Completeness and Redundancy in Constrained Clause Logic

Reinhard Pichler *
Technische Universität Wien

Abstract

In [CZ 91], a resolution-based inference system on c-clauses (i.e. constrained clauses) was introduced, incorporating powerful deletion rules for redundancy elimination. This inference system was extended to resolution refinements in subsequent papers of Caferra et al. (e.g. [CP 95] and [CP 96]). The completeness proofs given for the purely refutational calculi (i.e.: the inference systems without deletion rules) are basically "translations" of the corresponding results from standard clause logic to constraint clause logic.

This work focusses on the deletion rules of the calculi of Caferra et al. and, in particular, on the c-dissubsumption rule, which is considerably more powerful than the usual subsumption concept in standard clause logic. We will show that the "conventional" method for proving the completeness of (standard clause) resolution refinements with subsumption fails when the powerful deletion rules of Caferra et al. are considered. Therefore, in order to prove the completeness of the c-clause calculi, a different strategy is required. To this end we shall use the well-known concept of semantic trees, which can be easily extended from standard clause logic to c-clause logic.

In general, purely non-deterministic application of the inference rules is not sufficient to ensure refutational completeness. It is intuitively clear, that some sort of "fairness" must be required. The completeness proof via semantic trees gives us a hint for defining precisely what it means for a rule application strategy to be "fair".

Finally other methods for proving completeness and defining redundancy criteria are contrasted with completeness via semantic trees and c-dissubsumption. In particular, it is shown that the redundancy criterion of Bachmair/Ganzinger (cf. [BG 94]) is incomparable with c-dissubsumption.

1 Introduction

In [CZ 91], a refutational calculus based on factoring and resolution is carried over in a natural way from standard clause logic to constrained clause logic. This inference system was extended to resolution refinements in subsequent papers of Caferra et al. (e.g. [CP 95] and [CP 96]). The completeness of the resulting calculi is proven in 2 steps: First the *ground completeness* is proven, exploiting the fact that ground resolution on c-clauses coincides with ground resolution on standard clauses. Then the completeness for (general) c-clauses is shown by proving the *lifting property* of the refutational calculus consisting of c-factoring and (possibly refined) c-resolution.

However, one is not contented with the purely refutational calculus. In order to increase efficiency, deletion rules have to be added to the inference system, so as to delete

*Technische Universität Wien, Institut für Computersprachen, Abteilung für Anwendungen der Formalen Logik, Resselgasse 3/1/3, A-1040 Wien, AUSTRIA, reini@logic.at

"redundant" clauses, e.g.: rules for deleting tautologies and subsumed clauses. But this deletion of (c-)clauses may, in principle, destroy the completeness of the original refutational calculus. Hence, the compatibility of the deletion rules incorporated into the inference system with the original refutational rules is not trivial and must be proven separately for every refutational calculus, e.g.: in [Lei 97] the completeness of hyperresolution and of A-ordering resolution, respectively, when combined with tautology deletion and subsumption is proven for standard clauses.

The classical deletion rules in standard clause logic aim at the deletion of the "whole" clause, if it is detected to be redundant. Of course, these concepts of redundancy elimination can be easily translated to c-clause logic. However, the additional expressive power provided by c-clause logic not only allows the deletion of redundant c-clauses as a whole but also the deletion of redundant "parts" of a c-clause by introducing additional constraints. This is what the so-called model construction rules defined by Caferra et al. actually do (cf. [CZ 91]).

Obviously, on the one hand, the additional power of the deletion rules will, in general, make the resulting inference system on c-clauses more efficient (if we take into account only the number of derived clauses and if we do not care about the actual cost for checking the redundancy), since the deletion of only parts of the ground instances of a c-clause may already suffice to prevent the deduction of certain resolvents. On the other hand, the refutational completeness of the resulting inference system is by no means trivial. An example in chapter 4 illustrates that the completeness proof in [Lei 97] for A-ordering resolution together with subsumption is no longer valid if c-dissubsumption is applied. Hence, the original proof idea for the purely refutational calculus (i.e.: first proving the ground completeness and then the lifting property) will no longer suffice and, therefore, a different strategy for the completeness proof is called for. To this end we use the well-known concept of semantic trees, which can be easily extended from standard clause logic to c-clause logic.

