J. Symbolic Computation (1995) **11**, 1–32

Theorem Proving with Ordering and Equality Constrained Clauses

ROBERT NIEUWENHUIS AND ALBERT RUBIO †

Universitat Politècnica de Catalunya

Pau Gargallo 5

08028 Barcelona, Spain

(Received 2 February 1994)

Deduction methods for first-order constrained clauses with equality are described within an abstract framework: *constraint strategies*, consisting of an inference system, a *constraint inheritance strategy* and *redundancy criteria* for clauses and inferences. We give simple conditions for such a constraint strategy to be complete (refutationally and in the sense of Knuth-Bendix-like completion). This allows to prove in a uniform way the completeness of several instantiations of the framework with concrete strategies. For example, strategies in which *equality constraints* are inherited are *basic*: no inferences are needed on subterms introduced by unifiers of previous inferences. *Ordering constraints* reduce the search space by inheriting the ordering restrictions of previous inferences and increase the expressive power of the logic.

† To appear in spring 1995 (no. JSC1389). This work has been partially supported by the Esprit Working Group CCL, ref. 6028. Authors' e-mail:{roberto,rubio}@lsi.upc.es.

0747–7171/90/000000 + 00 $03.00/0

1. Introduction

A clause with a symbolic constraint can be seen as a shorthand for the (normally infinite) set of its ground instances satisfying the constraint. Such constraints not only increase the expressive power of the logic, but they are also very useful for making parts of the theorem proving process explicit, leading to new insight in the deduction strategies, cf. the paper by Kirchner, Kirchner and Rusinowitch (1990) where many of these ideas were introduced.

Here we give methods for dealing with constrained clauses that do not require *propagation* of information from the constraints to the clause part, as in (Kirchner *et al.*, 1990). For example, if '$=$' is syntactic equality, by such a propagation step a constrained equation $f(x) \simeq a\,[\![x = g(y)]\!]$, could be replaced by the logically equivalent one $f(g(y)) \simeq a\,[\![true]\!]$. For ordering constraints, propagation normally amounts to enumerating the solutions of the constraint. Therefore, avoiding propagation is essential for exploiting the constraints from the efficiency point of view. For example, constrained (equational) superposition can then be expressed like:

$$\frac{s' \simeq t'\,[\![T']\!] \qquad s \simeq t\,[\![T]\!]}{s[t']_p \simeq t\,[\![T' \wedge T \wedge s' \succ t' \wedge s \succ t \wedge s|_p = s']\!]} \quad \text{where } s|_p \notin \mathcal{V}ars(s)$$

This provides an elegant and also powerful representation for ordered inference rules, where information from the meta-level (the ordering and unifiability restrictions) is kept and inherited to restrict future inferences: clauses with unsatisfiable constraints are tautologies, hence redundant. These inference rules are *basic* (i.e. no inferences are needed on subterms introduced by the unifiers of previous inferences), because the accumulated unifiers are kept in the equality constraints (no propagation is needed) and future superpositions can take place only on non-variable subterms of the clause part. Our completeness proofs are based on the *model generation* framework with its *abstract redundancy notions* for detecting redundant clauses and inferences during the theorem proving process, developed by Bachmair and Ganzinger (1991).

To illustrate the additional difficulties with constrained clauses, consider $a \succ b \succ c$ and the following inconsistent set of clauses S:

1. $a \simeq c$
2. $P(x)\,[\![x \succ b]\!]$
3. $\neg P(c)$

No ordered paramodulation (or resolution) inferences that are compatible with the constraint of the second clause can be made. However, S does not contain the empty clause. This incompleteness is caused by the falsity in the constraint case of lifting lemmas (like the critical pair lemma) based on the existence of *all* ground instances of the clauses, which is not the case here, since $P(c)$ is not a ground instance of the second clause. From this example we can also conclude that the usual ordered resolution and paramodulation inference rules are incomplete when the *initial* clauses have arbitrary constraints. In this paper we also analyze the kind of inital constrained clauses for which completeness is preserved. Particular examples of admissible initial sets of clauses are the ones without equality literals, or sets of clauses without initial constraints in which variables are upper bounded. This allows us to describe and reason about infinite sets of normal formulae: for example the clause $\rightarrow P(x, y)\,[\![a \succ x]\!]$, under the lexicographic path ordering induced by the precedence $a \succ f \succ b$, denotes the infinite set of clauses $\rightarrow P(b, y)$, $\rightarrow P(f(b), y)$, $\rightarrow P(f(f(b)), y), \ldots$.

In section 3 of this paper we define a comprehensive abstract framework for deduction with ordering and equality constrained clauses by means of *constraint strategies*. Such a constraint strategy consists of an inference system, a *constraint inheritance strategy* and *redundancy criteria* for clauses and inferences. After this, simple conditions for such a constraint strategy to be complete (refutationally and in the sense of Knuth-Bendix-like completion) are given.

In section 4 we introduce a superposition-based inference system which will be used as ingredient in the constraint strategies of later sections. The purpose of section 5 is to clarify our methods by means of a short and simple example in which the constraints do not yet play any role, namely the case of ground clauses. In section 6 the framework is instantiated with the most powerful constraint inheritance strategy, i.e. the one in which all information of the inference rules is kept. At the end of this section some general conditions are given under which practical simplification methods can be applied in this framework.

In the following section the interest of *saturated* sets of constrained clauses is analyzed. For instance the applicability to Knuth-Bendix completion of these techniques is shown. We also give (in section 8) two results for fast ordering constraint solving and define a new semantics for ordering constraints which is shown to be necessary for theorem proving in the presence of additional symbols (like Skolem constants, or when combining saturated sets). The satisfiability of ordering constraints under these new semantics is proved decidable. The role of (possibly initial) ordering constraints for expressing infinite sets of formulae is also analyzed here.

The completeness of these ordering and equality constrained deduction methods, which we here further develop in a uniform framework, was originally reported in our preliminary papers (Nieuwenhuis and Rubio, 1992a) and (Nieuwenhuis and Rubio, 1992b). As said, the completeness of the basic strategy in ordered paramodulation and its compatibility with concrete simplification and deletion methods turns out to be a particular case of these results for (equality) constraint inheritance and the corresponding abstract redundancy notions. Bachmair et al. (1992) have independently developed a comprehensive framework for the basic strategy in combination with techniques for simplification and restrictions based e.g. on term selection rules and redex orderings, using *closure substitutions* which play the role of our equality constraints. Lynch and Snyder (1993) apply equality, disequality and irreducibility constraints to obtain more powerful redundancy criteria in equational completion. The constraint paradigm has also been applied to theorem proving in (Huet, 1972) and (Peterson, 1990). *Basic narrowing* was introduced in (Hullot, 1980). All techniques outlined in this paper, including the constraint solving algorithms for both semantics, have been successfully implemented within the *Saturate* system (Nivela and Nieuwenhuis, 1993). In fact, the idea of modularly combining constraint inheritance strategies is a result of this experimentation and indeed coincides with the final implementation choosen in this system†.

† The *Saturate* system is freely available by FTP: contact the authors.

2. Clauses with constraints and their semantics

We adopt the standard notations and definitions for term rewriting given in (Dershowitz and Jouannaud, 1990). The set of terms is denoted by $\mathcal{T}(\mathcal{F},\mathcal{X})$ (for a given *signature* $\mathcal{F}$ and set of variables $\mathcal{X}$). Variable-free terms, denoted $\mathcal{T}(\mathcal{F})$, are called *ground*. An *equation* is a multiset of terms $\{s,t\}$, which will be written in the form $s \simeq t$. A first-order clause is a pair of (finite) multisets of equations Γ and Δ, denoted by $\Gamma \rightarrow \Delta$. The sets Γ and Δ are called respectively the *antecedent* and the *succedent* of the clause. Here distinct clauses are supposed not to share variables. The set of variables of a term t (resp. an equation e or a clause C) is denoted by $\mathcal{V}ars(t)$ (resp. $\mathcal{V}ars(e)$ or $\mathcal{V}ars(C)$).

A *substitution* is a mapping from variables to terms, denoted as $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. The application of the substitution σ to a term t (an equation e or a clause C) is denoted by $t\sigma$ ($e\sigma$ or $C\sigma$). A substitution is ground if its range is in $\mathcal{T}(\mathcal{F})$. Here we suppose that ground substitutions are also *grounding*: when we apply a ground substitution σ to a term t (equation e or a clause C), we obtain a ground term $t\sigma$ (ground equation $e\sigma$ or a ground clause $C\sigma$), i.e. we assume that $\mathcal{V}ars(t)$ ($\mathcal{V}ars(e)$ or $\mathcal{V}ars(C)$) is a subset of $Dom(\sigma)$.

The results in this paper are parameterized by a given reduction ordering that is total on ground terms (because of its totality, such a reduction ordering must be a simplification ordering, i.e. it fulfils the subterm property). A typical example of such an ordering is the *lexicographic path ordering* (LPO), denoted by $\succ_{lpo}^{\mathcal{F}}$ (or simply $\succ_{lpo}$). It is generated by extending $\succ_{\mathcal{F}}$, a total *precedence* ordering on the function symbols in $\mathcal{F}$, and defined as follows:
$s = f(s_1,\ldots,s_m) \succ_{lpo} g(t_1,\ldots,t_n) = t$ if

$s_i \succeq_{lpo} t$, for some i with $1 \leq i \leq m$ or
$f \succ_{\mathcal{F}} g$, and $s \succ_{lpo} t_j$, for all j with $1 \leq j \leq n$ or
$f = g$, $(s_1,\ldots,s_m) \succ_{lpo}^{lex} (t_1,\ldots,t_n)$, and $s \succ_{lpo} t_j$, for all j with $1 \leq j \leq n$

where $(s_1,\ldots,s_m) \succ_{lpo}^{lex} (t_1,\ldots,t_n)$ if $\exists j \leq n, \forall i < j, s_i = t_i$ and $s_j \succ_{lpo} t_j$.

By *(ordering) constraints* we mean quantifier-free first-order formulae built over the binary predicate symbols '$\succ$' and '$=$' relating terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$, where '$=$' denotes syntactic equality of terms, and '$\succ$' is interpreted as the given total reduction ordering on ground terms. An *equality constraint* is a particular case of such a constraint, consisting of a conjunction of equality literals $s = t$.

Sometimes, instead of considering the given reduction ordering "on ground terms" it is necessary to be more precise, and for the given signature $\mathcal{F}$, we denote by $\succ^{\mathcal{F}}$ a total ordering on $\mathcal{T}(\mathcal{F})$ and write pairs $(\mathcal{F},\succ^{\mathcal{F}})$. By an *extension* $(\mathcal{F}',\succ^{\mathcal{F}'})$ of $(\mathcal{F},\succ^{\mathcal{F}})$ we mean a set of function symbols $\mathcal{F}'$ with $\mathcal{F}' \supseteq \mathcal{F}$ and a total ordering $\succ^{\mathcal{F}'}$ including $\succ^{\mathcal{F}}$. A solution in $(\mathcal{F}',\succ^{\mathcal{F}'})$ of a constraint T is a ground substitution σ with range in $\mathcal{T}(\mathcal{F}')$ and whose domain is a set of variables containing the variables of T, such that $T\sigma$ evaluates to true, denoted by $T\sigma \equiv true$, when $\succ$ is interpreted as the ordering $\succ^{\mathcal{F}'}$. Then we say that T is *satisfiable* in $(\mathcal{F}',\succ^{\mathcal{F}'})$. A constraint T with terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$ is *satisfiable under the given signature* if it has a solution in $(\mathcal{F},\succ^{\mathcal{F}})$. We will call the constraint T *satisfiable under extended signatures* if there exists some extension $(\mathcal{F}',\succ^{\mathcal{F}'})$ of $(\mathcal{F},\succ^{\mathcal{F}})$ in which T is satisfiable.

The satisfiability under the given signature of LPO-constraints, i.e. constraints where $\succ$ is interpreted as $\succ_{lpo}^{\mathcal{F}}$, was first proved to be decidable by Comon (1990). Satisfiability of LPO-constraints under extended signatures is proved decidable here, in section 8.1, where this notion of satisfiability is also shown to be the adequate semantics in most theorem provers with constrained clauses. For the contents of the rest of this paper the choice between the two possible semantics for constraints is not relevant, and we will simply speak about satisfiability of the constraints. For a further discussion about this, see section 7.

Now we define a total ordering on ground clauses where the terms appearing in antecedents of clauses are slightly more complex than the ones in succedents. An *occurrence of an equation* $t \simeq t'$ in an antecendent is (the two-fold multiset) $\{\{t, t'\}\}$, and in a succedent it is $\{\{t\}, \{t'\}\}$. The two-fold multiset extension of the given total reduction ordering $\succ$ is a total ordering on such occurrences of ground equations, and the three-fold multiset extension of $\succ$ is a total ordering on ground clauses (seen as multisets of occurences of equations). From now on we will ambiguously use $\succ$ to denote these orderings on terms, on equations and on clauses.

A term t is called *maximal* (resp. *strict maximal*) in a set of terms $\{t_1, \ldots, t_n\}$, if there is a ground substitution σ s.t. $t\sigma \succeq t_i\sigma$ (resp. $t\sigma \succ t_i\sigma$) for $i = 1 \ldots n$. Note that this is equivalent to saying that the constraint $[\![t \succeq t_1 \wedge \ldots \wedge t \succeq t_n]\!]$ (resp. $[\![t \succ t_1 \wedge \ldots \wedge t \succ t_n]\!]$) is satisfiable. Similarly we can extend the definition of maximality and strict maximality to occurences of equations in clauses: an occurrence of an equation e in a clause C is called *maximal* (resp. *strict maximal*) in C if there is a substitution σ s.t. $e\sigma \succeq e'\sigma$ (resp. $e\sigma \succ e'\sigma$) for all occurrences e' different from e in C. Let e be an occurrence of an equation and let C be a clause. We will denote by $max(e, C)$ (resp. $smax(e, C)$) the ordering constraint expressing the maximality (resp. strictly maximality) of e in C, i.e. such a constraint has as solutions those ground instances σ in which $e\sigma$ is maximal in $C\sigma$. Note that this can be encoded in a straightforward manner.

A constrained clause is a pair (C, T), denoted $C[\![T]\!]$, where C is a first-order clause and T is a constraint. Such a pair denotes the set of *ground instances* of $C[\![T]\!]$: those ground clauses $C\sigma$ such that $T\sigma \equiv true$.

Given a set of ground clauses S and a ground clause C, we define $S^{\prec C}$ (resp. $S^{\preceq C}$) as the subset of S containing only the clauses smaller wrt. $\succ$ than C (resp. smaller or equal to C).

An interpretation I is a congruence on ground terms. It satisfies a ground clause $\Gamma \rightarrow \Delta$, denoted $I \models \Gamma \rightarrow \Delta$, if $I \not\supseteq \Gamma$ or else $I \cap \Delta \neq \emptyset$. The empty clause is therefore satisfied by no interpretation. An interpretation I satisfies (is a model of) $C[\![T]\!]$, denoted $I \models C[\![T]\!]$, if it satisfies every ground instance of $C[\![T]\!]$, i.e. clauses with unsatisfiable constraints are tautologies. Therefore, $C[\![T]\!]$ is said to be an *empty clause* only if C is empty and T is satisfiable. I satisfies a set of clauses S, denoted by $I \models S$, if it satisfies every clause in S. A clause C *follows* from a set of clauses S (denoted by $S \models C$), if C is satisfied by every model of S.

