal{U}_{\mathcal{E}}(\Gamma_2)$. Then $\mathcal{U}_{\mathcal{E}}(\Gamma) = \Theta_1 \Uparrow_{\mathcal{E}} \Theta_2$.*

We note that it is much more complex to evaluate $\Uparrow_{\mathcal{E}}$ than the much simpler operation $\Uparrow$. However, equational parallel composition can be redefined in terms of 'most general' unifiers in the case of *finitary* theories, for which the solutions to an $\mathcal{E}$-unification problem Γ can always be represented by a complete and minimal finite set $\mu\mathcal{U}_{\mathcal{E}}(\Gamma)$ of (maximally general) $\mathcal{E}$-unifiers, which is unique up to equivalence [39]. Equational theories which are of finitary *unification type* play an important role in logic programming with equality [24]. In general, the unification type of an equational theory is undecidable. On the other hand, for a finitary theory the minimality requirement is often too strong, since an algorithm which generates a superset of $\mu\mathcal{U}_{\mathcal{E}}(\Gamma)$ may be far more efficient than a minimal one and hence sometimes preferable. In the following section, we will show that we can still work with ordinary parallel composition $\Uparrow$ when we consider the class of (level-)canonical equational theories, for which the problem of $\mathcal{E}$-unification reduces to ordinary (syntactic) unification plus narrowing [21].

4 Compositional Conditional Narrowing

In this section we present a compositional structured operational semantics for our language. Following the standard definitions, what we *observe* about a goal $\Leftarrow \mathbf{g}$ in a program $\mathcal{R}$ is the success set. We describe the *success* set of a goal, i.e. the set of the computed answer substitutions corresponding to all successful narrowing derivations, by a formal operational semantics $\mathcal{O} : \mathbf{Goal} \mapsto \wp\,\mathbf{Sub}$, based on a transition relation, which associates a set of substitutions with a goal.

Basic (conditional) narrowing is a restricted form of (conditional) narrowing where only terms at *basic* occurrences are considered to be narrowed [21, 31]. Informally, a basic occurrence is a nonvariable occurrence of the original goal or one that was introduced into the goal by the nonvariable part of the right-hand side or the condition of a rule applied in a preceeding narrowing step. The idea behind the concept of *basic* is to avoid narrowing steps on subterms that are introduced by instantiation. Basic conditional narrowing is a complete method for solving equations in the theory defined by a level-canonical conditional term rewriting system [31], and has been proposed as the operational model of equational logic programs in [20]. In the rest of this paper, we assume the level-canonical CTRS $\mathcal{R}$ to be fixed.

We formulate basic conditional narrowing according to the partition of equational goals into a *skeleton* and an *environment* part, as in [20]. The *skeleton* part is a set of equations $\mathbf{g}$ and the *environment* part is a substitution θ. Substitutions are composed in the environment part, but are not applied to the equations in the skeleton part, as opposed to ordinary (unrestricted) narrowing (cf. Section 2). Due to this representation, the basic occurrences in $\mathbf{g}\theta$ are all in $\mathbf{g}$, whereas the non-basic occurrences are all in the codomain of θ. This ensures that no narrowing step will reduce any expression brought by a substitution computed in a previous step. The calculus is defined as a transition system $(\mathbf{State}, \rightsquigarrow)$ whose transition relation $\rightsquigarrow \subseteq \mathbf{State} \times \mathbf{State}$ formalizes the computation steps [34]. We define the basic conditional narrowing relation $\rightsquigarrow$ as the smallest relation satisfying

$$\frac{\mathbf{e} \in \mathbf{g} \ \wedge\ \mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}) \ \wedge\ (\lambda \to \rho \Leftarrow \mathbf{C}) \ll \mathcal{R} \ \wedge\ \sigma = \mathbf{mgu}(\{\mathbf{e}_{|\mathbf{u}}\theta = \lambda\})}{\langle \Leftarrow \mathbf{g}, \theta\rangle \rightsquigarrow \langle \Leftarrow (\mathbf{g} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C}, \theta\sigma\rangle}$$

A *basic conditional narrowing derivation* is a sequence of states $\mathbf{s}_1 \rightsquigarrow \mathbf{s}_2 \rightsquigarrow \ldots$, where $\mathbf{s_i} \equiv \langle \Leftarrow \mathbf{g_i}, \theta_\mathbf{i}\rangle$ is the $\mathbf{i}$th state in the sequence.

Based on this transition system, we define the operational semantics of an equational goal $\Leftarrow \mathbf{g}$ in the CTRS $\mathcal{R} \cup \{\mathbf{x} = \mathbf{x} \to \mathbf{true} \Leftarrow\}$ by the (non ground) set (*success set*)

$$\mathcal{O}_\mathcal{R}(\Leftarrow \mathbf{g}) = \{\theta_{\restriction \mathbf{Var(g)}} \mid \langle \Leftarrow \mathbf{g}, \epsilon\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{true}, \theta\rangle\}.$$

We often write $\mathcal{O}(\Leftarrow \mathbf{g})$ instead of $\mathcal{O}_\mathcal{R}(\Leftarrow \mathbf{g})$ when $\mathcal{R}$ is understood.

We want to give a compositional description of the answers computed by a goal. Our aim is to formulate a compositional calculus based on basic conditional narrowing. Informally, we consider the basic strategy here because, in order to prove the equivalence between the (standard) operational semantics and its compositional version, we need the standard semantics to be compositional w.r.t. the AND operator. It is easy to prove that this condition holds for basic conditional narrowing. The next theorem shows that the standard success set semantics $\mathcal{O}$ is compositional w.r.t. the AND operator[1].

[1] Here we do not consider the problem of the independent 'standardization apart' of the parallel computations for $\mathbf{g}_1$ and $\mathbf{g}_2$

Theorem 4.1 $\mathcal{O}(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \mathcal{O}(\Leftarrow \mathbf{g}_1) \Uparrow \mathcal{O}(\Leftarrow \mathbf{g}_2)$

The proof of Theorem 4.1 heavily depends on the fact we are dealing with the basic strategy. We note that this result does not hold for ordinary (unrestricted) narrowing, i.e. ordinary narrowing is not compositional using the parallel composition operator $\Uparrow$. Roughly speaking, what is wrong with ordinary narrowing is the fact that it transfers terms from the substitution part into the goal, thus introducing narrowing steps (at *non-basic* positions) that might not be proven when the subgoals are solved independently. The following example illustrates this point.

Example 2 *Let $\mathcal{R}$ be the following program*

$$\begin{array}{rcl} \mathcal{R} = \{ \quad \mathbf{z}(\mathbf{s}(0)) & \rightarrow & 0 \Leftarrow \\ \mathbf{z}(\mathbf{one}(\mathbf{X})) & \rightarrow & 0 \Leftarrow \\ \mathbf{one}(0) & \rightarrow & \mathbf{s}(0) \Leftarrow \\ \mathbf{one}(\mathbf{s}(\mathbf{X})) & \rightarrow & \mathbf{one}(\mathbf{X}) \Leftarrow \ \}, \end{array}$$

and consider the goal $\Leftarrow \mathbf{g} \ \equiv \Leftarrow (\mathbf{g}_1, \mathbf{g}_2) \ \equiv \Leftarrow (\mathbf{X} = \mathbf{s}(0),\ \mathbf{z}(\mathbf{X}) = 0)$. *Then there exists the following (ordinary) narrowing derivation:*

$$\begin{array}{l} \Leftarrow (\mathbf{X} = \mathbf{s}(0),\ \mathbf{z}(\mathbf{X}) = 0) \overset{\{\mathbf{X}/\mathbf{one}(\mathbf{Y})\}}{\hookrightarrow} \Leftarrow (\mathbf{one}(\mathbf{Y}) = \mathbf{s}(0),\ 0 = 0) \overset{\{\mathbf{Y}/0\}}{\hookrightarrow} \\ \Leftarrow (\mathbf{s}(0) = \mathbf{s}(0),\ 0 = 0) \overset{\epsilon}{\hookrightarrow} \Leftarrow (\mathbf{true},\ 0 = 0) \overset{\epsilon}{\hookrightarrow} \Leftarrow \mathbf{true} \end{array}$$

with computed answer substitution $\theta = \{\mathbf{X}/\mathbf{one}(0)\}$. *However, since the goal* $\Leftarrow \mathbf{g}_1 \equiv \Leftarrow \mathbf{X} = \mathbf{s}(0)$ *only computes the substitution* $\{\mathbf{X}/\mathbf{s}(0)\}$, *the solution* θ *cannot be obtained from the (syntactic) composition of the computed answers substitutions of* $\Leftarrow \mathbf{g}_1$ *and* $\Leftarrow \mathbf{g}_2$. *Note that basic conditional narrowing does not compute the answer substitution* $\{\mathbf{X}/\mathbf{one}(0)\}$ *for the goal* $\Leftarrow \mathbf{g}$.

Now we are ready to give a compositional characterization of the operational semantics of equational Horn programs in a style similar to that of [33] for logic programs.

Definition 4.2 (Compositional Conditional Narrowing).
We define compositional conditional narrowing *as a transition system* $(\mathbf{State}, \longmapsto)$ *whose transition relation* $\longmapsto \subseteq \mathbf{State} \times \mathbf{State}$ *is the smallest relation which satisfies*

$$(1) \quad \frac{\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}) \ \wedge \ (\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \ll \mathcal{R} \ \wedge \ \sigma = \mathbf{mgu}(\{\mathbf{e}_{|\mathbf{u}}\theta = \lambda\})}{\langle \Leftarrow \{\mathbf{e}\}, \theta\rangle \longmapsto \langle \Leftarrow \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C}, \theta\sigma\rangle}$$

$$(2) \quad \frac{\langle \Leftarrow \mathbf{g}_1, \theta\rangle \longmapsto \langle \Leftarrow \mathbf{g}_1', \theta\theta_1\rangle \ \wedge \ \langle \Leftarrow \mathbf{g}_2, \theta\rangle \longmapsto \langle \Leftarrow \mathbf{g}_2', \theta\theta_2\rangle \ \wedge \ \theta_1 \Uparrow \theta_2 \not\equiv \mathbf{fail}}{\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \theta\rangle \longmapsto \langle \Leftarrow (\mathbf{g}_1', \mathbf{g}_2'), \theta(\theta_1 \Uparrow \theta_2)\rangle}$$

Roughly speaking, in the computation model formalized by the transition system above, all equations in the equational goal to be solved are reduced at the same time. Then, the substitutions resulting from these local computations are combined by means of the operator of *parallel composition* to obtain the global result of the computation.

which might cause the clash of variables in the operational semantics of the parallel subgoals. For solutions to this problem, we refer to [9, 37].

Example 3 *Let $\mathcal{R}$ be the program of Example 2 and consider the goal $\Leftarrow \mathbf{g} \equiv \Leftarrow (\mathbf{W} = \mathbf{one}(\mathbf{Y}),\ \mathbf{z}(\mathbf{Y}) = 0)$. Then there exists the following compositional narrowing derivation:*

$\langle \Leftarrow (\mathbf{W} = \mathbf{one}(\mathbf{Y}),\ \mathbf{z}(\mathbf{Y}) = 0), \epsilon\rangle \longmapsto \langle \Leftarrow (\mathbf{W} = \mathbf{one}(\mathbf{X}),\ 0 = 0), \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle$ (using rule 2)

(*since* $\langle \Leftarrow \mathbf{W} = \mathbf{one}(\mathbf{Y}), \epsilon\rangle \longmapsto \langle \Leftarrow \mathbf{W} = \mathbf{one}(\mathbf{X}), \{\mathbf{Y}/\mathbf{s}(\mathbf{X})\}\rangle$ (using rule 1) *and*
$\langle \Leftarrow \mathbf{z}(\mathbf{Y}) = 0, \epsilon\rangle \longmapsto \langle \Leftarrow 0 = 0, \{\mathbf{Y}/\mathbf{s}(0)\}\rangle$ (using rule 1) *and*
$\{\mathbf{Y}/\mathbf{s}(\mathbf{X})\} \Uparrow \{\mathbf{Y}/\mathbf{s}(0)\} = \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\} \not\equiv \mathbf{fail}$)

$\langle \Leftarrow (\mathbf{W} = \mathbf{one}(\mathbf{X}),\ 0 = 0), \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle \longmapsto \langle \Leftarrow \mathbf{W} = \mathbf{s}(0), \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle$ (using rule 2)

(*since* $\langle \Leftarrow \mathbf{W} = \mathbf{one}(\mathbf{X}), \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle \longmapsto \langle \Leftarrow \mathbf{W} = \mathbf{s}(0), \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle$ (using rule 1) *and*
$\langle \Leftarrow 0 = 0, \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle \longmapsto \langle \Leftarrow \mathbf{true}, \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle$ (using rule 1))

$\langle \Leftarrow \mathbf{W} = \mathbf{s}(0), \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0\}\rangle \longmapsto \langle \Leftarrow \mathbf{true}, \{\mathbf{Y}/\mathbf{s}(0), \mathbf{X}/0, \mathbf{W}/\mathbf{s}(0)\}\rangle$ (using rule 1)

with computed answer substitution $\theta = \{\mathbf{Y}/\mathbf{s}(0), \mathbf{W}/\mathbf{s}(0)\}$.

The computation model formalized in Definition 4.2 could be taken as a basis for an AND-parallel computation model of equational Horn programs. We note that the model has not been devised to achieve maximal parallelism in the sense that not all redexes in a given goal are allowed to perform one narrowing step independently. Namely, redexes which occur in a same equation are not reduced in parallel, while they could. To overcome this lack, it suffices to introduce the following *flattening* rule, which preserves the reachable solutions

$$(3) \quad \frac{\mathbf{e} \in \mathbf{g} \ \wedge\ \mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}) \ \wedge\ \mathbf{x} \text{ is a new variable}}{\langle \Leftarrow \mathbf{g}, \theta\rangle \longmapsto \langle \Leftarrow (\mathbf{g} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\mathbf{x}]_{\mathbf{u}}\} \cup \{\mathbf{e}_{|\mathbf{u}} = \mathbf{x}\}, \theta\rangle}$$

provided that both $\mathbf{e}_{|\mathbf{u}}$ and $\mathbf{e}[\mathbf{x}]_{\mathbf{u}}$ contain at least one function symbol [32]. Note that we need not determine the level of granularity [27] (as it neither affects correctness nor completeness); this we consider to be an implementation issue that could enable (more) effective parallelizations.

Our approach differs from other AND-parallel execution models, such as e.g. [27], where subexpressions are only narrowed in parallel if they are *independent*, i.e. if they do not share (unbound) variables. A 'need-driven' synchronization model is imposed which compels processes to wait for the value of a parallel subexpression if such a value is needed. In [26], a *dependent* AND-parallel execution model is exploited, but the imposed synchronization mechanisms produce too much overhead.

A new operational semantics of an equational goal $\Leftarrow \mathbf{g}$ in the CTRS $\mathcal{R} \cup \{\mathbf{x} = \mathbf{x} \rightarrow \mathbf{true} \Leftarrow\}$ can now be defined by

Definition 4.3 $\mathcal{O}'(\Leftarrow \mathbf{g}) = \{\theta_{\restriction \mathbf{Var}(\mathbf{g})} \mid \langle \Leftarrow \mathbf{g}, \epsilon\rangle \longmapsto^* \langle \Leftarrow \mathbf{true}, \theta\rangle\}$.

The new success set semantics $\mathcal{O}'$ is compositional w.r.t. the AND operator. Formally,

Theorem 4.4 $\mathcal{O}'(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \mathcal{O}'(\Leftarrow \mathbf{g}_1) \Uparrow \mathcal{O}'(\Leftarrow \mathbf{g}_2)$.

The following result states that the compositional conditional semantics and the standard basic conditional semantics coincide. The correspondence is restricted to successes, namely to the substitutions computed by all successfully terminating derivations.

Corollary 4.5 $\mathcal{O}(\Leftarrow \mathbf{g}) = \mathcal{O}'(\Leftarrow \mathbf{g})$.

As a consequence of Corollary 4.5, every solution found by basic conditional narrowing is found by compositional (basic) conditional narrowing as well. Hence, we have the following corollary for level-canonical systems.

Corollary 4.6 (completeness).
Let $\mathcal{E}$ be a level-canonical equational Horn theory. The set $\mathcal{O}'(\Leftarrow \mathbf{g})$ is a complete set of $\mathcal{E}$-unifiers of $\mathbf{g}$.

We note that, by forcing the join of the parallel solutions every time that all equations in the goal have performed a single step independently, the compositional execution model formalized by Definition 4.2 might not couch all the exploitable AND-parallelism. Corollary 4.5 suggests that many different computation schemes are possible. For instance, we can solve in parallel all (sub-)goals, joining the AND-parallel (sub-)goals when the (sub-)goals are completely solved, instead of forcing the syncronization of the AND-parallel branches at every single reduction step. In the following section, we show how this execution scheme can be efficiently exploited for program analysis.

5 Abstract Basic Conditional Narrowing

Abstract interpretation is a theory of semantic approximation which is used to provide statically sound answers to some questions about the run-time behaviour of programs [8]. The 'concrete' data and semantic operators are approximated and replaced by corresponding 'abstract' data and operators. The 'answers' obtained by using the abstract data and operators have to be proven sound by exploiting the correspondence with the concrete data and operators. In this section, we recall the framework of abstract interpretation for analysis of equational unsatisfiability we defined in [2]. Then we extend this framework by defining a compositional abstract semantics which safely approximates the observables.

Our analysis of unsatisfiability is an abstraction of the transition system semantics for basic conditional narrowing we have introduced in Section 4. We first recall the abstract domains and the associated abstract operators together with some previous results concerning them.

5.1 Abstract Domains and Operators

Some of the definitions at the beginning of this section are already in [2] and are reported for completeness. A difference with respect to [2] is that we present Definition 5.8 as parametric with respect to a functional dependency graph (or loop-check). The corresponding definition in [2] is an instance for the specific dependency graph considered there.

A *description* is the association of an *abstract domain* $(\mathbf{D}, \leq)$ (a poset) with a *concrete domain* $(\mathbf{E}, \leq)$ (a poset). When $\mathbf{E} = \mathbf{Eqn}$, $\mathbf{E} = \mathbf{Sub}$ or $\mathbf{E} = \mathbf{State}$, the description is called an *equation description*, a *substitution description* or a *state description*, respectively. The correspondence between the abstract and concrete domain is established through a monotonic 'concretization' function $\gamma : \mathbf{D} \to \wp\mathbf{E}$ [7]. We say that $\mathbf{d}$ *approximates* $\mathbf{e}$, written $\mathbf{d} \propto \mathbf{e}$, iff $\mathbf{e} \in \gamma(\mathbf{d})$. The approximation relation can be lifted to relations and cross products as usual [7].

We approximate the behaviour of a CTRS and initial state by an abstract transition system which can be viewed as a finite transition graph with nodes labeled by state descriptions, where transitions are proved by (abstract) narrowing reduction [2]. State descriptions consist of a set of equations with substitution

descriptions. The descriptions for equations, substitutions and term rewriting systems are defined as follows. By abuse of notation, we denote in the same way a preorder and the corresponding partial ordering induced on the equivalence classes of the equivalence relation associated with the preorder.