In general, purely non-deterministic application of the inference rules is not sufficient to ensure refutational completeness. It is intuitively clear, that some sort of "fairness" must be required. The completeness proof via semantic trees gives us a hint for defining an appropriate rule application strategy, thus making the concept of "fairness" precise. The principal result of this work is summarized in definition 3.3 and theorem 3.4, i.e.: a precise definition of an inference system based on a c-resolution refinement and a sufficient criterion for its refutational completeness. Theorem 3.4 can then be applied to prove the completeness of inference systems based on two common c-resolution refinements, namely semantic clash c-resolution and A-ordering c-resolution (cf. theorem 3.5).

In [FL 96], the concept of H-subsumption was introduced, which is a much stronger redundancy criterion than the familiar subsumption rule. Example 4.3 shows that in general the usual completeness proof for resolution refinements with subsumption no longer works, if H-subsumption is applied. By showing that H-subsumption is a special case of the c-dissubsumption rule of Caferra et al., the completeness of semantic clash resolution or A-ordering resolution in combination with H-subsumption is an immediate consequence of theorem 3.5.

Finally the completeness criterion via semantic trees and the rules for redundancy elimination of Caferra et al. are compared with other approaches in the literature. In particular, it is shown that the redundancy criterion of Bachmair/Ganzinger (cf. [BG 94]) is incomparable with c-dissubsumption, i.e.: On the one hand, there exist clauses redundant in the sense of [BG 94], which cannot be deleted by c-dissubsumption. On the other hand, there are clauses which can be deleted by c-dissubsumption but whose redundancy cannot

be established by the criterion of [BG 94].

This work is structured as follows: In section 2, some principal aspects of the rule system of Caferra et al. and the concept of semantic trees are revised very briefly. In section 3, the completeness of inference sytems based on some c-resolution refinement and the non-refutational rules of Caferra et al. is investigated. In section 4, subsumption on standard clauses is revisited. Finally, in chapter 5, the completeness criterion via semantic trees and the rules for redundancy elimination of Caferra et al. are compared with other approaches in the literature. In chapter 6, the main results of this work are summarized and directions for future research are indicated.

2 Basic Concepts

2.1 A Rule System on Constrained Clauses

Caferra et al. defined a calculus based on unrefined c-resolution for constraint clauses (cf. [CZ 91]) which was then further developed to c-resolution refinements like semantic c-resolution and semantic clash c-resolution (cf. [CP 95] and [CP 96]). Furthermore, ordering refinements were already discussed in the original paper [CZ 91]. Analogously to standard clause logic, many more c-resolution refinements are conceivable. Throughout this paper, we shall therefore use the notation cRes_x to denote an arbitrary c-resolution refinement.

The calculus of Caferra et al. comprises 3 kinds of rules, namely refutational rules, model construction rules and structural rules. The rule definitions of the refutational rules and of those model construction rules which aim at the deletion of parts of c-clauses, play the most important role as far as completeness is concerned. Their definition is recalled below. In contrast to the cDtaut- and cDsub-rule, the other model construction rules (i.e.: the cDfact-, cDres-, GPL-, EGPL-, GMPL-rule) and the structural rules do not delete any ground instances of c-clauses, i.e.: The cDfact-rule and the cDres-rule replace a c-clause by two new c-clauses which contain together exactly the same set of ground instances as the original c-clause. Moreover, the GPL-, EGPL- and GMPL-rule add new c-clauses and the structural rules allow the transformation of a c-clause into an equivalent c-clause. For details on these rules, the original papers have to be referred to (e.g.: [CZ 91], [CP 95], [CP 96]). Moreover, a short overview on some principle aspects of constrained clauses and equational problem can also be found in [Pic 98a]. Note that in contrast to the papers of Caferra et al., we distinguish between the notation "$\equiv$" (to denote syntactical identity of equational problems) and "$\approx$" (for the equivalence of equational problems).