For dealing with non-equality predicates, we express atoms A by equations $A \simeq true$ where *true* is a special symbol, i.e. we treat atoms as terms. No significant changes have to be made for distinguishing between atoms and terms in e.g. inference rules or orderings. In the ordering $\succ$, the special symbol *true* is the smallest symbol.

We use the definitions of (Dershowitz and Jouannaud, 1990) for rewriting-related notions like *normal form, confluence, convergence, reducibility,* etc. However, to avoid confusion with the arrow $\rightarrow$ of clauses, we denote *ground rewrite rules* (ground equations $t \simeq t'$ with $t \succ t'$) by $t \Rightarrow t'$. The congruence generated by a set of ground rewrite rules R (which is an interpretation) will be denoted by R^*.

A *normal form* of a substitution σ wrt. a set of rewrite rules R is a substitution σ' with the same domain as σ, and such that $x\sigma'$ is a normal form wrt. R of $x\sigma$, for every variable x in the domain. Given a set of equations E and and a total reduction ordering $\succ$, the *ordered rewriting* relation $\rightarrow_E$ is defined as: $s \rightarrow_E t$ if there is an equation $l \simeq r$ in E s.t. $s|_p = l\sigma$, $t = s[r\sigma]_p$ and $l\sigma \succ r\sigma$, for some substitution σ.

3. Abstract constraint strategies and saturation

Given a signature $\mathcal{F}$, below we denote by $\mathcal{S}$ the set of all clauses built over $\mathcal{F}$, and similarly by $\mathcal{C}$ the set of all constraints, and by $\mathcal{EC}$ the set of all equality constraints (which is a subset of $\mathcal{C}$).

DEFINITION 3.1. *An inference rule $\mathcal{IR}$ is a mapping of n-tuples of clauses to sets of triples containing a clause, a constraint and an equality constraint:*

$$\mathcal{IR}: \quad \mathcal{S}^n \quad \longrightarrow \quad \mathcal{P}(\langle \mathcal{S}, \mathcal{C}, \mathcal{EC} \rangle)$$

An inference system is a set of inference rules.

DEFINITION 3.2. *A constraint inheritance strategy is a function mapping a clause, two constraints and an equality constraint to a clause and a constraint:*

$$\mathcal{H}: \quad \mathcal{S} \times \mathcal{C} \times \mathcal{C} \times \mathcal{EC} \quad \longrightarrow \quad \mathcal{S} \times \mathcal{C}$$

Inference systems and constraint inheritance strategies are combined to produce inferences in the usual sense: given constrained clauses $C_1 [\![T_1]\!], \ldots, C_n [\![T_n]\!]$, we obtain a conclusion $C [\![T]\!]$ as follows. Applying an inference rule to $C_1, \ldots, C_n$ we obtain a triple $\langle D, OT, ET \rangle$. Then the constraint inheritance strategy determines which parts of the ordering constraint OT and the equality constraint ET generated in the inference rule are added to $T_1 \wedge \ldots \wedge T_n$ to form T, which parts are included in the clause part D to produce the part C of the conclusion $C [\![T]\!]$ and which parts are disregarded. In the following formal definition, let $\mathcal{IR}$ be an inference rule of an inference system $\mathcal{I}$ and let $\mathcal{H}$ be a constraint inheritance strategy.

DEFINITION 3.3. *Let $\langle C_1 [\![T_1]\!], \ldots, C_n [\![T_n]\!] \rangle$ be a sequence of constrained clauses. If*

$$\langle D, OT, ET \rangle \in \mathcal{IR}(\langle C_1, \ldots, C_n \rangle) \text{ and } \mathcal{H}(D, T_1 \wedge \ldots \wedge T_n, OT, ET) = \langle C, T \rangle$$

we say that by an inference by $\mathcal{I}$ and $\mathcal{H}$ the conclusion $C [\![T]\!]$ *is obtained from the* premises $C_1 [\![T_1]\!], \ldots, C_n [\![T_n]\!]$. *We will use $\mathcal{I}nf\,(\mathcal{I}, \mathcal{H})$ or simply $\mathcal{I}nf$ to denote the set of all such inferences.*

To improve readability, instead of $\langle C, OT, ET\rangle \in \mathcal{IR}(\langle C_1,\ldots,C_n\rangle)$ we will normally write

$$\mathcal{IR}: \quad \frac{C_1 \ldots C_n}{C} \qquad \begin{array}{lcl} \mathcal{C} & : & OT \\ \mathcal{EC} & : & ET \end{array}$$

Similarly, the fact that $C\,[\![T]\!]$ is obtained from $\langle C_1\,[\![T_1]\!],\ldots,C_n\,[\![T_n]\!]\rangle$ in an *inference* by $\mathcal{I}$ and $\mathcal{H}$ will be denoted

$$\frac{C_1\,[\![T_1]\!],\ldots,C_n\,[\![T_n]\!]}{C\,[\![T]\!]}$$

EXAMPLE 3.4. *The inference by equational basic superposition shown in the introduction, i.e.*

$$\frac{s' \simeq t'\,[\![T']\!] \qquad s \simeq t\,[\![T'']\!]}{s[t']_p \simeq t\,[\![T' \wedge T'' \wedge s' \succ t' \wedge s \succ t \wedge s|_p = s']\!]} \qquad \textit{where } s|_p \notin \mathcal{V}ars(s)$$

would be obtained by combining the inference rule

$$\textit{superposition}: \quad \frac{s' \simeq t' \quad s \simeq t}{s[t']_p \simeq t} \qquad \begin{array}{lcl} \mathcal{C} & : & s' \succ t' \wedge s \succ t \\ \mathcal{EC} & : & s|_p = s' \end{array} \qquad \textit{where } s|_p \notin \mathcal{V}ars(s)$$

with the constraint inheritance strategy:

$$\mathcal{H}(C, T, OT, ET) = \langle C, T \wedge OT \wedge ET\rangle$$

This inference corresponds to a strategy in which all equality and ordering constraints are inherited. But many other combinations are possible. One may decide to inherit only (parts of) the ordering constraints or only (parts of) the equality constraints, probably depending on the constraint solving algorithms used and their efficiency (this may be especially relevant when deduction is performed modulo equational theories, like AC).

For instance, if we decide to inherit only the ordering restrictions then the new inference would be

$$\frac{s' \simeq t'\,[\![T']\!] \qquad s \simeq t\,[\![T'']\!]}{s[t']_p\sigma \simeq t\sigma\,[\![(T' \wedge T'' \wedge s' \succ t' \wedge s \succ t)\sigma]\!]} \qquad \textit{where } s|_p \notin \mathcal{V}ars(s) \textit{ and } \sigma = mgu(s|_p, s')$$

For this, we do not change the inference rule, but only the constraint inheritance strategy:

$$\mathcal{H}(C, T, OT, ET) = \begin{cases} \langle C\sigma, T\sigma \wedge OT\sigma\rangle & \textit{if } ET \textit{ has a mgu } \sigma \\ \langle C, \textit{false}\rangle & \textit{otherwise} \end{cases}$$

We check whether the equality restriction is satisfiable and in such a case apply the most general unifier to the clause and to the constraints. If the equality restriction is not satisfiable then the conclusion must be a tautology. □

We now introduce the remaining components of what we have called *constraint strategies*, namely the abstract redundancy criteria for clauses and inferences:

DEFINITION 3.5.
A redundancy criterion for clauses is a function from sets of clauses to sets of clauses:

$$\mathcal{RC}: \mathcal{P}(\mathcal{S}) \longrightarrow \mathcal{P}(\mathcal{S})$$

such that $S \models \mathcal{RC}(S)$ and $S \subseteq \mathcal{RC}(S)$ for all sets of constrained clauses S.

A redundancy criterion for inferences is a function from sets of clauses to sets of inferences:

$$\mathcal{RI} : \mathcal{P}(\mathcal{S}) \longrightarrow \mathcal{P}(\mathcal{I}nf\,)$$

such that $\pi \in \mathcal{RI}(S)$ *if the conclusion of* π *is in* S.

A constraint strategy $\mathcal{CS}$ *is a tuple of the form* $\langle \mathcal{I}, \mathcal{H}, \mathcal{RI}, \mathcal{RC}\rangle$, *where* $\mathcal{I}$ *is an inference system,* $\mathcal{H}$ *is a constraint inheritance strategy, and* $\mathcal{RI}$ *and* $\mathcal{RC}$ *are redundancy criteria for inferences and clauses respectively.*

This framework allows to get different saturation procedures only by instantiating the different components of a given constraint strategy $\mathcal{CS}$, and generating in some *fair* manner a $\mathcal{CS}$-*derivation*:

Definition 3.6.

1 *A* $\mathcal{CS}$*-derivation is a sequence of sets of (constrained) clauses* $S_1, S_2, \ldots$ *where each* S_{i+1} *is obtained from* S_i *by adding a constrained clause* $C\,[\![T]\!]$ *following from* S_i *or by removing a constrained clause* $C\,[\![T]\!]$ *that is redundant in* $S_i \setminus \{C\,[\![T]\!]\}$.
2 *The set* S_∞ *of persistent clauses in* $S_0, S_1, \ldots$ *is defined as* $\cup_j(\cap_{k\succeq j} S_k)$.
3 *A* $\mathcal{CS}$*-derivation* $S_0, S_1, \ldots$ *is fair if, for every inference* π *with persistent premises, there is some* S_j *s.t.* $\pi \in \mathcal{RI}(S_j)$.

We now provide sufficient conditions that have to be fulfilled by the constraint strategy to ensure that any fair $\mathcal{CS}$-derivation is a refutationally complete theorem proving procedure. We first need some additional definitions like the ones of *pure* sets of constrained clauses or of $\mathcal{CS}$-*saturated* sets of constrained clauses:

Definition 3.7.

1 *A variable* x *is* not lower bounded *by a constraint* T *if* $T\sigma \equiv true$ *implies* $T\sigma' \equiv true$, *for every pair of ground substitutions* σ *and* σ', *s.t.* $x\sigma \succ x\sigma'$ *and* $y\sigma' = y\sigma$ *for any other variable* y *in the domain of* σ.
2 *A set* S *of constrained clauses is pure if for every clause* $C\,[\![T]\!]$ *in* S, *there are no variables in* $\mathcal{V}ars(C)$ *lower bounded by* T.

A trivial example of pure set is a set containing only clauses with empty (or trivially true) constraints, i.e. $C\,[\![true]\!]$, or with constraints where variables appear only in right hand sides of inequalities, e.g. $C\,[\![a \succ x]\!]$.

Definition 3.8. *A set of constrained clauses* S *is* $\mathcal{CS}$*-saturated if*

$\pi \in \mathcal{RI}(S)$, *for every inference* π *by* $\mathcal{I}$ *and* $\mathcal{H}$ *with premises in* S *and*
either all clauses in S *contain only atoms (i.e. all literals are of the form* $t \simeq true$*)*
or there exists a pure set S' *s.t.* $S' \models S$ *and* $S' \subseteq \mathcal{RC}(S)$.

The first condition that has to be fulfilled by a constraint strategy is the *correctness* of the pair of redundancy criteria;

DEFINITION 3.9. *A pair of redundancy criteria* $(\mathcal{RC},\mathcal{RI})$ *is* correct *if for all sets of constrained clauses* S *and* S' *and for every* $\mathcal{CS}$*-derivation* $S_0, S_1, \ldots$ *and* $j \geq 0$ *it holds that*

$$\mathcal{RC}(S) \subseteq \mathcal{RC}(S \cup S') \quad \textit{and} \quad \mathcal{RC}(S_j) \subseteq \mathcal{RC}(S_\infty)$$
$$\mathcal{RI}(S) \subseteq \mathcal{RI}(S \cup S') \quad \textit{and} \quad \mathcal{RI}(S_j) \subseteq \mathcal{RI}(S_\infty)$$

LEMMA 3.10. *Let* $S_0, S_1, \ldots$ *be a fair* $\mathcal{CS}$*-derivation and let* $\mathcal{RC}$ *and* $\mathcal{RI}$ *be correct. If* S_∞ *contains only atoms or* S_0 *is pure then* S_∞ *is* $\mathcal{CS}$*-saturated and logically equivalent to* S_i *for all* i.

PROOF. We first prove that S_∞ is $\mathcal{CS}$-saturated. By fairness of the derivation, we have that for every inference π with premises in S_∞ there is some S_j s.t. $\pi \in \mathcal{RI}(S_j)$. But, since $\mathcal{RI}(S_j) \subseteq \mathcal{RI}(S_\infty)$ by the correctness of $\mathcal{RC}$ and $\mathcal{RI}$, π also is in $\mathcal{RI}(S_\infty)$. The second part of the definition of $\mathcal{CS}$-saturatedness is fulfilled if S_∞ contains only atoms or else there exists a pure set (in this case S_0) s.t. $S_0 \models S_\infty$ and $S_0 \subseteq \mathcal{RC}(S_\infty)$. Indeed $S_0 \models S_\infty$ since in the derivation only clauses following from S_0 are added. Furthermore, $S_0 \subseteq \mathcal{RC}(S_\infty)$ follows from the facts that $S_0 \subseteq \mathcal{RC}(S_0)$ by definition of $\mathcal{RC}$, and $\mathcal{RC}(S_j) \subseteq \mathcal{RC}(S_\infty)$ for $j \geq 0$ (by correctness of $\mathcal{RC}$ and $\mathcal{RI}$). Therefore S_∞ is $\mathcal{CS}$-saturated.

Now we prove that S_∞ is logically equivalent to S_i for all i. All sets S_i are logically equivalent by definition of $\mathcal{CS}$-derivation, and similarly $S_0 \models S_\infty$. It remains to be shown that $S_\infty \models S_0$. Now note that for all $C\,[\![T]\!] \in S_0 \setminus S_\infty$ there is a j such that $C\,[\![T]\!] \in \mathcal{RC}(S_j)$. But, by correctness of $(\mathcal{RC},\mathcal{RI})$, we have $\mathcal{RC}(S_j) \subseteq \mathcal{RC}(S_\infty)$, which implies $S_0 \setminus S_\infty \subseteq \mathcal{RC}(S_\infty)$, and, since by definition of $\mathcal{RC}$ we have $S_\infty \models \mathcal{RC}(S_\infty)$, we obtain $S_\infty \models S_0 \setminus S_\infty$ and therefore $S_\infty \models S_0$, which means that S_∞ is logically equivalent to S_i for all i. □

DEFINITION 3.11. *A constraint strategy* $\mathcal{CS}$ *is complete if* $\mathcal{RC}$ *and* $\mathcal{RI}$ *are correct and every* $\mathcal{CS}$*-saturated inconsistent set of constrained clauses contains the empty clause.*

The previous definition and lemma 3.10, give us the exact tools for building refutationally complete (or complete in the sense of Knuth-Bendix completion, as we will see) procedures for pure sets of clauses S_0. Let $\mathcal{CS}$ be a complete constrained strategy. If we compute a $\mathcal{CS}$-derivation starting with S_0 then S_∞ will be $\mathcal{CS}$-saturated and logically equivalent to S_0, i.e. S_∞ will contain the empty clause iff S_0 is inconsistent. But since S_∞ is the set of persistent clauses, in that case the empty clause should be contained in some set S_j of the derivation.