Definition 5.1 (term, equation poset).
By $\mathcal{T} = (\tau(\Sigma \cup \mathbf{V}), \leq)$*, we denote the standard domain of (equivalence classes of) terms ordered by the standard partial order* $\leq$ *induced by the preorder on terms given by the relation of being 'more general'. Let* $\perp$ *be an irreducible symbol, where* $\perp \notin \Sigma$*. Let* $\mathcal{T}_\mathcal{A} = (\tau(\Sigma \cup \mathbf{V} \cup \{\perp\}), \preceq)$ *be the domain of terms over the signature augmented by* $\perp$ *(abstract terms), where the partial order* $\preceq$ *is defined as follows:*

(a) $\forall \mathbf{t} \in \mathcal{T}_\mathcal{A}. \perp \preceq \mathbf{t}$ *and* $\mathbf{t} \preceq \mathbf{t}$ *and*

(b) $\forall \mathbf{s_1}, \ldots, \mathbf{s_n}, \mathbf{s'_1}, \ldots, \mathbf{s'_n} \in \mathcal{T}_\mathcal{A}, \forall \mathbf{f/n} \in \Sigma.\ \mathbf{s'_1} \preceq \mathbf{s_1} \wedge \ldots \wedge \mathbf{s'_n} \preceq \mathbf{s_n} \Rightarrow \mathbf{f}(\mathbf{s'_1}, \ldots, \mathbf{s'_n}) \preceq \mathbf{f}(\mathbf{s_1}, \ldots, \mathbf{s_n})$

This order can be extended to (abstract) equations: $\mathbf{s'} = \mathbf{t'} \preceq \mathbf{s} = \mathbf{t}$ *iff* $\mathbf{s'} \preceq \mathbf{s}$ *and* $\mathbf{t'} \preceq \mathbf{t}$ *and to (abstract) sets of equations* $\mathbf{S}, \mathbf{S'}$*:*
$\mathbf{S'} \preceq \mathbf{S}$ *iff* $\forall \mathbf{e'} \in \mathbf{S'}. \exists \mathbf{e} \in \mathbf{S}$ *such that* $\mathbf{e'} \preceq \mathbf{e}$*. Note that* $\mathbf{S'} \preceq \mathbf{true} \Rightarrow \mathbf{S'} \equiv \mathbf{true}$*.*

Roughly speaking, we introduce the special symbol $\perp$ in the abstract domains to represent any concrete term. Logically, $\perp$ stands for an existentially quantified variable [2, 29, 30]. Define $[\![\mathbf{S}]\!] = \mathbf{S'}$, where the n-tuple of occurrences of $\perp$ in $\mathbf{S}$ is replaced by an n-tuple of existentially quantified fresh variables in $\mathbf{S'}$.

The following proposition shows how the preorder $\preceq$ defined on the set of abstract equation sets and the implication ordering $\leq$ defined on **Eqn** relate.

Proposition 5.2 [2] *Let* $\mathbf{S}, \mathbf{S'}$ *be two abstract equation sets s.t.* $\mathbf{S'} \preceq \mathbf{S}$*. Then* $[\![\mathbf{S'}]\!] \leq [\![\mathbf{S}]\!]$*, that is,* $[\![\mathbf{S}]\!] \Rightarrow [\![\mathbf{S'}]\!]$*.*

Definition 5.3 (abstract substitution).
An abstract substitution is a set of the form $\{\mathbf{x_1/t_1}, \ldots, \mathbf{x_n/t_n}\}$ *where, for each* $\mathbf{i} = 1, \ldots, \mathbf{n}$*,* $\mathbf{x_i}$ *is a distinct variable in* $\mathbf{V}$ *not occurring in any of the terms* $\mathbf{t_1}, \ldots, \mathbf{t_n}$ *and* $\mathbf{t_i} \in \tau(\Sigma \cup \mathbf{V} \cup \{\perp\})$*. The preorder on abstract substitutions can be given as logical implication: let* $\theta, \kappa \in \mathbf{Sub}_\mathcal{A}$*,* $\kappa \preceq \theta$ *iff* $[\![\widehat{\theta}]\!] \Rightarrow [\![\widehat{\kappa}]\!]$*.*

Let us introduce the abstract domains which we will use in our analysis.

Definition 5.4 (term, equation, substitution, state description).
Let $\mathcal{T} = (\tau(\Sigma \cup \mathbf{V}), \leq)$ *and* $\mathcal{T}_\mathcal{A} = (\tau(\Sigma \cup \mathbf{V} \cup \{\perp\}), \preceq)$*. The term description is* $\langle \mathcal{T}_\mathcal{A}, \gamma, \mathcal{T} \rangle$ *where* $\gamma : \mathcal{T}_\mathcal{A} \to \wp\mathcal{T}$ *is defined by:* $\gamma(\mathbf{t'}) = \{\mathbf{t} \in \mathcal{T} \mid \mathbf{t'} \preceq \mathbf{t}\}$*.*

Let **Eqn** *be the set of finite sets of equations over* $\tau(\Sigma \cup \mathbf{V})$ *and* $\mathbf{Eqn}_\mathcal{A}$ *be the set of finite sets of equations over* $\tau(\Sigma \cup \mathbf{V} \cup \{\perp\})$*. The equation description is* $\langle (\mathbf{Eqn}_\mathcal{A}, \preceq), \gamma, (\mathbf{Eqn}, \leq) \rangle$*, where* $\gamma : \mathbf{Eqn}_\mathcal{A} \to \wp\,\mathbf{Eqn}$ *is defined by:* $\gamma(\mathbf{g'}) = \{\mathbf{g} \in \mathbf{Eqn} \mid \mathbf{g'} \preceq \mathbf{g}$ *and* $\mathbf{g}$ *is unquantified* $\}$*.*

Let **Sub** *be the set of substitutions over* $\tau(\Sigma \cup \mathbf{V})$ *and* $\mathbf{Sub}_\mathcal{A}$ *be the set of substitutions over* $\tau(\Sigma \cup \mathbf{V} \cup \{\perp\})$*. The substitution description is* $\langle (\mathbf{Sub}_\mathcal{A}, \preceq),\ \gamma, (\mathbf{Sub}, \leq) \rangle$*, where* $\gamma : \mathbf{Sub}_\mathcal{A} \to \wp\,\mathbf{Sub}$ *is defined by:* $\gamma(\kappa) = \{\theta \in \mathbf{Sub} \mid \kappa \preceq \theta\}$*.*

Define the abstract state domain $\mathbf{State}_\mathcal{A}$ *induced by* $\mathbf{Eqn}_\mathcal{A}$ *and* $\mathbf{Sub}_\mathcal{A}$ *to be* $\mathbf{State}_\mathcal{A} = \{\langle \Leftarrow \mathbf{g}, \kappa \rangle \mid \mathbf{g} \in \mathbf{Eqn}_\mathcal{A}, \kappa \in \mathbf{Sub}_\mathcal{A}\}$*.*

An *abstract term rewriting system* is a finite set of abstract reduction rules of the form $(\lambda \to \rho \Leftarrow \mathbf{C})$, $\lambda \in \tau(\Sigma \cup \mathbf{V})$, $\rho \in \tau(\Sigma \cup \mathbf{V} \cup \{\perp\})$, $\lambda \notin \mathbf{V}, \mathbf{Var}(\rho) \subseteq \mathbf{Var}(\lambda)$ and $\mathbf{C} \in \mathbf{Eqn}_\mathcal{A}$. We let $Ctrs_\mathcal{A}$ denote the set of abstract term rewriting systems.

In the following, we formalize the idea that abstract narrowing reduction approximates narrowing reduction by replacing concrete states, unification and term rewriting systems with abstract states, abstract unification and abstract term rewriting systems. We define the abstract most general unifier for an equation set $\mathbf{E}' \in \mathbf{Eqn}_{\mathcal{A}}$ as follows. First replace all occurrences of $\bot$ in $\mathbf{E}'$ by existentially quantified fresh variables. Then take a solved form of the resulting quantified equation set and finally replace the existentially quantified variables again by $\bot$. Formally: let $\exists \mathbf{y_1} \ldots \exists \mathbf{y_n}.\mathbf{E} = \mathbf{solve}([\![\mathbf{E}']\!])$ and $\kappa = \{\mathbf{y_1}/\bot, \ldots, \mathbf{y_n}/\bot\}$. Then $\mathbf{mgu}_{\mathcal{A}}(\mathbf{E}') = \mathbf{sub}(\mathbf{E}\kappa)$.

Example 4 *Let* $\mathbf{E} = \{\mathbf{f}(\mathbf{X}, \mathbf{X}) = \mathbf{f}(\mathbf{Y}, \bot),\ \mathbf{X} = \mathbf{a}\}$. *Then,* $\mathbf{mgu}_{\mathcal{A}}(\mathbf{E}) = \{\mathbf{X}/\mathbf{a},\ \mathbf{Y}/\mathbf{a}\}$.

We now extend the notion of parallel composition from substitutions to abstract substitutions by replacing unification by abstract unification.

Definition 5.5 *Let* $\kappa_1, \kappa_2 \in \mathbf{Sub}_{\mathcal{A}}$. *We define the abstract parallel composition* $\Uparrow_{\mathcal{A}}$: $\mathbf{Sub}_{\mathcal{A}} \times \mathbf{Sub}_{\mathcal{A}} \rightarrow \mathbf{Sub}_{\mathcal{A}}$ *by:*

$$\kappa_1 \ \Uparrow_{\mathcal{A}} \kappa_2 \ = \mathbf{mgu}_{\mathcal{A}}(\widehat{\kappa}_1 \cup \widehat{\kappa}_2).$$

It is straightforward to show that abstract parallel composition is idempotent, commutative, associative and has a null element **fail** and an identical element ϵ. $\Uparrow_{\mathcal{A}}$ can be lifted to sets of abstract substitutions in the obvious way.

Our notion of abstract term rewriting system is parametric with respect to a loop-check, i.e. a finite graph of functional dependencies built from the equational theory, which helps to recognize the narrowing derivations which definitely terminate. The purpose of a loop-check is to reduce the search space to end up with a finite search space.

Definition 5.6 (loop-check).
A loop-check is a graph $\mathcal{G}_{\mathcal{R}}$ *associated with a term rewriting system* $\mathcal{R}$, *i.e. a relation consisting of a set of pairs of terms, such that: (1) The transitive closure* $\mathcal{G}^+_{\mathcal{R}}$ *is decidable, and (2) There is a function* $\mathring{\mathbf{t}} = \mathbf{t}'$ *which assigns to a term* $\mathbf{t}$ *some node* $\mathbf{t}'$ *in* $\mathcal{G}_{\mathcal{R}}$ *such that, if there is an infinite sequence:*

$\langle \Leftarrow \mathbf{g}_0, \theta_0 \rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}_1, \theta_1 \rangle \rightsquigarrow \ldots$

then

$$\exists \mathbf{i} \geq 0.\ \langle \mathring{\mathbf{t}}_\mathbf{i}, \mathring{\mathbf{t}}_\mathbf{i} \rangle \in \mathcal{G}^+_{\mathcal{R}},\ \textit{where}\ \mathbf{t_i} = \mathbf{e}_{|\mathbf{u}}\theta_\mathbf{i},\ \mathbf{e} \in \mathbf{g_i}\ \textit{and}\ \mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}).$$

(we refer to $\langle \mathring{\mathbf{t}}_\mathbf{i}, \mathring{\mathbf{t}}_\mathbf{i} \rangle$ *as a 'cycle' of* $\mathcal{G}_{\mathcal{R}}$.*)*

A loop-check can be thought of as a sort of 'oracle' whose usefulness in proving the termination of basic narrowing derivations is stated in the following.

Remark 5.7 *Let* $\mathcal{R}$ *be a term rewriting system and* $\mathcal{G}_{\mathcal{R}}$ *be a loop-check for* $\mathcal{R}$. *If there is no cycle in* $\mathcal{G}_{\mathcal{R}}$, *then every basic conditional narrowing derivation for* $\mathcal{R}$ *terminates.*

To illustrate our definition, we consider a simple example here.

Example 5 *Let* $\mathcal{R} = \{\mathbf{X} + 0 \to \mathbf{X} \Leftarrow,\ \mathbf{X} + \mathbf{s}(\mathbf{Y}) \to \mathbf{s}(\mathbf{X} + \mathbf{Y}) \Leftarrow\}$ *and define* $\overset{\circ}{\mathbf{t}} = \mathbf{t}'$ *the (partial) function which, given a graph, assigns to a term* $\mathbf{t}$ *some node* $\mathbf{t}'$ *in the graph such that* $\mathbf{t}'$ *unifies with* $\mathbf{t}$*, if there is some such node. Variables are implicitly renamed to be disjoint. Then the graph* $\mathcal{G} = \{\langle \mathbf{X}+\mathbf{s}(\mathbf{Y}), \mathbf{X}+\mathbf{s}(\mathbf{Y})\rangle\}$ *is a loop-check for* $\mathcal{R}$*.*

Most papers on loop-checking consider the application of loop-checks at run-time. Static loop-checks have not received that much attention yet. [6, 36] consider a graph of terms which allows the detection of some loops in the search tree which do not lead to any solution. The graphs are built using information about the equations being narrowed as well as the reduction rules being used for narrowing. In the following we show how a loop-check which does not depend on the equation set to be solved can be used to obtain a form of (compiled) abstract program which always terminates and in which the semantics of a given goal can be approximated safely. A CTRS is abstracted by simplifying the right-hand side and the body of each rule. This definition is given inductively on the structure of terms and equations. The main idea is that terms which are mapped to a cycle of the loop-check are drastically simplified by replacing them by $\bot$. This enforces termination.

Definition 5.8 (abstract term rewriting system).
Let $\mathcal{R}$ *be a CTRS. Let* $\mathcal{G}_\mathcal{R}$ *be a loop-check for* $\mathcal{R}$*. We define the abstraction of* $\mathcal{R}$ *using* $\mathcal{G}_\mathcal{R}$ *as follows:*

$\mathcal{R}_\mathcal{A} = \{\lambda \to \mathbf{sh}(\rho) \Leftarrow \mathbf{sh}(\mathbf{C}) \mid \lambda \to \rho \Leftarrow \mathbf{C} \in \mathcal{R}\}$ *(we also write* $\mathcal{R}_\mathcal{A} \propto \mathcal{R}$*), where the shell* $\mathbf{sh}(\mathbf{x})$ *of an expression* $\mathbf{x}$ *is defined inductively*

$$\mathbf{sh}(\mathbf{x}) = \begin{cases} \mathbf{x} & \text{if } \mathbf{x} \in \mathbf{V} \\ \mathbf{f}(\mathbf{sh}(\mathbf{t}_1), \ldots, \mathbf{sh}(\mathbf{t_k})) & \text{if } \mathbf{x} = \mathbf{f}(\mathbf{t}_1, \ldots, \mathbf{t_k}) \text{ and } \langle \overset{\circ}{\mathbf{x}}, \overset{\circ}{\mathbf{x}} \rangle \notin \mathcal{G}_\mathcal{R}^+ \\ \mathbf{sh}(\mathbf{l}) = \mathbf{sh}(\mathbf{r}) & \text{if } \mathbf{x} = (\mathbf{l} = \mathbf{r}) \\ \mathbf{sh}(\mathbf{e}_1), \ldots, \mathbf{sh}(\mathbf{e_n}) & \text{if } \mathbf{x} = \mathbf{e}_1, \ldots, \mathbf{e_n} \\ \bot & \text{otherwise} \end{cases}$$

Example 6 *(Continued from Example 5) The abstraction of* $\mathcal{R}$ *using the loop-check* $\mathcal{G}$ *is:*
$\mathcal{R}_\mathcal{A} = \{\mathbf{X} + 0 \to \mathbf{X} \Leftarrow,\ \mathbf{X} + \mathbf{s}(\mathbf{Y}) \to \mathbf{s}(\bot) \Leftarrow\}$.

In the following definition, we consider sets of CTRSs defined over a fixed signature Σ and we see a CTRS as a finite set of rules. Thus, CTRSs can be ordered by set inclusion. We define the term rewriting system description as follows.

Definition 5.9 *Let* $(\mathbf{Ctrs}, \supseteq)$ *be the poset of term rewriting systems and* $(\mathbf{Ctrs}_\mathcal{A}, \supseteq)$ *be the poset of abstract term rewriting systems ordered by set inclusion. The abstract term rewriting system description is* $\langle(\mathbf{Ctrs}_\mathcal{A}, \supseteq), \gamma, (\mathbf{Ctrs}, \supseteq)\rangle$*, where* $\gamma : \mathbf{Ctrs}_\mathcal{A} \to \wp(\mathbf{Ctrs})$ *is defined by:*

$\gamma(\mathcal{R}_\mathcal{A}) = \{\mathcal{R} \in \mathbf{Ctrs} \mid (\lambda \to \rho_\mathcal{A} \Leftarrow \mathbf{C}_\mathcal{A}) \in \mathcal{R}_\mathcal{A} \ \wedge\ \rho_\mathcal{A} = \mathbf{sh}(\rho) \ \wedge\ \mathbf{C}_\mathcal{A} = \mathbf{sh}(\mathbf{C}) \ \wedge\ (\lambda \to \rho \Leftarrow \mathbf{C}) \in \mathcal{R}\}$.

We give an example which can help to understand these definitions and motivates the remainder of the section. Let us introduce the auxiliary function $\lfloor \mathbf{t} \rfloor$, which inductively replaces by a fresh variable any term whose outermost symbol is not an irreducible function symbol, i.e.

$$\lfloor \mathbf{t} \rfloor = \begin{cases} \mathbf{c}(\lfloor \mathbf{t}_1 \rfloor, \ldots, \lfloor \mathbf{t_k} \rfloor) & \text{if } \mathbf{t} = \mathbf{c}(\mathbf{t}_1, \ldots, \mathbf{t_k}) \text{ and } \mathbf{c} \in \mathcal{C} \\ \mathbf{y} & \text{otherwise, where } \mathbf{y} \text{ is a fresh variable.} \end{cases}$$

The following definition introduces a particular case of dependency graph $\mathcal{I}_\mathcal{R}$ that is 'induced' by a term rewriting system $\mathcal{R}$. Then we show that $\mathcal{I}_\mathcal{R}$ is a loop-check. Another different, less accurate, loop-check can be found in [2].

Definition 5.10 *Let $\mathcal{R}$ be a CTRS. The following transformation defines a directed graph $\mathcal{I}_\mathcal{R}$ of functional dependencies induced by $\mathcal{R}$. Define $\overline{\mathbf{t}} = \mathbf{f}(\lfloor \mathbf{t_1} \rfloor, \ldots, \lfloor \mathbf{t_n} \rfloor)$ if $\mathbf{t} = \mathbf{f}(\mathbf{t_1}, \ldots, \mathbf{t_n})$. To build $\mathcal{I}_\mathcal{R}$, the algorithm starts with $\langle \mathcal{R}, \emptyset \rangle$ and applies the inference rules as long as they add new arrows. The symbol $\cup$ stands for set union.*

$$(1)\quad \frac{\mathbf{r} = (\lambda \to \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}}{\langle \mathcal{R}, \mathcal{I}_\mathcal{R} \rangle \;\longmapsto\; \langle \mathcal{R} - \{\mathbf{r}\}, \mathcal{I}_\mathcal{R} \cup \{\lambda \xrightarrow{\mathcal{R}} \overline{\mathbf{t}} \mid (\mathbf{t} = \rho_{|\mathbf{u}} \wedge \mathbf{u} \in \bar{\mathbf{O}}(\rho)) \text{ or } (\mathbf{t} = \mathbf{e}_{|\mathbf{u}} \wedge \mathbf{e} \in \mathbf{C} \wedge \mathbf{u} \in (\bar{\mathbf{O}}(\mathbf{e}) - \{\Lambda\}))\}\rangle}$$

$$(2)\quad \frac{(\lambda \xrightarrow{\mathcal{R}} \mathbf{r}) \in \mathcal{I}_\mathcal{R} \;\wedge\; (\lambda' \xrightarrow{\mathcal{R}} \mathbf{r}') \in \mathcal{I}_\mathcal{R} \;\wedge\; \mathbf{r} \stackrel{?}{=} \lambda'}{\langle \mathcal{R}, \mathcal{I}_\mathcal{R} \rangle \;\longmapsto\; \langle \mathcal{R}, \mathcal{I}_\mathcal{R} \cup \{\mathbf{r} \xrightarrow{\mathbf{u}} \lambda'\}\rangle}$$

Termination of this calculus is ensured since the number of terms occurring in the rules in $\mathcal{R}$ is finite. Roughly speaking, in Definition 5.10, for each rule $(\lambda \to \rho \Leftarrow \mathbf{C})$ in $\mathcal{R}$ and for each term $\mathbf{f}(\mathbf{t_1}, \ldots, \mathbf{t_n})$ occurring in ρ or in $\mathbf{C}$, rule (1) adds an arrow $\lambda \xrightarrow{\mathcal{R}} \mathbf{f}(\lfloor \mathbf{t_1} \rfloor, \ldots, \lfloor \mathbf{t_n} \rfloor)$ to $\mathcal{I}_\mathcal{R}$. Rule (2) adds an arrow $\mathbf{r} \xrightarrow{\mathbf{u}} \lambda'$ between the right-hand side $\mathbf{r}$ of an arrow $\lambda \xrightarrow{\mathcal{R}} \mathbf{r}$ in $\mathcal{I}_\mathcal{R}$ and the left-hand side λ' of each arrow $\lambda' \xrightarrow{\mathcal{R}} \mathbf{r}'$ with which $\mathbf{r}$ unifies. In the following, $\to^*$ denotes a path in the graph that may contain arrows $\xrightarrow{\mathcal{R}}$ and arrows $\xrightarrow{\mathbf{u}}$. Note that $\mathcal{I}_\mathcal{R}$ associates a path with every basic narrowing derivation issued from a given goal and that it does not require the inspection of the equation set to be solved, as opposed to [6].