Definition 2.1 (refutational rules and redundancy elimination rules) *Let $H(\mathcal{C})$ be a fixed Herbrand universe. Then the rules cFact, cRes, cDtaut and cDsub are defined as follows:*

1. *(binary) c-factorization: From the c-clause $C = [L(\overline{s}) \vee L(\overline{t}) \vee c : X]$ over the Herbrand universe $H(\mathcal{C})$, the following c-clause may be derived:*

$$\frac{[L(\overline{s}) \vee L(\overline{t}) \vee c : X]}{[L(\overline{s}) \vee c : X \wedge \overline{s} = \overline{t}]} \quad cFact(C)$$

2. *c-resolution: Let $C_1 = [L(\overline{s}) \vee c : X]$ and $C_2 = [L^d(\overline{t}) \vee d : Y]$ be c-clauses over $H(\mathcal{C})$. Then the following c-clause may be derived:*

$$\frac{[L(\overline{s}) \vee c : X] \qquad [L^d(\overline{t}) \vee d : Y]}{[c \vee d : X \wedge Y \wedge \overline{s} = \overline{t}]} \quad cRes(C_1, C_2)$$

3. *c-distautology: Let* $C = [L_{i_1}(\overline{s}_{i_1}) \vee L_{i_2}(\overline{s}_{i_2}) \vee c : X]$ *be a c-clause over* $H(\mathcal{C})$*, where the order of the literals is chosen s.t. the first 2 literal symbols are complementary, i.e.:* $L_{i_1} = L^d_{i_2}$*. By the c-distautology rule, the original c-clause* C *may be replaced by the following c-clause "*$cDtaut(C)$*":*

$$\frac{[L_{i_1}(\overline{s}_{i_1}) \vee L_{i_2}(\overline{s}_{i_2}) \vee c : X]}{[L_{i_1}(\overline{s}_{i_1}) \vee L_{i_2}(\overline{s}_{i_2}) \vee c : X \wedge \overline{s}_{i_1} \neq \overline{s}_{i_2}]} \quad cDtaut(C)$$

4. *c-dissubsumption: Let* $C_1 = [L_1(\overline{s}_1) \vee \ldots \vee L_n(\overline{s}_n) : X]$ *be a c-clause over* $H(\mathcal{C})$ *with a fixed order of the literals and let* $\overline{x}$ *denote the free variables of the constraint* X*. Furthermore let* $C_2 = [M_{i_1}(\overline{t}_{i_1}) \vee \ldots \vee M_{i_n}(\overline{t}_{i_n}) \vee c : Y]$ *be c-clause over* $H(\mathcal{C})$ *with some appropriately chosen order of the literals, s.t. the literal symbols* L_j *and* M_{i_j} *are identical. Then, by the c-dissubsumption rule, the original c-clause* C_2 *may be replaced by the following c-clause "*$cDsub(C_1, C_2)$*":*

$$\frac{[L_1(\overline{s}_1) \vee \ldots \vee L_n(\overline{s}_n) : X] \qquad [M_{i_1}(\overline{t}_{i_1}) \vee \ldots \vee M_{i_n}(\overline{t}_{i_n}) \vee c : Y]}{[M_{i_1}(\overline{t}_{i_1}) \vee \ldots \vee M_{i_n}(\overline{t}_{i_n}) \vee c : Y \wedge Z]} \quad cDsub(C_1, C_2)$$

where $Z \equiv \neg(\exists\, \overline{x})(X \wedge \overline{s}_1 = \overline{t}_{i_1} \wedge \ldots \wedge \overline{s}_n = \overline{t}_{i_n})$.

The 2 redundancy elimination rules defined above have the following meaning: The cDtaut-rule eliminates all tautological ground instances $C\tau$ from the original c-clause C. Likewise, the cDsub-rule allows the deletion of those ground instances $C_2\tau$ from C_2, which are subsumed by the c-clause C_1.

2.2 Semantic Trees in c-Clause Logic

In [KH 69], the concept of *semantic trees* is introduced and several definitions like *failure node*, *inference node* and *closed semantic tree* are given for standard clause logic. Since these definitions can be taken over almost literally from the standard clause case to constraint clause logic, they are not repeated here. Definition 2.2 of completeness via semantic trees is slightly modified w.r.t. the original definition in [KH 69].

Definition 2.2 (completeness via semantic trees) *Let* $cRes_x$ *be a c-resolution refinement. Then* $cRes_x$ *is called* complete via semantic trees, *iff for every unsatisfiable set of c-clauses* $\mathcal{C}$ *the following conditions hold:*

1. *There is a finite semantic tree* T *with some particular property* $\mathcal{P}$*, s.t.* T *is closed for* $\mathcal{C}$*.*

2. *Let* T *be a closed, finite semantic tree* T *with property* $\mathcal{P}$*. Then there is an inference node* N *in* T *and there are c-clauses* $C_1, \ldots, C_k$ *in* $\mathcal{C}$ *which fail at nodes* $N_1, \ldots, N_k$ *immediately below* N *but not at* N *itself, s.t. there is a c-clause* D *which is derived by* $cRes_x$ *from c-factors of* $C_1, \ldots, C_k$ *and which fails at* N*.*

