On Narrowing, Refutation Proofs and Constraints

Robert Nieuwenhuis*

Technical University of Catalonia
Pau Gargallo 5, 08028 Barcelona, Spain.
E-mail: `roberto@lsi.upc.es`.

Abstract. We develop a proof technique for dealing with narrowing and refutational theorem proving in a uniform way, clarifying the exact relationship between the existing results in both fields and allowing us to obtain several new results. Refinements of narrowing (basic, LSE, etc.) are instances of the technique, but are also defined here for arbitrary (possibly ordering and/or equality constrained or not yet convergent or saturated) Horn clauses, and shown compatible with simplification and other redundancy notions. By narrowing modulo equational theories like AC, compact representations of solutions, expressed by AC-equality constraints, can be obtained. Computing AC-unifiers is only needed at the end if one wants to "uncompress" such a constraint into its (doubly exponentially many) concrete substitutions.

1 Introduction

Answer computation in some logic is at the heart of many applications in computer science, such as (functional) logic programming, automated theorem proving/discovering or deductive data bases. Well-known simple examples of such mechanisms are Prolog's SLD-resolution, where the accumulated unifiers are kept as answers, or E-unification procedures for equational (or more general) theories E in which, given a goal $s = t$, answers σ are computed such that $E \models s\sigma = t\sigma$.

Narrowing was originally devised as an efficient E-unification procedure using a convergent (confluent and terminating) set of rewrite rules R for E ([Fay79, Hul80]). A rewrite rule $l \to r$ can be applied to narrow a term s into $s\sigma[r\sigma]_p$, denoted $s \rightsquigarrow s\sigma[r\sigma]_p$, if σ is the mgu of $s|_p$ and l. Assume we have two new symbols *goal* and *true*, and in R the additional rule $goal(x, x) \to true$. Then (naive unconditional) narrowing is complete in the sense that for each (irreducible wrt. R) solution σ of $s = t$, there is a narrowing derivation $goal(s, t) \rightsquigarrow^* true$

* This work was presented in a preliminary stage at the Unif'94 workshop in june 1994 in Val d'Ajol, France. I wish to thank Albert Rubio and Christopher Lynch for several useful discussions. The author is partially supported by the ESPRIT Basic Research working group 6028, CCL.

in which the variables of the goal $s = t$ are instantiated with some accumulated solution θ more general than σ. To find all such solutions θ, the possible derivations can be explored by building a (possibly infinite, but finitely branching) narrowing *tree*. Many extensions (to e.g. conditional TRS's) and optimizations of narrowing have been proposed during the last years (see e.g. [RKL85, BGM88, Höl89, NRS89, BKW92, BW94]).

In the literature, most completeness proofs of narrowing strategies are based on lifting arguments applied to rewrite proofs ([Hul80]). For example, to prove the completeness of *basic* narrowing (no steps needed on subterms introduced by unifiers of previous steps), one easily shows that for each (irreducible) solution σ every rewrite proof $goal(s\sigma, t\sigma) \rightarrow^* true$ can be *lifted*, i.e. each rewrite step has a corresponding basic narrowing step in a narrowing derivation $goal(s,t) \leadsto^* true$ in which some solution θ more general than σ is computed. For a rewrite strategy with unique rewrite proofs one may obtain a corresponding narrowing strategy which is *optimal* in the sense that each solution will be computed exactly once. This is in essence the idea behind LSE-narrowing ([BKW92, BW94]), in which leftmost-innermost rewrite proofs, with a priority ordering on the rewrite rules, are lifted.

However, the proof technique of lifting rewrite proofs has severe limitations. For more efficient strategies, more general clauses without restrictions on the occurrences of variables, more general simplification and redundancy notions, or with constrained clauses, it quickly becomes extremely complicated or impossible (see [CAR93, Cha94] for the case of constrained equations). Here we follow an alternative approach, based on the well-known fact that in refutation theorem proving, each refutation proof provides *one* answer, like in SLD-resolution. The main difficulty thereby lies in proving that *all* answers can be obtained, even if one considers arbitrary Horn clauses with equality, more restrictive strategies, or constrained clauses. 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, defined by Bachmair and Ganzinger in [BG94]. As clauses may have both equality and non-equality predicates, the technique uniformly covers E-unification-like methods and Prolog-like resolution strategies, and we believe that it can be applied to all known major narrowing strategies parameterized by a total (on ground terms) reduction ordering.

Symbolic constraints à la [KKR90] are very useful in our framework. On one hand, ordering and equality constraints prune the search tree in several ways ([NR92b]). Furthermore, equality constraints have also turned out to be ideally suited for expressing basicness restrictions [NR92a], and more "operational" formalisms like clauses with a set of *marked* positions produced by earlier substitutions can be avoided. Moreover, the concept of equality constraints in which equality is interpreted *modulo* an equational theory like associativity and commutativity (AC), allows us to express sets of solutions in a compact way. In this context, computing AC-unifiers is completely unnecessary during the deduction process. In fact, AC-unification can be seen in this framework as the operation of "uncompressing" a compact set of solutions, expressed by a constraint, into its (doubly exponentially many) concrete substitutions.

This paper is structured as follows. Section 2 gives basic definitions and notation. In section 3 we revisit model generation for constrained clauses in order to more carefully analyse the permissible kind of initial constrained clauses. This is needed in section 4 to prove that a narrowing derivation can be formulated as a refutation proof of such admissible constrained clauses, which leads to our main completeness results. After this, in section 5 we relate some existing results from narrowing and refutational theorem proving; for example, it becomes clear that the "natural" notion (from the refutation point of view) of basic normal narrowing is indeed complete, and that left-to-right narrowing strategies generalize to the *redex orderings* of [BGLS92]. Section 6 is on narrowing modulo equational theories like AC.