Moreover, if we start with a set S_0 where all clauses contain only atoms (i.e. no equality literals, but with arbitrary constraints), and we ensure that in the derivation no clauses with equality literals are added, then, since in S_∞ there are only clauses containing atoms, S_∞ will be $\mathcal{CS}$-saturated, and, as before, if S_0 is inconsistent the empty clause should be contained in some set S_j of the derivation.

A practical theorem prover can choose any strategy to produce fair derivations. Usually it will keep the clauses in some date structure which allows to add conclusions of inferences, choosing the premises using some heuristics like their size, but ensuring fairness by considering sooner or later any inference having an (at least apparently) persisting clause as premise. Note that adding the conclusion of such an inference to some S_j, by definition of redundancy criterion for inferences, suffices to make the inference redundant in this S_j, i.e. to assure fairness. Simplifying e.g. some clause C into C' can be done by first adding C' and then removing C which has become redundant. Similarly, simplification and deletion methods involving larger numbers of clauses also fit into the framework.

There is an interesting conceptual difference here with previous versions of Bachmair and Ganzinger's abstract redundancy notions (apart from the inclusion of constraints in the framework), namely that we consider that a clause C can be deleted in a derivation from a set S if it is redundant in $S \setminus \{C\}$ (instead of doing this if it is redundant in S). This allows to work with stronger redundancy notions for clauses which have advantages like the inclusion of *subsumption* without the need of combining the underlying orderings with special subsumption orderings.

4. A concrete inference system: the strict superposition calculus

The superposition-based inference system introduced in this section will be used as ingredient in all the constraint strategies of later sections.

In this inference system, the ordering restrictions encode (cf. section 2) the facts that the equations $s \simeq t$ and $s' \simeq t'$ are maximal wrt. $\succ$ in the premise to which they belong (strictly maximal if they belong to the succedent, except in the factoring rule, where only $s \simeq t$ is maximal), with $s \succ t$ and $s' \succ t'$ (except in equality resolution). This implies that in the strict superposition inferences (when considering only their ground instances satisfying the constraints) always the right premise is required to be bigger than the left one, and that the conclusion will be smaller than the maximal premise. No additional assumptions are made: no functional reflexivity axioms or paramodulation into variables are needed. Remind that the restrictions expressing comparisons between equations depend on their occurrence in the clauses (cf. section 2).

Our inference rule for factoring is a generalization to the equality case of "normal" factoring. For instance, if s and s' are atoms, then both t and t' are the symbol *true* and the equation $true \simeq true$ can be omitted in the antecedent.

DEFINITION 4.1. *The inference rules of the inference system $\mathcal{I}$ are the following:*

1 strict superposition right:

$$\frac{\Gamma' \rightarrow \Delta', s' \simeq t' \qquad \Gamma \rightarrow \Delta, s \simeq t}{\Gamma', \Gamma \rightarrow \Delta', \Delta, s[t']_p \simeq t} \qquad \begin{array}{lll} \mathcal{C} & : & s' \succ t' \wedge s \succ t \wedge (s \simeq t \succ s' \simeq t') \wedge \\ & & smax(s' \simeq t',\ (\ \Gamma' \rightarrow \Delta', s' \simeq t'\ )) \wedge \\ & & smax(s \simeq t,\ (\ \Gamma \rightarrow \Delta, s \simeq t\ )) \\ \mathcal{EC} & : & s|_p = s' \end{array}$$

where $s|_p \notin \mathcal{V}ars(s)$

2 strict superposition left:

$$\frac{\Gamma' \rightarrow \Delta', s' \simeq t' \qquad \Gamma, s \simeq t \rightarrow \Delta}{\Gamma', \Gamma, s[t']_p \simeq t \rightarrow \Delta', \Delta} \qquad \begin{array}{lll} \mathcal{C} & : & s' \succ t' \wedge s \succ t \wedge (s \simeq t \succ s' \simeq t') \wedge \\ & & smax(s' \simeq t',\ (\ \Gamma' \rightarrow \Delta', s' \simeq t'\ )) \wedge \\ & & max(s \simeq t,\ (\ \Gamma, s \simeq t \rightarrow \Delta\ )) \\ \mathcal{EC} & : & s|_p = s' \end{array}$$

where $s|_p \notin \mathcal{V}ars(s)$

3 equality resolution:

$$\frac{\Gamma, s \simeq t \rightarrow \Delta}{\Gamma \rightarrow \Delta} \qquad \begin{array}{lll} \mathcal{C} & : & max(s \simeq t,\ (\ \Gamma, s \simeq t \rightarrow \Delta\ )) \\ \mathcal{EC} & : & s = t \end{array}$$

4 factoring:

$$\frac{\Gamma \rightarrow \Delta, s \simeq t, s' \simeq t'}{\Gamma, t \simeq t' \rightarrow \Delta, s \simeq t} \qquad \begin{array}{lll} \mathcal{C} & : & s \succ t \wedge s' \succ t' \wedge \\ & & max(s \simeq t,\ (\ \Gamma \rightarrow \Delta, s \simeq t, s' \simeq t'\ )) \\ \mathcal{EC} & : & s = s' \end{array}$$

5. A simple example of a $\mathcal{CS}$-strategy: the ground case

In this section we will apply, as a simple example, the framework developed in the sections above to *ground* first-order clauses with equality. Ground constraints must be equivalent to *true* or *false*, i.e. we can assume that all ground constrained clauses are of the form $C [\![true]\!]$ or $C [\![false]\!]$. Since $[\![true]\!]$ is the weakest constraint, we sometimes call it the *empty* constraint, and write C instead of $C [\![true]\!]$. We now define a $\mathcal{CS}$-strategy $\mathcal{CS}_{\mathcal{G}}$ for (constrained) ground clauses: an inference system $\mathcal{I}_{\mathcal{G}}$, a constraint inheritance strategy $\mathcal{H}_{\mathcal{G}}$, a redundancy criterion for inferences $\mathcal{RI}_{\mathcal{G}}$ and a redundancy criterion for clauses $\mathcal{RC}_{\mathcal{G}}$.

DEFINITION 5.1. *The $\mathcal{CS}$-strategy $\mathcal{CS}_{\mathcal{G}} = \langle \mathcal{I}_{\mathcal{G}}, \mathcal{H}_{\mathcal{G}}, \mathcal{RI}_{\mathcal{G}}, \mathcal{RC}_{\mathcal{G}} \rangle$ is defined as follows:*

The inference system $\mathcal{I}_{\mathcal{G}}$ is the strict superposition calculus $\mathcal{I}$.
$\mathcal{H}_{\mathcal{G}}(C, T, OT, ET) = \langle C, T \wedge OT \wedge ET \rangle$
$\mathcal{RC}_{\mathcal{G}}$ is defined by: $C \in \mathcal{RC}_{\mathcal{G}}(S)$ iff $S^{\preceq C} \models C$.
$\mathcal{RI}_{\mathcal{G}}$ is defined by: $\pi \in \mathcal{RI}_{\mathcal{G}}(S)$ iff $S^{\prec D} \models D'$.

if π is an inference (by $\mathcal{I}_{\mathcal{G}}$ and $\mathcal{H}_{\mathcal{G}}$) with conclusion D' and maximal premise D.

Note that, as each ground constraint is equivalent to *true* or *false*, the constraint inheritance strategy is very simple: if one of the restrictions does not hold the inherited constraint will be false and the conclusion will be a tautology. Furthermore, let us remark that by this definition a ground clause is redundant in a set S if either it is in S or it follows from smaller clauses in S, and, similarly, an inference is redundant if its conclusion follows from clauses smaller than the maximal premise.

LEMMA 5.2. *$\mathcal{RC}_{\mathcal{G}}$ and $\mathcal{RI}_{\mathcal{G}}$ are correct redundancy criteria for clauses and inferences.*

PROOF. Both $S \models \mathcal{RC}_{\mathcal{G}}(S)$ and $S \subseteq \mathcal{RC}_{\mathcal{G}}(S)$ are immediate consequences of the definition of $\mathcal{RC}_{\mathcal{G}}$, since a clause is in $\mathcal{RC}_{\mathcal{G}}(S)$ if either it is already in S or it follows from smaller clauses in S. Therefore, $\mathcal{RC}_{\mathcal{G}}$ is a redundancy criterion for clauses.

$\mathcal{RI}_{\mathcal{G}}$ is a redundancy criterion for inferences, since for all inferences by $\mathcal{I}_{\mathcal{G}}$ and $\mathcal{H}_{\mathcal{G}}$ the conclusion is smaller than the maximal premise, which by definition of $\mathcal{RI}_{\mathcal{G}}$ implies that $\pi \in \mathcal{RI}(S)$ if the conclusion of π is in S.

Conditions $\mathcal{RC}_{\mathcal{G}}(S) \subseteq \mathcal{RC}_{\mathcal{G}}(S \cup S')$ and $\mathcal{RI}_{\mathcal{G}}(S) \subseteq \mathcal{RI}_{\mathcal{G}}(S \cup S')$ of the correctness of $(\mathcal{RC}_{\mathcal{G}}, \mathcal{RI}_{\mathcal{G}})$ follow trivially from the definition.

To prove that for every derivation $S_0, S_1, \ldots, \mathcal{RC}_{\mathcal{G}}(S_j) \subseteq \mathcal{RC}_{\mathcal{G}}(S_\infty)$, for $j \geq 0$, we will prove that $\mathcal{RC}_{\mathcal{G}}(\cup S_j) \subseteq \mathcal{RC}_{\mathcal{G}}(S_\infty)$, which is a stronger property. We derive a contradiction from the existence of a minimal (wrt. $\succ$) ground clause C in $\mathcal{RC}_{\mathcal{G}}(\cup S_j)$ s.t. C is not in $\mathcal{RC}_{\mathcal{G}}(S_\infty)$. There are two possibilities. If $C \in \cup S_j$ then C cannot be in S_∞ (since this would contradict $C \notin \mathcal{RC}_{\mathcal{G}}(S_\infty)$) and therefore there must exist some $k \geq 0$ s.t. $C \in S_k$, $C \in \mathcal{RC}_{\mathcal{G}}(S_k \setminus \{C\})$ and $S_{k+1} = S_k \setminus \{C\}$, and hence there are clauses $C_1, \ldots, C_m$ in $S_{k+1}^{\prec C}$ s.t. $\{C_1, \ldots, C_m\} \models C$. Since $C_i \in \mathcal{RC}_{\mathcal{G}}(\cup S_j)$, by minimality of C, we have $C_i \in \mathcal{RC}_{\mathcal{G}}(S_\infty)$ and therefore $C \in \mathcal{RC}_{\mathcal{G}}(S_\infty)$. Finally if $C \notin \cup S_j$ then there are clauses $C_1, \ldots, C_m$ in $\cup S_j^{\prec C}$ s.t. $\{C_1, \ldots, C_m\} \models C$, and as in the previous case $C \in \mathcal{RC}_{\mathcal{G}}(S_\infty)$ holds.

Now we will prove that for every derivation $S_0, S_1, \ldots$, it holds that $\mathcal{RI}_{\mathcal{G}}(S_j) \subseteq \mathcal{RI}_{\mathcal{G}}(S_\infty)$, for $j \geq 0$. Suppose $\pi \in \mathcal{RI}_{\mathcal{G}}(S_j)$ with maximal (wrt. $\succ$) premise D and conclusion D'. Then there are clauses $D_1, \ldots, D_m \in S_j^{\prec D}$ s.t. $\{D_1, \ldots, D_m\} \models D'$. Since $D_k \in \mathcal{RC}_{\mathcal{G}}(S_j)$ (for $k = 1 \ldots m$), by the previous result, we have $D_k \in \mathcal{RC}_{\mathcal{G}}(S_\infty)$ and therefore there are clauses $C_1, \ldots, C_n$ in $S_\infty^{\prec D}$ s.t. $\{C_1, \ldots, C_n\} \models D'$, i.e. $\pi \in \mathcal{RI}_{\mathcal{G}}(S)$. □

As $\mathcal{RC}_{\mathcal{G}}$ and $\mathcal{RI}_{\mathcal{G}}$ have been proved correct, for the completeness of $\mathcal{CS}_{\mathcal{G}}$ it only remains to be proved that every inconsistent $\mathcal{CS}_{\mathcal{G}}$-saturated set contains the empty clause. To do this, we apply Bachmair and Ganzinger's model generation technique, i.e. we define a method for associating to a set of ground clauses S an interpretation I_S, which will be expressed by a canonical set of ground rewrite rules R_S. This interpretation will be a model for every $\mathcal{CS}_{\mathcal{G}}$-saturated set S that does not contain the empty clause, which implies that any such S contains the empty clause iff it is inconsistent.

Definition 5.3. *Let S be a set of ground clauses, and let C be a clause $\Gamma \to \Delta, s \simeq t$ in S. Then C generates a rule $s \Rightarrow t$ if*

1 $R_C \not\models C$ (i.e. $R_C^* \supseteq \Gamma$ and $R_S^* \cap (\Delta, s \simeq t) = \emptyset$),
2 $s \simeq t$ is maximal (wrt. $\succ$) in C with $s \succ t$,
3 $R_C^* \not\models t \simeq t'$, for every $s \simeq t'$ in Δ, and
4 s is irreducible by R_C,

where R_C is the set of rules generated by clauses in S smaller than C (wrt. $\succ$). The set of rules generated by all clauses in S is denoted by R_S, and the interpretation I_S is defined as the congruence generated by R_S. For that reason, from now on we will sometimes write R_S^* instead of I_S.

Condition 3 above, under conditions 1 and 2, is necesary and sufficient to ensure that $R_C \cup \{s \Rightarrow t\} \not\models \Gamma \to \Delta$, i.e. by adding the rule another literal in Δ comes true: If Δ is $\Delta', s \simeq t'$ and $R_C^* \models t \simeq t'$ then $(R_C \cup \{s \Rightarrow t\})^* \models s \simeq t'$, which implies $(R_C \cup \{s \Rightarrow t\})^* \cap \Delta \neq \emptyset$. Conversely, since $R_C \not\models C$ and, by maximality, the rule $s \Rightarrow t$ can only be used in equations like $s \simeq t'$, the only way for $R_C \cup \{s \Rightarrow t\} \models \Gamma \to \Delta$ is having a literal $s \simeq t'$ in Δ s.t. $R_C^* \models t \simeq t'$. Therefore under condition 1 and 2, $R_C \cup \{s \Rightarrow t\} \models \Gamma \to \Delta$ iff there is some $s \simeq t'$ in Δ s.t. $R_C^* \models t \simeq t'$.

Similarly note that condition 2 together with condition 3 implies *strict* maximality of $s \simeq t$ in C since if it is maximal but not strict maximal is because it occurs twice in the succedent which would contradict condition 3.

LEMMA 5.4. *For every set of ground clauses S the set R_S is canonical.*

PROOF. Obviously, by the way it is constructed, R_S is terminating. Now we prove that there are no overlappings between left hand sides of rules in R_S. If a clause C generates a rule $s \Rightarrow t$ then, by definition, there are no overlappings between s and left hand sides of rules in R_C. Therefore, only rules in $R_S \setminus (R_C \cup \{s \Rightarrow t\})$ can overlap with $s \Rightarrow t$. But, since all rules $s' \Rightarrow t'$ in $R_S \setminus (R_C \cup \{s \Rightarrow t\})$ have been generated by clauses greater than C and have a left hand side which is irreducible by $R_C \cup \{s \Rightarrow t\}$, it holds that $s' \succ s$, which implies that s' cannot be a subterm of s (since $\succ$ is a simplification ordering and hence fulfills the subterm property). $\square$

LEMMA 5.5. *Let C be a clause in S of the form $\Gamma \rightarrow \Delta, s \simeq t$. If C generates a rule $s \Rightarrow t$ then $R_S^* \supseteq \Gamma$ and $R_S^* \cap \Delta = \emptyset$.*

PROOF. Since C has generated a rule, we have $R_C \not\models C$, which implies that $R_C^* \supseteq \Gamma$. Then, as $R_C \subseteq R_S$, it holds that $R_S^* \supseteq \Gamma$.