Also note that $\mathcal{I}_\mathcal{R}$ consists of the graph of top symbols in [2] when $\overline{\mathbf{t}}$ $(\overset{\circ}{\mathbf{t}})$ is defined as the function $\overline{\mathbf{t}} = \mathbf{f}$ if $\mathbf{t} = \mathbf{f}(\mathbf{t_1}, \ldots, \mathbf{t_n})$. In general, $\mathcal{I}_\mathcal{R}$ expresses more fine-grained representations of the recursive structure in a program. Our notion of loop-check $\mathcal{I}_\mathcal{R}$ resembles that of the U-graphs [41] of pure logic programming, which differ from the standard predicate dependency graphs in the sense that unification is also taken into account [10]. U-graphs contain a node for each call (each atom) in the program, and also contain two types of arrows: clause arrows and unification arrows. There is a clause arrow from atom $\mathbf{H}$ to atom $\mathbf{B}$, if $\mathbf{H}$ is the head of a clause and $\mathbf{B}$ is a body atom of the same clause. There is an unification arrow from atom $\mathbf{B}$ to atom $\mathbf{H}$ if $\mathbf{B}$ is a body atom which (after renaming) unifies with the head $\mathbf{H}$ of another (or the same) clause.

The following proposition shows that the induced dependency graph $\mathcal{I}_\mathcal{R}$ is a loop-check.

Proposition 5.11 *Let $\mathcal{R}$ be a CTRS and $\mathcal{I}_\mathcal{R}$ be the graph of terms induced by $\mathcal{R}$. Then $\mathcal{I}_\mathcal{R}$ is a loop-check.*

Example 7 *Let us consider the following level-canonical CTRS $\mathcal{R}$ and its abstraction $\mathcal{R}_\mathcal{A}$.*

$\mathcal{R} = \{$
- $\mathbf{r1)}$ $\mathbf{h}(0) \to 0.$
- $\mathbf{r2)}$ $\mathbf{f}(0) \to 0.$
- $\mathbf{r3)}$ $\mathbf{f}(\mathbf{c}(\mathbf{X})) \to \mathbf{c}(\mathbf{f}(\mathbf{X})) \Leftarrow \mathbf{g}(\mathbf{X}) = \mathbf{X}.$
- $\mathbf{r4)}$ $\mathbf{g}(\mathbf{c}(\mathbf{X})) \to \mathbf{c}(\mathbf{X}).$ $\}$

$\mathcal{R}_\mathcal{A} = \{$
- $\mathbf{r1}_\mathcal{A})$ $\mathbf{h}(0) \to 0.$
- $\mathbf{r2}_\mathcal{A})$ $\mathbf{f}(0) \to 0.$
- $\mathbf{r3}_\mathcal{A})$ $\mathbf{f}(\mathbf{c}(\mathbf{X})) \to \mathbf{c}(\bot) \Leftarrow \mathbf{g}(\mathbf{X}) = \mathbf{X}.$
- $\mathbf{r4}_\mathcal{A})$ $\mathbf{g}(\mathbf{c}(\mathbf{X})) \to \mathbf{c}(\mathbf{X}).$ $\}$

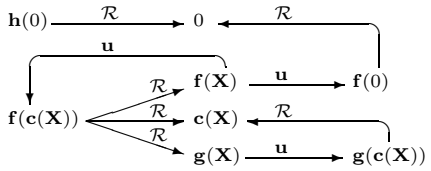

Figure 1: *Graph of functional dependencies*

We depict in Figure 1 the dependency graph induced by $\mathcal{R}$. *There is a cycle* $\mathbf{f}(\mathbf{X}) \rightarrow^* \mathbf{f}(\mathbf{X}) \rightarrow^* \ldots$ *in the graph.*

5.2 Abstract Compositional Narrowing

Now we are ready to formalize an abstract compositional operational semantics. We first introduce abstract (basic) narrowing.

In the sequel, for $\mathbf{e} \in \mathbf{Eqn}_{\mathcal{A}}$, we let $\bar{\mathbf{O}}(\mathbf{e})$ denote the set of the occurrences $\mathbf{u}$ of $\mathbf{e}$ s.t. $\mathbf{e}_{|\mathbf{u}} \notin (\mathbf{V} \cup \{\bot\})$.

Definition 5.12 *Let* $\mathcal{R}_{\mathcal{A}}$ *be an abstract* **CTRS**. *We define abstract (basic) narrowing as a transition system* $(\mathbf{State}_{\mathcal{A}}, \leadsto_{\mathcal{A}})$ *whose transition relation* $\leadsto_{\mathcal{A}} \subseteq \mathbf{State}_{\mathcal{A}} \times \mathbf{State}_{\mathcal{A}}$ *is defined as the smallest relation satisfying:*

$$\frac{\begin{array}{l}\mathbf{e} \in \mathbf{g} \quad \wedge \quad \mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}) \quad \wedge \quad (\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}_{\mathcal{A}} \ \wedge \\ \sigma = \mathbf{mgu}_{\mathcal{A}}(\{\mathbf{e}_{|\mathbf{u}} = \lambda\}) \quad \wedge \quad \kappa \Uparrow_{\mathcal{A}} \sigma \not\equiv \mathbf{fail}\end{array}}{\langle \Leftarrow \mathbf{g}, \kappa \rangle \quad \leadsto_{\mathcal{A}} \quad \langle \Leftarrow (\mathbf{g} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C}, \kappa \Uparrow_{\mathcal{A}} \sigma \rangle}$$

The following definition formalizes the *abstract basic narrowing semantics* for the success set.

Definition 5.13 (abstract semantics)

$$\Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}) = \{\kappa_{\restriction \mathbf{Var}(\mathbf{g})} \mid \langle \Leftarrow \mathbf{g}, \epsilon \rangle \leadsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \rangle\}.$$

The main purpose of introducing abstract basic narrowing here is to suggest a mechanism for the static analysis of the run-time behaviour of programs. We now establish some preliminary results that clarify our interest in abstract basic narrowing reduction. The following theorem can be proven as an immediate consequence of Lemma 6.3 of [2]. It basically states that in the abstract computations no solutions are lost, that is, each concrete computed answer is still 'represented' by a more general answer in the abstract semantics.

Theorem 5.14 *Let* $\mathcal{R}_{\mathcal{A}} \propto \mathcal{R}$ *and* $\mathbf{g}' \propto \mathbf{g}$. *Then, for every solution* $\theta \in \mathcal{O}_{\mathcal{R}}(\Leftarrow \mathbf{g})$ *there exists* $\kappa \in \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}')$ *such that* $\kappa \propto \theta$.

Abstract narrowing computes, for a given goal and a program, a description of a superset of the computed answers. Therefore, if the superset is empty, then the goal fails w.r.t. the program. Our analysis of unsatisfiability is formalized in the following theorem.

Corollary 5.15 *If* $\Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}) = \emptyset$, *then* $\mathbf{g}$ *is unsatisfiable in* $\mathcal{R}$.

The following theorem basically states that abstract narrowing is compositional w.r.t. the AND operator.

Theorem 5.16 $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}_1) \Uparrow_\mathcal{A} \Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}_2)$.

This theorem suggests the following method to extend abstract basic narrowing to a compositional rule.

Definition 5.17 *Let $\mathcal{R}_\mathcal{A}$ be an abstract* **CTRS**. *We define abstract compositional (basic) narrowing as a transition system $(\mathbf{State}_\mathcal{A}, \longmapsto_\mathcal{A})$ whose transition relation $\longmapsto_\mathcal{A} \subseteq \mathbf{State}_\mathcal{A} \times \mathbf{State}_\mathcal{A}$ is defined as the smallest relation satisfying*[2,3]*:*

(1) $$\frac{\langle \Leftarrow \mathbf{g}_1, \kappa\rangle \longmapsto^*_\mathcal{A} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_\mathcal{A} \kappa_1\rangle \ \wedge\ \langle \Leftarrow \mathbf{g}_2, \kappa\rangle \longmapsto^*_\mathcal{A} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_\mathcal{A} \kappa_2\rangle \wedge \kappa' = (\kappa_1 \Uparrow_\mathcal{A} \kappa_2) \ \wedge\ \kappa' \not\equiv \mathbf{fail}}{\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa\rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_\mathcal{A} \kappa'\rangle}$$

(2) $$\frac{\langle \Leftarrow \{\mathbf{e}\}, \kappa\rangle \rightsquigarrow_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle}{\langle \Leftarrow \{\mathbf{e}\}, \kappa\rangle \longmapsto_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle}$$

(3) select don't care (3.1) or (3.2):

(3.1) $$\frac{\langle \Leftarrow \mathbf{g}, \kappa\rangle \rightsquigarrow_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle}{\langle \Leftarrow \mathbf{g}, \kappa\rangle \longmapsto_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle}$$ (3.2) $$\frac{\langle \Leftarrow \mathbf{g}, \kappa\rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{g}', \kappa'\rangle}{\langle \Leftarrow \mathbf{g}, \kappa\rangle \longmapsto_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle}$$

Roughly speaking, the calculus in Definition 5.17 basically considers two cases: the equation set $\mathbf{g}$ is a singleton or not. In the first case (rule 2), abstract compositional narrowing just proceeds like abstract basic narrowing ($\rightsquigarrow_\mathcal{A}$). In the case when $\mathbf{g}$ has more than one equations (rule 3), $\mathbf{g}$ can be narrowed using abstract basic narrowing, or it can be split two ways, and each of the separate subgoals $\mathbf{g}_1$ and $\mathbf{g}_2$ can be solved independently (using the auxiliary arrow $\longmapsto_{\|}$, rule 1). The following propositions show the soundness and completeness of the compositional abstract narrower.

Proposition 5.18 (soundness).
If $\langle \Leftarrow \mathbf{g}, \kappa\rangle \longmapsto^*_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle$ *then* $\langle \Leftarrow \mathbf{g}, \kappa\rangle \rightsquigarrow^*_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa'\rangle$.

Proposition 5.19 (completeness).
If $\langle \Leftarrow \mathbf{g}, \kappa\rangle \rightsquigarrow^*_\mathcal{A} \langle \Leftarrow \mathbf{true}, \kappa'\rangle$ *then* $\langle \Leftarrow \mathbf{g}, \kappa\rangle \longmapsto^*_\mathcal{A} \langle \Leftarrow \mathbf{true}, \kappa'\rangle$.

Using these results, the compositionality of abstract basic narrowing can be easily lifted to the case of abstract compositional narrowing. The following result expresses this more formally.

Corollary 5.20 *Let* $\Delta'_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}) = \{\kappa_{|\mathbf{Var}(\mathbf{g})} \mid \langle \Leftarrow \mathbf{g}, \epsilon\rangle \longmapsto^*_\mathcal{A} \langle \Leftarrow \mathbf{true}, \kappa\rangle\}$. *Then*

$$\Delta'_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \Delta'_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}_1) \Uparrow_\mathcal{A} \Delta'_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}_2).$$

As a consequence of Proposition 5.18 and Proposition 5.19, the analysis for a specific goal (the abstract meaning of a goal) can be determined by exploiting the AND-compositionality of the basic narrowing semantics and its abstract version. In Section 6 we will formalize the idea that the compositionality of the abstract semantics w.r.t. the union of $\mathcal{E}$-unification problems, as established in Theorem 5.16 and Corollary 5.20, provides for incrementality when dealing with constraint satisfaction problems in the framework of constraint logic programming, where sets of constraints are incrementally added to a solver.

[2] For the sake of simplicity, in rules (3.1) and (3.2), $\mathbf{g}$ is not singleton and, in rule (1), $\mathbf{g}_1$ and $\mathbf{g}_2$ denote non empty equation sets.

[3] An implementation of rule (3) could always choose rule (3.1), or rule (3.2), or it could select arbitrarily one of them (but not the other) at each step.

6 Incremental Equational Analyzer

In the context of constraint logic programming [19, 23], incremental search consists of proving the solvability of a sequence of constraint problems by transforming the existing solution to each previously solved problem into a solution to the next problem [17].

When dealing with equational constraints [1], the tests of solvability can be extremely redundant. Termination is not even guaranteed. In [2] we propose a lazy resolution procedure [19] which incorporates an analysis of unsatisfiability which allows us to avoid some useless computations. To achieve efficiency, the analyses also need to be incremental, that is, when adding a new equation set $\widetilde{\mathbf{c}}$ to an already tested set $\mathbf{c}$ of constraints, the analysis should not start checking the accumulated constraint $\mathbf{c} \cup \widetilde{\mathbf{c}}$ from scratch. In this section, we formulate an incremental algorithm for analyzing the unsatisfiability of equation sets within a constraint setting [1]. The kernel of the algorithm is the calculus of abstract compositional (basic) narrowing reduction as formulated in Section 5.

We assume that constraints monotonically grow as long as the computation proceeds, and the question we consider is how to deal efficiently with the test of unsatisfiability for the accumulated constraints as long as new equations are added.

Definition 6.1 (incremental constraint satisfaction problem).
Let $\mathbf{c}_0, \mathbf{c}_1, \widetilde{\mathbf{c}}_1, \ldots, \mathbf{c}_\mathbf{n}, \widetilde{\mathbf{c}}_\mathbf{n}$ *be constraints, where* $\mathbf{c_i} = \mathbf{c_{i-1}} \cup \widetilde{\mathbf{c}}_\mathbf{i}$. *The incremental constraint satisfaction problem consists of (efficiently) checking the (un)satisfiability of* $\mathbf{c_i}$ *by using some information from the computations of* $\mathbf{c}_0, \ldots, \mathbf{c_{i-1}}$, $\mathbf{i} = 1, \ldots, \mathbf{n}$.

The idea here is to compute the abstract success set of $\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}}$ by combining the sets $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c})$ ($\Delta'_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c})$) and $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \widetilde{\mathbf{c}})$ ($\Delta'_{\mathcal{R}_\mathcal{A}}(\Leftarrow \widetilde{\mathbf{c}})$) which describe the successes of $\Leftarrow \mathbf{c}$ and $\Leftarrow \widetilde{\mathbf{c}}$, respectively.

We define an **i***ncremental* **E***quational* **A***nalyzer (*iEA*)* as follows.

Definition 6.2 *An* **iEA***-state is a pair* $\langle \mathbf{c}, \Theta \rangle$, *where* $\mathbf{c}$ *is a set of equations and* Θ *is a set of abstract substitutions. The empty* **iEA***-state is* $\langle \emptyset, \emptyset \rangle$.

Definition 6.3 *(*iEA transition relation $\xrightarrow{\tilde{\mathbf{c}}}_{\mathbf{iEA}}$*)*

$$\frac{\Theta' = \Theta \Uparrow_\mathcal{A} \Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \widetilde{\mathbf{c}})}{\langle \mathbf{c}, \Theta \rangle \quad \xrightarrow{\tilde{\mathbf{c}}}_{\mathbf{iEA}} \quad \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Theta' \rangle}$$

We note that, if the accumulated abstract success set $\Theta' = \emptyset$ then $\mathbf{c} \cup \widetilde{\mathbf{c}}$ is unsatisfiable by Corollary 5.15. Our strategy proves the unsatisfiability of $\mathbf{c} \cup \widetilde{\mathbf{c}}$, or it builds the (non-empty) abstract success set $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}})$, as stated by:

Theorem 6.4 *Let* $\mathbf{c}$ *be a constraint and* $\Theta = \Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c}) \neq \emptyset$. *Then,*

1. *if a transition* $\langle \mathbf{c}, \Theta \rangle \xrightarrow{\tilde{\mathbf{c}}}_{\mathbf{iEA}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Theta' \rangle$ *is proven, then* $\Theta' = \Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}})$;
2. *if a transition* $\langle \mathbf{c}, \Theta \rangle \xrightarrow{\tilde{\mathbf{c}}}_{\mathbf{iEA}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \emptyset \rangle$ *is proven, then the constraint* $\mathbf{c} \cup \widetilde{\mathbf{c}}$ *is unsatisfiable.*

We note that the computed abstract answer set $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}})$ can be used to guide the final execution of a 'full' narrower which can find the concrete solutions and possibly recognize the unsatisfiability not detected by this lazy procedure, as described in [3].

constraints	CAn	ICAn	AbNar	APCom	Speedup
`Z = 0, parity(X) = even`	0.52	0.54	0.42	0.00	
`Y + Z = s`^{2}`(0)`	3.38	1.62	0.62	0.86	
`Y = X + Z`	**10.62**	**5.24**	0.44	4.76	2
`parity(X) = even`	0.42	0.42	0.30	0.00	
`parity(Y) = even`	3.88	2.80	0.32	1.20	
`X + Y = s`^{2}`(0)`	**18.24**	**3.44**	0.44	2.92	5.3
`parity(X) = even`	0.40	0.40	0.28	0.00	
`Y = s(0), Z = X + Y`	3.78	1.52	0.46	0.66	
`X + Z = s`^{3}`(0)`	**26.36**	**4.44**	**fail**		5.9
`X + Y = s`^{4}`(0)`	0.72	0.72	0.54	0.00	
`parity(X) = even`	3.66	2.10	0.32	1.46	
`parity(Y) = even`	**19.48**	**3.64**	0.30	2.62	5.34
`parity(X) = parity(Y)`	3.62	3.66	2.30	0.00	
`X + Y = s(Z)`	23.18	12.00	0.46	9.52	
`Z = 0`	**20.26**	**1.14**	**fail**		17.8

CAn Constraint Analyzer
ICAn Incremental Constraint Analyzer
AbNar Abstract Narrowing
APCom Abstract Parallel Composition

Table 1: *Incremental* vs. *Non-Incremental* constraint analyzer times (secs, using BIM-Prolog, SUN 3/80).