2 Clauses with Equality and Constraints

For more details on basic notions see [DJ90] (rewriting), [BG94] (first-order clauses and model generation) and [KKR90, NR92b, NR94] (constraints and constrained deduction).

Here an *equation* is a multiset of terms $\{s, t\}$, which will be written in the form $s \simeq t$. A (first-order) Horn clause C is a pair of (finite) multisets of equations Γ (the *antecedent*) and Δ (the *succedent*), denoted by $\Gamma \rightarrow \Delta$, where Δ consists of exactly one equation (then C is a *program* clause) or Δ is empty (*goal* clause). We sometimes also write C like $\neg e_1 \vee \ldots \vee \neg e_n \vee e'_1 \vee \ldots \vee e'_n$ if Γ is $\{e_1, \ldots, e_n\}$ and Δ is $\{e'_1, \ldots, e'_n\}$.

Ordering and equality constraints are quantifier-free first-order formulae built over the binary predicate symbols $\succ$ and $=$ relating terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, where $=$ is interpreted as equality of ground terms in $\mathcal{T}(\mathcal{F})$ (sometimes modulo an equational theory like AC), and $\succ$ as a (possibly AC-compatible) reduction ordering total on ground terms. A ground substitution σ *satisfies* a constraint T (is a *solution* of T) if $T\sigma$ evaluates to true (denoted $T\sigma \equiv true$) in the given interpretation. We extend $\succ$ to an ordering on ground equations (in fact, to their *occurrences* in clauses) and to clauses by using standard multiset extensions (see [BG94]). We will ambiguously use $\succ$ to denote all these (total) orderings on ground terms, equations and clauses. A pair consisting of a clause C and a constraint T is a *constrained clause*, written $C [\![T]\!]$.

An interpretation I is a congruence on ground terms. The congruence on $\mathcal{T}(\mathcal{F})$ generated by a set of equations (or rewrite rules) E (which is an interpretation) will be denoted by E^*. I satisfies a ground clause $\Gamma \rightarrow \Delta$, denoted $I \models \Gamma \rightarrow \Delta$, if $I \not\supseteq \Gamma$ or else $I \cap \Delta \neq \emptyset$. An interpretation I satisfies (is a model of) a constrained clause $C [\![T]\!]$, denoted $I \models C [\![T]\!]$, if it satisfies every ground instance of $C [\![T]\!]$, i.e. every $C\sigma$ such that σ is ground and $T\sigma$ is true. Therefore, clauses with unsatisfiable constraints are tautologies, and $\Gamma \rightarrow \Delta [\![T]\!]$ is an *empty clause* denoting an inconsistence iff $\Gamma \cup \Delta = \emptyset$ 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 (and is called a *logical consequence* of S), denoted $S \models C$, if $I \models C$ whenever $I \models S$. For dealing with non-equality predicates, atoms A can be expressed by equations $A \simeq \mathit{true}$ (sometimes denoted simply by A), where true is a special symbol (minimal in $\succ$).

3 Model Generation for Narrowing

Bachmair and Ganzinger's model generation method [BG94] defines a way to build from a set S of clauses a (convergent) set of ground rewrite rules R_S. Then it is shown that if S is a *saturated* set of clauses (i.e. closed up to redundant inferences under a given superposition inference system), then either the empty clause is in S or else S has a model, namely R_S^*, the congruence generated by R_S, thus proving the refutational completeness of saturation. In [NR92b] this method is applied to constrained clauses, and in [NR94] to constrained clauses modulo equational theories like AC.

Here we give an adapted version of it for the case of constrained Horn clauses. The essential modification wrt. [NR92b] for our completeness proofs of narrowing is that most notions are parameterized by R, a set of ground rewrite rules (e.g. a clause is redundant if all its *irreducible* (by R) instances follow from R and other smaller irreducible instances). This parameterization by R allows us to more carefully analyse the kind of initial constrained clauses for which the methods are complete. It is known that arbitrary initial constraints cannot be dealt with:

Example 1. Let S be the set of clauses $\{\ a \simeq c,\ P(x)\,[\![x \succ b]\!],\ \neg P(c)\ \}$ and suppose $a \succ b \succ c$. Then S is inconsistent, since we have $P(a)$ as an instance of $P(x)\,[\![x \succ b]\!]$ contradicting $\neg P(c)$ and $a \simeq c$. However, the empty clause is not in S and no superposition inferences exist: the only possible paramodulation inferences are either below the variable position x in $P(x)$ or involve the small side of $a \simeq c$ (applied to $P(c)$) or lead to a tautology (when unifying $P(x)$ and $P(c)$, getting a clause with the unsatisfiable constraint $[\![c \succ b]\!]$; note that this inference can also be seen as a resolution step). $\square$

Definition 1. Let R be a set of ground rewrite rules.

1. A ground substitution σ is *irreducible* wrt. R if $x\sigma$ is irreducible wrt. R, for every variable x in the domain of σ.
2. A clause $C\sigma$ is a *ground instance with* σ of a constrained clause $C\,[\![T]\!]$ if σ is a ground substitution s.t. $T\sigma \equiv \mathit{true}$.
3. Similarly, if π is an inference with premises $C_1\,[\![T_1]\!], \ldots C_n\,[\![T_n]\!]$ and conclusion $C\,[\![T]\!]$ and σ is a ground substitution σ satisfying T, then the inference $\pi\sigma$ with premises $C_1\sigma, \ldots, C_n\sigma$ and conclusion $D\sigma$ is a *ground instance with* σ of π.
4. The set of ground instances of a constrained clause $C\,[\![T]\!]$ with substitutions that are irreducible wrt. R is denoted by $\mathit{Ir}_R(C\,[\![T]\!])$. Similarly, we write $\mathit{Ir}_R(\pi)$ for inferences π and $\mathit{Ir}_R(S)$ for the irreducible instances of clauses of a set S.