To prove $R_S^* \cap \Delta = \emptyset$, suppose $s' \simeq t'$ is an equation in Δ such that $s' \simeq t' \in R_S^*$, and wlog. $s' \succ t'$. Then, since R_S is canonical there would be a rewrite proof for $s' \simeq t'$. As seen in the proof of the previous lemma, all rules in $R_S \setminus R_C \cup \{s \Rightarrow t\}$ have left hand sides greater than s, and therefore they cannot be used in a rewrite proof of $s' \simeq t'$. Furthermore, the rules of R_C alone do not suffice, since this would imply $R_C \models C$, contradicting the first condition for generating the rule $s \Rightarrow t$. So the rule $s \Rightarrow t$ has to be applicable to $s' \simeq t'$, and, moreover, since $s \simeq t$ is maximal in C, it can only be applied at the topmost position of s' (note that then s' and s are the same term), obtaining $t \simeq t'$, which should follow from R_C (since $s \Rightarrow t$ cannot be used any more as $s \succ t$ and $s \succ t'$), but this would contradict the third condition for the generation of $s \Rightarrow t$. $\square$

LEMMA 5.6. *Let S be a $\mathcal{CS}_{\mathcal{G}}$-saturated set of clauses not containing the empty clause. Then $R_S^* \models S$.*

PROOF. Let C be the minimal (wrt. $\succ$) clause in S, such that $R_S^* \not\models C$. We will derive a contradiction from the existence of such a clause. There are several cases to be analyzed, depending on the maximal equation in C:

1 Since $R_S^* \not\models C$, the clause C cannot be a clause with a trivial maximal equation of the form $t \simeq t$ in its succedent.

2 Let C be a clause $\Gamma \rightarrow \Delta, s \simeq t$, with a maximal equation $s \simeq t$, and $s \succ t$. Since $R_S^* \not\models C$, we have $R_S^* \supseteq \Gamma$ and $R_S^* \cap (\Delta, s \simeq t) = \emptyset$. Then the clause C has not generated the rule $s \Rightarrow t$. This must be because conditions 3) or 4) of definition 5.3 do not hold:

(a) If condition 3) does not hold, then Δ must be of the form $\Delta', s \simeq t'$ and $R^*_C \models t \simeq t'$ (and therefore $R^*_S \models t \simeq t'$). In this case the following inference by factoring

$$\frac{\Gamma \rightarrow \Delta', s \simeq t, s \simeq t'}{\Gamma, t \simeq t' \rightarrow \Delta', s \simeq t}$$

can be made. As $R^*_S \not\models C$ and $R^*_S \models t \simeq t'$, its conclusion D is not satisfied by R^*_S.
Since S is $\mathcal{CS}_\mathcal{G}$-saturated, the inference π is redundant in S, i.e. D follows from clauses D_i in $S^{\prec C}$. The fact that D is not satisfied by R^*_S implies that at least one of the D_i is not satisfied in R^*_S either, which contradicts the fact that C is the minimal such clause.

(b) If condition 4) does not hold, let s be reducible by R_C, with a rule $s' \Rightarrow t'$ generated by a clause C' smaller than C. Let C' be a clause $\Gamma' \rightarrow \Delta', s' \simeq t'$ and $s|_p = s'$. Note that, by lemma 5.5, $R^*_S \supseteq \Gamma'$ and $R^*_S \cap \Delta' = \emptyset$. Then the inference π by strict superposition right

$$\frac{\Gamma' \rightarrow \Delta', s' \simeq t' \qquad \Gamma \rightarrow \Delta, s \simeq t}{\Gamma', \Gamma \rightarrow \Delta', \Delta, s[t']_p \simeq t}$$

can be made, since all the conditions for its application hold: we assume that $s \simeq t$ and $s' \simeq t'$ are strictly maximal in C and C' respectively, since if they are maximal in their clause but not strict maximal, this is because they occur twice in the succedent which contradicts condition 3. Moreover its conclusion D is not satisfied by R^*_S: $R^*_S \supseteq (\Gamma', \Gamma)$ and $R^*_S \cap (\Delta', \Delta, s[t']_p \simeq t) = \emptyset$ (since $R^*_S \models s' \simeq t'$).
Since S is $\mathcal{CS}_\mathcal{G}$-saturated, the inference is redundant in S, i.e. D follows from clauses in $S^{\prec C}$, which leads to a contradiction as in the previous case.

3 If C is a clause $\Gamma, t \simeq t \rightarrow \Delta$, where $t \simeq t$ is maximal in C then consider the following equality resolution inference:

$$\frac{\Gamma, t \simeq t \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$

The conclusion of this inference is not satisfied by R^*_S and, since the inference is redundant, as before a contradiction is obtained.

4 The only remaining case is that C is a clause $\Gamma, s \simeq t \rightarrow \Delta$, where $s \simeq t$ is maximal in C and $s \succ t$. In this case $R^*_S \supseteq \Gamma$, $R^*_S \cap \Delta = \emptyset$ and $R^*_S \models s \simeq t$, because $R^*_S \not\models C$. Then, since R_S is canonical the term s must be reducible in some position p by a rule $s' \Rightarrow t'$ in R_S (i.e. $s|_p = s'$) generated by a clause C' of the form $\Gamma' \rightarrow \Delta', s' \simeq t'$. Note that, as in case 2.b, $R^*_S \supseteq \Gamma'$ and $R^*_S \cap \Delta' = \emptyset$. Then the following inference π by strict superposition left can be made:

$$\frac{\Gamma' \rightarrow \Delta', s' \simeq t' \qquad \Gamma, s \simeq t \rightarrow \Delta}{\Gamma', \Gamma, s[t']_p \simeq t \rightarrow \Delta', \Delta}$$

In this inference, C is the maximal premise, and, as in case 2.b, its conclusion is not satisfied by R^*_S. This implies as before that, since the inference is redundant, a contradiction is obtained. $\square$

THEOREM 5.7. *The constraint strategy $\mathcal{CS}_{\mathcal{G}}$ is complete.*

PROOF. By lemma 5.2 we know that $\mathcal{RC}_{\mathcal{G}}$ and $\mathcal{RI}_{\mathcal{G}}$ are correct, and by the previous lemma every $\mathcal{CS}_{\mathcal{G}}$-saturated set of clauses S not containing the empty clause has a model, namely R_S^*, and therefore S will be inconsistent iff it contains the empty clause. □

Note that for proving that $\mathcal{CS}_{\mathcal{G}}$-saturated inconsistent sets S contain the empty clause we have not used the second property of the definition of $\mathcal{CS}_{\mathcal{G}}$-saturated sets, i.e. the existence of a pure set S' s.t. $S' \models S$ and S' is redundant in S. The reason is that this property is not needed if the constraints do not play any role. In fact, this property trivially holds for every set of ground clauses S, since they are all pure.

6. Saturation with constrained clauses

In this section we extend the results given in the previous section to general clauses with constraints by means of a $\mathcal{CS}$-strategy $\mathcal{CS}_{\mathcal{OB}} = \langle \mathcal{I}_{\mathcal{OB}}, \mathcal{H}_{\mathcal{OB}}, \mathcal{RC}_{\mathcal{OB}}, \mathcal{RI}_{\mathcal{OB}} \rangle$. In this strategy, $\mathcal{I}_{\mathcal{OB}}$ will be again the strict superposition calculus given in section 4.

Concerning the constraint inheritance strategy $\mathcal{H}_{\mathcal{OB}}$, the proofs in this section also apply to many other possibilities, like the inheritance strategies $\mathcal{H}_{\mathcal{O}}$, $\mathcal{H}_{\mathcal{B}}$ and $\mathcal{H}_{\mathcal{E}}$ given below, but here we develop the proofs for $\mathcal{H}_{\mathcal{OB}}$, which is the most restrictive case, namely the one in which the maximum amount of restrictions produced by the inferences is inherited (that is the reason why this strategy is called $\mathcal{CS}_{\mathcal{OB}}$: ordering constraint inheritance and basicness, i.e. equality constraint inheritance).

The constraint inheritance strategies given below depend on an algorithm `unsat` that is supposed to perform some finite test to check whether a given constraint is unsatisfiable. We do not need this constraint check to be complete, but only sound, i.e. if $\texttt{unsat}(T)$ is true then the constraint T is indeed unsatisfiable. Such constraint checkers turn out to be very useful in practice, as shown in our experiments with the *Saturate* system, because the satisfiability of this kind of constraints is decidable but NP-complete (Nieuwenhuis, 1993). Of course there are also many cases where indeed the full power of a complete constraint solver is preferable, like when the aim is to compute as few as possible additional clauses, for example when trying to obtain a saturated set for a given theory (which is very useful, as we will see). Cf. section 8 for a further discussion of constraint solving.

The following strategies correspond to, respectively, inheritance of both the ordering and equality restrictions ($\mathcal{H}_{\mathcal{OB}}$), only the ordering restrictions ($\mathcal{H}_{\mathcal{O}}$), only the equality restrictions ($\mathcal{H}_{\mathcal{B}}$) and no restrictions ($\mathcal{H}_{\mathcal{E}}$). In fact many other strategies are possible by inheriting *parts* of the restrictions, depending on the constraint satisfaction tests. For example one can perform a test eliminating some more easily solvable or simplifiable parts of the constraints, and inherit the remaining part. In fact, this is the most usual practical way to deal with constraints, where constraint solving is done incrementally (e.g. in the *Saturate* system).

$$\mathcal{H}_{\mathcal{OB}}(C,T,OT,ET) \quad = \quad \langle C,\ T \wedge ET \wedge OT\rangle$$

$$\mathcal{H}_{\mathcal{O}}(C,T,OT,ET) \quad = \begin{cases} \langle C\sigma,\ (T \wedge OT)\sigma\rangle & \text{if } ET \text{ has a } mgu\ \sigma \\ \langle C,\ \mathit{false}\rangle & \text{otherwise} \end{cases}$$

$$\mathcal{H}_{\mathcal{B}}(C,T,OT,ET) \quad = \begin{cases} \langle C,\ T \wedge ET\rangle & \text{if } T \wedge ET \text{ has a } mgu\ \sigma \wedge\neg\mathtt{unsat}(OT\sigma) \\ \langle C,\ \mathit{false}\rangle & \text{otherwise} \end{cases}$$

$$\mathcal{H}_{\mathcal{E}}(C,\mathit{true},OT,ET) \quad = \begin{cases} \langle C\sigma,\ \mathit{true}\rangle & \text{if } ET \text{ has a } mgu\ \sigma \wedge\neg\mathtt{unsat}(OT\sigma) \\ \langle C,\ \mathit{false}\rangle & \text{otherwise} \end{cases}$$

As we can see, the inherited ordering constraints can become quickly very restrictive, since many conditions are encoded within the part of the constraints generated in every inference.

To prove the completeness of $\mathcal{CS}_{\mathcal{OB}}$ we apply a new lifting technique, consisting in first proving our results for the particular set of instances with, in some sense, *irreducible substitutions* and, after that, generalizing these results to *all* instances. This allows to avoid the lifting problems of examples like the second one of the introduction. Apart from this difference, to prove that $\mathcal{CS}_{\mathcal{OB}}$ is complete we proceed like in the ground case. Given a saturated set of constrained clauses S, we generate an interpretation for the set of irreducible ground instances of S (which is a set of ground clauses like in the previous section). This interpretation R^*_S is shown to be a model of the set of all irreducible ground instances of S. Finally, using the second part of the definition of saturated sets, namely the existence of a pure set for S (or the lack of equality literals in S), we prove that R^*_S is a model of *all* instances of S.

Let us first define more precisely what we mean by irreducible instances:

DEFINITION 6.1. *Given a clause C and a ground substitution σ, x is a succedent-top-left variable of C wrt. σ, denoted by $x \in stlvars(C,\sigma)$, if x only appears in the succedent of C in equations of the form $x \simeq t$, where $x\sigma \succ t\sigma$.*

Note that if x is a succedent-top-left variable of C then x cannot occur as argument of any function symbol.

DEFINITION 6.2. *Let R be a set of ground rewrite rules, let $C\sigma$ be a ground instance of a clause C and let x be a variable in $\mathcal{V}ars(C)$. Then x is said to be variable irreducible in $C\sigma$ wrt. R if,*

1 *$x\sigma$ is irreducible wrt. R, or*
2 *$x \in stlvars(C,\sigma)$ and $x\sigma$ is irreducible wrt. all rules $l \Rightarrow r \in R$ s.t. $x\sigma \simeq t\sigma \succ l \simeq r$ for all $x \simeq t$ in C.*

If this property holds for all x in $\mathcal{V}ars(C)$ then $C\sigma$ is variable irreducible wrt. R.

DEFINITION 6.3. *Let π be an inference (by $\mathcal{I}_{\mathcal{OB}}$ and $\mathcal{H}_{\mathcal{OB}}$) with conclusion $C[\![T]\!]$ and* *premises* $C_1 [\![T_1]\!], \ldots, C_n [\![T_n]\!]$. *Furthermore, let* σ *be a ground substitution such that* $T_1\sigma \wedge \ldots \wedge T_n\sigma \wedge T\sigma \equiv true$.

Then, if there exists an inference by $\mathcal{I}_{\mathcal{OB}}$ *and* $\mathcal{H}_{\mathcal{OB}}$ *with premises* $C_1\sigma \ldots C_n\sigma$ *and conclusion* $C\sigma$*, this inference, denoted* $\pi\sigma$*, is called a* ground instance *with* σ *of* π.

Definition 6.4. *Let* R *be a set of ground rewrite rules,* $C [\![T]\!]$ *a constrained clause,* S *a set of constrained clauses and* π *an inference (by* $\mathcal{I}_{\mathcal{OB}}$ *and* $\mathcal{H}_{\mathcal{OB}}$*).*

1 *The set of variable irreducible (wrt.* R*) ground instances of* $C [\![T]\!]$ *is denoted by* $irred_R(C [\![T]\!])$*, i.e.*
$irred_R(C [\![T]\!]) = \{C\sigma \mid T\sigma \equiv true \wedge \sigma \text{ ground} \wedge C\sigma \text{ variable irreducible wrt. } R\}$.
2 *Similarly,* $irred_R(S) = \{C\sigma \mid C [\![T]\!] \in S \wedge C\sigma \in irred_R(C [\![T]\!])\}$.
3 $irred_R(\pi)$ *is the set of ground instances* $\pi\sigma$ *of* π *such that* $C\sigma \in irred_R(C [\![T]\!])$ *for each* $C [\![T]\!]$ *that is premise or conclusion of* π.