6.1 Performance Results

The concepts presented regarding compositionality can be used in practice to obtain speedups with respect to the original, non-incremental, execution. The analysis above has been implemented in Prolog and tested on several programs with good results. To demonstrate this point empirically, in this section we present actual run-times for a simple benchmark (`parity/1`).

```
X + 0       →  X          ⇐
X + s(Y)    →  s(X + Y)   ⇐
parity(X)   →  even       ⇐  X = Y + Y
```

Table 1 gives execution times for the program running on a prototype sequential implementation and its optimized, incremental, version. We have not tried with the parallel interpreter yet. In order to test the relative performance of incremental and non-incremental analysis, we timed the analysis adding the equations one by one. That is, the analysis was first run for the first equation of each constraint benchmark. Then the next equation was added and the resulting accumulated constraint (re-)analyzed. The total time involved in this process is given in Table 1 by **ICAn**, for the case of incremental analysis, and by **CAn**, for the case of restarting the analysis from scratch every time an equation is added. We compare the time performances of the incremental vs. the non-incremental analyzers. The column **Speedup** (CAn/ICAn) shows the speedup obtained by incremental analysis. Given the current naïve implementation, the speedups are quite encouraging. If the incrementality was exploited, our interpreter was able to achieve up to 95% gain in efficiency. The time in the second column (ICAn) is the result of the sum of the time in the third and fourth columns: AbNar (time of Abstract Narrowing) and APCom (time of Abstract Parallel Composition) plus some extratime for some simplification rules which are only relevant for the implementation.

7 Conclusion and further research

The contribution of this paper is twofold. We have presented a formal compositional semantics for the success set of equational logic programs which is suitable for AND-parallel implementations. We have then shown that this semantics leads to compositional analyses and have given an example of an enhanced analysis of unsatisfiability which is suitable for theories where equations are considered as constraints.

The approach which we have taken is of general interest. In particular it applies to any kind of analysis where we look for properties which are satisfied by all (or some) success paths. For specific analyses, it will be necessary to provide the appropriate abstract domains and approximation of the term rewriting system. A groundness analysis which follows the approach proposed here is defined in [3].

Acknowledgements

We thank Catuscia Palamidessi for her detailed comments on a draft of this paper, and the anonymous referees for many useful suggestions. Part of this work was performed in the context of the CICYT project TIC 92-0793-C02-02. The final version of the paper was produced under partial support from CICYT project TIC 95-0433-C03-03.

References

[1] M. Alpuente, M. Falaschi, and G. Levi. Incremental Constraint Satisfaction for Equational Logic Programming. *Theoretical Computer Science*, 142(1):27–57, 1995.

[2] M. Alpuente, M. Falaschi, and F. Manzo. Analyses of Unsatisfiability for Equational Logic Programming. *Journal of Logic Programming*, 22(3):221–252, 1995.

[3] M. Alpuente, M. Falaschi, M.J. Ramis, and G. Vidal. Optimization of Equational Logic Programs Using Abstract Narrowing. Technical Report DSIC-II/1/93, UPV, 1993. Short version in J. Penjam and M. Bruynooghe, editors, *Proc. of PLILP'93*, volume 714 of *Lecture Notes in Computer Science*, pages 391-409, Springer-Verlag, Berlin, 1993.

[4] M. Alpuente, M. Falaschi, M.J. Ramis, and G. Vidal. A Compositional Semantics for Conditional Term Rewriting Systems. In H.E. Bal, editor, *Proc. Sixth IEEE Int'l Conf. on Computer Languages ICCL'94*, pages 171–182. IEEE Computer Society Press, 1994.

[5] M. Alpuente, M. Falaschi, and G. Vidal. Compositional Analysis for Equational Horn Programs. In G. Levi and M. Rodríguez-Artalejo, editors, *Proc. of Fifth Int'l Conf. on Algebraic and Logic Programming, ALP'94*, volume 850 of *Lecture Notes in Computer Science*, pages 77–94. Springer-Verlag, Berlin, 1994.

[6] J. Chabin and P. Réty. Narrowing directed by a graph of terms. In G. Goos and J. Hartmanis, editors, *Proc. of RTA'91*, volume 488 of *Lecture Notes in Computer Science*, pages 112–123. Springer-Verlag, Berlin, 1991.

[7] M. Codish, M. Falaschi, and K. Marriott. Suspension Analyses for Concurrent Logic Programs. *ACM Transactions on Programming Languages and Systems*, 16(3):649–686, 1994.

[8] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. Fourth ACM Symp. Principles of Programming Languages*, pages 238–252, 1977.

[9] F. S. de Boer and C. Palamidessi. On the asynchronous nature of communication in concurrent logic languages: A fully abstract model based on sequences. In J.C.M. Baeten and J.W. Klop, editors, *Proc. of Concur 90*, volume 458 of *Lecture Notes in Computer Science*, pages 175–194. Springer-Verlag, Berlin, 1990.

[10] D. de Schreye and S. Decorte. Termination of Logic Programs: the Never Ending Story. *Journal of Logic Programming*, 19/20:199–260, 1994.

[11] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 243–320. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.

[12] N. Dershowitz and N. Lindenstrauss. An Abstract Concurrent Machine for Rewriting. In H. Kirchner and W. Wechler, editors, *Proc. Second Int'l Conf. on Algebraic and Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 318–331. Springer-Verlag, Berlin, 1990.

[13] M. Gabbrielli and G. Levi. On the Semantics of Logic Programs. In J. Leach Albert, B. Monien, and M. Rodriguez Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium*, volume 510 of *Lecture Notes in Computer Science*, pages 1–19. Springer-Verlag, Berlin, 1991.

[14] J.H. Gallier and S. Raatz. Extending SLD-resolution to equational Horn clauses using E-unification. *Journal of Logic Programming*, 6:3–43, 1989.

[15] E. Giovannetti and C. Moiso. A completeness result for E-unification algorithms based on Conditional Narrowing. In M. Boscarol, L. Carlucci, and G. Levi, editors, *Foundations of Logic and Functional Programming*, volume 306 of *Lecture Notes in Computer Science*, pages 157–167. Springer-Verlag, Berlin, 1986.

[16] M. Hanus. The Integration of Functions into Logic Programming: From Theory to Practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

[17] P. Van Hentenryck and T. Le Provost. Incremental Search in Constraint Logic Programming. *New Generation Computing*, 9:257–275, 1991.

[18] M. Hermenegildo and F. Rossi. Strict and Non-Strict Independent And-Parallelism in Logic Programs: Correctness, Efficiency and Compile-Time Conditions. *Journal of Logic Programming*, 22(1):1–45, 1995.

[19] M. Höhfeld and G. Smolka. Definite relations over constraint languages. Technical report, IBM Deutschland GmbH, Stuttgart, 1988.

[20] S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 1989.

[21] J.M. Hullot. Canonical Forms and Unification. In *5th Int'l Conf. on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer-Verlag, Berlin, 1980.

[22] J.-M. Jacquet. *Conclog: A Methodological Approach to Concurrent Logic Programming*. PhD thesis, University of Namur, Belgium, 1989.

[23] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. In *Proc. Fourteenth Annual ACM Symp. on Principles of Programming Languages*, pages 111–119. ACM, 1987.

[24] J. Jaffar, J.-L. Lassez, and M.J. Maher. A theory of complete logic programs with equality. *Journal of Logic Programming*, 3:211–223, 1984.

[25] J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume I, pages 1–112. Oxford University Press, 1992.

[26] H. Kuchen and W. Hans. An AND-Parallel Implementation of the Functional Logic Language Babel. In *Aachener Informatik-Bericht*, volume 12, pages 119–139, RWTH Aachen, 1991.

[27] H. Kuchen, J.J. Moreno-Navarro, and M. Hermenegildo. Independent AND-Parallel Implementation of Narrowing. In M. Bruynooghe and M. Wirsing, editors, *Proc. of PLILP'92, Leuven (Belgium)*, volume 631 of *Lecture Notes in Computer Science*, pages 24–38. Springer-Verlag, Berlin, 1992.

[28] J.-L. Lassez, M. J. Maher, and K. Marriott. Unification Revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 587–625. Morgan Kaufmann, Los Altos, Ca., 1988.

[29] M. J. Maher. Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees. In *Proc. Third IEEE Symp. on Logic In Computer Science*, pages 348–357. Computer Science Press, New York, 1988.

[30] M. J. Maher. On parameterized substitutions. Technical Report RC 16042, IBM - T.J. Watson Research Center, Yorktown Heights, NY, 1990.

[31] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *AAECC*, 5:213–253, 1994.

[32] W. Nutt, P. Réty, and G. Smolka. Basic narrowing revisited. *Journal of Symbolic Computation*, 7:295–317, 1989.

[33] C. Palamidessi. Algebraic properties of idempotent substitutions. In M. S. Paterson, editor, *Proc. of the 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 386–399. Springer-Verlag, Berlin, 1990.

[34] G. Plotkin. A structured approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[35] U.S. Reddy. Narrowing as the Operational Semantics of Functional Languages. In *Proc. Second IEEE Int'l Symp. on Logic Programming*, pages 138–151. IEEE, 1985.

[36] P. Réty, C. Kirchner, H. Kirchner, and P. Lescanne. NARROWER: A new algorithm for unification and its applications to logic programming. In *Proc. of RTA'85*, volume 202 of *Lecture Notes in Computer Science*, pages 141–157. Springer-Verlag, Berlin, 1985.

[37] V. Saraswat, M. Rinard, and P. Panangaden. Semantic Foundation of Concurrent Constraint Programming. In *Proc. Eighteenth Annual ACM Symp. on Principles of Programming Languages POPL'91*, pages 333–352, 1991.

[38] D. Scott. Domains for Denotational Semantics. In *Proc. ICALP'82*, volume 140 of *Lecture Notes in Computer Science*, pages 577–613. Springer-Verlag, Berlin, 1982.

[39] J.H. Siekmann. Unification Theory. *Journal of Symbolic Computation*, 7:207–274, 1989.

[40] J.E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* MIT Press, 1977.

[41] B. Wang and R.K. Shyamasundar. Methodology for Proving the Termination of Logic Programs. In *Proc. of PLILP'90*, volume 456 of *Lecture Notes in Computer Science*, pages 204–221. Springer-Verlag, Berlin, 1990.

A Appendix

In this appendix we give the proofs of the theorems we previously stated. In doing so, we will also state and prove a number of technical results.

Proof of Proposition 3.3

Proposition 3.3 Let $\Gamma = \Gamma_1 \cup \Gamma_2$, $\Theta_1 = \mathcal{U}_\mathcal{E}(\Gamma_1)$ and $\Theta_2 = \mathcal{U}_\mathcal{E}(\Gamma_2)$. Then $\mathcal{U}_\mathcal{E}(\Gamma) = \Theta_1 \Uparrow_\mathcal{E} \Theta_2$.

Proof.

($\subseteq$) Let $\vartheta \in \mathcal{U}_\mathcal{E}(\Gamma_1 \cup \Gamma_2)$. Then $\vartheta \in \mathcal{U}_\mathcal{E}(\Gamma_1) = \Theta_1$ and $\vartheta \in \mathcal{U}_\mathcal{E}(\Gamma_2) = \Theta_2$. Thus $\vartheta \in \vartheta \Uparrow_\mathcal{E} \vartheta \subseteq \Theta_1 \Uparrow_\mathcal{E} \Theta_2$.

($\supseteq$) Let $\theta \in \Theta_1 \Uparrow_\mathcal{E} \Theta_2$. Then, by Definition 3.2, $\exists \theta_1 \in \Theta_1, \exists \theta_2 \in \Theta_2$ such that $\theta \in \mathcal{U}_\mathcal{E}(\widehat{\theta_1} \cup \widehat{\theta_2})$. Since

$$\theta \in \mathcal{U}_\mathcal{E}(\widehat{\theta_1}) \Rightarrow \theta \in \mathcal{U}_\mathcal{E}(\Gamma_1)$$
$$\theta \in \mathcal{U}_\mathcal{E}(\widehat{\theta_2}) \Rightarrow \theta \in \mathcal{U}_\mathcal{E}(\Gamma_2)$$

then $\theta \in \mathcal{U}_\mathcal{E}(\Gamma_1 \cup \Gamma_2)$.

□

Proof of Theorem 4.1

In order to prove Theorem 4.1 we need some technical lemmata.

Lemma A.1 [28, 33] *Let ϑ_1, ϑ_2 be idempotent substitutions. Then*

$$\vartheta_1 \Uparrow \vartheta_2 = \vartheta_1 \mathbf{mgu}(\widehat{\vartheta_2}\vartheta_1) = \vartheta_2 \mathbf{mgu}(\widehat{\vartheta_1}\vartheta_2).$$

Lemma A.2 *Let θ, σ be idempotent substitutions. Then $\theta \leq (\theta \Uparrow \sigma)$.*

Proof. [proof of Lemma A.2]
By Lemma A.1, $\theta \Uparrow \sigma = \theta \mathbf{mgu}(\widehat{\sigma}\theta) \geq \theta$. □

Lemma A.3 *Let θ, σ be idempotent substitutions. If $\mathbf{Var}(\sigma) \cap \mathbf{Dom}(\theta) = \emptyset$ then $\theta \Uparrow \sigma = \theta\sigma$.*

Proof. [proof of Lemma A.3]
We have the following equalities.

$\theta \Uparrow \sigma$	=	(by Lemma A.1)
$\theta \mathbf{mgu}(\widehat{\sigma}\theta)$	=	(since $\mathbf{Var}(\sigma) \cap \mathbf{Dom}(\theta) = \emptyset$)
$\theta \mathbf{mgu}(\widehat{\sigma})$	=	(since σ is idempotent)
$\theta\sigma$.		

□

Lemma A.4 *Let θ, θ' be idempotent substitutions. Then $\theta \leq \theta' \Rightarrow \mathbf{e}\theta' = \mathbf{e}\theta\theta'$.*

Proof. [proof of Lemma A.4]
Since $\theta \leq \theta'$ then there exists γ such that $\theta' = \theta\gamma$. Hence $\mathbf{e}\theta\theta' = \mathbf{e}\theta\theta\gamma = \mathbf{e}\theta\gamma = \mathbf{e}\theta'$. □

Lemma A.5 *Let* $\mathbf{e}$ *be a set of equations and* θ *be an idempotent substitution. Then* $\mathbf{mgu}(\mathbf{e}\theta) = \mathbf{mgu}(\widehat{\mathbf{mgu}}(\mathbf{e})\theta)$.

PROOF. [proof of Lemma A.5]
$\mathbf{mgu}(\mathbf{e}\theta) = \mathbf{mgu}(\mathbf{e} \wedge \widehat{\theta})_{\restriction \mathbf{Var(e)}} = \mathbf{mgu}(\widehat{\mathbf{mgu}}(\mathbf{e}) \wedge \widehat{\mathbf{mgu}}(\widehat{\theta}))_{\restriction \mathbf{Var(e)}} = \mathbf{mgu}(\widehat{\mathbf{mgu}}(\mathbf{e})\theta)$. □

Lemma A.6
$\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$ *iff*
$\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ *and* $\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{g}'_2, \sigma\theta_2\rangle$,
$\theta = \theta_1 \Uparrow \theta_2 \not\equiv \mathbf{fail}$ *and* $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

PROOF. [proof of Lemma A.6]

($\Rightarrow$) Let $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \equiv \langle \Leftarrow \mathbf{h}_1, \sigma\vartheta_1\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{h_n}, \sigma\vartheta_\mathbf{n}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$, $(\vartheta_1 \equiv \epsilon)$. The proof is done by induction on the length $\mathbf{n}$ of the derivation.

(n=1) Straightforward.

Let us consider the inductive case.
If $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \equiv \langle \Leftarrow \mathbf{h}_1, \sigma\vartheta_1\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{h_n}, \sigma\vartheta_\mathbf{n}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}', \sigma\vartheta_\mathbf{n}\delta\rangle \equiv \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$,
then there exist $(\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}$, $\mathbf{e} \in \mathbf{h_n}$ and $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$ such that $\delta = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_\mathbf{n}\}) \not\equiv \mathbf{fail}$ and $\mathbf{g}' = (\mathbf{h_n} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C}$.

By the inductive hypothesis, there exist
$\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{11}, \sigma\vartheta_{11}\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{g_{1m}}, \sigma\vartheta_{1\mathbf{m}}\rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{21}, \sigma\vartheta_{21}\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{g_{2k}}, \sigma\vartheta_{2\mathbf{k}}\rangle$, $(\vartheta_{11} \equiv \vartheta_{21} \equiv \epsilon)$,
such that $\vartheta_\mathbf{n} = \vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}} \not\equiv \mathbf{fail}$ and $\mathbf{h_n} = \mathbf{g_{1m}} \cup \mathbf{g_{2k}}$.
Since $\mathbf{h_n} = \mathbf{g_{1m}} \cup \mathbf{g_{2k}}$ then $\mathbf{e} \in \mathbf{g_{1m}}$ or $\mathbf{e} \in \mathbf{g_{2k}}$. Let $\mathbf{e} \in \mathbf{g_{1m}}$ (the case when $\mathbf{e} \in \mathbf{g_{2k}}$ is perfectly analogous). It suffices to show that $\langle \Leftarrow \mathbf{g_{1m}}, \sigma\vartheta_{1\mathbf{m}}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ and $\theta = \theta_1 \Uparrow \vartheta_{2\mathbf{k}} \not\equiv \mathbf{fail}$ and $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g_{2k}}$.

Since $\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma(\vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}})\}) = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_\mathbf{n}\}) = \delta \not\equiv \mathbf{fail}$, then
$\delta' = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\}) \not\equiv \mathbf{fail}$. Therefore $\langle \Leftarrow \mathbf{g_{1m}}, \sigma\vartheta_{1\mathbf{m}}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{m}}\delta'\rangle \equiv \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$, where $\theta_1 = \vartheta_{1\mathbf{m}}\delta'$ and $\mathbf{g}'_1 = (\mathbf{g_{1m}} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C}$. Since $\theta_1 = \vartheta_{1\mathbf{m}}\delta'$ then

$\theta_1 \Uparrow \vartheta_{2\mathbf{k}}$	=	
$(\vartheta_{1\mathbf{m}}\delta') \Uparrow \vartheta_{2\mathbf{k}}$	=	(by Lemma A.3, since $\mathbf{Var}(\delta') \cap \mathbf{Dom}(\vartheta_{1\mathbf{m}}) = \emptyset$)
$\vartheta_{1\mathbf{m}} \Uparrow \delta' \Uparrow \vartheta_{2\mathbf{k}}$	=	(by commutativity of $\Uparrow$)
$\vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}} \Uparrow \delta'$	=	(since $\vartheta_\mathbf{n} = \vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}}$)
$\vartheta_\mathbf{n} \Uparrow \delta'$	=	(by Lemma A.1)
$\vartheta_\mathbf{n}\mathbf{mgu}(\widehat{\delta'}\vartheta_\mathbf{n})$	=	
$\vartheta_\mathbf{n}\mathbf{mgu}(\widehat{\mathbf{mgu}}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\})\vartheta_\mathbf{n})$	=	(by Lemma A.5)
$\vartheta_\mathbf{n}\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\}\vartheta_\mathbf{n})$	=	
$\vartheta_\mathbf{n}\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\vartheta_\mathbf{n}\})$	=	(since $\vartheta_\mathbf{n} = \vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}}$)
$\vartheta_\mathbf{n}\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}(\vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}})\})$	=	(by Lemma A.4)
$\vartheta_\mathbf{n}\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma(\vartheta_{1\mathbf{m}} \Uparrow \vartheta_{2\mathbf{k}})\})$	=	
$\vartheta_\mathbf{n}\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_\mathbf{n}\})$	=	
$\vartheta_\mathbf{n}\delta$	=	
θ	$\not\equiv$	**fail**

Finally we prove that $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}_{2\mathbf{k}}$. Since $\mathbf{h}_\mathbf{n} = \mathbf{g}_{1\mathbf{m}} \cup \mathbf{g}_{2\mathbf{k}}$ then $\mathbf{g}' = (\mathbf{h}_\mathbf{n} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C} = ((\mathbf{g}_{1\mathbf{m}} \cup \mathbf{g}_{2\mathbf{k}}) - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C} = \mathbf{g}'_1 \cup \mathbf{g}_{2\mathbf{k}}$.