5. A set of constrained clauses S is *pure* wrt. R if $Ir_R(S) \cup R \models S$.
6. If C is a ground clause and S is a set of clauses, then we denote by $S^{\prec C}$ (resp. $S^{\preceq C}$) the set of ground instances of clauses in S that are smaller (resp. smaller or equal) wrt. $\succ$ than C.
7. A constrained clause $D\,[\![T]\!]$ is *redundant* wrt. R in a set of constrained clauses S if $Ir_R(S)^{\preceq C} \cup R \models C$ for all C in $Ir_R(D\,[\![T]\!])$.
8. An inference π is *redundant* wrt. R in a set S if $Ir_R(S)^{\prec C} \cup R \models D$ for all inferences in $Ir_R(\pi)$ with maximal premise C and conclusion D.

Pure sets of *initial* clauses (for a certain R) are the ones for which completeness is preserved. Note that the set of the previous example is not pure if $a \to c$ is in R: then the reducible instance $P(a)$ of $P(x)\,[\![x \succ b]\!]$ does not follow from R and the irreducible instances, since $P(c)$ is not an instance of $P(x)\,[\![x \succ b]\!]$.

Now we define the set of rewrite rules R_S for a set S by induction. Each instance C may *generate* a rule depending on the set R_C of rules generated by smaller instances:

Definition 2. Let S be a set of constrained clauses and let C be a ground instance $\Gamma \to s \simeq t$ with σ of a clause in S. Then C *generates* a rule $s \to t$ if:

1. $R_C^* \not\models C$,
2. $s \succ t$ and $s \succ t'$ for all terms t' occurring in Γ and
3. both s and σ are irreducible by R_C.

where R_C and R_S are the sets of rules generated by $S^{\prec C}$ and S respectively.

Definition 3. The inference system $\mathcal{I}$ consists of the following two inference rules for constrained clauses, where OC denotes the ordering constraint stating that the inferences take place with the maximal equations of each premise, and, in the case of superposition, with the maximal terms and where $D|_p$ is not a variable:

superposition:
$$\frac{C \vee s \simeq t\,[\![T]\!] \qquad D\,[\![T']\!]}{C \vee D[t]_p\,[\![T \wedge T' \wedge OC \wedge D|_p = s]\!]}$$

equality resolution:
$$\frac{C \vee s \not\simeq t\,[\![T]\!]}{C\,[\![T \wedge OC \wedge s = t]\!]}$$

Definition 4. A set of constrained clauses S is *saturated* if all inferences of $\mathcal{I}$ with premises in S are redundant wrt. R_S in S.

Lemma 5. *Let S be a saturated set of constrained clauses. Then either the empty clause is in S or else $R_S^* \models Ir_{R_S}(S)$.*

Proof. The proof is rather similar to [NR92b]. Here we only sketch the main idea. Suppose there is a minimal (non-empty) counter example C in $Ir_{R_S}(S)$, such that $R_S^* \not\models C$. If the maximal equation $s \simeq t$ of C is negative and $s \succ t$, then, since $R_S^* \not\models C$ we have $R_S \models s \simeq t$, so s must be reducible by a rule $l\theta \to r\theta$ of the (convergent) system R_S. But then there is an inference by superposition, with the clause $D \vee l \simeq r$ of S that has generated the rule, on the clause of which C is an instance. This inference is redundant since S is saturated. From the definition of redundancy it follows that then there must be a smaller clause C' in $Ir_{R_S}(S)$ such that $R_S^* \not\models C'$, contradicting the minimality of C. Similar arguments apply in the remaining cases: if the maximal equation of C is a negative equation $s \simeq s$ (inference by equality resolution) or a positive equation $s \simeq t$ (superposition).

□

Corollary 6. *Let S be a saturated set of constrained clauses that is pure wrt. R_S. Then either the empty clause is in S or else $R_S^* \models S$, i.e. S is consistent.*

Now, since we have the previous result for saturated sets, our aim will be to compute such sets, by means of *theorem proving derivations*. The following definition of such derivations is still parameterized by R, so from the definition it is not clear how to compute derivations if R is not known, because e.g. the redundancy of clauses depends on R. Later on this becomes clear, and sufficient conditions for redundancy will be given.

Definition 7. Let R be a ground rewrite system.

1. A *theorem proving derivation* wrt. R is a sequence of sets of constrained clauses $S_0, S_1, \ldots$, such that each S_{i+1} is obtained from S_i by adding a logical consequence of S_i or by removing a clause $C [\![T]\!]$ that is redundant wrt. R in $S_i \setminus C [\![T]\!]$.
2. The set S_∞ of *persistent* clauses in $S_0, S_1, \ldots$ is defined as $\cup_j (\cap_{k \geq j} S_k)$.
3. A theorem proving derivation wrt. R is *fair* if every inference of $\mathcal{I}$ with persisting premises is redundant wrt. R in some S_j.

Lemma 8. *Let $S_0, S_1, \ldots$ be a theorem proving derivation wrt. some R s.t. S_0 is pure wrt. R. Then S_∞ is also pure wrt. R.*

Proof. We first prove $R \cup Ir_R(S_\infty) \models Ir_R(S_0)$, by showing something more general, namely $R \cup Ir_R(S_\infty) \models Ir_R(S_k)$ for all k. Let C be the smallest counter example, i.e. C is the minimal clause of all $Ir_R(S_k)$ such that $R \cup Ir_R(S_\infty) \not\models C$.