Definition 6.5. *The* $\mathcal{CS}$*-strategy* $\mathcal{CS}_{\mathcal{OB}} = \langle \mathcal{I}_{\mathcal{OB}}, \mathcal{H}_{\mathcal{OB}}, rcob, \mathcal{RI}_{\mathcal{OB}}\rangle$ *is defined by:*

The inference system $\mathcal{I}_{\mathcal{OB}}$ *is the strict superposition calculus* $\mathcal{I}$ *(cf. 4).*
$\mathcal{H}_{\mathcal{OB}}(C, T, ET, OT) = \langle C,\ T \wedge ET \wedge OT\rangle$
$C [\![T]\!] \in \mathcal{RC}_{\mathcal{OB}}(S)$ *iff* $R \cup irred_R(S)^{\preceq C\sigma} \models C\sigma$
$\pi \in \mathcal{RI}_{\mathcal{OB}}(S)$ *iff* $R \cup irred_R(S)^{\prec D} \models D'$

for every set of rules R *s.t.* $\rightarrow_R \subseteq \succ$ *and* $C\sigma \in irred_R(C [\![T]\!])$*, and for every* $\pi\sigma \in irred_R(\pi)$ *with conclusion* D' *and maximal premise (wrt.* $\succ$*)* D.

Note that these definitions of $\mathcal{RC}_{\mathcal{OB}}$ and $\mathcal{RI}_{\mathcal{OB}}$ provide abstract redundancy criteria. At the end of this section we will define more practical sufficient conditions for proving the redundancy of clauses and inferences.

Lemma 6.6. $\mathcal{RC}_{\mathcal{OB}}$ *and* $\mathcal{RI}_{\mathcal{OB}}$ *are correct redundancy criteria for clauses and inferences.*

Proof. First we prove that $\mathcal{RC}_{\mathcal{OB}}$ is a redundancy criterion for clauses: $S \subseteq \mathcal{RC}_{\mathcal{OB}}(S)$ is an immediate consequence of the definition of $\mathcal{RC}_{\mathcal{OB}}$. For $S \models \mathcal{RC}_{\mathcal{OB}}(S)$, suppose that $C [\![T]\!] \in \mathcal{RC}_{\mathcal{OB}}(S)$ then, by definition, for every set of rules R s.t. $\rightarrow_R \subseteq \succ$ and for every instance $C\sigma \in irred_R(C [\![T]\!])$, there are clauses $C_1, \ldots, C_n$ in $irred_R(S)^{\preceq C}$ s.t. $R \cup \{C_1, \ldots, C_m\} \models C$. But then, since the empty set of rules $\emptyset$ satisfies $\rightarrow_\emptyset \subseteq \succ$, and that all ground instances of a clause are variable irreducible wrt. $\emptyset$, we have that for every ground instance $C\sigma$ of $C [\![T]\!]$ there are instances $C_1, \ldots, C_n$ of clauses in S s.t. $\{C_1, \ldots, C_m\} \models C$ (note that we are considering the empty set of rules), which implies that $S \models C [\![T]\!]$.

$\mathcal{RI}_{\mathcal{OB}}$ is a redundancy criterion for inferences, since for all ground instances of an inference by $\mathcal{I}_{\mathcal{OB}}$ and $\mathcal{H}_{\mathcal{OB}}$ the conclusion is smaller than the maximal premise, which by definition of $\mathcal{RI}_{\mathcal{OB}}$ implies that $\pi \in \mathcal{RI}(S)$ if the conclusion of π is in S.

Conditions $\mathcal{RC}_{\mathcal{OB}}(S) \subseteq \mathcal{RC}_{\mathcal{OB}}(S \cup S')$ and $\mathcal{RI}_{\mathcal{OB}}(S) \subseteq \mathcal{RI}_{\mathcal{OB}}(S \cup S')$ of the correctness of $(\mathcal{RC}_{\mathcal{OB}}, \mathcal{RI}_{\mathcal{OB}})$ follow trivially from the definition.

To prove $\mathcal{RC}_{\mathcal{OB}}(S_j) \subseteq \mathcal{RC}_{\mathcal{OB}}(S_\infty)$, for $j \geq 0$ and for every derivation $S_0, S_1, \ldots$, we will prove that $\mathcal{RC}_{\mathcal{OB}}(\cup S_j) \subseteq \mathcal{RC}_{\mathcal{OB}}(S_\infty)$, which is a stronger property. To show it, we will derive a contradiction from the existence of a minimal (wrt. $\succ$) instance $C\sigma$ of a clause in $\cup S_j$ s.t. there exists a rewrite set R (with $\rightarrow_R \subseteq \succ$) s.t. $C\sigma$ is variable irreducible wrt. R and

there are $D_1, \ldots, D_m$ in $irred_R(\cup S_j)^{\preceq C\sigma}$ s.t. $R \cup \{D_1, \ldots, D_m\} \models C\sigma$, but
there are no $D'_1, \ldots, D'_n$ in $irred_R(S_\infty)^{\preceq C\sigma}$ s.t. $R \cup \{D'_1, \ldots, D'_m\} \models C\sigma$.

Then there are two possibilities. If $C\sigma \in irred_R(\cup S_j)$ then $C\sigma$ cannot be in $irred_R(S_\infty)$ (since this would contradict the assumption) and therefore there must exists some $k \geq 0$ s.t. $C[\![T]\!] \in S_k$, (where $C\sigma$ is an instance of $C[\![T]\!]$), $C[\![T]\!] \in \mathcal{RC}_{\mathcal{OB}}(S_k \setminus \{C[\![T]\!]\})$, S_{k+1} is $S_k \setminus \{C[\![T]\!]\}$ and $C\sigma \notin irred_R(S_{k+1})$, and hence there are clauses $C_1, \ldots, C_p$ in $irred_R(S_{k+1})^{\prec C\sigma}$ s.t. $R \cup \{C_1, \ldots, C_p\} \models C\sigma$. Since $C_i \in irred_R(\cup S_j)$, by minimality of $C\sigma$, we have that there are instances of clauses $C'_1, \ldots, C'_q$ in $irred_R(S_\infty)^{\prec C\sigma}$ s.t. $R \cup \{C'_1, \ldots, C'_q\} \models \{C_1, \ldots, C_p\}$, and therefore $R \cup \{C'_1, \ldots, C'_q\} \models C\sigma$, contradicting the assumption. Finally, if $C\sigma \notin \cup irred_R(\cup S_j)$ there are instances of clauses $C_1, \ldots, C_p$ in $irred_R(\cup S_j)^{\prec C\sigma}$ s.t. $R \cup \{C_1, \ldots, C_p\} \models C\sigma$, and as in the previous case a contradiction is obtained.

Now we prove that for every derivation $S_0, S_1, \ldots$, $\mathcal{RI}_{\mathcal{OB}}(S_j) \subseteq \mathcal{RI}_{\mathcal{OB}}(S_\infty)$, for $j \geq 0$. Suppose $\pi \in \mathcal{RI}_{\mathcal{OB}}(S_j)$, then for every set of rules R s.t. $\rightarrow_R \subseteq \succ$ and for every ground instance $\pi\sigma \in irred_R(\pi)$, with conclusion D' and maximal premise (wrt. $\succ$) D, there exist instances $D_i\sigma_i$ in $irred_R(S_j)^{\prec D}$, for $i : 1 \ldots m$, such that $R \cup \{D_1\sigma_1, \ldots, D_m\sigma_m\} \models D'$. Now, since $S_j \subseteq \mathcal{RC}_{\mathcal{OB}}(S_j)$ and $\mathcal{RC}_{\mathcal{OB}}(S_j) \subseteq \mathcal{RC}_{\mathcal{OB}}(S_\infty)$, $S_j \subseteq \mathcal{RC}_{\mathcal{OB}}(S_\infty)$ and then for each $D_i\sigma_i$ there must be clauses $C_1, \ldots, C_n$ in $irred_R(S_\infty)$ with $D_i\sigma_i \succeq C_k$, for $k : 1 \ldots n$ (and hence $C_k \in irred_R(S_\infty)^{\prec D}$), s.t. $R \cup \{C_1, \ldots, C_n\} \models D_i\sigma_i$. Therefore π is redundant in S_∞. □

DEFINITION 6.7. *Let S be a set of constrained clauses, and let $C\sigma$ be a ground instance of a clause $C[\![T]\!]$ in S, s.t. $C\sigma$ is variable irreducible wrt. $R_{C\sigma}$. Then $C\sigma$ generates a rule $s\sigma \Rightarrow t\sigma$ iff it generates the rule as specified in definition 5.3.*

As in the previous section, the set of rules generated by all ground instances of clauses in S is denoted by R_S.

Note that as for the ground case, R_S fulfils an adapted version of lemmas 5.4 and 5.5:

LEMMA 6.8. *For every set of constrained clauses S the set R_S is canonical.*

LEMMA 6.9. *If a ground instance $\Gamma \rightarrow \Delta, s \simeq t$ of a constrained clause in S generates a rule $s \Rightarrow t$ then $R_S^* \supseteq \Gamma$ and $R_S^* \cap \Delta = \emptyset$.*

LEMMA 6.10. *Let S be a set of constrained clauses. If a ground instance $C\sigma$ of a clause $C[\![T]\!]$ in S, with C of the form $\Gamma \rightarrow \Delta, s \simeq t$ generates a rule $s\sigma \Rightarrow t\sigma$ then $C\sigma$ is variable irreducible wrt. R_S.*

PROOF. We know $C\sigma$ is variable irreducible wrt. $R_{C\sigma}$, since it has generated a rule, so we only have to show the variable irreducibility of x in $\mathcal{V}ars(C)$ wrt. rules $l \Rightarrow r$ in

$R_S \setminus R_{C\sigma}$. Now if $s\sigma \succ x\sigma$ then no such rule $l \Rightarrow r$ reduces $x\sigma$, since $l \succeq s\sigma \succ x\sigma$. Otherwise, since $s\sigma$ is a maximal term in $C\sigma$ and $s\sigma \simeq t\sigma$ is strict maximal in $C\sigma$, we have that $x\sigma = s\sigma$ and that x can only occur in the succedent of C in equations like $x \simeq t'$ with $x\sigma \succ t'\sigma$, i.e. $x \in stlvars(C, \sigma)$. Then the only rule that can reduce $x\sigma$ is $s\sigma \Rightarrow t\sigma$, but since $s\sigma \simeq t\sigma \succeq x\sigma \simeq t'\sigma$ for every $x \simeq t'$ in C (by strict maximality of $s\sigma \simeq t\sigma$ in $C\sigma$) and $x \in stlvars(C, \sigma)$ this rule cannot be considered, i.e. x is variable irreducible in $C\sigma$ wrt. R_S. □

LEMMA 6.11. *Let S be a $\mathcal{CS}_{\mathcal{OB}}$-saturated set of constrained clauses not containing the empty clause. Then $R_S^* \models irred_{R_S}(S)$.*

PROOF. We proceed as in lemma 5.6 for the ground case, but considering variable irreducible instances of clauses. We will derive a contradiction from the existence of a minimal (wrt. $\succ$) instance $C\sigma$ in $irred_{R_S}(S)$ of a clause $C\,[\![T]\!]$ in S, such that $R_S^* \not\models C\sigma$. As in the ground case there are several cases to be analyzed, depending on the maximal equation in $C\sigma$:

1 Since $R_S^* \not\models C\sigma$, the clause $C\sigma$ cannot be a clause with a maximal equation of the form $t\sigma \simeq t'\sigma$, where $t\sigma$ and $t'\sigma$ are the same terms, in its succedent.

2 Let $C\sigma$ be a clause $\Gamma\sigma \rightarrow \Delta\sigma, s\sigma \simeq t\sigma$, with a maximal equation $s\sigma \simeq t\sigma$, and $s\sigma \succ t\sigma$. Since $R_S^* \not\models C\sigma$, the clause $C\sigma$ has not generated the rule $s\sigma \Rightarrow t\sigma$. This must be because conditions 3) or 4) of definition 5.3 do not hold:

   (a) If condition 3) does not hold, then Δ must be of the form $\Delta', s' \simeq t'$ where $s'\sigma$ is the same term as $s\sigma$ and $R_C^* \models t\sigma \simeq t'\sigma$. In this case the following inference by factoring

$$\frac{\Gamma \rightarrow \Delta, s \simeq t, s' \simeq t'\,[\![T]\!]}{\Gamma, t \simeq t' \rightarrow \Delta, s \simeq t\,[\![T \wedge OC \wedge s = s']\!]}$$

   can be made and its conclusion has a ground instance $D\sigma$ which is not satisfied by R_S^*. Moreover $D\sigma$ is variable irreducible wrt. R_S: if $x \notin stlvars(C, \sigma)$ then $x\sigma$ is irreducible by R_S. Otherwise, either $x \notin \mathcal{V}ars(D)$ or x is also in $stlvars(D, \sigma)$ and appears in the same literals as in C.
   Since S is $\mathcal{CS}_{\mathcal{OB}}$-saturated, the inference π is redundant in S, therefore there exist clauses D_i in $irred_{R_S}(S)^{\prec C\sigma}$ such that $R_S \cup \{D_1, \ldots, D_m\} \models D\sigma$ (note that $\rightarrow_{R_S} \subseteq \succ$ by construction). The fact that $D\sigma$ is not satisfied by R_S^* implies that at least one of the D_i is not satisfied in R_S^* either, which contradicts the fact that $C\sigma$ is a minimal such clause.

   (b) If condition 4) does not hold, let $s\sigma$ be reducible by $R_{C\sigma}$, with a rule $s'\sigma \Rightarrow t'\sigma$ generated by a clause $C'\sigma$ smaller than $C\sigma$ (we can use the same σ since C and C' do not share variables). Let C' be a clause $\Gamma' \rightarrow \Delta', s' \simeq t'$ and $s\sigma|_p = s'\sigma$. Then $s\sigma|_p$ cannot be bellow a variable, as $C\sigma$ is variable irreducible wrt. R_S: if s were in $stlvars(C, \sigma)$, then it would be irreducible wrt. rules smaller than $s\sigma \simeq t\sigma$, which is not the case; Otherwise, $x\sigma$ is irreducible wrt. R_S, for all $x \in \mathcal{V}ars(s)$.

Then the inference π by strict superposition right

$$\frac{\Gamma' \to \Delta', s' \simeq t' \,[\![T']\!] \qquad \Gamma \to \Delta, s \simeq t \,[\![T]\!]}{\Gamma', \Gamma \to \Delta', \Delta, s[t']_p \simeq t \,[\![T \wedge T' \wedge OC \wedge s|_p = s']\!]}$$

can be made, and, since all the conditions for its application hold (we assume that $s\sigma \simeq t\sigma$ and $s'\sigma \simeq t'\sigma$ are strictly maximal $C\sigma$ and $C'\sigma$ respectively, since otherwise condition 3 would fail in both cases), its conclusion D has a ground instance $D\sigma$ of the form $\Gamma'\sigma, \Gamma\sigma \to \Delta'\sigma, \Delta\sigma, s\sigma[t'\sigma]_p \simeq t\sigma$ that is not satisfied by R_S^*. Moreover $D\sigma$ is irreducible wrt. R_S: since (by lemma 6.10) $C'\sigma$ is also variable irreducible, for every variable $x \in \mathcal{V}ars(D)$, either $x\sigma$ is irreducible wrt. R_S or $x \in stlvars(C, \sigma)$ or $x \in stlvars(C', \sigma)$ in some $x \simeq t_x$ Then also $x \in stlvars(D, \sigma)$ in the same $x \simeq t_x$, and therefore x is variable irreducible in $D\sigma$ wrt. R_S.