3. *The semantic tree which results from deleting the nodes immediately below* N *from* T *(*N*,* T *according to condition 2 above) can be reduced to a semantic tree* T' *which is closed for* $\mathcal{C} \cup \{D\}$ *and which again has property* $\mathcal{P}$*.*

Although our definition 2.2 differs slightly from the one given in [KH 69], the following theorem on the relationship between refutational completeness and completeness via semantic trees can be proven by exactly the same ideas as theorem 2 in [KH 69].

Theorem 2.3 (refutational completeness) *Let $cRes_x$ be a c-resolution refinement which is complete via semantic trees. Then $cRes_x$ is refutationally complete, i.e.: for every unsatisfiable set $\mathcal{C}$ of c-clauses the empty c-clause $\underline{\square}$ can be derived in the refutational calculus consisting of the cFact-rule and the $cRes_x$-rule (and, possibly, structural rules).*

3 Complete Inference Systems

By theorem 2.3, completeness via semantic trees is a sufficient criterion for the completeness of the purely refutational calculus. In order to investigate the effect of the non-refutational rules of Caferra et al. on a closed semantic tree, we introduce a binary relation "$\Rightarrow$" on c-clauses in a derivation. Roughly speaking, "$C \Rightarrow D$" means that C is either replaced by D or c-dissubsumed by D. In theorem 3.2 we shall show that, whenever a c-clause C, that fails at some node N, is deleted through a non-refutational rule, then there is a c-clause D, s.t. $C \Rightarrow D$ and D fails at N.

Definition 3.1 (relations "$\Rightarrow$" and "$\Rightarrow^*$") *Let $\mathcal{C}_0, \mathcal{C}_1, \ldots$ be the sets of c-clauses in a derivation. We define a binary relation "$\Rightarrow$" on c-clauses as follows: Suppose that for some c-clause C there exists an $i \geq 0$ s.t. $C \in \mathcal{C}_i$ but $C \notin \mathcal{C}_{i+1}$. Then we distinguish 2 cases:*

1. *c-dissubsumption is applied to C: Suppose that there is a c-clause $D \in \mathcal{C}_i$ s.t. the cDsub-rule w.r.t. D is applied to C and $C' = cDsub(D, C)$. Then $C \Rightarrow D$ and $C \Rightarrow C'$.*

2. *any other non-refutational rule is applied to C: If there is a single c-clause $C' \in \mathcal{C}_{i+1}$ s.t. C is replaced by C', then $C \Rightarrow C'$. Likewise, if C is replaced by 2 c-clauses $C_1, C_2 \in \mathcal{C}_{i+1}$, then $C \Rightarrow C_1$ and $C \Rightarrow C_2$.*

By "$\Rightarrow^*$" we denote the reflexive and transitive closure of the relation "$\Rightarrow$".

Theorem 3.2 (effect of non-refutational rules) *Let $\mathcal{C}$ be an unsatisfiable set of c-clauses and T be a finite semantic tree which is closed for $\mathcal{C}$. Let $C \in \mathcal{C}$ be a c-clause which fails at a leaf node N of T. Furthermore suppose that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by applying some model construction rule (i.e.: cDtaut, cDsub, cDfact, cDres) or structural rule to C s.t. $C \notin \mathcal{C}'$. Then there is a c-clause $D \in \mathcal{C}'$, s.t. $C \Rightarrow D$ and D fails at N.*

Proof (sketch): Note that failure of a c-clause C at some node N only depends on the set of ground instances contained in C and not on a specific representation of C. The structural rules leave the set of ground instances of a c-clause unchanged. The model construction rules of c-disfactorization and c-disresolution replace the original c-clause C by 2 new c-clauses C_1 and C_2 s.t. the ground instances of C are partitioned into the 2 sets of ground instances of C_1 and C_2. Therefore, we only consider c-distautolgy and c-dissubsumption here, since these are the only rules which lead to the actual deletion of ground instances. Let $C = [c : X]$ and let $c\sigma$ be an instance of C which fails at N.