C is not an instance of any clause in S_∞, so there is some j such that C is an instance of some clause in S_j but not of any clause in S_{j+1}. This means $R \cup Ir_R(S_{j+1})^{\prec C} \models C$, which contradicts the minimality of C.

Now we have proved $R \cup Ir_R(S_\infty) \models Ir_R(S_0)$. By pureness of S_0 (i.e. $R \cup Ir_R(S_0) \models S_0$), this implies $R \cup Ir_R(S_\infty) \models S_0$ which in turn, by soundness of the derivation (only consequences are added) implies $R \cup Ir_R(S_\infty) \models S_\infty$, i.e. S_∞ is pure wrt. R. □

If we instantiate R by the set R_{S_∞} for some derivation $S_0, S_1, \ldots$, then we get the following theorem. Again, note that R_{S_∞} is not known in advance, and that therefore in practice sufficient conditions for pureness and redundancy have to be used:

Theorem 9. *If $S_0, S_1, \ldots$ is a fair theorem proving derivation wrt. R_{S_∞} and S_0 is pure wrt. R_{S_∞}, then (i) S_∞ is saturated, (ii) S_∞ is pure wrt. R_{S_∞}, and (iii) either the empty clause is in S_∞ or else $R^*_{S_\infty} \models S_\infty$, hence S_∞ in consistent.*

Proof. (i) follows from fairness and the fact that inferences that are redundant in some S_j remain redundant in S_∞ (see [NR92b] for details); (ii) is an instance of the previous lemma where R is R_{S_∞} and (iii) follows from (i) and (ii) by Corollary 6. □

Before applying these results to narrowing, let us see how theorem proving derivations can be computed in practice, i.e. how a practical theorem prover can be an instance of the framework. For the notions of redundancy, we have the following sufficient conditions:

Lemma 10. *A constrained clause $C[\![T]\!]$ is redundant (wrt. every R) in a set of constrained clauses S if $un(S)^{\preceq C\sigma} \models C\sigma$ for all its ground instances $C\sigma$, where $un(S)$ denotes the subset of unconstrained clauses of S.*

This lemma is powerful enough to allow simplification by rewriting and subsumption with unconstrained clauses, deletion of tautologies, etc. For instance, a simplification step of a clause C into D by rewriting (demodulation) fits into the framework by adding the new clause D and then removing C, which has become redundant. This allows us in the next section to prove the completeness of narrowing techniques with simplification steps and other redundancy proofs.

In fact, the previous lemma can be sharpened (see [NR92b]), e.g. by replacing $un(S)$ by the subset of S with constraints that do not lower bound any variables. If also this condition fails one can *propagate* information from the constraint to the clause part until it holds (ordering constraints generated by the inferences can be discarded without propagation to the clause part). For redundancy of inferences similar sufficient conditions exist (see [NR92b] for details). When applying these sufficient conditions for redundancy, one can speak about fair derivations in general (wrt. *any* R, so the concrete R can be omitted).

For pureness of initial clauses, obviously all sets S of clauses without constraints are pure for all R: we have $R \cup Ir_R(S) \models S$, and again this can be sharpened by allowing constraints that do not lower bound any variables.

Finally, let us say some words about finite saturated sets. If in a theorem proving derivation $S_0, S_1, \ldots$ a subset S of the initial clauses S_0 is a finite saturated set (all inferences between clauses in S are redundant in S), then these inferences are also redundant in the superset S_0 of S, i.e. no inferences between clauses of S have to be considered in the derivation. This will be applied below for narrowing in the theory defined by S.

4 Narrowing and Constraints

In the following, let S be an arbitrary consistent set of Horn clauses, and let $\Gamma \rightarrow$ be a goal clause. Our aim is to find all *irreducible* solutions σ such that $S \models \Gamma\sigma$. Here "irreducible" refers to the following:

Definition 11. Let S be an arbitrary consistent set of Horn clauses. Then a ground substitution σ is called *irreducible* for S if for each x in the domain of σ, the term $x\sigma$ is minimal wrt. $\succ$ in its congruence class[2] of the unique minimal (*initial*) Herbrand model of S.

Note that this notion of irreducibility coincides with the usual notion in the particular case where S is a convergent conditional TRS, which is irreducibility wrt. conditional rewriting with S. We also have the following not too surprising fact, which implies that for saturated sets S it also coincides with irreducibility wrt. the convergent set of rewrite rules R_S defined as in the previous section:

Lemma 12. *Let S be a pure (wrt. R_S) consistent saturated set of Horn clauses. Then R_S^*, the congruence generated by R_S, is the (unique) minimal model of S.*

Proof. By Corollary 6, we have $R_S^* \models S$, and by construction of R_S we have $S \models R_S$. This implies that R_S^* is indeed the minimal Herbrand model of S. $\square$

For finding all solutions, we will proceed as follows. As in e.g. Prolog, we could start a theorem proving derivation from an initial set containing S plus the goal clause $\Gamma \rightarrow$; then, the accumulated unifiers of each refutation proof (i.e. each proof of the empty clause) provide a solution. However, we cannot do exactly this because we would not be able to obtain all solutions. For example, our redundancy notions would allow to subsume all clauses as soon as the first empty clause appears. Furthermore, we don't want to accumulate unifiers, because sometimes (e.g. in the AC case, see section 6) it is preferable not to compute eagerly all solutions of the unification problems. Also, for technical reasons we prefer to keep the whole set of clauses (including the goal clause) consistent.