Since S is $\mathcal{CS}_{\mathcal{OB}}$-saturated, the inference is redundant in S, therefore there exist clauses D_i in $irred_{R_S}(S)^{\prec C\sigma}$ such that $R_S \cup \{D_1, \ldots, D_m\} \models D\sigma$, which leads to a contradiction as in the previous case.

3 If C is a clause $\Gamma, s \simeq t \to \Delta$, where $s\sigma \simeq t\sigma$ is maximal in $C\sigma$, and $s\sigma$ is $t\sigma$ then consider the following equality resolution inference:

$$\frac{\Gamma, s \simeq t \to \Delta \,[\![T]\!]}{\Gamma \to \Delta \,[\![T \wedge s = t]\!]}$$

The conclusion D of this inference has a ground instance $D\sigma$ which is not satisfied by R_S^*. Moreover, the variable irreducibility of $C\sigma$ wrt. R_S implies the variable irreducibility of $D\sigma$. Since the inference is redundant, as before a contradiction is obtained.

4 The only remaining case is that C is a clause $\Gamma, s \simeq t \to \Delta$, where $s\sigma \simeq t\sigma$ is maximal in $C\sigma$ and $s\sigma \succ t\sigma$. In this case $R_S^* \models s\sigma \simeq t\sigma$, because $R_S^* \not\models C\sigma$. Then $s\sigma$ must be reducible by a rule $s'\sigma \Rightarrow t'\sigma$ in R_S generated by a clause C' of the form $\Gamma' \to \Delta', s' \simeq t'$ (we can use the same σ since C and C' do not share variables), where $s\sigma|_p = s'\sigma$ and $s\sigma|_p$ cannot be bellow a variable since for all $x \in \mathcal{V}ars(s)$ we have $x \notin stlvars(C, \sigma)$ and therefore the variable irreducibility of $C\sigma$ wrt. R_S implies that $x\sigma$ is irreducible wrt. R_S.

Then the following inference π by strict superposition left can be made:

$$\frac{\Gamma' \to \Delta', s' \simeq t' \,[\![T']\!] \qquad \Gamma, s \simeq t \to \Delta \,[\![T]\!]}{\Gamma', \Gamma, s[t']_p \simeq t \to \Delta', \Delta \,[\![T \wedge T' \wedge OC \wedge s|_p = s']\!]}$$

In this inference, for the corresponding instance $C\sigma$ is the maximal premise, and, as in case 2.b, its conclusion D has a ground instance $D\sigma$ which is variable irreducible and also not satisfied by R_S^*. This implies as before that, since the inference is redundant, a contradiction is obtained.

□

LEMMA 6.12. *If S is a pure set of clauses then for every ground set of rules R s.t. $\to_R \subseteq \succ$, we have $R \cup irred_R(S) \models S$.*

PROOF. If S is a pure set then for all clauses $C\,[\![T]\!]$ in S the constraint T has no lower bounded variables: $T\sigma \equiv true$ implies $T\sigma' \equiv true$ for every ground set of rules R s.t. $\rightarrow_R \subseteq \succ$ and for every pair of substitutions σ and σ' s.t. σ' is the normal form of σ wrt. R This holds (by induction on the number of variables in T), since if $\rightarrow_R \subseteq \succ$ then $x\sigma \succ x\sigma'$ for every variable in T. Then for every ground instance $C\sigma$ of a clause in S, there is a ground instance $C\sigma'$ in $irred_R(S)$ s.t. $R \cup C\sigma' \models C\sigma$. Therefore $R \cup irred_R(S) \models S$. □

LEMMA 6.13. *Let S be a $\mathcal{CS}_{\mathcal{OB}}$-saturated set of clauses. Then S is inconsistent if, and only if, the empty clause belongs to S.*

PROOF. The right-to-left implication is trivial. For the reverse implication, we prove that S is consistent, i.e. it has a model, if the empty clause does not belong to S. In that case, by lemma 6.11, $R_S^* \models irred_{R_S}(S)$ because S is a $\mathcal{CS}_{\mathcal{OB}}$-saturated set, and does not contain the empty clause. Moreover, its completeness implies that either all clauses in S contain only atoms (i.e. all literals are of the form $t \simeq true$, where t is not a variable) or there exists a pure set S' s.t. $S' \models S$ and S' is redundant in S.

In the first case R_S only contains rules $t \Rightarrow true$ where the top-symbol of t is a predicate symbol. Therefore, since a variable never can be instantiated by a term with a predicate top-symbol, it holds that $irred_{R_S}(S)$ is S and hence $R_S^* \models S$.

If we are in the second case, then, since S' is in $\mathcal{RC}_{\mathcal{OB}}(S)$, we have $R_S \cup irred_{R_S}(S) \models irred_{R_S}(S')$ (note that R_S is a ground set of rules s.t. $\rightarrow_{R_S} \subseteq \succ$) and hence $R_S^* \models irred_{R_S}(S')$, since $R_S^* \models irred_{R_S}(S)$. By lemma 6.12, $R_S \cup irred_{R_S}(S') \models S'$, and therefore $R_S^* \models S'$. Now, since $S' \models S$, it follows that $R_S^* \models S$. □

As a direct consequence of lemmas 6.6 and 6.13 the completeness of $\mathcal{CS}_{\mathcal{OB}}$ holds:

THEOREM 6.14. *The constraint strategy $\mathcal{CS}_{\mathcal{OB}}$ is complete.*

6.1. Redundancy notions for constrained saturation

In this section we give sufficient conditions to ensure the redundancy of clauses and inferences. Practical methods for proving the redundancy of clauses and inferences during the theorem proving procedures can be easily designed according to these notions (cf. Nivela and Nieuwenhuis, 1993 for examples).

First, to get some intuition, let us look at an example showing that the usual simplification techniques are not compatible with constraint inheritance, even if the initial set has no constraints at all. For simplicity, we consider here the constraint inheritance strategy $\mathcal{H}_{\mathcal{B}}$, i.e. the pure basic strategy, and a simplification step in which $f(g(a))$ is simplified into $f(b)$ by demodulation with the instance $g(a) \simeq b$ of $g(x) \simeq b\,[\![x = a]\!]$. This is indeed a very natural kind of simplification, which is perfectly admissible in the case without inheriting constraints (i.e. the strategy $\mathcal{H}_{\mathcal{E}}$) as we will see later on.

EXAMPLE 6.15. *Consider an LPO with $f \succ g \succ a \succ b$ and the inconsistent set of four initial clauses:*

1.		$\rightarrow a \simeq b$
2.		$\rightarrow f(g(x)) \simeq g(x)$
3.		$\rightarrow f(g(a)) \simeq b$
4.	$g(b) \simeq b$	$\rightarrow$

Now we could generate a saturation process as follows:

5.	$\rightarrow g(x) \simeq b$	$[\![x = a]\!]$	*(by superposition of* 3 *on* 2*)*
6.	$\rightarrow f(b) \simeq g(x)$	$[\![x = a]\!]$	*(by superposition of* 5 *on* 2*)*
$3'$.	$\rightarrow f(b) \simeq b$		*(simplifying* 3 *by* 5*)*

Finally, the set $\{1, 2, 3', 4, 5, 6\}$ *is closed under the inference rules, but the empty clause has not been generated.* □

LEMMA 6.16. *Let* S *be a set of constrained clauses. A clause* $C\,[\![T]\!]$ *is in* $\mathcal{RC}_{\mathcal{OB}}(S)$ *if for every ground instance* $C\sigma$ *of it, there are ground instances* $D_i\sigma_i$ *of clauses* $D_i\,[\![T_i]\!]$ *in* $S^{\preceq C\sigma}$ *with* $\{D_1\sigma_1, \ldots, D_m\sigma_m\} \models C\sigma$ *and for all* $i = 1 \ldots m$ *and variable* x *in* D_i*:*

1 *x is not lower bounded by T_i, or else*

2 *there is a variable $y \in \mathcal{V}ars(C)$ s.t. $x\sigma_i = y\sigma$ and $y \notin stlvars(C, \sigma)$.*

PROOF. We have to prove that $R \cup irred_R(S)^{\preceq C\sigma} \models C\sigma$ for every set of rules R s.t. $\rightarrow_R \subseteq \succ$ and $C\sigma \in irred_R(C\,[\![T]\!])$.

We know that $S^{\preceq C} \models C\sigma$ for all ground instances $C\sigma$ of C, so in particular for all R we have $R \cup S^{\preceq C} \models C\sigma$ if $C\sigma$ is an irreducible instance wrt. R of C. It remains to be shown that there are indeed instances in $irred_R(S)^{\preceq C\sigma}$ that can be used, in case of any of the $D_i\sigma_i$ is not in $irred_R(S)^{\preceq C\sigma}$.

Now let σ'_i be the ground substitution s.t. $x\sigma'_i$ is the normal form of $x\sigma$ wrt. R if x is not lower bounded by T_i and $x\sigma'_i = x\sigma_i$, otherwise. Since $T_i\sigma_i \equiv true$, $T_i\sigma'_i$ must be also $true$. By the restriction imposed on R (i.e. $\rightarrow_R \subseteq \succ$) we also have $D_i\sigma_i \succeq D_i\sigma'_i$ and obviously $R \cup D_i\sigma'_i \models D_i\sigma_i$, which implies $R \cup D_i\sigma'_i \models C\sigma$.

Now, to prove that these $D_i\sigma'_i$ can indeed be used, i.e. they are in $irred_R(S)$, we distinguish two cases on the variables x in D_i: (i) if x is not bounded by T_i then $x\sigma'_i$ is in normal form wrt. R and, (ii) if x is bounded by T_i then $x\sigma'_i = x\sigma_i = y\sigma$ (where y is a variable of C s.t $y \notin stlvars(C, \sigma)$), which, by definition, since $C\sigma$ is irreducible, must be irreducible by R. Then, since $x\sigma'_i$ is irreducible by R for every x in $\mathcal{V}ars(D_i)$, obviously $D_i\sigma'_i$ is variable irreducible by R. □

First of all, note that for the "classical" saturation methods without constraint inheritance, the previous lemma shows that the usual redundancy notions are correct, since the first condition is always true, i.e. a clause C is redundant in S if $S^{\preceq C\sigma} \models C\sigma$ for every ground σ. This means that our framework can deal uniformly with constrained and unconstrained clauses, obtaining exactly the known results for the unconstrained case. In fact, our modified concept of redundant clause (in previous work, $S^{\prec C\sigma} \models C\sigma$ was required) allows to include practical redundancy methods like subsumption without the need of combining the underlying orderings with special subsumption orderings.

In the constraint case, if both conditions of the previous lemma on the variables fail for some constraint T_i of a clause D_i, and we still want to carry out the redundancy proof, then we can always *weaken* T_i to fulfil the conditions: we can eliminate the part of the constraint that establishes the lower bounds in T_i.

EXAMPLE 6.17. *Following the previous example if we want to apply the simplification step then we have to weaken the constraint of* $\rightarrow g(x) \simeq b\,[\![x = a]\!]$, *which in this case means substituting* x *by* a *in the clause. So after this step the set of clauses is:*

$$
\begin{array}{llll}
1. & & \rightarrow a \simeq b & \\
2. & & \rightarrow f(g(x)) \simeq g(x) & \\
3'. & & \rightarrow f(b) \simeq b & \\
4. & g(b) \simeq b & \rightarrow & \\
5'. & & \rightarrow g(a) \simeq b & \\
6. & & \rightarrow f(b) \simeq g(x) & [\![x = a]\!]
\end{array}
$$

Which is not closed under the inference system, since by superposition of 1 *on* $5'$ *we obtain* $\rightarrow g(a) \simeq b$, *which together with clause* 4 *leads to the empty clause.* □

Note that it is sound to do this only if we propagate the information of the removed literals in the equality constraint part and we have started with an initial set of clauses without constraints. On the other hand, weakening constraints may increase the number of inferences. Therefore there is sometimes a trade-off between the possibility of carrying out redundancy proofs and restrictedness of the constraints.

A result similar to the previous one holds for the redundancy criterion for inferences:

LEMMA 6.18. *Let* S *be a set of constrained clauses and let* π *be an inference (by* $\mathcal{I}_{\mathcal{OB}}$ *and* $\mathcal{H}_{\mathcal{OB}}$*) with conclusion* $C\,[\![T]\!]$. *The inference* π *is in* $\mathcal{RI}_{\mathcal{OB}}(S)$ *if for every ground instance* $\pi\sigma$ *with maximal premise* D, *there are ground instances* $D_i\sigma_i$ *of clauses* $D_i\,[\![T_i]\!]$ *in* $S^{\prec D}$ *with* $\{D_1\sigma_1, \ldots, D_m\sigma_m\} \models C\sigma$ *and for all* $i = 1 \ldots m$ *and variable* x *in* D_i:

1 x *is not lower bounded by* T_i, *or else*
2 there is a variable $y \in \mathcal{V}ars(C)$ *s.t.* $x\sigma_i = y\sigma$ *and* $y \notin stlvars(C, \sigma)$.

7. Saturated sets

Due to the restrictive inference rules and the powerful redundancy notions, it is sometimes possible to compute a finite saturated set (not containing the empty clause) for a given theory. This happens when at a certain point of the derivation a set S_n is obtained such that fairness does not require any further inferences to be computed. In this section the interest of such finite $\mathcal{CS}$-saturated sets of clauses is analyzed. In what follows, let S be such a finite $\mathcal{CS}$-saturated set without the empty clause, where $\mathcal{CS}$ is one of the strategies that have been proved complete, like $\mathcal{CS}_{\mathcal{OB}}$.

The first consequence is that, since $I_S \models S$, obtaining such an S proves the consistency of the theory (one can normally only prove inconsistencies). This kind of consistency proving (for full first-order clauses with equality) is closely related to inductive theorem proving. It has many applications but has received very little attention up to now, since most systems normally only prove inconsistencies. For some classes of first-order clauses, like the *monadic class with equality*, saturation can always be made terminating, producing decidability results (Bachmair *et al.*, 1993).

Below we analyze some less immediate applications of saturated sets: for theorem proving in theories expressed by saturated sets of axioms more efficient proof strategies become (refutationally) complete (e.g. the set-of-support strategy, which is incomplete in general for ordered inference systems and also for equality clauses). Saturated sets fulfilling some syntactic properties even provide decision procedures, like rewrite proofs do in the (conditional) equational case.

7.1. Compatibility with the set-of-support strategy

Lemma 7.1. *Let S be a finite $\mathcal{CS}$-saturated set of clauses without the empty clause, for some complete constrained strategy $\mathcal{CS}$.*

Then, for refutation of a set $S \cup S'$, the strategy $\mathcal{CS}$ remains complete if S' is considered as set of support, i.e. no inferences are considered between clauses of S if (i) S has been obtained from a pure set S_0 and S' is pure, or (ii) $S \cup S'$ contains no equality literals and no clauses with equality literals are added in any step of the derivation.

Proof. Consider a fair derivation that starts as the (finite) derivation saturating S of the form $S_0 \cup S', \ldots, S \cup S' = S'_0, S'_1, \ldots$ In this derivation, no inferences between clauses of S have to be computed, since, by correctness of the redundancy criteria, all inferences redundant in S remain redundant in any set containing S.