($\Leftarrow$) Let $\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{10}, \sigma\vartheta_{10}\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{g}_{1\mathbf{m}}, \sigma\vartheta_{1\mathbf{m}}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{m}}\delta\rangle \equiv \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{20}, \sigma\vartheta_{20}\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{g}_{2\mathbf{k}}, \sigma\vartheta_{2\mathbf{k}}\rangle \equiv \langle \Leftarrow \mathbf{g}'_2, \sigma\theta_2\rangle$, $(\vartheta_{10} \equiv \vartheta_{20} \equiv \epsilon)$,
and $(\mathbf{m}+1) + \mathbf{k} = \mathbf{n}$, $\mathbf{n} \geq 1$.
The proof is done by induction on the sum $\mathbf{n}$ of the lengths of the derivations.

(n=1) Straightforward.

Let us consider the inductive case. Since $\langle \Leftarrow \mathbf{g}_{1\mathbf{m}}, \sigma\vartheta_{1\mathbf{m}}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{m}}\delta\rangle$, then there exist
$(\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}$, $\mathbf{e} \in \mathbf{g}_{1\mathbf{m}}$ and $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$ such that $\delta = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\})$ and
$\mathbf{g}'_1 = (\mathbf{g}_{1\mathbf{m}} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C}$.
By the inductive hypothesis, there exists a derivation $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \rightsquigarrow^* \langle \Leftarrow (\mathbf{g}_{1\mathbf{m}}, \mathbf{g}'_2), \sigma(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\rangle$.
Then, it suffices to show that $\langle \Leftarrow (\mathbf{g}_{1\mathbf{m}}, \mathbf{g}'_2), \sigma(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\rangle \rightsquigarrow \langle \Leftarrow (\mathbf{g}'_1, \mathbf{g}'_2), \sigma(\theta_1 \Uparrow \theta_2)\rangle$.

Since

$(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\})$	$=$	(by Lemma A.2 and Lemma A.4)
$(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\})$	$=$	(by Lemma A.5)
$(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\mathbf{mgu}(\widehat{\mathbf{mgu}}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\})(\vartheta_{1\mathbf{m}} \Uparrow \theta_2))$	$=$	(by Lemma A.1)
$\vartheta_{1\mathbf{m}} \Uparrow \theta_2 \Uparrow \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma\vartheta_{1\mathbf{m}}\})$	$=$	
$\vartheta_{1\mathbf{m}} \Uparrow \theta_2 \Uparrow \delta$	$=$	(by commutativity of $\Uparrow$)
$\vartheta_{1\mathbf{m}} \Uparrow \delta \Uparrow \theta_2$	$=$	(by Lemma A.3, since $\mathbf{Var}(\delta) \cap \mathbf{Dom}(\vartheta_{1\mathbf{m}}) = \emptyset$)
$(\vartheta_{1\mathbf{m}}\delta) \Uparrow \theta_2$	$=$	
$\theta_1 \Uparrow \theta_2$	$\not\equiv$	**fail**,

then $\delta = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\sigma(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\}) \not\equiv \mathbf{fail}$, and thus
$\langle \Leftarrow (\mathbf{g}_{1\mathbf{m}}, \mathbf{g}'_2), \sigma(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\rangle \rightsquigarrow \langle \Leftarrow (\mathbf{g}'_1, \mathbf{g}'_2), \sigma(\vartheta_{1\mathbf{m}} \Uparrow \theta_2)\delta\rangle \equiv \langle \Leftarrow (\mathbf{g}'_1, \mathbf{g}'_2), \sigma(\theta_1 \Uparrow \theta_2)\rangle$, which is as desired.

□

We let $\mid \mathcal{D} \mid$ denote the length of the derivation $\mathcal{D}$. We have the following property.

Lemma A.7 *Let $\mathbf{g}_1$ and $\mathbf{g}_2$ be non empty equation sets and $\mathcal{D} \equiv \langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{true}, \sigma\theta\rangle$.
Then there exist $\mathcal{D}_1 \equiv \langle \Leftarrow \mathbf{g}_1, \sigma\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{true}, \sigma\theta_1\rangle$ and $\mathcal{D}_2 \equiv \langle \Leftarrow \mathbf{g}_2, \sigma\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{true}, \sigma\theta_2\rangle$,
with $\theta = \theta_1 \Uparrow \theta_2 \not\equiv \mathbf{fail}$, $\mid \mathcal{D} \mid > \mid \mathcal{D}_1 \mid$ and $\mid \mathcal{D} \mid > \mid \mathcal{D}_2 \mid$.*

Proof. [proof of Lemma A.7]
The proof is done along the lines of the proof of Lemma A.6. Routine. □

Now we can prove the desired result.

Theorem 4.1 $\mathcal{O}(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \mathcal{O}(\Leftarrow \mathbf{g}_1) \Uparrow \mathcal{O}(\Leftarrow \mathbf{g}_2)$.

Proof.
The result is a particular case of Lemma A.6 for $\sigma \equiv \epsilon$ and $\mathbf{g}'_1 \equiv \mathbf{g}'_2 \equiv \mathbf{true}$. □

Proof of Theorem 4.4

To prove Theorem 4.4 we first need the following lemmata. ⋆

separo el lemma en dos porque en esta direccion ⇒ las longitudes no tienen porque coincidir (aunque en la prueba no se considera el caso en que $\mathbf{g}'_1$ o $\mathbf{g}'_2$ sean vacios...)

Lemma A.8
If $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \longmapsto^* \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$ *then*
$\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \longmapsto^* \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ *and* $\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \longmapsto^* \langle \Leftarrow \mathbf{g}'_2, \sigma\theta_2\rangle$,
where $\theta = \theta_1 \Uparrow \theta_2 \not\equiv \mathbf{fail}$ *and* $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

Proof.
Let $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \equiv \langle \Leftarrow \mathbf{h}_1, \sigma\vartheta_1\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{h_n}, \sigma\vartheta_\mathbf{n}\rangle \longmapsto \langle \Leftarrow \mathbf{g}', \sigma\vartheta_\mathbf{n}\delta\rangle \equiv \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$, $(\vartheta_1 \equiv \epsilon)$, and assume $\mathbf{h_n} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k_n}} \{\mathbf{e_i}\}$, $\mathbf{k_n} \geq 1$. The proof is done by induction on the length of the derivation.

Let $\mathbf{n} = 1$ and consider the sets $\mathbf{I}_1 = \{\mathbf{i} \mid \mathbf{e_i} \in \mathbf{g}_1\} = \{\mathbf{i}_1, \ldots, \mathbf{i_m}\}$ and $\mathbf{I}_2 = \{\mathbf{j} \mid \mathbf{e_j} \in \mathbf{g}_2\} = \{\mathbf{j}_1, \ldots, \mathbf{j_p}\}$, $\mathbf{I}_1 \subseteq \{1, \ldots, \mathbf{k}\}$, $\mathbf{I}_2 \subseteq \{1, \ldots, \mathbf{k}\}$, $\mathbf{I}_1 \cap \mathbf{I}_2 = \emptyset$ and $\mathbf{I}_1 \cup \mathbf{I}_2 = \{1, \ldots, \mathbf{k}\}$, with $\mathbf{k} = \mathbf{k}_1$. If $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \longmapsto \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$ then, by rules (1) and (2) of Definition 4.2, there exist $(\lambda_\mathbf{i} \to \rho_\mathbf{i} \Leftarrow \mathbf{C_i}) \ll \mathcal{R}$, $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$ **and** $\delta_\mathbf{i} = \mathbf{mgu}(\{\lambda_\mathbf{i} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\sigma\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \{1, \ldots, \mathbf{k}\}$ such that $\langle \Leftarrow \{\mathbf{e_i}\}, \sigma\rangle \longmapsto \langle \Leftarrow \{\mathbf{e_i}[\rho_\mathbf{i}]_\mathbf{u_i}\} \cup \mathbf{C_i}, \sigma\delta_\mathbf{i}\rangle \equiv \langle \Leftarrow \mathbf{c_i}, \sigma\delta_\mathbf{i}\rangle$, $\mathbf{g}' = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \mathbf{c_i}$ and $\theta = \delta_1 \Uparrow \ldots \Uparrow \delta_\mathbf{k} \not\equiv \mathbf{fail}$. Then, $\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \longmapsto \langle \Leftarrow \bigcup_{\mathbf{i}\in\mathbf{I}_1} \mathbf{c_i}, \sigma\theta_1\rangle \equiv \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ and $\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \longmapsto \langle \Leftarrow \bigcup_{\mathbf{i}\in\mathbf{I}_2} \mathbf{c_i}, \sigma\theta_2\rangle \equiv \langle \Leftarrow \mathbf{g}'_2, \sigma\theta_2\rangle$, $\theta_1 = \delta_{\mathbf{i}_1} \Uparrow \ldots \Uparrow \delta_{\mathbf{i_m}}$, $\theta_2 = \delta_{\mathbf{j}_1} \Uparrow \ldots \Uparrow \delta_{\mathbf{j_p}}$, $\theta_1 \Uparrow \theta_2 = \theta$ and $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

Now we consider the inductive case. By the inductive hypothesis, there exist
$\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{11}, \sigma\vartheta_{11}\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{g}_{1\mathbf{n}}, \sigma\vartheta_{1\mathbf{n}}\rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{21}, \sigma\vartheta_{21}\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{g}_{2\mathbf{n}}, \sigma\vartheta_{2\mathbf{n}}\rangle$, $(\vartheta_{11} \equiv \vartheta_{21} \equiv \epsilon)$
such that $\vartheta_\mathbf{n} = \vartheta_{1\mathbf{n}} \Uparrow \vartheta_{2\mathbf{n}} \not\equiv \mathbf{fail}$ and $\mathbf{h_n} = \mathbf{g}_{1\mathbf{n}} \cup \mathbf{g}_{2\mathbf{n}}$. Let $\mathbf{I}_1 = \{\mathbf{i} \mid \mathbf{e_i} \in \mathbf{g}_{1\mathbf{n}}\} = \{\mathbf{i}_1, \ldots, \mathbf{i_m}\}$ and $\mathbf{I}_2 = \{\mathbf{j} \mid \mathbf{e_j} \in \mathbf{g}_{2\mathbf{n}}\} = \{\mathbf{j}_1, \ldots, \mathbf{j_p}\}$, $\mathbf{I}_1 \subseteq \{1, \ldots, \mathbf{k_n}\}$, $\mathbf{I}_2 \subseteq \{1, \ldots, \mathbf{k_n}\}$, $\mathbf{I}_1 \cap \mathbf{I}_2 = \emptyset$ and $\mathbf{I}_1 \cup \mathbf{I}_2 = \{1, \ldots, \mathbf{k_n}\}$.

Now it suffices to show that $\langle \Leftarrow \mathbf{g}_{1\mathbf{n}}, \sigma\vartheta_{1\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{n}}\delta'\rangle$ and $\langle \Leftarrow \mathbf{g}_{2\mathbf{n}}, \sigma\vartheta_{2\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_2, \sigma\vartheta_{2\mathbf{n}}\delta''\rangle$ and $\vartheta_\mathbf{n}\delta = (\vartheta_{1\mathbf{n}}\delta') \Uparrow (\vartheta_{2\mathbf{n}}\delta'')$ and $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

If $\langle \Leftarrow \mathbf{h_n}, \sigma\vartheta_\mathbf{n}\rangle \longmapsto \langle \Leftarrow \mathbf{g}', \sigma\vartheta_\mathbf{n}\delta\rangle$ then there exist $(\lambda_\mathbf{i} \to \rho_\mathbf{i} \Leftarrow \mathbf{C_i}) \ll \mathcal{R}$, $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$ and $\delta_\mathbf{i} = \mathbf{mgu}(\{\lambda_\mathbf{i} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\sigma\vartheta_\mathbf{n}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \{1, \ldots, \mathbf{k_n}\}$ such that $\langle \Leftarrow \{\mathbf{e_i}\}, \sigma\vartheta_\mathbf{n}\rangle \longmapsto \langle \Leftarrow \{\mathbf{e_i}[\rho_\mathbf{i}]_\mathbf{u_i}\} \cup \mathbf{C_i}, \sigma\vartheta_\mathbf{n}\delta_\mathbf{i}\rangle \equiv \langle \Leftarrow \mathbf{c_i}, \sigma\vartheta_\mathbf{n}\delta_\mathbf{i}\rangle$, $\mathbf{g}' = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k_n}} \mathbf{c_i}$ and $\delta = \delta_1 \Uparrow \ldots \Uparrow \delta_{\mathbf{k_n}} \not\equiv \mathbf{fail}$.

Since $\delta_\mathbf{i} = \mathbf{mgu}(\{\lambda_\mathbf{i} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\sigma\vartheta_\mathbf{n}\}) \not\equiv \mathbf{fail}$ and $\vartheta_\mathbf{n} = \vartheta_{1\mathbf{n}} \Uparrow \vartheta_{2\mathbf{n}}$ then there exist $\delta'_\mathbf{i} = \mathbf{mgu}(\{\lambda_\mathbf{i} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\sigma\vartheta_{1\mathbf{n}}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \mathbf{I}_1$ and $\delta''_\mathbf{i} = \mathbf{mgu}(\{\lambda_\mathbf{i} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\sigma\vartheta_{2\mathbf{n}}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \mathbf{I}_2$. Therefore, by an argument similar to the basic case, $\langle \Leftarrow \mathbf{g}_{1\mathbf{n}}, \sigma\vartheta_{1\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{n}}\delta'\rangle$ and $\langle \Leftarrow \mathbf{g}_{2\mathbf{n}}, \sigma\vartheta_{2\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_2, \sigma\vartheta_{2\mathbf{n}}\delta''\rangle$, $\delta' = \delta'_{\mathbf{i}_1} \Uparrow \ldots \Uparrow \delta'_{\mathbf{i_m}}$, $\delta'' = \delta''_{\mathbf{j}_1} \Uparrow \ldots \Uparrow \delta''_{\mathbf{j_p}}$ and $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

Finally, we prove that $(\vartheta_{1\mathbf{n}}\delta') \Uparrow (\vartheta_{2\mathbf{n}}\delta'') = \vartheta_\mathbf{n}\delta$.

$(\vartheta_{1\mathbf{n}}\delta') \Uparrow (\vartheta_{2\mathbf{n}}\delta'')$ = (by Lemma A.3, since $\mathbf{Var}(\delta') \cap \mathbf{Dom}(\vartheta_{1\mathbf{n}}) = \emptyset$)

$\vartheta_{1\mathbf{n}} \Uparrow \delta' \Uparrow (\vartheta_{2\mathbf{n}}\delta'')$ = (by Lemma A.3, since $\mathbf{Var}(\delta'') \cap \mathbf{Dom}(\vartheta_{2\mathbf{n}}) = \emptyset$)

$\vartheta_{1\mathbf{n}} \Uparrow \delta' \Uparrow \vartheta_{2\mathbf{n}} \Uparrow \delta''$ = (by commutativity of $\Uparrow$)

$\vartheta_{1\mathbf{n}} \Uparrow \vartheta_{2\mathbf{n}} \Uparrow \delta' \Uparrow \delta''$ =

$\vartheta_{\mathbf{n}} \Uparrow \delta' \Uparrow \delta''$ =

$\vartheta_{\mathbf{n}} \Uparrow (\delta'_{\mathbf{i}_1} \Uparrow \ldots \Uparrow \delta'_{\mathbf{i_m}}) \Uparrow (\delta''_{\mathbf{j}_1} \Uparrow \ldots \Uparrow \delta''_{\mathbf{j_p}})$ =

$(\vartheta_{\mathbf{n}} \Uparrow \delta'_{\mathbf{i}_1}) \Uparrow \ldots \Uparrow (\vartheta_{\mathbf{n}} \Uparrow \delta'_{\mathbf{i_m}}) \Uparrow (\vartheta_{\mathbf{n}} \Uparrow \delta''_{\mathbf{j}_1}) \Uparrow \ldots \Uparrow (\vartheta_{\mathbf{n}} \Uparrow \delta''_{\mathbf{j_p}})$ = (by Lemma A.1)

$\vartheta_{\mathbf{n}}\mathbf{mgu}(\widehat{\delta'_{\mathbf{i}_1}\vartheta_{\mathbf{n}}}) \Uparrow \ldots \Uparrow \vartheta_{\mathbf{n}}\mathbf{mgu}(\widehat{\delta''_{\mathbf{j_p}}\vartheta_{\mathbf{n}}})$ =

$\vartheta_{\mathbf{n}}\mathbf{mgu}(\widehat{\mathbf{mgu}}(\{\mathbf{e}_{\mathbf{i}_1|\mathbf{u}_{\mathbf{i}_1}}\sigma\vartheta_{1\mathbf{n}}\})\vartheta_{\mathbf{n}}) \Uparrow \ldots$
$\ldots \Uparrow \vartheta_{\mathbf{n}}\mathbf{mgu}(\widehat{\mathbf{mgu}}(\{\mathbf{e}_{\mathbf{j_p}|\mathbf{u}_{\mathbf{j_p}}}\sigma\vartheta_{2\mathbf{n}}\})\vartheta_{\mathbf{n}})$ = (by Lemma A.5)

$\vartheta_{\mathbf{n}}\mathbf{mgu}(\{\mathbf{e}_{\mathbf{i}_1|\mathbf{u}_{\mathbf{i}_1}}\sigma\vartheta_{1\mathbf{n}}\}\vartheta_{\mathbf{n}}) \Uparrow \ldots \Uparrow \vartheta_{\mathbf{n}}\mathbf{mgu}(\{\mathbf{e}_{\mathbf{j_p}|\mathbf{u}_{\mathbf{j_p}}}\sigma\vartheta_{2\mathbf{n}}\}\vartheta_{\mathbf{n}})$ =

$\vartheta_{\mathbf{n}}\mathbf{mgu}(\{\mathbf{e}_{\mathbf{i}_1|\mathbf{u}_{\mathbf{i}_1}}\sigma\vartheta_{1\mathbf{n}}\vartheta_{\mathbf{n}}\}) \Uparrow \ldots \Uparrow \vartheta_{\mathbf{n}}\mathbf{mgu}(\{\mathbf{e}_{\mathbf{j_p}|\mathbf{u}_{\mathbf{j_p}}}\sigma\vartheta_{2\mathbf{n}}\vartheta_{\mathbf{n}})\}$ = (by Lemmata A.2 and A.4)

$\vartheta_{\mathbf{n}}\mathbf{mgu}(\{\mathbf{e}_{\mathbf{i}_1|\mathbf{u}_{\mathbf{i}_1}}\sigma\vartheta_{\mathbf{n}}\}) \Uparrow \ldots \Uparrow \vartheta_{\mathbf{n}}\mathbf{mgu}(\{\mathbf{e}_{\mathbf{j_p}|\mathbf{u}_{\mathbf{j_p}}}\sigma\vartheta_{\mathbf{n}})\}$ =

$(\vartheta_{\mathbf{n}}\delta_{\mathbf{i}_1}) \Uparrow \ldots \Uparrow (\vartheta_{\mathbf{n}}\delta_{\mathbf{i_m}}) \Uparrow (\vartheta_{\mathbf{n}}\delta_{\mathbf{j}_1}) \Uparrow \ldots \Uparrow (\vartheta_{\mathbf{n}}\delta_{\mathbf{j_p}})$ = (by Lemma A.3, since $\mathbf{Var}(\delta_{\mathbf{i}}) \cap \mathbf{Dom}(\vartheta_{\mathbf{n}}) = \emptyset$)

$(\vartheta_{\mathbf{n}} \Uparrow \delta_{\mathbf{i}_1}) \Uparrow \ldots \Uparrow (\vartheta_{\mathbf{n}} \Uparrow \delta_{\mathbf{i_m}}) \Uparrow (\vartheta_{\mathbf{n}} \Uparrow \delta_{\mathbf{j}_1}) \Uparrow \ldots \Uparrow (\vartheta_{\mathbf{n}} \Uparrow \delta_{\mathbf{j_p}})$ =

$\vartheta_{\mathbf{n}} \Uparrow (\delta_{\mathbf{i}_1} \Uparrow \ldots \Uparrow \delta_{\mathbf{i_m}} \Uparrow \delta_{\mathbf{j}_1} \Uparrow \ldots \Uparrow \delta_{\mathbf{j_p}})$ =

$\vartheta_{\mathbf{n}} \Uparrow \delta$ = (by Lemma A.3, since $\mathbf{Var}(\delta) \cap \mathbf{Dom}(\vartheta_{\mathbf{n}}) = \emptyset$)

$\vartheta_{\mathbf{n}}\delta$,
and therefore the thesis follows. □

Lemma A.9
If $\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \longmapsto^{\mathbf{n}} \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ *and* $\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \longmapsto^{\mathbf{n}} \langle \Leftarrow \mathbf{g}'_2, \sigma\theta_2\rangle$ *then*
$\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \longmapsto^{\mathbf{n}} \langle \Leftarrow \mathbf{g}', \sigma\theta\rangle$,
where $\theta = \theta_1 \Uparrow \theta_2 \not\equiv \mathbf{fail}$ *and* $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

Proof. Let $\langle \Leftarrow \mathbf{g}_1, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{11}, \sigma\vartheta_{11}\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{g}_{1\mathbf{n}}, \sigma\vartheta_{1\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{n}}\delta'\rangle \equiv \langle \Leftarrow \mathbf{g}'_1, \sigma\theta_1\rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \sigma\rangle \equiv \langle \Leftarrow \mathbf{g}_{21}, \sigma\vartheta_{21}\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{g}_{2\mathbf{n}}, \sigma\vartheta_{2\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_2, \sigma\vartheta_{2\mathbf{n}}\delta''\rangle \equiv \langle \Leftarrow \mathbf{g}'_2, \sigma\theta_2\rangle$,
$(\vartheta_{11} \equiv \vartheta_{21} \equiv \epsilon)$,
such that $\theta_1 \Uparrow \theta_2 \not\equiv \mathbf{fail}$, and assume $\mathbf{g}_{1\mathbf{n}} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{q_n}} \{\mathbf{e}_{\mathbf{i}}\}$, $\mathbf{q_n} \geq 1$ and $\mathbf{g}_{2\mathbf{n}} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{p_n}} \{\mathbf{e}_{\mathbf{i}}\}$, $\mathbf{p_n} \geq 1$.