Therefore we will add an answer literal to each goal clause $\Gamma \rightarrow$, which becomes $answer(x_1, \ldots, x_n), \Gamma \rightarrow$, where $\{x_1, \ldots, x_n\} = \mathcal{V}ars(\Gamma)$. The new predicate $answer$ does not appear in the program, so $S \cup \{\ answer(x_1, \ldots, x_n), \Gamma \rightarrow \}$ is consistent. Furthermore, this answer literal is not taken into account wrt. the ordering $\succ$ on equations and clauses (for ordering purposes, assume it is the minimal constant $true$). The following definition tells us how answers can be read off from clauses of the form $answer(x_1, \ldots, x_n) \rightarrow [\![T]\!]$ (also see the examples in the next sections):

Definition 13. Let $S_0, S_1, \ldots$ be a theorem proving derivation. A ground answer σ *is computed* in the derivation if there is some clause $answer(x_1, \ldots, x_n) \rightarrow [\![T]\!]$ in some S_i, and σ is a solution of T restricted to the domain $\{x_1, \ldots, x_n\}$.

Regarding the previous definition, note that in fact all solutions of only the accumulated equality constraint part $\mathcal{E}$ of T are correct solutions, but completeness is preserved even if we require σ to be solution of the whole constraint T, including the ordering constraint part.

[2] Remind that we require $\succ$ to be (extendable to) a total ordering on ground terms. This is the case for all general-purpose orderings like LPO, RPO, polynomial orderings, etc.

Theorem 14. *Let $S_0, S_1, \ldots$ be a fair theorem proving derivation wrt. R_{S_∞} s.t. S_0 is a consistent set of Horn clauses that is pure wrt. R_{S_∞}.*

For every goal clause answer$(x_1, \ldots, x_n), \Gamma \rightarrow$ *in S_0, every irreducible for S_0 answer σ such that $S_0 \models \Gamma\sigma$ is computed in the derivation.*

Proof. We derive a contradiction from the existence of a minimal ground (wrt. $\succ$) instance C with an irreducible σ of a clause $answer(x_1, \ldots, x_n), \Gamma \rightarrow [\![T]\!]$ in $\cup_j S_j$ such that $S_0 \models \Gamma\sigma$ and σ is not computed. Note that irreducibility for S_0 of σ is equivalent by Lemma 12 to irreducibility wrt. R_{S_∞}. There are several cases to be considered:

1. If C is not an instance of S_∞, then there is some i such that C is an instance of S_{i-1} but not of S_i, i.e. C is redundant in S_i and therefore $irred_{R_{S_\infty}}(S_i)^{\prec C} \cup R_{S_\infty} \models C$.

This implies that $irred_{R_{S_\infty}}(S_i)^{\prec C} \cup R_{S_\infty} \cup \Gamma\sigma \cup \{answer(x_1, \ldots, x_n)\sigma\}$ is inconsistent, so in particular R^* is not a model of it, if R is the rewrite system $R_{S_\infty} \cup \{answer(x_1, \ldots, x_n)\sigma \rightarrow true\}$. But we know (applying the results of the previous section) that $R^*_{S_\infty} \models irred_{R_{S_\infty}}(S_i)^{\prec C} \cup R_{S_\infty} \cup \Gamma\sigma$, and that R is convergent because σ is irreducible wrt. R_{S_∞}. This means that there must be some goal clause D in $irred_{R_{S_\infty}}(S_i)^{\prec C}$ such that $R^* \not\models D$ (note that adding the rewrite rule $answer(x_1, \ldots, x_n)\sigma \rightarrow true$ can only affect the satisfaction of clauses with answer literals, i.e. goal clauses).

Let D be of the form $answer(x_1, \ldots, x_n)\sigma', \Gamma'\sigma' \rightarrow$, where σ' is irreducible wrt. R_{S_∞}. From $R^* \not\models D$ we have (i) $R \models answer(x_1, \ldots, x_n)\sigma' \simeq true$ and (ii) $R \models \Gamma'\sigma'$. The only rewrite rule of R applicable for (i) is $answer(x_1, \ldots, x_n)\sigma \rightarrow true$, because σ' is irreducible wrt. R_{S_∞}, and therefore $answer(x_1, \ldots, x_n)\sigma$ is $answer(x_1, \ldots, x_n)\sigma'$. We also have $R^*_{S_\infty} \models \Gamma'\sigma'$, because the rule $answer(x_1, \ldots, x_n)\sigma \rightarrow true$ is not applicable in a rewrite proof of $\Gamma'\sigma'$.

So if Γ' is empty, this contradicts the fact that σ is not computed, and if Γ' is not empty, then $C \succ D$ contradicts the minimality of C.

2. If C is an instance of S_∞ then let $s\sigma \simeq t\sigma$ be the maximal equation in $\Gamma\sigma$. Since $S_0 \models \Gamma\sigma$, also $R^*_{S_\infty} \models \Gamma\sigma$ and therefore $R^*_{S_\infty} \models s\sigma \simeq t\sigma$.

1. If $s\sigma \succ t\sigma$, then $s\sigma$ must be reducible by some rule $l\sigma \rightarrow r\sigma$ in the convergent set of rules R_{S_∞}, and $l\sigma \rightarrow r\sigma$ is generated by an instance C' of a clause $\Gamma' \rightarrow l \simeq r[\![T']\!]$ in S_∞ (note that we can use σ for both clauses since they can be supposed not to share any variables). Since σ is irreducible by R_{S_∞}, then there exists an inference by superposition between the clauses of which C and C' are instances, with a conclusion with an instance D of the form $\Gamma'\sigma, answer(x_1, \ldots, x_n)\sigma, \Gamma\sigma[r\sigma] \rightarrow$ such that $R^*_{S_\infty} \models \Gamma'\sigma, \Gamma\sigma[r\sigma]$. By fairness of the derivation, this inference is redundant in some S_i. This leads, as in the previous point 1., to a contradiction (either σ is generated or C is not minimal).
2. Finally, if $s\sigma$ is the same term as $t\sigma$, then there exists an inference by equality resolution that is redundant in some S_i, leading to a contradiction as in the previous case.