Furthermore, if S has been obtained from a pure set S_0 and S' is pure, then $S_0 \cup S'$ is pure and as in lemma 3.10 the persistent set of clauses S'_∞ of this new derivation is $\mathcal{CS}$-saturated and S'_∞ is logically equivalent to S'_i for all i. Therefore, if S'_0 is inconsistent the empty clause must belong to some S'_j of the derivation. The same arguments apply if S'_0 contains only atoms and no clauses with equality literals are added in any step of the derivation. □

In the previous proof we have implicitly applied the fact that S is saturated under the semantics in which the satisfiability of the constraints of the clauses of S has been considered wrt. extended signatures (cf. section 2). Otherwise, the following situation could occur:

Example 7.2. *Suppose $\mathcal{F}$ is $\{f, g, a\}$ and S is the set of equations:*

$$\{\rightarrow f(x) \simeq a, \qquad \rightarrow g(a) \simeq a\}$$

under a lexicographic path ordering with the precedence $g \succ a \succ f$.

This set is saturated under the given signature (i.e. it is ground complete): the only possible inference generates $g(f(x)) \simeq a\,[\![a \succ f(x)]\!]$, which is a tautology since $a \succ f(x)$ is unsatisfiable because a is the smallest constant symbol.

But if we want to refute a set $S' = \{g(f(b)) \simeq a \rightarrow\}$ under the new extension of the precedence $g \succ a \succ f \succ b$ no inferences can be computed between S and S' but $S \cup S'$ is inconsistent.

If S is saturated wrt. extended signatures then the clause $g(f(x)) \simeq a\,[\![a \succ f(x)]\!]$ is not redundant, and this incompleteness problem does not appear. □

Of course this is only necessary if in S' appear new function symbols. In fact, if new symbols appear, another assumption is also implicitly made: that the total reduction ordering can be extended to include the new symbols of S'. Actually LPO's are, as far as we know, the only total ordering used in practice for such theorem proving purposes, and these orderings can always be extended by extending the precedence as done in the example above. But in the other cases it is also necessary to obtain a total reduction ordering $\succ_e$ extending $\succ$, which is always possible (Rubio, 1994).

Under the previous suppositions it is indeed possible to deal with Skolem function symbols:

EXAMPLE 7.3. *Suppose we want to prove that $S \models F$, where F is an arbitrary first-order formula and where S is a finite $\mathcal{CS}$-saturated set of clauses without the empty clause as above.*

Let F_1 be the prenex normal form of $\neg F$ and let F_2 be the conjunctive normal form of the Skolemization of F_1. If F_2 (omitting the universal quantifiers) is of the form $C_1 \wedge \ldots \wedge C_n$ then $S \models F$ iff $S \cup \{C_1 [\![true]\!], \ldots, C_n [\![true]\!]\}$ is inconsistent.

The ordering $\succ$ can be extended to a total reduction ordering $\succ_e$ on the new signature, and then any fair $\mathcal{CS}$-derivation starting with $S \cup \{C_1 [\![true]\!], \ldots, C_n [\![true]\!]\}$, where no inferences between clauses in S are considered (i.e. applying the set of support strategy), will generate the empty clause iff $S \models F$. □

7.2. COMBINING SATURATED SETS

Let us now study the case when we combine two finite sets of constrained clauses S_1 and S_2 (built over $\mathcal{F}_1$ and $\mathcal{F}_2$ resp.) $\mathcal{CS}$-saturated wrt. $\succ_1$ and $\succ_2$ respectively. We will suppose that either both S_1 and S_2 contain clauses with only atoms or else they have both been derived from pure sets.

Then a similar result to the one of completeness of the set-of-support-strategy (with a very similar proof) applies: in this case no inferences have to be considered in which all premises are in S_1 or all premises belong to S_2. Therefore, if $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ then $S_1 \cup S_2$ is $\mathcal{CS}$-saturated.

Again here it is needed that S_1 and S_2 are saturated under the semantics in which the satisfiability of the constraints has been considered wrt. extended signatures (cf. section 8.1), or at least wrt. a signature containing $\mathcal{F}_1 \cup \mathcal{F}_2$. Otherwise, situations similar to the example 7.2 can again appear:

EXAMPLE 7.4. *Assume $\mathcal{F}_1 = \{g \succ a \succ f\}$ for the set S_1 and $\mathcal{F}_2 = \{i \succ b \succ h\}$ in S_2:*

$$\begin{array}{llll} S_1: & 1. & \rightarrow f(x) \simeq a & \quad S_2: & 1. & \rightarrow h(x) \simeq b \\ & 2. & \rightarrow g(a) \simeq a & & 2. & \rightarrow i(b) \simeq b \end{array}$$

As in example 7.2, both sets are saturated due to the fact that a and b are the smallest constants in $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively, but in this case, when considering $\mathcal{F}_1 \cup \mathcal{F}_2$ as a new set of function symbols, for any total precedence extending $\succ_{\mathcal{F}_1}$ and $\succ_{\mathcal{F}_2}$ at least one of the sets S_1 and S_2 must become non-saturated. □

This implies that, when working with the union of S_1 and S_2, if we have not considered the semantics of extended signatures then apart from computing inferences between equations form S_1 and S_2 as usual, it becomes also necessary to compute new inferences between equations from the same set.

Furthermore, the same assuption as in the previous section about the existence of a total reduction ordering $\succ_e$ on the new signature $\mathcal{F}_1 \cup \mathcal{F}_2$ extending $\succ_1$ and $\succ_2$ has to be made, which e.g. holds for LPO's.

7.3. Decision procedures applying saturated sets

In some cases, depending on the syntactic properties of the given finite saturated set S, decision procedures for the theory are obtained. This is the case e.g. for $\mathcal{CS}_{\mathcal{OB}}$-saturated sets of equations E, which are convergent for rewriting. Note that in this case saturation behaves like a Knuth-Bendix completion procedure.

Theorem 7.5. *Let E be a consistent $\mathcal{CS}_{\mathcal{OB}}$-saturated set of equations. Then $E \models s \simeq t$ iff $t \rightarrow_E \circ \leftarrow_E s$, for all terms s and t.*

Proof. We prove that every ground term u (possibly with new Skolem constants to which any simplification ordering $\succ$ can be extended) can be rewritten into the minimal (wrt. $\succ$) representative of its E-congruence class. Then the theorem holds since $E \models s \simeq t$ iff the Skolemized versions of s and t have the same minimal representative.

We proceed by induction wrt. $\succ$, i.e. it suffices to prove the reducibility wrt. $\rightarrow_E$ of a non-minimal u.

Let v be such a minimal representative of the congruence class of u. If $u \succ v$, the only inference rules that can be applied in a refutation of $u \simeq v \rightarrow$ are strict superposition left steps on u with some equation $l \simeq r\,[\![T]\!]$ of E. Then the conclusion is a clause C of the form $u[r]_p \simeq v \rightarrow\ [\![T \wedge u \succ v \wedge l \succ r \wedge s|_p = l]\!]$. Such a step must exist, since $E \models u \simeq v$, and a ground instance $C\sigma$ must exist such that the constraint is true, which implies that $l\sigma \succ r\sigma$. But then u is reducible by ordered rewriting, as $u|_p = l\sigma$ for some p and $l\sigma \succ r\sigma$. □

Let us remark that, if we want to use a $\mathcal{CS}_{\mathcal{OB}}$-saturated set of equations for rewriting, since we are not going to superpose on that equations anymore, there is no reason for keeping the equality constraint part in the constraint of the equations (we do not need to block such possitions). We can remove the equality constraints by applying their mgu to the equations.

A similar decision result (by refutation or by conditional rewriting) can be shown to hold for ground queries $S \models s_1 \simeq t_1 \wedge \ldots \wedge s_n \simeq t_n$ if S contains only Horn clauses $\Gamma \rightarrow [\![T]\!]$ (which need not to be used) and Horn clauses $\Gamma \rightarrow s \simeq t\,[\![T]\!]$ where for each solution θ of T, $s\theta$ is strictly maximal in the clause (which ensures termination of the process).

8. Efficient LPO-constraint solving

In our completion procedure the ordering constraints quickly become very restrictive, since in every inference many conditions are added to the constraints inherited from the premises. Therefore we need efficient techniques for proving the unsatisfiability of ordering constraints, since clauses with unsatisfiable constraints can be deleted. The first method for deciding the satisfiability under a given signature of ordering constraints was defined by Comon (1990) for the case in which $\succ$ is interpreted as the lexicographic path ordering (LPO). Here we cannot use its extension to the recursive path ordering with status (RPOS), given by Jouannaud and Okada (1991), since RPO is not total on ground terms. However, since these methods are quite inefficient in practice, we have developed a new faster algorithm for the LPO case that can be used to decide satisfiability under a given signature and under extended signatures, which is, as we will see in the second part of this section, the adequate semantic for satisfiability in complete refutation procedures for ordering constrained clauses. Satisfiability of LPO-constraints for both semantics has been proved NP-complete (Nieuwenhuis, 1993). This method has shown its practical value in the *Saturate* system (Nivela and Nieuwenhuis, 1993).

In what follows we denote variables by $x, y \ldots$ and terms (with variables) by s and t. Every ordering constraint can be expressed by an equivalent disjunction of *solved forms* (Comon, 1990), by keeping it in disjunctive normal form, eliminating negations with $(t \not\succ t') \equiv (t' \succ t \vee t = t')$ and $(t \neq t') \equiv (t' \succ t \vee t \succ t')$, and applying the definition of LPO:

Definition 8.1. *A solved form I is either $\top$, $\perp$ or a formula*

$$x_1 \succ t_1 \wedge \ldots \wedge x_n \succ t_n \quad \wedge \quad t'_1 \succ x'_1 \wedge \ldots \wedge t'_m \succ x'_m \quad \wedge \quad y_1 = s_1 \wedge \ldots \wedge y_r = s_r$$

where t'_j is not a variable, $x_i \notin \mathcal{V}ars(t_i)$, $x'_j \notin \mathcal{V}ars(t'_j)$ and y_k appears only once in I for $i = 1 \ldots n$, $j = 1 \ldots m$ and $k = 1 \ldots r$.

A constraint T is satisfiable if and only if one of its solved forms is satisfiable. The equalities $y_i = s_i$ moreover can be ignored if we are not interested in obtaining an actual solution of the constraint but only in deciding whether it is satisfiable or not, i.e. deleting these equalities does not affect the satisfiability of a solved form.

For such a solved form I, we call the literals $x_i \succ t_i$ *right term* literals, and literals $t'_j \succ x'_j$ (where t'_j is not a variable) *left term* literals. We call $\{t_1, \ldots, t_n\}$ the *set of right terms* of I, and a right term is called *maximal* wrt. $\succ$ in a solved form I if it is maximal wrt. $\succ$ in $\{t_1\sigma, \ldots, t_n\sigma\}$, for some solution σ of I.

From now on, we consider a set of function (and predicate) symbols $\mathcal{F}$ (containing at least one constant) whose elements are totally ordered by $\succ_{\mathcal{F}}$. Let f and 0 be respectively the smallest (wrt. $\succ_{\mathcal{F}}$) non-constant and constant symbols in $\mathcal{F}$ (if there are no such a symbols then the problem is trivial) and suppose moreover that there is no constant c with $c \neq 0$ such that $f \succ_{\mathcal{F}} c$ (note that $\succ_{\mathcal{F}}$ can be arbitrarily chosen). As said, by $\succ^{\mathcal{F}}_{lpo}$ we denote the extension of $\succ_{\mathcal{F}}$ to the lexicographic path ordering on terms. Now we define the *successor* of a ground term t as the smallest ground term greater than t:

Definition 8.2. *Given a ground term t (i.e. $t \in \mathcal{T}(\mathcal{F})$), the successor of t denoted by $succ_{\mathcal{F}}(t)$, is a ground term s.t. there is no ground term t' with $succ_{\mathcal{F}}(t) \succ^{\mathcal{F}}_{lpo} t' \succ^{\mathcal{F}}_{lpo} t$.*

Since f is the smallest non-constant symbol and there is at most one constant symbol smaller than f, it is easy to prove the following lemma:

LEMMA 8.3. *Let t be a term in $\mathcal{T}(\mathcal{F})$. Then $succ_{\mathcal{F}}(t) = f(0,\ldots,0,t)$.*

PROOF. We proceed by induction on the size wrt. $\succ^{\mathcal{F}}_{lpo}$ of t. Suppose that there is a term t' s.t. $f(0,\ldots,0,t) \succ^{\mathcal{F}}_{lpo} t' \succ^{\mathcal{F}}_{lpo} t$. Then $\mathcal{H}ead(t') = f$ (otherwise $t \succ^{\mathcal{F}}_{lpo} t' \succ^{\mathcal{F}}_{lpo} t$ can be derived). Moreover, t' must be $f(0,\ldots,0,t'')$ for some t'' with $f(0,\ldots,0,t'') \succ^{\mathcal{F}}_{lpo} t \succ^{\mathcal{F}}_{lpo} t''$, which is not possible by induction hypothesis. □

EXAMPLE 8.4. *Assume $\mathcal{F} = \{g \succ f \succ 0\}$ where f is a unary function symbol. Intuitively, if the constraint $[\![x \succ y \wedge g(z) \succ x]\!]$ is satisfiable, then the constraint obtained by replacing x by $f(y)$, i.e. $[\![f(y) \succ y \;\wedge\; g(z) \succ f(y)]\!]$ is satisfiable too, since, by the previous lemma, $f(y)$ is the smallest term greater than y. Less clear is the situation when there are also (left term) literals of the form $t[x]_p \succ z$ with $p \neq \lambda$ (e.g. $f(x) \succ z$), since maybe we need a bigger substitution for x.* □

The following lemma states that a solved form (different from $\perp$) with no right term literals is always satisfiable:

LEMMA 8.5. *Let I be a solved form $t'_1 \succ x'_1 \wedge \ldots \wedge t'_m \succ x'_m$, where t'_j is not a variable (i.e. I has no right term literals). Then the substitution α defined by $\alpha(x) = 0 \quad \forall x \in \mathcal{V}ars(I)$ is a solution for I.*

PROOF. Since t'_j is not a variable, we have $t'_j\sigma \succ^{\mathcal{F}}_{lpo} 0$ for all $j : 1 \ldots m$ and substitution σ (s.t. $t'_j\sigma \in \mathcal{T}(\mathcal{F})$), and hence in particular for α. □

In the following we will sometimes use set notation to express the deletion or the addition of literals in a constraint containing only conjuctions.

THEOREM 8.6. *Let I be a solved form $x_1 \succ t_1 \wedge \ldots \wedge x_n \succ t_n \wedge t'_1 \succ x'_1 \wedge \ldots \wedge t'_m \succ x'_m$ with $n > 0$. Then I is satisfiable in $(\mathcal{F}, \succ^{\mathcal{F}}_{lpo})$ iff there exists a maximal wrt. $\succ_{lpo}$ right term t_i in I such that*

$$I' = I \setminus \{t'_j[x_i]_p \succ x'_j \mid p \neq \lambda\} \cup \{t_i \succeq t_j \mid j = 1 \ldots n\} \cup \{x_i = f(0,\ldots,0,t_i)\}$$

is satisfiable. Moreover if I and I' are satisfiable then they have a common solution.

PROOF. For the left to right implication suppose that σ is a solution of I, s.t. $t_i\sigma$ is maximal wrt. $\succ_{lpo}$ in $\{t_1\sigma,\ldots,t_n\sigma\}$, then x_i cannot appear in any t_j, and therefore there is an I' for t_i, where x_i only appears in literals as $x_i \succ t_j$ or $t'_j \succ x_i$ (note that literals $t'_j[x_i]_p \succ x'_j$ with $p \neq \lambda$ have been removed). Then we can build a substitution σ' which is equal to σ for all variables different from x_i and where $x_i\sigma' = f(0,\ldots,0,t_i\sigma)$, such that σ' is a solution for I' (note that, since $succ_{\mathcal{F}}(t_i\sigma) = f(0,\ldots,0,t_i\sigma)$, i.e. $f(0,\ldots,0,t_i\sigma)$ is the smallest term that is bigger than $t_i\sigma$, we have $x_i\sigma \succeq x_i\sigma'$).