We prove the claim by induction on the length of the derivations.

Let $\mathbf{n} = 1$. It follows trivially by rule (2) of Definition 4.2.

Now we consider the inductive case. By the inductive hypothesis, there exist a derivation
$\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \sigma\rangle \longmapsto^{\mathbf{n}-1} \langle \Leftarrow (\mathbf{g}_{1\mathbf{n}}, \mathbf{g}_{2\mathbf{n}}), \sigma(\vartheta_{1\mathbf{n}} \Uparrow \vartheta_{2\mathbf{n}})\rangle$.
Then it suffices to show that $\langle \Leftarrow (\mathbf{g}_{1\mathbf{n}}, \mathbf{g}_{2\mathbf{n}}), \sigma(\vartheta_{1\mathbf{n}} \Uparrow \vartheta_{2\mathbf{n}})\rangle \longmapsto \langle \Leftarrow (\mathbf{g}'_1, \mathbf{g}'_2), \sigma(\theta_1 \Uparrow \theta_2)\rangle$.
If $\langle \Leftarrow \mathbf{g}_{1\mathbf{n}}, \sigma\vartheta_{1\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g}'_1, \sigma\vartheta_{1\mathbf{n}}\delta'\rangle$ then there exist $(\lambda_{\mathbf{i}} \to \rho_{\mathbf{i}} \Leftarrow \mathbf{C}_{\mathbf{i}}) \ll \mathcal{R}$, $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$, $\mathbf{e_i} \in \mathbf{g}_{1\mathbf{n}}$ and $\delta'_{\mathbf{i}} = \mathbf{mgu}(\{\lambda_{\mathbf{i}} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\sigma\vartheta_{1\mathbf{n}}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \{1, \ldots, \mathbf{q_n}\}$ such that $\langle \Leftarrow \{\mathbf{e_i}\}, \sigma\vartheta_{1\mathbf{n}}\rangle \longmapsto \langle \Leftarrow \{\mathbf{e_i}[\rho_{\mathbf{i}}]_{\mathbf{u_i}}\} \cup \mathbf{C_i}, \sigma\vartheta_{1\mathbf{n}}\delta'_{\mathbf{i}}\rangle \equiv$

$\langle \Leftarrow \mathbf{c'_i}, \sigma\vartheta_{\mathbf{1n}}\delta'_{\mathbf{i}}\rangle$, $\mathbf{g'_1} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{q_n}} \mathbf{c'_i}$ and $\delta' = \delta'_1 \Uparrow \ldots \Uparrow \delta'_{\mathbf{q_n}} \not\equiv \mathbf{fail}$. Analogously, if $\langle \Leftarrow \mathbf{g_{2n}}, \sigma\vartheta_{\mathbf{2n}}\rangle \longmapsto \langle \Leftarrow \mathbf{g'_2}, \sigma\vartheta_{\mathbf{2n}}\delta''\rangle$ then there exist $(\lambda_{\mathbf{i}} \to \rho_{\mathbf{i}} \Leftarrow \mathbf{C_i}) \ll \mathcal{R}$, $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$, $\mathbf{e_i} \in \mathbf{g_{2n}}$ and $\delta''_{\mathbf{i}} = \mathbf{mgu}(\{\lambda_{\mathbf{i}} = \mathbf{e_{i|u_i}}\sigma\vartheta_{\mathbf{2n}}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \{1,\ldots,\mathbf{p_n}\}$ such that $\langle \Leftarrow \{\mathbf{e_i}\}, \sigma\vartheta_{\mathbf{2n}}\rangle \longmapsto \langle \Leftarrow \{\mathbf{e_i}[\rho_{\mathbf{i}}]_{\mathbf{u_i}}\} \cup \mathbf{C_i}, \sigma\vartheta_{\mathbf{2n}}\delta''_{\mathbf{i}}\rangle \equiv \langle \Leftarrow \mathbf{c''_i}, \sigma\vartheta_{\mathbf{2n}}\delta''_{\mathbf{i}}\rangle$, $\mathbf{g'_2} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{p_n}} \mathbf{c''_i}$ and $\delta'' = \delta''_1 \Uparrow \ldots \Uparrow \delta''_{\mathbf{p_n}} \not\equiv \mathbf{fail}$.

Assume that $\mathbf{g_{1n}} \cup \mathbf{g_{2n}} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k_n}} \mathbf{e_i}$. Now suppose that, for all $\mathbf{i}$ in $\{1,\ldots,\mathbf{k_n}\}$, there exist $(\lambda_{\mathbf{i}} \to \rho_{\mathbf{i}} \Leftarrow \mathbf{C_i}) \ll \mathcal{R}$ and $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$ such that $\delta_{\mathbf{i}} = \mathbf{mgu}(\{\lambda_{\mathbf{i}} = \mathbf{e_{i|u_i}}\sigma\vartheta_{\mathbf{n}}\})$, $\delta = \delta_1 \Uparrow \ldots \Uparrow \delta_{\mathbf{k_n}}$. By a similar argument as in the proof of Lemma A.8, $\vartheta_{\mathbf{n}}(\delta_1 \Uparrow \ldots \Uparrow \delta_{\mathbf{k_n}}) = (\vartheta_{\mathbf{1n}}\delta') \Uparrow (\vartheta_{\mathbf{2n}}\delta'') = \theta_1 \Uparrow \theta_2$ and then we have
$\langle \Leftarrow (\mathbf{g_{1n}}, \mathbf{g_{2n}}), \sigma\vartheta_{\mathbf{n}}\rangle \longmapsto \langle \Leftarrow (\bigcup_{\mathbf{i}=1,\ldots,\mathbf{q_n}} \mathbf{c'_i}, \bigcup_{\mathbf{i}=1,\ldots,\mathbf{p_n}} \mathbf{c''_i}), \sigma\vartheta_{\mathbf{n}}(\delta_1 \Uparrow \ldots \Uparrow \delta_{\mathbf{k_n}})\rangle \equiv \langle \Leftarrow (\mathbf{g'_1}, \mathbf{g'_2}), \sigma(\theta_1 \Uparrow \theta_2)\rangle$, which gives the claim.

□

The following result is now immediate.

Theorem 4.4 $\mathcal{O}'(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \mathcal{O}'(\Leftarrow \mathbf{g}_1) \Uparrow \mathcal{O}'(\Leftarrow \mathbf{g}_2)$.

Proof.
The result is a particular case of Lemma A.8 for $\sigma \equiv \epsilon$ and $\mathbf{g'_1} \equiv \mathbf{g'_2} \equiv \mathbf{true}$ and Lemma A.9 for $\sigma \equiv \epsilon$, $\mathbf{g'_1} \equiv \mathbf{g'_2} \equiv \mathbf{true}$ and considering as many transitions $\langle \Leftarrow \mathbf{true}, \theta\rangle \longmapsto \langle \Leftarrow \mathbf{true}, \theta\rangle$ as needed to make the length of the (independent) derivations coincide. □

Proof of Corollary 4.5 and Corollary 4.6

The following corollary shows that there is a correspondence between (successful) basic conditional narrowing derivations and compositional conditional narrowing derivations. By this correspondence, we get an indirect proof of the completeness of compositional conditional narrowing.

Corollary 4.5 $\mathcal{O}(\Leftarrow \mathbf{g}) = \mathcal{O}'(\Leftarrow \mathbf{g})$.

Proof.

(⊆) We prove that, for every successful basic narrowing sequence $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{true}, \sigma\vartheta_{\mathbf{m}}\rangle$ such that $\vartheta_{\mathbf{m}} \not\equiv \mathbf{fail}$ there exists a corresponding successful compositional conditional narrowing sequence $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{true}, \sigma\theta_{\mathbf{n_m}}\rangle$ such that $\vartheta_{\mathbf{m}} = \theta_{\mathbf{n_m}}$. The proof is done by induction on the length $\mathbf{m}$ of the former derivation. For the sake of simplicity, we assume that the clause renaming chosen for basic conditional narrowing reduction is the same as in compositional narrowing reduction.

Let $\mathbf{m} = 1$. If $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \rightsquigarrow \langle \Leftarrow \mathbf{true}, \sigma\vartheta_1\rangle$, then the rule $(\mathbf{x} = \mathbf{x} \to \mathbf{true} \Leftarrow)$ has been applied, and there exists an equation $\mathbf{e}$ such that $\mathbf{g}_0 = \{\mathbf{e}\}$ and $\vartheta_1 = \mathbf{mgu}(\{\mathbf{e}\sigma\})$. By rule (1) of Definition 4.2,

$\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \equiv \langle \Leftarrow \{\mathbf{e}\}, \sigma\rangle \longmapsto \langle \Leftarrow \mathbf{true}, \sigma\theta\rangle$, with $\theta = \mathbf{mgu}(\{\mathbf{e}\sigma\}) = \vartheta_1$.

Now we consider the inductive case. If $\mathbf{m} > 1$ then $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}_1, \sigma\vartheta_1\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{true}, \sigma\vartheta_{\mathbf{m}}\rangle$, with $\vartheta_{\mathbf{m}} \not\equiv \mathbf{fail}$. We consider two cases:

(i) Let $\mathbf{g}_0 = \{\mathbf{e}\}$. By the inductive hypothesis, there exists $\langle \Leftarrow \mathbf{g}_1, \sigma\vartheta_1\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{true}, \sigma\theta_{\mathbf{n_m}}\rangle$ such that $\vartheta_{\mathbf{m}} = \theta_{\mathbf{n_m}}$. Since $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}_1, \sigma\vartheta_1\rangle$ and $\mathbf{g}_0 = \{\mathbf{e}\}$, then by rule (1) of Definition 4.2, $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \longmapsto \langle \Leftarrow \mathbf{g}_1, \sigma\theta_1\rangle$ with $\theta_1 = \vartheta_1$.

(ii) Let $\mathbf{g}_0 = \mathbf{g}'_0 \cup \mathbf{g}''_0$, where $\mathbf{g}'_0$ and $\mathbf{g}''_0$ are non empty equation sets. Then, by Lemma A.7, there exist $\mathcal{D}_1 \equiv \langle \Leftarrow \mathbf{g}'_0, \sigma\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{true}, \sigma\vartheta'_{\mathbf{i}}\rangle$ and $\mathcal{D}_2 \equiv \langle \Leftarrow \mathbf{g}''_0, \sigma\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{true}, \sigma\vartheta''_{\mathbf{k}}\rangle$ such that $\vartheta_{\mathbf{m}} = \vartheta'_{\mathbf{i}} \Uparrow \vartheta''_{\mathbf{k}}$, $\vartheta_{\mathbf{m}} \not\equiv \mathbf{fail}$, $\mathbf{m} > \mid \mathcal{D}_1 \mid$ and $\mathbf{m} > \mid \mathcal{D}_2 \mid$. By the inductive hypothesis, there exist $\langle \Leftarrow \mathbf{g}'_0, \sigma\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{true}, \sigma\theta'_{\mathbf{n_i}}\rangle$ and $\langle \Leftarrow \mathbf{g}''_0, \sigma\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{true}, \sigma\theta''_{\mathbf{n_k}}\rangle$ such that $\vartheta'_{\mathbf{i}} = \theta'_{\mathbf{n_i}}$ and $\vartheta''_{\mathbf{k}} = \theta''_{\mathbf{n_k}}$. Finally, from Lemma A.8 ($\Leftarrow$), there exists the derivation $\langle \Leftarrow \mathbf{g}_0, \sigma\rangle \equiv \langle \Leftarrow (\mathbf{g}'_0, \mathbf{g}''_0), \sigma\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{true}, \sigma\theta_{\mathbf{n_m}}\rangle$ such that $\theta_{\mathbf{n_m}} = \theta'_{\mathbf{n_i}} \Uparrow \theta''_{\mathbf{n_k}} = \vartheta'_{\mathbf{i}} \Uparrow \vartheta''_{\mathbf{k}} = \vartheta_{\mathbf{m}}$.

($\supseteq$) We prove the more general assertion that, for every compositional conditional narrowing sequence $\langle \Leftarrow \mathbf{g}_0, \epsilon\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{g_n}, \theta_{\mathbf{n}}\rangle$ such that $\theta_{\mathbf{n}} \not\equiv \mathbf{fail}$, there exists a corresponding basic conditional narrowing sequence $\langle \Leftarrow \mathbf{g}'_0, \epsilon\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{g}'_{\mathbf{m_n}}, \vartheta_{\mathbf{m_n}}\rangle$ such that $\mathbf{g}'_{\mathbf{m_n}} = \mathbf{g_n}$ and $\vartheta_{\mathbf{m_n}} = \theta_{\mathbf{n}}$. The proof is done by induction on the length $\mathbf{n}$ of the former derivation. We make the same assumption about clauses renaming as above.

Let $\mathbf{n} = 1$ and assume $\mathbf{g}_0 = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \{\mathbf{e_i}\}$, $\mathbf{k} \geq 1$. If $\langle \Leftarrow \mathbf{g}_0, \epsilon\rangle \longmapsto \langle \Leftarrow \mathbf{g}_1, \theta_1\rangle$ then, by rules (1) and (2) of Definition 4.2, there exist $(\lambda_{\mathbf{i}} \to \rho_{\mathbf{i}} \Leftarrow \mathbf{C_i}) \ll \mathcal{R}$, $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$ **and** $\sigma_{\mathbf{i}} = \mathbf{mgu}(\{\lambda_{\mathbf{i}} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \{1, \ldots, \mathbf{k}\}$ such that $\langle \Leftarrow \{\mathbf{e_i}\}, \epsilon\rangle \longmapsto \langle \Leftarrow \{\mathbf{e_i}[\rho_{\mathbf{i}}]_{\mathbf{u_i}}\} \cup \mathbf{C_i}, \sigma_{\mathbf{i}}\rangle \equiv \langle \Leftarrow \mathbf{c_i}, \sigma_{\mathbf{i}}\rangle$ and $\mathbf{g}_1 = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \mathbf{c_i}$, $\theta_1 = \sigma_1 \Uparrow \ldots \Uparrow \sigma_{\mathbf{k}}$, $\theta_1 \not\equiv \mathbf{fail}$. Then by the definition of basic conditional narrowing and Lemma A.6 ($\Leftarrow$), $\langle \Leftarrow \mathbf{g}_0, \epsilon\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{g}_{\mathbf{m}_1}, \vartheta_{\mathbf{m}_1}\rangle$, $\mathbf{g}_{\mathbf{m}_1} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \mathbf{c_i} = \mathbf{g}_1$ and $\vartheta_{\mathbf{m}_1} = \sigma_1 \Uparrow \ldots \Uparrow \sigma_{\mathbf{k}} = \theta_1$.

Let us consider the inductive case. If $\mathbf{n} > 1$ then $\langle \Leftarrow \mathbf{g}_0, \epsilon\rangle \longmapsto \ldots \longmapsto \langle \Leftarrow \mathbf{g_{n-1}}, \theta_{\mathbf{n}-1}\rangle \longmapsto \langle \Leftarrow \mathbf{g_n}, \theta_{\mathbf{n}}\rangle$, with $\theta_{\mathbf{n}} \not\equiv \mathbf{fail}$, and by the inductive hypothesis, there exists $\langle \Leftarrow \mathbf{g}'_0, \epsilon\rangle \rightsquigarrow \ldots \rightsquigarrow \langle \Leftarrow \mathbf{g}'_{\mathbf{m_{n-1}}}, \vartheta_{\mathbf{m_{n-1}}}\rangle$ such that $\mathbf{g}'_{\mathbf{m_{n-1}}} = \mathbf{g_{n-1}}$ and $\vartheta_{\mathbf{m_{n-1}}} = \theta_{\mathbf{n}-1}$. Assume that $\mathbf{g_{n-1}} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \{\mathbf{e_i}\}$, $\mathbf{k} \geq 1$. If $\langle \Leftarrow \mathbf{g_{n-1}}, \theta_{\mathbf{n}-1}\rangle \longmapsto \langle \Leftarrow \mathbf{g_n}, \theta_{\mathbf{n}}\rangle$ then, by rules (1) and (2) of Definition 4.2, there exist $(\lambda_{\mathbf{i}} \to \rho_{\mathbf{i}} \Leftarrow \mathbf{C_i}) \ll \mathcal{R}$, $\mathbf{u_i} \in \bar{\mathbf{O}}(\mathbf{e_i})$ and $\sigma_{\mathbf{i}} = \mathbf{mgu}(\{\lambda_{\mathbf{i}} = \mathbf{e}_{\mathbf{i}|\mathbf{u_i}}\theta_{\mathbf{n}-1}\}) \not\equiv \mathbf{fail}$, $\mathbf{i} \in \{1, \ldots, \mathbf{k}\}$ such that $\langle \Leftarrow \{\mathbf{e_i}\}, \theta_{\mathbf{n}-1}\rangle \longmapsto \langle \Leftarrow \{\mathbf{e_i}[\rho_{\mathbf{i}}]_{\mathbf{u_i}}\} \cup \mathbf{C_i}, \theta_{\mathbf{n}-1}\sigma_{\mathbf{i}}\rangle \equiv \langle \Leftarrow \mathbf{c_i}, \theta_{\mathbf{n}-1}\sigma_{\mathbf{i}}\rangle$, $\mathbf{g_n} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \mathbf{c_i}$ and $\theta_{\mathbf{n}} = \theta_{\mathbf{n}-1}(\sigma_1 \Uparrow \ldots \Uparrow \sigma_{\mathbf{k}}) \not\equiv \mathbf{fail}$. Then, by the definition of basic conditional narrowing, $\langle \Leftarrow \{\mathbf{e_i}\}, \theta_{\mathbf{n}-1}\rangle \rightsquigarrow \langle \Leftarrow \mathbf{c_i}, \theta_{\mathbf{n}-1}\sigma_{\mathbf{i}}\rangle$ and, by Lemma A.6 ($\Leftarrow$), $\langle \Leftarrow \mathbf{g}'_{\mathbf{m_{n-1}}}, \vartheta_{\mathbf{m_{n-1}}}\rangle \rightsquigarrow^* \langle \Leftarrow \mathbf{g}'_{\mathbf{m_n}}, \vartheta_{\mathbf{m_n}}\rangle$, $\mathbf{g}_{\mathbf{m_n}} = \bigcup_{\mathbf{i}=1,\ldots,\mathbf{k}} \mathbf{c_i} = \mathbf{g_n}$ and $\vartheta_{\mathbf{m_n}} = \theta_{\mathbf{n}-1}(\sigma_1 \Uparrow \ldots \Uparrow \sigma_{\mathbf{k}}) = \theta_{\mathbf{n}}$, as desired.