□

5 Relationship between Narrowing and Refutation

The previous theorem has several implications. On one hand, as it states the completeness of narrowing techniques for arbitrary sets S of Horn clauses, a particular case of it applies when S is a saturated set. This is the case if S is a convergent set of conditional rewrite rules, but there are also saturated sets that do not fall under any existing notion of convergent TRS. If S is saturated, then no new inferences have to be computed between the clauses of S, and the theorem proving derivation amounts to the construction of a narrowing tree. The notion of fair theorem proving derivation does not impose any concrete strategy wrt. the order in which the narrowing tree is built. Inferences by superposition correspond to narrowing steps with other clauses, and inferences by equality resolution correspond to steps with reflexivity of equality, like in usual narrowing. Completeness is also preserved if in a clause C an arbitrary negative literal is *selected* and no inferences involving other literals of C are computed (see e.g. [BG94]). Simplification steps by rewriting correspond to normalization of the goal. Also other redundancy proofs apply, like constrained and clausal rewriting, case analysis, etc. [NN93].

If S is not a saturated set, then in the theorem proving derivation a saturated set for S is computed, while building at the same time the narrowing tree for the goal. In the particular case of (unconstrained) pure equations, this corresponds to narrowing combined with unfailing Knuth-Bendix completion.

Another implication of the theorem is that constraints can be inherited between clauses, which allows us to cut the search space by exploiting information from the meta-level (the ordering and unifiability restrictions). This information is kept and inherited to restrict future inferences: clauses with unsatisfiable constraints are tautologies, hence redundant. This leads to basicness restrictions (no inferences are needed on subterms introduced by the unifiers of previous inferences) because the accumulated unification problems 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.

In the remaining part of this section, our aim is to have a closer look, from the viewpoint of our proof technique which deals uniformly with narrowing and refutation proofs, to the results that have been independently obtained in both fields and to clarify the relationship between them. Here, because of the lack of space, we cannot analyse all cases, so let us only mention two of them: basic normal strategies and the relationship between redex orderings and left-to-right narrowing.

5.1 Basic Normal Narrowing

Example 2. Let R be the following program (which is in fact a set of two rewrite rules): $\{\ f(a) \simeq a,\ g(x) \simeq x\ \}$, and consider the goal $g(f(y)) \simeq y$. The only irreducible answer is here $\{y \mapsto a\}$.

In *normal narrowing* one originally used to view a rewrite (or normalization) step as a particular case of a narrowing step where the unification can be done by matching, in which case one does not need to consider alternative choices of narrowing steps (i.e. one has don't care non-determinism in normalization steps and don't know non-determinism in other narrowing steps). In our example, $g(f(y)) \simeq y$ normalizes into $f(y) \simeq y$, which narrows into $a \simeq a$ finding the answer $\{y \mapsto a\}$.

If one extends this viewpoint to *basic normal* narrowing, after the first (normalization) narrowing step obtaining $f(y) \simeq y$, the subterm $f(y)$ is blocked by the basicness restriction since it is part of the unifier (formulated with constraints, starting with $answer(y), g(f(y)) \simeq y \rightarrow [\![true]\!]$, we would obtain the clause $answer(y), x \simeq y \rightarrow [\![x = f(y)]\!]$). Then no further inferences can be made and the solution is not found. This illustrates the classical incompleteness result for (naive) basic normal narrowing.

From the viewpoint of refutational theorem proving, this is not surprising: normalization is in fact simplification, that is, a redundancy proof. (If it were an inference, one could not remove the simplified clause, i.e. one could not get don't care non-determinism.) For obtaining completeness of basic normal narrowing in the sense mentioned above, the clause $answer(y), x \simeq y \rightarrow [\![x = f(y)]\!]$, together with $g(x) \simeq x$, should make $answer(y), g(f(y)) \simeq y \rightarrow [\![true]\!]$ redundant. This is not the case: the instance of $answer(y), g(f(y)) \simeq y \rightarrow$ with the irreducible substitution $\{y \mapsto a\}$ does not follow from irreducible instances of $g(x) \simeq x$ and $answer(y), x \simeq y \rightarrow [\![x = f(y)]\!]$ (for which if y is a we get $\{y \mapsto a, x \mapsto f(a)\}$, which is reducible).

On the other hand, a normalization step, i.e. a classical simplification step adding the clause $answer(y), f(y) \simeq y \rightarrow [\![true]\!]$, does indeed make the clause $answer(y), g(f(y)) \simeq y \rightarrow [\![true]\!]$ redundant in the presence of $g(x) \simeq x$, so the natural and complete notion of basic normal narrowing from the refutational point of view is to have basic inferences and redundancy proofs.

5.2 Left-to-right Narrowing vs. Redex Orderings

In [BGLS92] the concept of *redex orderings* is introduced to further restrict the search space in refutation proofs for full first-order clauses with equality without loosing completeness. The idea is that inferences are only needed at positions which are minimal wrt. an arbitrary redex ordering which orders independent positions of the clause. A particular case of redex ordering is the one which orders positions in a leftmost-innermost way.