1. cDtaut: Suppose that C is replaced by the c-clause $C' = \text{cDtaut}(C) = [c : X \wedge Y]$. But a tautology cannot fail at any node and, hence, the ground instance $c\sigma$ is still contained in C'. Furthermore, by definition 3.1, $C \Rightarrow C'$.

2. cDsub: Suppose that some of the instances of C are deleted by applying the cDsub-rule to C w.r.t. some c-clause $D \in \mathcal{C}$. Then, by definition 3.1, $C \Rightarrow D$ and $C \Rightarrow C'$, where $C' = cDsub(D, C)$. We have to distinguish 2 cases:

   (a) If $c\sigma$ is not among the deleted ground instances, then the resulting c-clause C' still fails at N.

   (b) If $D = [d : Y]$ subsumes $c\sigma$, then there exists a ground instance $d\tau$ of D, s.t. all literals of $d\tau$ are contained in the literals of $c\sigma$. But then $d\tau$ fails at any node at which $c\sigma$ fails. Hence, also D fails at N. $\diamond$

We are now ready to formulate the main result of this work, namely a precise definition of an inference system in constrained clause logic (including, in particular, a concrete definition of the notion of "fairness") and a sufficient condition for its refutational completeness (cf. definition 3.3 and theorem 3.4):

Definition 3.3 (inference system based on a c-resolution refinement) *Let $cRes_x$ be a c-resolution refinement. Then we define the inference system $\mathcal{I}_x$ through the following rule system and rule application strategy.*

- **rule system:**

  1. *refutational rules: cFact, $cRes_x$*
  2. *model construction rules: cDsub, cDtaut, cDfact, cDres*
  3. *structural rules: normalization rules, variable elimination rules*

- **rule application strategy:**
  The rules may be applied non-deterministically. However, the following 3 restrictions must be complied with:

  1. *fairness w.r.t. $cRes_x$: Suppose that the current c-clause set $\mathcal{C}_t$ at some stage in the deduction process contains the c-clauses $C_1, \ldots, C_k$ to which $cRes_x$ can be applied. Then, after a finite number of steps, the $cRes_x$-resolvent D of appropriate c-factors $C'_1, \ldots, C'_k$ must be actually derived. If a c-resolution step is due and some c-clause C_i is no longer contained in the new c-clause set $\mathcal{C}_{t'}$, then all $cRes_x$-resolvents which exist for (c-factors of) $D_1, \ldots, D_k$ have to be derived instead, where $D_i \in \mathcal{C}_{t'}$ and $C_i \Rightarrow^* D_i$.*
  2. *availability of appropriate c-factors: If a resolution step is to be carried out on the c-clauses $C_1, \ldots, C_k$, then appropriate c-factors must be available, i.e.: If L_i is the literal to be resolved upon in c-clause C_i, then all c-factors of C_i which result from unifying L_i with any subset of literals in C_i have to be derived before.*
  3. *normalization of the empty clause: If a c-clause $[\Box, X]$ is derived, s.t. $X \not\approx \bot$, then the normalization rule yielding the empty clause $\underline{\Box}$ has to be applied.*

Remark: The above condition of "fairness w.r.t. cRes_x" of a rule application strategy is a generalization of the usual level saturation strategy, i.e.: A c-clause set $\mathcal{C}_i$ is transformed to the set $\mathcal{C}_{i+1}$ by first deriving all possible (refined) c-resolvents from $\mathcal{C}_i$ and then simplifying the resulting set by finitely many applications of structural and model construction rules.

Theorem 3.4 (completeness of an inference system based on a c-resolution refinement) *Let $cRes_x$ be a c-resolution refinement and $\mathcal{I}_x$ be the inference system based on $cRes_x$ according to definition 3.3. If $cRes_x$ is* complete via semantic trees*, then the inference system $\mathcal{I}_x$ is* refutationally complete*, i.e.: If $\mathcal{C}$ is an unsatisfiable set of c-clauses and the rules of $\mathcal{I}_x$ are applied to $\mathcal{C}$ according to the rule application strategy of $\mathcal{I}_x$, then eventually the empty c-clause $\underline{\square}$ is derived.*

Proof (sketch): Let $\mathcal{C}$ be an unsatisfiable set of c-clauses. By assumption, the c-resolution refinement cRes_x is complete via semantic trees. Hence, by condition 1 of definition 2.2, there is a finite semantic tree T with some particular property $\mathcal{P}$, s.t. T is closed for $\mathcal{C}$. We prove the refutational completeness of $\mathcal{I}_x$ by induction on the number of nodes n of T:

If T has 1 node (i.e.: T consists of the root node only), then some c-clause in $\mathcal{C}$ must fail at the root of T. But only a c-clause of the form $[\square, X] \in \mathcal{C}$ with $X \not\approx \bot$ can fail at the root node. By condition 3 of the rule application strategy, we have to apply the normalization rule, thus deriving the empty c-clause $\underline{\square}$.