□

Corollary 4.6 (completeness).

Let $\mathcal{E}$ be a level-canonical equational Horn theory. The set $\mathcal{O}'(\Leftarrow \mathbf{g})$ is a complete set of $\mathcal{E}$-unifiers of $\mathbf{g}$.

PROOF.
Immediate from Corollary 4.5. □

Proof of Proposition 5.11

Proposition 5.11 Let $\mathcal{R}$ be a CTRS and $\mathcal{I}_{\mathcal{R}}$ be the graph of terms induced by $\mathcal{R}$. Then $\mathcal{I}_{\mathcal{R}}$ is a loop-check.

PROOF.
The proof is based on a well-founded complexity measure on goals whose is decreased by narrowing steps. Consider the infinite derivation:

$\mathcal{D} \equiv \langle \Leftarrow \mathbf{g}_0, \theta_0 \rangle \rightsquigarrow \langle \Leftarrow \mathbf{g}_1, \theta_1 \rangle \rightsquigarrow \ldots$

and assume, by contradiction, that $\mathcal{I}_{\mathcal{R}}$ has no cycle associated with any of the terms occurring in $\mathcal{D}$. For $\mathbf{t} \in \mathcal{T}$, let $\mathbf{m}(\mathbf{t})$ denote the maximal length of the paths in $\mathcal{I}_{\mathcal{R}}$ starting from any node $\mathbf{t}'$ in the left-hand side of an arrow $\stackrel{\mathcal{R}}{\rightarrow}$ such that $\mathbf{t}'$ unifies with $\overline{\mathbf{t}}$ (variables are implicitly renamed to be disjoint). Let $\overline{\mathbf{t}}$ be one of such nodes, with a path which length is $\mathbf{m}(\mathbf{t})$. We associate with each $\langle \Leftarrow \mathbf{g_i}, \theta_\mathbf{i} \rangle$ in the derivation a set $\mathcal{H}_\mathbf{i} = \{\langle \mathbf{e}, \mathbf{u}, \mathbf{n} \rangle \mid \mathbf{e} \in \mathbf{g_i}, \mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}), \mathbf{e}[\mathbf{u}] \in \mathcal{F},\ \mathbf{n} = \mathbf{m}(\mathbf{e}_{|\mathbf{u}}\theta_\mathbf{i})\}$, $\mathbf{i} \geq 0$. We define the complexity $\mathcal{M}_\mathbf{i}$ of the goal $\langle \Leftarrow \mathbf{g_i}, \theta_\mathbf{i} \rangle$ as the finite multiset of natural numbers consisting of the third components of the triples in the set $\mathcal{H}_\mathbf{i}$. Since $\mathcal{I}_{\mathcal{R}}$ has no cycle associated with any term occurring in $\mathcal{D}$, these third components are finite for all $\mathbf{i} \geq 0$.

Let us define a well-founded total ordering $<_{\mathbf{mul}}$ over multiset complexities by extending the well-founded ordering $<$ on $\mathbf{N}$ to the set $\mathbf{M}(\mathbf{N})$ of finite multisets over $\mathbf{N}$. The set $\mathbf{M}(\mathbf{N})$ is well-founded under the ordering $<_{\mathbf{mul}}$ since $\mathbf{N}$ is well-founded under $<$ [11, 25]. Let $\mathcal{M}, \mathcal{M}'$ be multiset complexities. $\mathcal{M} <_{\mathbf{mul}} \mathcal{M}' \leftrightarrow \exists \mathbf{X} \subseteq \mathcal{M}, \mathbf{X}' \subseteq \mathcal{M}'$ such that $\mathcal{M} = (\mathcal{M}' - \mathbf{X}') \cup \mathbf{X}$ and $\forall \mathbf{n} \in \mathbf{X}\ \exists \mathbf{n}' \in \mathbf{X}'.\ \mathbf{n} < \mathbf{n}'$.

From the definition of basic conditional narrowing, at each derivation step $\langle \Leftarrow \mathbf{g_i}, \theta_\mathbf{i} \rangle \rightsquigarrow \langle \Leftarrow \mathbf{g_{i+1}}, \theta_{\mathbf{i}+1} \rangle$, the selected occurrence either comes from the nonvariable occurrences of the equations in $\mathbf{g}_0$ or it comes from the right-hand side or from the condition of a rewrite rule used in a previous step. Let $\mathbf{e}$ and $\mathbf{u}$ denote, respectively, the selected equation and occurrence in $\mathbf{g_i}$. By definition of $\mathcal{I}_{\mathcal{R}}$, for all $\mathbf{i}$, $\mathbf{i} \geq 0$, $\forall \langle \mathbf{e}', \mathbf{u}', \mathbf{n}' \rangle \in (\mathcal{H}_{\mathbf{i}+1} - \mathcal{H}_\mathbf{i})$ there exists $\mathbf{l} \stackrel{\mathcal{R}}{\rightarrow} \mathbf{r} \in \mathcal{I}_{\mathcal{R}}$ such that $\mathbf{l} \stackrel{?}{=} \mathbf{e}_{|\mathbf{u}}\theta_\mathbf{i}$ and $\mathbf{r} \stackrel{?}{=} \overline{\mathbf{e}'_{|\mathbf{u}'}}$. Since there exists $\mathbf{r} \stackrel{\mathbf{u}}{\rightarrow} \lambda'$, for each λ' such that $\mathbf{r} \stackrel{?}{=} \lambda'$, then $\mathbf{m}(\mathbf{e}_{|\mathbf{u}}\theta_\mathbf{i}) = 2 + \mathbf{m}(\mathbf{e}'_{|\mathbf{u}'})$ and thus $\forall \langle \mathbf{e}', \mathbf{u}', \mathbf{n}' \rangle \in (\mathcal{H}_{\mathbf{i}+1} - \mathcal{H}_\mathbf{i})$. $\mathbf{n}' < \mathbf{m}(\mathbf{e}_{|\mathbf{u}}\theta_\mathbf{i})$. Hence, by the definition of the ordering $<_{\mathbf{mul}}$, it is immediate that for all $\mathbf{i}$, $\mathbf{i} \geq 0$, $\mathcal{M}_{\mathbf{i}+1} <_{\mathbf{mul}} \mathcal{M}_\mathbf{i}$. From the definition of basic conditional narrowing, if there is a $\mathbf{k} > 0$ such that every element in $\mathcal{M}_\mathbf{k}$ is 0 then no more narrowing steps are possible. Thus, by the well-foundness of the multiset ordering $<_{\mathbf{mul}}$ over $\mathbf{M}(\mathbf{N})$, only finite decreasing chains are possible and the narrowing derivation terminates. □

Proof of Theorem 5.14 and Corollary 5.15

Theorem 5.14 can easily be proved as a consequence of the following lemmata.

Lemma A.10 [2] *Let $\mathbf{g}, \mathbf{g}'$ be (unquantified) finite sets of equations over $\tau(\Sigma \cup \mathbf{V})$ and $\tau(\Sigma \cup \mathbf{V} \cup \{\bot\})$, respectively. Let $\theta \in \mathbf{Sub}$ and $\theta' \in \mathbf{Sub}_\mathcal{A}$. If $\theta' \propto \theta$, and $\mathbf{g}' \propto \mathbf{g}$, then $\mathbf{mgu}_\mathcal{A}(\mathbf{g}' \cup \widehat{\theta'}) \propto \mathbf{mgu}(\mathbf{g} \cup \widehat{\theta})$.*

Lemma A.11 [2] *Let* $\mathbf{s} \equiv \langle \Leftarrow \mathbf{g}, \theta \rangle$, $\mathbf{s}' \equiv \langle \Leftarrow \mathbf{g}', \theta' \rangle$ *and* $\mathbf{s}' \propto \mathbf{s}$. *Assume that* $\mathbf{s} \leadsto \mathbf{t}$ *(by a basic narrowing step), and let* $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$ *be the occurrence in* $\mathbf{e} \in \mathbf{g}$ *selected for this step. If* $\nexists\, \mathbf{e}' \in \mathbf{g}'$ *such that* $\mathbf{e}' \propto \mathbf{e}$ *and* $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}')$, *then* $\mathbf{s}' \propto \mathbf{t}$.

Lemma A.12 *Let* $\mathbf{s}' \propto \mathbf{s}$. *If there is a (basic) narrowing reduction from* $\mathbf{s}$ *to* $\mathbf{t}$, *then there is an abstract narrowing reduction from* $\mathbf{s}'$ *to some* $\mathbf{t}'$ *such that* $\mathbf{t}' \propto \mathbf{t}$, *or* $\mathbf{s}' \propto \mathbf{t}$.

Proof. The proof is similar to the proof of Lemma 6.3 of [2].
Let $\mathbf{s} \equiv \langle \Leftarrow \mathbf{g}, \theta \rangle$ and $\mathbf{s}' \equiv \langle \Leftarrow \mathbf{g}', \kappa \rangle$. Since $\mathbf{s}' \propto \mathbf{s}$, then $\mathbf{g}' \propto \mathbf{g}$ and $\kappa \propto \theta$. Now, since $\mathbf{s} \leadsto \mathbf{t}$, then $\mathbf{t} = \langle (\mathbf{g} \sim \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C}, \theta\sigma \rangle$, where $\mathbf{e} \in \mathbf{g}$, $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$, $(\lambda \to \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}$ and $\sigma = \mathbf{mgu}(\{(\mathbf{e}_{|\mathbf{u}})\theta = \lambda\})$. We consider two cases.

1. Assume that there exists $\mathbf{e}' \in \mathbf{g}'$ s.t. $\mathbf{e}' \propto \mathbf{e}$ and $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}')$. Thus it is immediate that $(\mathbf{e}'_{|\mathbf{u}}) \propto (\mathbf{e}_{|\mathbf{u}})$. By Lemma A.10, there exists $\mathbf{mgu}_{\mathcal{A}}(\{(\mathbf{e}'_{|\mathbf{u}}) = \lambda\} \cup \widehat{\kappa}) \propto \mathbf{mgu}(\{(\mathbf{e}_{|\mathbf{u}}) = \lambda\} \cup \widehat{\theta}) \Leftrightarrow \theta \circ \mathbf{mgu}(\{(\mathbf{e}_{|\mathbf{u}})\theta = \lambda\}) = \theta\sigma$. Let $\sigma' = \mathbf{mgu}_{\mathcal{A}}(\{(\mathbf{e}'_{|\mathbf{u}}) = \lambda\})$. Then $\kappa \Uparrow_{\mathcal{A}} \sigma' = \mathbf{mgu}_{\mathcal{A}}(\widehat{\kappa} \cup \widehat{\sigma'}) = \mathbf{mgu}_{\mathcal{A}}(\widehat{\kappa} \cup \widehat{\mathbf{mgu}}_{\mathcal{A}}(\{(\mathbf{e}'_{|\mathbf{u}}) = \lambda\})) = \mathbf{mgu}_{\mathcal{A}}(\{(\mathbf{e}'_{|\mathbf{u}}) = \lambda\} \cup \widehat{\kappa}) \propto \theta\sigma$. Since $(\kappa \Uparrow_{\mathcal{A}} \sigma') \propto \theta\sigma$ and $\theta\sigma \not\equiv \mathbf{fail}$, then $(\kappa \Uparrow_{\mathcal{A}} \sigma') \not\equiv \mathbf{fail}$. Therefore, by Definition 5.12, $\mathbf{s}' \leadsto_{\mathcal{A}} \langle \Leftarrow (\mathbf{g}' \sim \{\mathbf{e}'\}) \cup \{\mathbf{e}'[\mathbf{sh}(\rho)]_{\mathbf{u}}\} \cup \mathbf{sh}(\mathbf{C}), \kappa \Uparrow_{\mathcal{A}} \sigma' \rangle$. From Definition 5.8[4] it is immediate that $\mathbf{sh}(\rho) \propto \rho$ and $\mathbf{sh}(\mathbf{C}) \propto \mathbf{C}$. Hence $\mathbf{t}' \propto \mathbf{t}$.

2. If there does not exist $\mathbf{e}' \in \mathbf{g}'$ such that $\mathbf{e}' \propto \mathbf{e}$ and $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e}')$, then the lemma follows immediately from Lemma A.11.

□

Lemma A.13 *Let* $\mathbf{s}'_1 \propto \mathbf{s}_1$. *If there exists a basic narrowing derivation* $\mathbf{s}_1 \leadsto \ldots \leadsto \mathbf{s}_{\mathbf{n}}$, *then there exists an abstract basic narrowing derivation* $\mathbf{s}'_1 \leadsto_{\mathcal{A}} \ldots \leadsto_{\mathcal{A}} \mathbf{s}'_{\mathbf{m}}$ *such that* $\mathbf{s}'_{\mathbf{m}} \propto \mathbf{s}_{\mathbf{n}}$, $\mathbf{m} \leq \mathbf{n}$.

Proof. We prove the lemma by induction on the length $\mathbf{n}$ of the basic narrowing derivation.
Let $\mathbf{n} = 1$. The claim follows directly from Lemma A.12.
Let us consider the inductive case $\mathbf{n} > 1$. Since $\mathbf{s}'_1 \propto \mathbf{s}_1$ and $\mathbf{s}_1 \leadsto \ldots \leadsto \mathbf{s}_{\mathbf{n}-1}$, by the inductive hypothesis, there exists an abstract basic narrowing derivation $\mathbf{s}'_1 \leadsto_{\mathcal{A}} \ldots \leadsto_{\mathcal{A}} \mathbf{s}'_{\mathbf{k}}$, $\mathbf{k} \leq \mathbf{n}-1$, such that $\mathbf{s}'_{\mathbf{k}} \propto \mathbf{s}_{\mathbf{n}-1}$. Since $\mathbf{s}'_{\mathbf{k}} \propto \mathbf{s}_{\mathbf{n}-1}$ and $\mathbf{s}_{\mathbf{n}-1} \leadsto \mathbf{s}_{\mathbf{n}}$, then by Lemma A.12, either 1) $\mathbf{s}'_{\mathbf{k}} \leadsto_{\mathcal{A}} \mathbf{s}'_{\mathbf{k}+1}$ and $\mathbf{s}'_{\mathbf{k}+1} \propto \mathbf{s}_{\mathbf{n}}$, or 2) $\mathbf{s}'_{\mathbf{k}} \propto \mathbf{s}_{\mathbf{n}}$. First, we consider case 1). In this case, it suffices to take $\mathbf{m} = \mathbf{k}+1$ and we have $\mathbf{s}'_{\mathbf{m}} \propto \mathbf{s}_{\mathbf{n}}$, $\mathbf{m} \leq \mathbf{n}$. Now we consider case 2). In this case, we take $\mathbf{m} = \mathbf{k}$, and have $\mathbf{s}'_{\mathbf{m}} \propto \mathbf{s}_{\mathbf{n}}$, $\mathbf{m} \leq \mathbf{n}-1 \leq \mathbf{n}$, which proves the claim.
□

Now we can prove the desired result.

Theorem 5.14 Let $\mathcal{R}_{\mathcal{A}} \propto \mathcal{R}$ and $\mathbf{g}' \propto \mathbf{g}$. Then, for every solution $\theta \in \mathcal{O}_{\mathcal{R}}(\Leftarrow \mathbf{g})$, there exists $\kappa \in \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}')$ such that $\kappa \propto \theta$.

[4] We assume for the sake of simplicity that the clause renaming chosen for the abstract narrowing reduction is the same as in the (basic) narrowing reduction.

PROOF. The result is a particular case of Lemma A.13 for $\mathbf{s}_1 \equiv \langle \Leftarrow \mathbf{g}, \epsilon \rangle$, $\mathbf{s_n} \equiv \langle \Leftarrow \mathbf{true}, \theta \rangle$, $\mathbf{s}'_1 \equiv \langle \Leftarrow \mathbf{g}', \epsilon \rangle \propto \mathbf{s}_1$, and $\mathbf{s}'_\mathbf{m} \equiv \langle \Leftarrow \mathbf{g}'_\mathbf{m}, \kappa \rangle \propto \mathbf{s_n}$. Hence, $\mathbf{s}'_\mathbf{m} = \langle \Leftarrow \mathbf{true}, \kappa \rangle$, since $\mathbf{g}'_\mathbf{m} \propto \mathbf{true}$ implies that $\mathbf{g}'_\mathbf{m} \equiv \mathbf{true}$. □

Corollary 5.15 If $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{g}) = \emptyset$, then $\mathbf{g}$ is unsatisfiable in $\mathcal{R}$.

PROOF. Immediate from Theorem 5.14. □

Proof of Theorem 5.16

In order to prove Theorem 5.16 we need the following lemma.

Lemma A.14
$\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa \rangle \leadsto^*_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa \Uparrow_\mathcal{A} \theta \rangle$ *iff*
$\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \leadsto^*_\mathcal{A} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_\mathcal{A} \theta_1 \rangle$ *and* $\langle \Leftarrow \mathbf{g}_2, \kappa \rangle \leadsto^*_\mathcal{A} \langle \Leftarrow \mathbf{g}'_2, \kappa \Uparrow_\mathcal{A} \theta_2 \rangle$,
$\theta = \theta_1 \Uparrow_\mathcal{A} \theta_2 \not\equiv \mathbf{fail}$ *and* $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}'_2$.

PROOF. [proof of Lemma A.14]

($\Rightarrow$) Let $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa \rangle \equiv \langle \Leftarrow \mathbf{h}_1, \kappa \Uparrow_\mathcal{A} \vartheta_1 \rangle \leadsto_\mathcal{A} \ldots \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{h_n}, \kappa \Uparrow_\mathcal{A} \vartheta_\mathbf{n} \rangle \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa \Uparrow_\mathcal{A} \theta \rangle$, $(\vartheta_1 \equiv \epsilon)$. The proof is done by induction on the length $\mathbf{n}$ of the derivation.

(n=1) Straightforward.

Let us consider the inductive case.
If $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa \rangle \equiv \langle \Leftarrow \mathbf{h}_1, \kappa \Uparrow_\mathcal{A} \vartheta_1 \rangle \leadsto_\mathcal{A} \ldots \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{h_n}, \kappa \Uparrow_\mathcal{A} \vartheta_\mathbf{n} \rangle \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}', \kappa \Uparrow_\mathcal{A} \vartheta_\mathbf{n} \Uparrow_\mathcal{A} \delta \rangle \equiv \langle \Leftarrow \mathbf{g}', \kappa \Uparrow_\mathcal{A} \theta \rangle$, then there exist $(\lambda \to \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}_\mathcal{A}$, $\mathbf{e} \in \mathbf{h_n}$ and $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$ such that $\delta = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\}) \not\equiv \mathbf{fail}$ and $\mathbf{g}' = \mathbf{h_n} - \{\mathbf{e}\} \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C}$.