The completeness proof of this technique is also based on the model generation technique and is compatible with the one we apply here. Only our proof of lemma 5 has to be refined: instead of considering that a term s must be reducible by *some* rule of R_S, we pick the particular rule that reduces s in a minimal position in the redex ordering, which leads to the result that inferences on such minimal positions suffice. This implies that a particular case of our results given here is the completeness of narrowing strategies like left-to-right narrowing which are based on some particular redex ordering. Arbitrary redex orderings correspond to the selection orderings of *basic selection narrowing* [BGM88].

LSE-narrowing [BKW92, BW94] is a refined version of left-to-right narrowing in which in case of several rules reducing at the minimal position of the redex ordering, the minimal rule is chosen wrt. an ordering on the rewrite rules (e.g. by numbering them). This is also straightforwardly covered by our results by again considering in the lemma the rule (this time the one generated by the clause with minimal number) that reduces s in a minimal position in the redex ordering.

6 Narrowing Modulo Equational Theories

In some cases special treatments for some equational subset E of the axioms are preferable, since they generate many slightly different permuted versions of clauses, and for efficiency reasons it is many times better to treat all these clauses together as a single one representing the whole class, by working with special E-matching and E-unification algorithms (see [DJ90] for the basic notions and further references). A special attention has always been devoted to the case where E includes axioms of associativity (A) and commutativity (C), as they occur very frequency in practical specifications, and are well-suited for being built in due to their permutative nature. The traditional way of working modulo AC is with inferences in which one conclusion is added for each σ in $cU_{AC}(s|_p, s')$, a minimal complete set of AC-unifiers of $s|_p$ and s':

$$\frac{s' \simeq t' \qquad s \simeq t}{(s[t']_p \simeq t)\sigma} \quad \forall\, \sigma \in cU_{AC}(s|_p, s')$$

One of the drawbacks of building-in AC in theorem proving or narrowing is the complexity of AC-unification: there may be doubly exponentially many AC-unifiers for two terms (and therefore as many conclusions in an inference) whereas the decision problem of AC-unifiability is “only” NP-complete.

However, now that constraints have become available, these problems can be avoided. In the following inference system, the predicates $=$ and $\succ$ of the constraints are interpreted as AC-equality and an AC-compatible total ordering on terms (e.g. the ordering [RN93]) respectively.

Definition 15. The inference system $\mathcal{I}_{AC}$ consists of the inference rules of $\mathcal{I}$ plus the following two inference rules where f denotes an AC-function symbol, x and x' are new variables, and OC denotes the ordering constraint stating that the inferences take place with the maximal terms of the maximal equations of each premise and where $D|_p$ is headed with f:

AC-superposition:

$$\frac{C \vee s \simeq t\,[\![T]\!] \qquad D\,[\![T']\!]}{C \vee D[f(t,x)]_p\,[\![T \wedge T' \wedge OC \wedge D|_p = f(s,x)]\!]}$$

AC-top-superposition:

$$\frac{C \vee s \simeq t\,[\![T]\!] \qquad D \vee s' \simeq t'\,[\![T']\!]}{C \vee D \vee f(t',x') \simeq f(t,x)\,[\![T \wedge T' \wedge OC \wedge f(s,x) = f(s',x')]\!]}$$

Equality constraints turn out to be extremely useful in this context: instead of AC-unifying the terms, the unification problem is kept in an equality constraint. Apart from the fact that we obtain basic strategies in this way, dealing with a constrained clause $C\,[\![s = t]\!]$ is also much more efficient than having n clauses $C_1, \ldots, C_n$, one for each AC-unifier of s and t, since many inferences are computed at once, and each inference generates *one single conclusion* with an additional equality $s = t$ in its constraint. Such a constrained clause can be proved redundant by means of efficient incomplete methods detecting cases of non-AC-unifiability of its constraint.

In the context of narrowing, AC-equality constraints have the additional advantage that they can be used to express sets of solutions in a compact way.

Example 3. Suppose $a \succ b \succ c \succ d$ and the program R contains one single clause (in fact, a rewrite rule) $f(a,b) \simeq c$ where f is an AC symbol, and we have the goal $f(x,y) \simeq f(c,d)$.

If we do not want to work modulo AC, then we need to do narrowing while running the saturation procedure (like unfailing Knuth-Bendix completion), because no (ground) confluent system exists for $R \cup$ AC. Then, apart from the trivial solutions where one of x and y is c and the other one is d, we will obtain, in a rather inefficient procedure, many other irreducible solutions: $\{x \mapsto f(a,b), y \mapsto d\}$, $\{x \mapsto f(b,d), y \mapsto a\}$, $\{x \mapsto f(d,b), y \mapsto a\}$, $\{x \mapsto f(d,a), y \mapsto b\}$, etc.

If we work modulo AC and with constraints, the set R is a convergent rewrite system, i.e. R is saturated, so no inferences between program clauses are needed, and, starting with $answer(x,y), f(x,y) \simeq f(c,d) \rightarrow [\![true]\!]$, we obtain in one inference by equality resolution the answer clause: $answer(x,y) \rightarrow [\![f(x,y) = f(c,d)]\!]$ which represents the first two solutions. Furthermore, by one single inference of AC-superposition, we obtain the clause $answer(x,y), f(c,z) \simeq f(c,d) \rightarrow [\![f(f(a,b),z) = f(x,y)]\!]$ which in turn by equality resolution produces the answer clause with a (simplified) constraint $answer(x,y) \rightarrow [\![f(f(a,b),d) = f(x,y)]\!]$ representing in a compact way all other irreducible solutions. □

Computing AC-unifiers is needed only at the end if one wants to "uncompress" such a constraint into its (doubly exponentially many) concrete substitutions. In [NR94], by means of the model generation method, we proved the completeness of such a basic strategy for the AC-case combined with ordering constraints (see also [Vig94], and see [Rub94] for a detailed analysis of constrained theorem proving modulo equational theories). Indeed, when working modulo AC and considering the inference system $\mathcal{I}_{AC}$, Theorem 14 holds too.