For the right to left implication suppose that σ is a solution of I'. Then we build the following substitution σ': for all $y \in \mathcal{V}ars(I')$, $y\sigma' = y\sigma$ if $x_i\sigma \succeq y\sigma$ and $y\sigma' = x_i\sigma$ otherwise. Note that $x_i\sigma' \succeq y\sigma'$ for all $y \in \mathcal{V}ars(I')$.

Now we will prove that σ' is a solution for I and I'. First we prove that σ' satisfies

I'. Since σ is solution of I', we have $x_i\sigma = x_i\sigma' = f(0,\ldots,0,t_i\sigma)$ and $t_i\sigma \succeq t_j\sigma$, which implies $x_i\sigma \succ t_j\sigma$ and hence $t_j\sigma = t_j\sigma'$, for $j = 1\ldots n$. Therefore $t_i\sigma' \succeq t_j\sigma'$, for $j = 1\ldots n$. For literals $t' \succ y$ (where t' is not a variable) we distinguish two cases: if $t'\sigma \succ x_i\sigma$ then $t'\sigma' \succ x_i\sigma'$ (since t' is not a variable and either $t'\sigma = t'\sigma'$ or $x_i\sigma'$ is a proper subterm of $t'\sigma'$) and therefore $t'\sigma' \succ y\sigma'$ (as $x_i\sigma' \succeq y\sigma'$, for all $y \in \mathcal{V}ars(I')$). Otherwise, if $x_i\sigma \succeq t'\sigma$ then $t'\sigma = t'\sigma'$ and, since $t'\sigma \succ y\sigma$, $y\sigma = y\sigma'$, which implies $t'\sigma' \succ y\sigma'$. Finally, literals as $y \succ t_j$ are also satisfied by σ' since, as said, $x_i\sigma \succ t_j\sigma = t_j\sigma'$ (for all $j = 1\ldots n$) and either $y\sigma' = y\sigma$ or $y\sigma' = x_i\sigma$.

Therefore, since $x_i\sigma' \succeq y\sigma'$ for all $y \in \mathcal{V}ars(I')$, we have that σ' is also a solution for I (note that if there are variables in I that do not appear in I', as the constraint is satisfiable without taking them into acount, we can give them any value, for instance 0). □

LEMMA 8.7. *Let I be a solved form $x_1{\succ}t_1 \wedge \ldots \wedge x_n{\succ}t_n \wedge t'_1{\succ}x'_1 \wedge \ldots \wedge t'_m{\succ}x'_m$. Then I is satisfiable in $(\mathcal{F}, \succ^{\mathcal{F}}_{lpo})$ iff $n = 0$ or else there exists a maximal right term in I such that $I[x_i \mapsto f(0,\ldots,0,t_i)]$ is satisfiable.*

PROOF. For the right to left implication if $n = 0$ then by lemma 8.5 the ground substitution α is a solution for I; otherwise I is obviously satisfiable.

For the left to right implication if I is satisfiable and $n > 0$ then by theorem 8.6, $I' = I \setminus \{t'_j[x_i]_p \succ x'_j \mid p \neq \lambda\} \cup \{t_i \succeq t_j \mid j = 1\ldots n\} \cup \{x_i = f(0,\ldots,0,t_i)\}$ is satisfiable and moreover there is a solution σ for both I and I'. But clearly $x_i\sigma = f(0,\ldots,0,t_i)\sigma$ for every solution σ of I', which implies that there is a solution for I fulfilling $x_i = f(0,\ldots,0,t_i)$ and hence $I[x_i \mapsto f(0,\ldots,0,t_i)]$ is satisfiable. □

The previous lemma states the correctness of a constraint solving algorithm. For instance, in practice, to find a solution (or decide the satisfiability) of a given constraint we proceed as follows: for each maximal right term t_i in I we can decide the satisfiability of $I[x_i \leftarrow f(0,\ldots,0,t_i)]$ by computing its set of solved forms and applying the theorem recursively. This process terminates because the number of variables decreases in every step.

EXAMPLE 8.8. *We want to solve $[\![x \succ y \;\wedge\; x \succ z \;\wedge\; z \succ y]\!]$ with $\mathcal{F} = \{f \succ 0\}$ where f is a unary function symbol.*
First we can try replacing x by $f(y)$ obtaining $[\![f(y) \succ z \;\wedge\; z \succ y]\!]$, and then z by $f(y)$ obtaining $[\![f(y) \succ f(y)]\!]$, which has no solution. The only other possibility is to reconsider the first choice. This time we replace x by $f(z)$ getting $[\![f(z) \succ y \;\wedge\; z \succ y]\!]$, and z by $f(y)$ obtaining $[\![f(f(y)) \succ y]\!]$, which has α as a solution, since it has no right terms. □

LEMMA 8.9. *Let I be a solved form $x_1{\succ}t_1 \wedge \ldots \wedge x_n{\succ}t_n \wedge t'_1{\succ}x'_1 \wedge \ldots \wedge t'_m{\succ}x'_m$. Then I is satisfiable in $(\mathcal{F}, \succ^{\mathcal{F}}_{lpo})$ iff $n = 0$ or else there exists a maximal right term t_i in I such that $(I \setminus \{t'_j[x_i]_p \succ x'_j \mid p \neq \lambda\} \cup \{t_i \succeq t_j \mid j = 1\ldots n\})[x_i \mapsto f(0,\ldots,0,t_i)]$ is satisfiable.*

This lemma, which trivialy follows from theorem 8.6, gives us another agorithm for deciding the satisfiability of LPO-constraints, which, in some cases turns out to be more efficient: a part of the constraint is removed, but another part has to be considered in addition wrt. the algorithm corresponding to lemma 8.7.

8.1. Satisfiability of LPO-constraints under extended signatures

The satisfiability of ordering constraints depends on the set of function symbols $\mathcal{F}$ and the precedence $\succ_{\mathcal{F}}$. As seen, it may happen that a set of axioms S over $\mathcal{F}$ that is saturated wrt. $(\mathcal{F}, \succ^{\mathcal{F}})$ becomes non-saturated wrt. an extension of it $(\mathcal{F}', \succ^{\mathcal{F}'})$. Due to this problem in principle we can loose e.g. the completeness of the set of support strategy for the refutation of clauses with new Skolem constants when using a saturated set of clauses, or similarly, when using a saturated set E of equations for proving an equation $s \simeq t$ by rewriting with E when s and t contain (Skolemized) variables (cf. section 7). The following theorem provides a solution for this problem:

Theorem 8.10. *Let $(\mathcal{F}_0, \succ_{lpo}^{\mathcal{F}_0})$ be the extension of $(\mathcal{F}, \succ_{lpo}^{\mathcal{F}})$ s.t. $\mathcal{F}_0$ is $\mathcal{F} \cup \{f, 0\}$ with $f \succ_{\mathcal{F}_0} 0$ and $g \succ_{\mathcal{F}_0} f$ for every symbol g in $\mathcal{F}$.*
Then T is satisfiable in $(\mathcal{F}_0, \succ_{lpo}^{\mathcal{F}_0})$ if, and only if, there exists some extension $(\mathcal{F}', \succ_{lpo}^{\mathcal{F}'})$ of $(\mathcal{F}, \succ_{lpo}^{\mathcal{F}})$ such that T is satisfiable in $(\mathcal{F}', \succ_{lpo}^{\mathcal{F}'})$.

Proof. The left-to-right implication is trivial since $(\mathcal{F}_0, \succ_{lpo}^{\mathcal{F}_0})$ is an extension of $(\mathcal{F}, \succ_{lpo}^{\mathcal{F}}$). The reverse is proved as follows: if T is satisfiable in $(\mathcal{F}', \succ_{lpo}^{\mathcal{F}'})$ then T is satisfiable in $(\mathcal{F}'_0, \succ_{lpo}^{\mathcal{F}'_0})$ (where $\mathcal{F}'_0$ is $\mathcal{F}' \cup \{f, 0\}$ with smallest symbols f and 0 as in $\mathcal{F}_0$), since f and 0 are new symbols. But by applying any of the two algorithms proposed in lemmas 8.7 and 8.9 to check the satisfiability of T in $(\mathcal{F}'_0, \succ_{lpo}^{\mathcal{F}'_0})$ (note that $(\mathcal{F}'_0, \succ^{\mathcal{F}'_0})$ satisfies the required conditions on the precedence for any $(\mathcal{F}', \succ^{\mathcal{F}'})$), we obtain a solution in which only symbols from $\mathcal{F} \cup \{f, 0\}$ appear, which obviously implies that T is also satisfiable in $(\mathcal{F}_0, \succ_{lpo}^{\mathcal{F}_0})$. □

The previous theorem provides the first decision procedure for the satisfiability of constraints under extended signatures. It also states that sets of axioms S over $\mathcal{F}$ that are saturated wrt. $\mathcal{F}_0$ remain saturated under every extension of $\mathcal{F}$. Therefore, solving constraints during the completion process wrt. $\mathcal{F}_0$ is a method to overcome the problems mentioned above.

If we review example 7.2, but now working with $\mathcal{F}_0$ then we can check that now the constrained equation $g(f(x)) \simeq a\, [\![a \succ f(x)]\!]$ is not a tautology, since the constraint has a solution. Therefore the set S is *not* saturated wrt. $\mathcal{F}_0$.

The next result provides another algorithm for solving constraints, but only for deciding satisfiability of ordering constraints under extended signatures. This is done by adapting lemma 8.9 to the case we are solving the constraint wrt. $(\mathcal{F}_0, \succ_{lpo}^{\mathcal{F}_0})$, which by previous theorem assures the satisfiability under any extension of $(\mathcal{F}, \succ_{lpo}^{\mathcal{F}})$:

Corollary 8.11. *Let I be a solved form $x_1 \succ t_1 \wedge \ldots \wedge x_n \succ t_n \wedge t'_1 \succ x'_1 \wedge \ldots \wedge t'_m \succ x'_m$. Then I is satisfiable in $(\mathcal{F}_0, \succ_{lpo}^{\mathcal{F}_0})$ iff $n = 0$ or else there exists a maximal right term t_i in I such that $I \setminus \{t' \succ t \mid x_i \in \mathcal{V}ars(t' \succ t)\} \cup \{t_i \succeq t_j \mid j = 1 \ldots n\}\{t \succeq t_i \mid (t \succ x_i) \in I\}$ is satisfiable.*

9. Conclusions and future work

We have described an abstract framework, *constraint strategies*, which allows to express many strategies for theorem proving with constrained clauses and prove their completeness in a uniform way. Such a framework consists of an inference system, a *constraint inheritance strategy* and *redundancy criteria* for clauses and inferences.

As a particular case we obtain basic strategies (no inferences are needed on subterms introduced by unifiers of previous inferences) by inheriting equality constraints. Another case is the inheritance of *ordering constraints*, which further reduces the search space.

Our completeness proofs are based on Bachmair and Ganzinger's model generation framework with its abstract redundancy notions. The techniques have been proved compatible with such redundancy criteria for clauses and inferences, and sufficient conditions for practical simplifications and deletion methods to be applicable have been given. In particular, our notions coincide with the most powerful usual ones when the clauses applied in the redudancy proofs have no constraints.

Initial sets of axioms that do have non-trivial ordering constraints can also be handled. As said, this allows us to describe and reason about infinite sets of normal formulae: the clause $\rightarrow P(x,y)\,[\![a \succ x]\!]$, under the LPO induced by $a \succ f \succ b$ denotes the infinite set of clauses $\rightarrow P(b,y),\ \rightarrow P(f(b),y),\ \rightarrow P(f(f(b)),y),\ldots$. Indeed this is a simple case of a pure clause, since the constraint does not lower bound any variable. In fact, if the initial clauses do not have any equality literal then our methods remain complete even if they have arbitrary constraints. It would be interesting to study (although this is beyond the scope of this paper) the expressive power of this kind of ordering constrained clauses, both for the case of unrestricted initial constraints (i.e. first order clauses without equality) and for the case of clauses with equality (when only non-lower bounded variables are allowed in constraints).

Another interesting extension, but also out of the scope of this paper, is the applicability of these techniques to deduction modulo equational theories (like associativity and commutativity), since this could importantly reduce the number of conclusions of the inferences. Such strategies also fall into our abstract framework for constraint strategies as shown in (Rubio, 1994).

References

Bachmair, L., Ganzinger, H. (1991). Rewrite-based equational theorem proving with selection and simplification. Technical Report MPI-I-91-208, Max-Planck-Institut für Informatik, Saarbrücken. To appear in Journal of Logic and Computation.

Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W. (1992). Basic paramodulation and superposition. In Deepak Kapur, editor, *11th International Conference on Automated Deduction*, LNAI 607, pages 462–476, Saratoga Springs, New York, USA. Springer-Verlag. To appear in Information and Computation.

Bachmair, L., Ganzinger, H., Waldmann, U. (1993). Superposition with simplification as a decision procedure for the monadic class with equality. In *3rd Kurt Gödel Colloquium: Computational Logic and Proof Theory*. To appear.

Comon, H. (1990). Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411.

Dershowitz, N., Jouannaud, J.P. (1990). Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 6, pages 244–320. Elsevier Science Publishers B.V., Amsterdam, New York, Oxford, Tokyo.

Huet, G. (1972). *Constrained Resolution: A complete method for higher-order logic*. PhD thesis, Case Western Reserve University.

Hullot, J.M. (1980). *Compilation de Formes Canoniques dans les Teories Equationnelles.* PhD thesis, Universite de Paris Sud, France.

Jouannaud, J.P., Okada, M. (1991). Satisfiability of systems of ordinal notations with the subterm property is decidable. In *Automata, Languages and Programming, 18th International Colloquium*, LNCS 510, Madrid, Spain. Springer-Verlag.

Kirchner, C., Kirchner H., Rusinowitch, M. (1990). Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52.

Lynch, C., Snyder, W. (1993). Redundancy criteria for constrained completion. In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications*, LNCS 690, pages 2–16, Montreal, Canada. Springer-Verlag.

Nieuwenhuis, R., Rubio, A. (1992a) Basic superposition is complete. In B. Krieg-Brückner, editor, *European Symposium on Programming*, LNCS 582, pages 371–390, Rennes, France. Springer-Verlag.

Nieuwenhuis, R., Rubio, A. (1992b) Theorem proving with ordering constrained clauses. In Deepak Kapur, editor, *11th CADE*, LNAI 607, pages 477–491, Saratoga Springs, New York.

Nieuwenhuis, R. (1993). Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69.

Nivela, P., Nieuwenhuis, R. (1993). Practical results on the saturation of full first-order clauses: Experiments with the saturate system. (system description). In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications*, LNCS 690, Montreal, Canada. Springer-Verlag.

Peterson, G.E. (1990). Complete sets of reductions with constraints. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction*, LNAI 449, pages 381–395, Kaiserslautern, FRG. Springer-Verlag.

Rubio, A. (1994). Automated deduction with ordering and equality constrained clauses. PhD. Thesis, Technical University of Catalonia, Barcelona, Spain.