By the inductive hypothesis, there exist
$\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \equiv \langle \Leftarrow \mathbf{g}_{11}, \kappa \Uparrow_\mathcal{A} \vartheta_{11} \rangle \leadsto_\mathcal{A} \ldots \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}_{1\mathbf{m}}, \kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \kappa \rangle \equiv \langle \Leftarrow \mathbf{g}_{21}, \kappa \Uparrow_\mathcal{A} \vartheta_{21} \rangle \leadsto_\mathcal{A} \ldots \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}_{2\mathbf{p}}, \kappa \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} \rangle$, $(\vartheta_{11} \equiv \vartheta_{21} \equiv \epsilon)$,
such that $\vartheta_\mathbf{n} = \vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} \not\equiv \mathbf{fail}$ and $\mathbf{h_n} = \mathbf{g}_{1\mathbf{m}} \cup \mathbf{g}_{2\mathbf{p}}$.
Since $\mathbf{h_n} = \mathbf{g}_{1\mathbf{m}} \cup \mathbf{g}_{2\mathbf{p}}$, then $\mathbf{e} \in \mathbf{g}_{1\mathbf{m}}$ or $\mathbf{e} \in \mathbf{g}_{2\mathbf{p}}$. Let $\mathbf{e} \in \mathbf{g}_{1\mathbf{m}}$ (the case when $\mathbf{e} \in \mathbf{g}_{2\mathbf{p}}$ is perfectly analogous). It suffices to show that $\langle \Leftarrow \mathbf{g}_{1\mathbf{m}}, \kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \rangle \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_\mathcal{A} \theta_1 \rangle$ and $\theta = \theta_1 \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} \not\equiv \mathbf{fail}$ and $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}_{2\mathbf{p}}$.

Since $(\kappa \Uparrow_\mathcal{A} \vartheta_\mathbf{n} \Uparrow_\mathcal{A} \delta) \not\equiv \mathbf{fail}$ and $\vartheta_\mathbf{n} = (\vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}})$, then $(\kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \delta) \not\equiv \mathbf{fail}$ (by commutativity and associativity of $\Uparrow_\mathcal{A}$). Now, since $\mathbf{e} \in \mathbf{g}_{1\mathbf{m}}$, $\langle \Leftarrow \mathbf{g}_{1\mathbf{m}}, \kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \rangle \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \delta \rangle \equiv \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_\mathcal{A} \theta_1 \rangle$, where $\theta_1 = \vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \delta$ and $\mathbf{g}'_1 = (\mathbf{g}_{1\mathbf{m}} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C}$. Then, $\theta_1 \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} = (\vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \delta) \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} = \vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} \Uparrow_\mathcal{A} \delta = \vartheta_\mathbf{n} \Uparrow_\mathcal{A} \delta = \theta$. Finally, we prove that $\mathbf{g}' = \mathbf{g}'_1 \cup \mathbf{g}_{2\mathbf{p}}$. Since $\mathbf{h_n} = \mathbf{g}_{1\mathbf{m}} \cup \mathbf{g}_{2\mathbf{p}}$ then $\mathbf{g}' = (\mathbf{h_n} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C} = ((\mathbf{g}_{1\mathbf{m}} \cup \mathbf{g}_{2\mathbf{p}}) - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_\mathbf{u}\} \cup \mathbf{C} = \mathbf{g}'_1 \cup \mathbf{g}_{2\mathbf{p}}$.

($\Leftarrow$) Let $\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \equiv \langle \Leftarrow \mathbf{g}_{10}, \kappa \Uparrow_\mathcal{A} \vartheta_{10} \rangle \leadsto_\mathcal{A} \ldots \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}_{1\mathbf{m}}, \kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \rangle \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_\mathcal{A} \vartheta_{1\mathbf{m}} \Uparrow_\mathcal{A} \delta \rangle \equiv \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_\mathcal{A} \theta_1 \rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \kappa \rangle \equiv \langle \Leftarrow \mathbf{g}_{20}, \kappa \Uparrow_\mathcal{A} \vartheta_{20} \rangle \leadsto_\mathcal{A} \ldots \leadsto_\mathcal{A} \langle \Leftarrow \mathbf{g}_{2\mathbf{p}}, \kappa \Uparrow_\mathcal{A} \vartheta_{2\mathbf{p}} \rangle \equiv \langle \Leftarrow \mathbf{g}'_2, \kappa \Uparrow_\mathcal{A} \theta_2 \rangle$, $(\vartheta_{10} \equiv \vartheta_{20} \equiv \epsilon)$,
and $(\mathbf{m} + 1) + \mathbf{p} = \mathbf{n}$, $\mathbf{n} \geq 1$.

The proof is done by induction on the sum $\mathbf{n}$ of the lengths of the derivations.

(n=1) Straightforward.

Let us consider the inductive case. Since $\langle \Leftarrow \mathbf{g_{1m}}, \kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{1m}} \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \delta \rangle$, then there exist $(\lambda \rightarrow \rho \Leftarrow \mathbf{C}) \ll \mathcal{R}_{\mathcal{A}}$, $\mathbf{e} \in \mathbf{g_{1m}}$ and $\mathbf{u} \in \bar{\mathbf{O}}(\mathbf{e})$ such that $\delta = \mathbf{mgu}(\{\lambda = \mathbf{e}_{|\mathbf{u}}\})$ and
$\mathbf{g}'_1 = (\mathbf{g_{1m}} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C}$.

By the inductive hypothesis, there exists a derivation $\langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow (\mathbf{g_{1m}}, \mathbf{g}'_2), \kappa \Uparrow_{\mathcal{A}} (\vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2) \rangle$. Then, it suffices to show that $\langle \Leftarrow (\mathbf{g_{1m}}, \mathbf{g}'_2), \kappa \Uparrow_{\mathcal{A}} (\vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2) \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow (\mathbf{g}'_1, \mathbf{g}'_2), \kappa \Uparrow_{\mathcal{A}} (\theta_1 \Uparrow_{\mathcal{A}} \theta_2) \rangle$.

Since $(\kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \delta) \not\equiv \mathbf{fail}$, $(\kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2) \not\equiv \mathbf{fail}$, and $\theta = (\theta_1 \Uparrow_{\mathcal{A}} \theta_2) = (\vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \delta \Uparrow_{\mathcal{A}} \theta_2) \not\equiv \mathbf{fail}$, then $(\kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2 \Uparrow_{\mathcal{A}} \delta) \not\equiv \mathbf{fail}$ (by commutativity and associativity of $\Uparrow_{\mathcal{A}}$), and $\langle \Leftarrow (\mathbf{g_{1m}}, \mathbf{g}'_2), \kappa \Uparrow_{\mathcal{A}} (\vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2) \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow (\mathbf{g}'_1, \mathbf{g}'_2), \kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2 \Uparrow_{\mathcal{A}} \delta \rangle$, where $\mathbf{g}'_1 = (\mathbf{g_{1m}} - \{\mathbf{e}\}) \cup \{\mathbf{e}[\rho]_{\mathbf{u}}\} \cup \mathbf{C}$. Finally, since $\theta_1 = \vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \delta$, we have that $(\vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \theta_2) \Uparrow_{\mathcal{A}} \delta = (\vartheta_{\mathbf{1m}} \Uparrow_{\mathcal{A}} \delta) \Uparrow_{\mathcal{A}} \theta_2 = \theta_1 \Uparrow_{\mathcal{A}} \theta_2$, which proves the claim.

□

Now we can prove the desired result.

Theorem 5.16 $\Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}_1) \Uparrow_{\mathcal{A}} \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}_2)$.

Proof.
The result is a particular case of Lemma A.14 for $\kappa \equiv \epsilon$ and $\mathbf{g}'_1 \equiv \mathbf{g}'_2 \equiv \mathbf{true}$. □

Proof of Proposition 5.18 and Proposition 5.19

We proceed with Proposition 5.18 whose proof needs the following intermediate result.

Lemma A.15
If $\langle \Leftarrow \mathbf{g}, \kappa \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa' \rangle$ *then* $\langle \Leftarrow \mathbf{g}, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa' \rangle$.

Proof. [proof of Lemma A.15]
If $\langle \Leftarrow \mathbf{g}, \kappa \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta \rangle$, then by rule (1) of Definition 5.17, there exist $\mathbf{g}_1$ and $\mathbf{g}_2$ such that $\mathbf{g} = \mathbf{g}_1 \cup \mathbf{g}_2$ and
$\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$,
$\langle \Leftarrow \mathbf{g}_2, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$ and $\theta = \theta_1 \Uparrow_{\mathcal{A}} \theta_2 \not\equiv \mathbf{fail}$.
We prove the claim by induction on the number $\mathbf{n}$ of applications of rule (1) which prove these latter derivations.

$\mathbf{n} = 0$. Straightforward.
Let $\mathbf{n} = 1$ and assume
$\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$ and
$\langle \Leftarrow \mathbf{g}_2, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$ (the case when $\longmapsto_{\|}$ occurs in the derivation for $\mathbf{g}_2$ is perfectly analogous).

Since $\langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta \rangle$, then by rule (1) of Definition 5.17, there exist $\mathbf{g}'_{11}$ and $\mathbf{g}'_{12}$ such that $\mathbf{g}'_1 = \mathbf{g}'_{11} \cup \mathbf{g}'_{12}$, $\langle \Leftarrow \mathbf{g}'_{11}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta_1 \rangle$, $\langle \Leftarrow \mathbf{g}'_{12}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta_2 \rangle$ and $\delta = \delta_1 \Uparrow_{\mathcal{A}} \delta_2 \not\equiv \mathbf{fail}$. Then, by Lemma A.14, $\langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \equiv \langle \Leftarrow (\mathbf{g}'_{11}, \mathbf{g}'_{12}), \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$ and thus $\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$.
By Lemma A.14, $\langle \Leftarrow \mathbf{g}, \kappa \rangle \equiv \langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} (\theta_1 \Uparrow_{\mathcal{A}} \theta_2) \rangle$.

Let $\mathbf{n} > 1$. Since $\langle \Leftarrow \mathbf{g}_1, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$ and $\langle \Leftarrow \mathbf{g}_2, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$, then by rules (1) and (3) of Definition 5.17, either

$\begin{cases} \langle \Leftarrow \mathbf{g}_1, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta_1 \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle \text{ and} \\ \langle \Leftarrow \mathbf{g}_2, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_2, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \Uparrow_{\mathcal{A}} \delta_2 \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle, \text{ or} \end{cases}$

$\begin{cases} \langle \Leftarrow \mathbf{g}_1, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta_1 \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle \text{ and} \\ \langle \Leftarrow \mathbf{g}_2, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle, \text{ or} \end{cases}$

$\begin{cases} \langle \Leftarrow \mathbf{g}_1, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle \text{ and} \\ \langle \Leftarrow \mathbf{g}_2, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}'_2, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \Uparrow_{\mathcal{A}} \delta_2 \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle \end{cases}$

and the number of applications of rule (1) which prove the transitions $\langle \Leftarrow \mathbf{g}'_{\mathbf{i}}, \kappa \Uparrow_{\mathcal{A}} \kappa_{\mathbf{i}} \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_{\mathbf{i}} \Uparrow_{\mathcal{A}} \delta_{\mathbf{i}} \rangle, \mathbf{i} \in \{1, 2\}$ is less than $\mathbf{n}$. Let us consider only the first case since the other two cases are analogous.
Since $\langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \Uparrow_{\mathcal{A}} \delta_1 \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$ and
$\langle \Leftarrow \mathbf{g}'_2, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \Uparrow_{\mathcal{A}} \delta_2 \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$, then by the inductive hypothesis,
$\langle \Leftarrow \mathbf{g}'_1, \kappa \Uparrow_{\mathcal{A}} \kappa_1 \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$ and $\langle \Leftarrow \mathbf{g}'_2, \kappa \Uparrow_{\mathcal{A}} \kappa_2 \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$.
Since $\theta_1 \Uparrow_{\mathcal{A}} \theta_2 \not\equiv \mathbf{fail}$, then by Lemma A.14, $\langle \Leftarrow \mathbf{g}, \kappa \rangle \equiv \langle \Leftarrow (\mathbf{g}_1, \mathbf{g}_2), \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} (\theta_1 \Uparrow_{\mathcal{A}} \theta_2) \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta \rangle$, and the thesis follows. □

Proposition 5.18 (soundness).
If $\langle \Leftarrow \mathbf{g}, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}', \kappa' \rangle$ then $\langle \Leftarrow \mathbf{g}, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{g}', \kappa' \rangle$.

Proof.
Immediate from Lemma A.15. □

Now we proceed with the proof of Proposition 5.19.

Proposition 5.19 (completeness).
If $\langle \Leftarrow \mathbf{g}, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$ then $\langle \Leftarrow \mathbf{g}, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$.

Proof.
The proof is done by induction on the length $\mathbf{n}$ of the former derivation.

If $\mathbf{n} = 1$ and $\langle \Leftarrow \mathbf{g}, \kappa \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \sigma \rangle \equiv \langle \Leftarrow \mathbf{true}, \theta \rangle$ then the rule $(\mathbf{x} = \mathbf{x} \rightarrow \mathbf{true} \Leftarrow)$ has been applied and there exists an equation $\mathbf{e}$ such that $\mathbf{g} = \{\mathbf{e}\}$ and $\sigma = \mathbf{mgu}(\{\mathbf{e}\})$. Then, by rule (2) of Definition 5.17, $\langle \Leftarrow \{\mathbf{e}\}, \kappa \rangle \longmapsto_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \sigma \rangle$.

Now we consider the inductive case. If $\mathbf{n} > 1$ then
$\langle \Leftarrow \mathbf{g}, \kappa \rangle \equiv \langle \Leftarrow \mathbf{g}_0, \kappa \Uparrow_{\mathcal{A}} \vartheta_0 \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle \rightsquigarrow_{\mathcal{A}} \ldots \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{g}_{\mathbf{n}}, \kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{n}} \rangle \equiv \langle \Leftarrow \mathbf{true}, \theta \rangle, \ (\vartheta_0 \equiv \epsilon)$.

There are two cases to consider.

(1) Let $\mathbf{g}_0 = \{\mathbf{e}\}$. By the inductive hypothesis, there exists $\langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle \longmapsto_{\mathcal{A}} \ldots \longmapsto_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$. Since $\langle \Leftarrow \{\mathbf{e}\}, \kappa \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle$ then, by rule (2) of Definition 5.17, $\langle \Leftarrow \{\mathbf{e}\}, \kappa \rangle \longmapsto_{\mathcal{A}} \langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle$.

(2) Let $\mathbf{g}_0 = \mathbf{g}'_0 \cup \mathbf{g}''_0$ where $\mathbf{g}'_0$ and $\mathbf{g}''_0$ are non empty equation sets.

We have to prove that $\langle \Leftarrow \mathbf{g}, \kappa \rangle \longmapsto_{\mathcal{A}} \ldots \longmapsto_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$. According to rule (3) of Definition 5.17, we need to show that:
(a) $\langle \Leftarrow \mathbf{g}, \kappa \rangle \equiv \langle \Leftarrow \mathbf{g}_0, \kappa \Uparrow_{\mathcal{A}} \vartheta_0 \rangle \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle \longmapsto_{\mathcal{A}} \ldots \longmapsto_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$, or
(b) $\langle \Leftarrow \mathbf{g}, \kappa \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \theta \rangle$.

(a) Since $\langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle \rightsquigarrow_{\mathcal{A}} \ldots \rightsquigarrow_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$ then, by the inductive hypothesis, there exists $\langle \Leftarrow \mathbf{g}_1, \kappa \Uparrow_{\mathcal{A}} \vartheta_1 \rangle \longmapsto_{\mathcal{A}} \ldots \longmapsto_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \theta \rangle$.

(b) Since $\langle \Leftarrow \mathbf{g}, \kappa \rangle \equiv \langle \Leftarrow (\mathbf{g}'_0, \mathbf{g}''_0), \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{n}} \rangle \equiv \langle \Leftarrow \mathbf{true}, \theta \rangle$, then by Lemma A.14, there exist $\langle \Leftarrow \mathbf{g}'_0, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$, $\langle \Leftarrow \mathbf{g}''_0, \kappa \rangle \rightsquigarrow^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$ and $\vartheta_{\mathbf{n}} = \theta_1 \Uparrow_{\mathcal{A}} \theta_2$.
Then, by the inductive hypothesis, there exist $\langle \Leftarrow \mathbf{g}'_0, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_1 \rangle$ and $\langle \Leftarrow \mathbf{g}''_0, \kappa \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \theta_2 \rangle$. Therefore, by rule (1) of Definition 5.17,
$\langle \Leftarrow (\mathbf{g}'_0, \mathbf{g}''_0), \kappa \rangle \longmapsto_{\|} \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} (\theta_1 \Uparrow_{\mathcal{A}} \theta_2) \rangle \equiv \langle \Leftarrow \mathbf{true}, \kappa \Uparrow_{\mathcal{A}} \vartheta_{\mathbf{n}} \rangle \equiv \langle \Leftarrow \mathbf{true}, \theta \rangle$, and this completes the proof.

□

The following is an immediate conclusion of previous results.

Corollary 5.20 Let $\Delta'_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}) = \{\kappa_{|\mathbf{Var(g)}} \mid \langle \Leftarrow \mathbf{g}, \epsilon \rangle \longmapsto^*_{\mathcal{A}} \langle \Leftarrow \mathbf{true}, \kappa \rangle\}$. Then

$$\Delta'_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}_1, \mathbf{g}_2) = \Delta'_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}_1) \Uparrow_{\mathcal{A}} \Delta'_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{g}_2).$$

Proof.
Immediate from Theorem 5.16, Proposition 5.18 and Proposition 5.19. □

Proof of Theorem 6.4

Theorem 6.4 Let $\mathbf{c}$ be a constraint and $\Theta = \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{c}) \neq \emptyset$. Then,

1. if a transition $\langle \mathbf{c}, \Theta \rangle \xrightarrow[\mathbf{iEA}]{\tilde{\mathbf{c}}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Theta' \rangle$ is proven, then $\Theta' = \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}})$;
2. if a transition $\langle \mathbf{c}, \Theta \rangle \xrightarrow[\mathbf{iEA}]{\tilde{\mathbf{c}}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \emptyset \rangle$ is proven, then the constraint $\mathbf{c} \cup \widetilde{\mathbf{c}}$ is unsatisfiable.

Proof.
We prove the two claims separately.

1. From Definition 6.3, $\langle \mathbf{c}, \Theta \rangle \xrightarrow[\mathbf{iEA}]{\tilde{\mathbf{c}}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Theta \Uparrow_{\mathcal{A}} \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \widetilde{\mathbf{c}}) \rangle = \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{c}) \Uparrow_{\mathcal{A}} \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \widetilde{\mathbf{c}}) \rangle = \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Delta_{\mathcal{R}_{\mathcal{A}}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}}) \rangle$, by Theorem 5.16.

2. Assume, by contradiction, that the constraint $\mathbf{c} \cup \widetilde{\mathbf{c}}$ is satisfiable. It suffices to show that the transition $\langle \mathbf{c}, \Theta \rangle \xrightarrow{\tilde{\mathbf{c}}}_{\mathbf{iEA}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \emptyset \rangle$ cannot be proven for the hypothesis to be contradicted. By case 1, $\langle \mathbf{c}, \Theta \rangle \xrightarrow{\tilde{\mathbf{c}}}_{\mathbf{iEA}} \langle \mathbf{c} \cup \widetilde{\mathbf{c}}, \Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}}) \rangle$. By Corollary 5.15 and the assumption that $\mathbf{c} \cup \widetilde{\mathbf{c}}$ is satisfiable, $\Delta_{\mathcal{R}_\mathcal{A}}(\Leftarrow \mathbf{c} \cup \widetilde{\mathbf{c}}) \not\equiv \emptyset$.

□