7 Conclusions and Further Work

We have given a uniform completeness proof covering both refutational theorem proving and narrowing, and have analysed the relationship between some results obtained in both fields. We have proved several new results for answer computation with arbitrary Horn clauses (e.g. no restrictions on occurrences of variables) with constraints and with powerful abstract redundancy notions.

We hope that this work will help to avoid situations in which the same work is done twice (once for narrowing and once for refutational theorem proving), as it has happened with results on basic strategies or redex orderings.

In this paper we have focussed on Horn clauses. However, most techniques of model generation applied here can be extended to work for the non-Horn case. Also a natural semantics for answer computation in such general logic programs with equality exists, namely the *perfect model semantics* given in [BG91], where it is shown that the model generated by R_S for saturated (under an inference system similar to the one used here) sets S coincides with the unique perfect model for S under the given term ordering. It would be interesting to work out practical narrowing techniques which avoid the problem of *floundering* that appears in the theoretical method outlined in [BG91].

References

[BG91] Leo Bachmair and Harald Ganzinger. Perfect model semantics for logic programs with equality. In Koichi Furukawa, editor, *Logic Programming, Proceedings of the Eighth International Conference*, pages 645–659, Paris, France, June 24–28, 1991. The MIT Press.

[BG94] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):1–31, 1994.

[BGLS92] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation and superposition. In Deepak Kapur, editor, *11th International Conference on Automated Deduction*, LNAI 607, pages 462–476, Saratoga Springs, New York, USA, June 15–18, 1992. Springer-Verlag. To appear in Information and Computation.

[BGM88] P. Bosco, E Giovanetti, and C. Moiso. Narrowing vs. sld-resolution. *Theoretical Computer Science*, 2(59):3–23, 1988.

[BKW92] Alexander Bockmayr, Stefan Krischer, and Andreas Werner. An optimal narrowing strategy for general canonical systems. In M. Rusinowitch and J.-L. Rémy, editors, *The Third International Workshop on Conditional Term Rewriting Systems*, LNCS 656, Pont-à-Mousson, France, July 8–10, 1992. Springer-Verlag.

[BW94] Alexander Bockmayr and Andreas Werner. LSE narrowing for decreasing conditional term rewrite systems. In N. Dershowitz, editor, *The fourth International Workshop on Conditional Term Rewriting Systems*, Jerusalem, July 11–15, 1994.

[CAR93] J. Chabin, S. Anantharaman, and P. Réty. E-unification via constrained rewriting. In *7th Workshop on Unification*, Boston, USA, June 13–14, 1993. Boston University.

[Cha94] Jacques Chabin. Unification generale par surreduction ordonnee contrainte et surreduction dirigee. Thèse de Doctorat, Université d'Orleans, France, January 1994.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. 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, 1990.

[Fay79] M. Fay. First-order unification in an equational theory. In *Proceedings of the Fourth Workshop on Automated Deduction*, pages 161–167, Austin, TX, February 1979.

[Höl89] Steffen Hölldobler. *Foundations of equational logic programming*. LNCS 353. Springer-Verlag, 1989.

[Hul80] J. M. Hullot. Canonical forms and unification. In *Proc. 4th International Conference on Automated Deduction*, LNCS 87, Les Arcs, 1980.

[KKR90] Claude Kirchner, Hélène Kirchner, and Michaël Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990.

[NN93] Pilar Nivela and Robert Nieuwenhuis. 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, June 16–18, 1993. Springer-Verlag.

[NR92a] Robert Nieuwenhuis and Albert Rubio. Basic superposition is complete. In B. Krieg-Brückner, editor, *European Symposium on Programming*, LNCS 582, pages 371–390, Rennes, France, February 26–28, 1992. Springer-Verlag.

[NR92b] Robert Nieuwenhuis and Albert Rubio. Theorem proving with ordering constrained clauses. In Deepak Kapur, editor, *11th CADE*, LNAI 607, pages 477–491, Saratoga Springs, New York, 1992. Extended version to appear in Journal of Symbolic Computation.

[NR94] Robert Nieuwenhuis and Albert Rubio. AC-Superposition with constraints: No AC-unifiers needed. In Allan Bundy, editor, *12th International Conference on Automated Deduction*, LNAI, Nancy, France, June 1994. Springer-Verlag.

[NRS89] W. Nutt, P. Réty, and G Smolka. Basic narrowing revisited. *Journal of Symbolic Computation*, 7:295–317, 1989.

[RKL85] P. Rety, C. Kirchner, and P Lescanne. NARROWER: a new algorithm for unification and its application to logic programming. In Jean-Pierre Jouannaud, editor, *Rewriting Techniques and Applications, 1st International Conference*, LNCS 202, Dijon, France, May 20–22, 1985. Springer-Verlag.

[RN93] Albert Rubio and Robert Nieuwenhuis. A precedence-based total AC-compatible ordering. In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications*, LNCS 690, pages 374–388, Montreal, Canada, June 16–18, 1993. Extended version to appear in Theoretical Computer Science.

[Rub94] Albert Rubio. Automated deduction with ordering and equality constrained clauses. PhD. Thesis, Technical University of Catalonia, Barcelona, Spain, 1994.

[Vig94] Laurent Vigneron. Associative Commutative Deduction with constraints. In Allan Bundy, editor, *12th International Conference on Automated Deduction*, LNAI, Nancy, France, June 1994. Springer-Verlag.