If T has $n > 1$ nodes then by condition 2 of definition 2.2, there is an inference node N in T and there are c-clauses $C_1, \ldots, C_k$ in $\mathcal{C}$ which fail immediately below N s.t. there is a c-clause D derivable by cRes_x from c-factors of $C_1, \ldots, C_k$, s.t. D fails at N. According to condition 1 of the rule application strategy, this resolution step eventually has to be carried out. Now suppose that some c-clause C_i is deleted prior to this resolution step. Then theorem 3.2 together with a simple induction argument guarantees the existence of a c-clause D_i in the current c-clause set, s.t. $C_i \Rightarrow^* D_i$ and either D_i fails at N or D_i can take the place of C_i in the resolution step at the inference node N. In the latter case, the rule application strategy requires that this resolution step be actually carried out. But this leads to a closed semantic tree with less than n nodes, to which the induction hypothesis can be applied. ◇

We now put theorem 3.4 to work by applying it to inference systems based on two common c-resolution refinements, namely semantic clash resolution and A-ordering resolution. For a precise definition of these resolution refinements on c-clauses, cf. [CP 96] and [NR 95], respectively.

Theorem 3.5 (examples of complete inference systems) *Let H be a Herbrand universe. Furthermore let $\mathcal{M}$ be an interpretation over H and let $<_A$ be an A-ordering over H. Then the inference systems $\mathcal{I}_{\mathcal{M}}$ and $\mathcal{I}_{<_A}$ based on the c-resolution refinements $cRes_{\mathcal{M}}$ and $cRes_{<_A}$, respectively, are refutationally complete.*

Proof (sketch): By theorem 3.4 it suffices to prove that $cRes_{\mathcal{M}}$ and $cRes_{<_A}$ are complete via semantic trees. But the completeness proof for semantic clash resolution and A-ordering resolution on standard clauses given in [KH 69] can be easily extended to c-clauses. ◇

4 Subsumption on Standard Clauses

The c-dissubsumption rule defined for c-clauses in definition 2.1 restricts the admissible set of ground instances of a c-clause by strengthening the constraints. To this aim, usually, new disequations are added. In some cases this introduction of disequations can be simulated in standard clause logic by replacing the original clause through an appropriate instance

thereof. However, in general, this kind of simulation is not possible. Example 4.1 shows both cases.

Example 4.1 *Let $\mathcal{C} = \{C_1, C_2, C_3, C_4, C_5\}$ be a clause set over the Herbrand universe $H(\Sigma)$ with $\Sigma = \{a^0, f^1, P^2, Q^2\}$, and let the C_i's be defined as follows:*

$$\begin{array}{lll} C_1 = P(x,y) \vee Q(x,y) & C_3 = P(x,x) & C_5 = \neg P(a, f(a)) \\ C_2 = P(x,a) & C_4 = \neg Q(a, f(a)) & \end{array}$$

The clause C_1 corresponds to the c-clause $[P(x,y) \vee Q(x,y) : \top]$ and C_2 to $[P(x,a) : \top]$. Application of the cDsub-rule yields the c-clause $cDsub(C_2, C_1) = [P(x,y) \vee Q(x,y) : y \neq a] = [P(x,y) \vee Q(x,y) : (\exists z) y = f(z)]$, which corresponds to the standard clause $P(x, f(z)) \vee Q(x, f(z))$.

On the other hand, c-dissubsumption w.r.t. $[P(x,x) : \top]$ allows us to restrict the original c-clause $[P(x,y) \vee Q(x,y) : \top]$ to $[P(x,y) \vee Q(x,y) : x \neq y]$. But neither may the original clause $P(x,y) \vee Q(x,y)$ be deleted completely (without destroying the completeness) nor can the resulting c-clause be represented by a finite set of standard clauses.

Hence, the c-dissubsumption rule given in definition 2.1 is naturally more powerful than an analogous subsumption rule defined on standard clauses. But even if the application of c-dissubsumption is restricted to cases where it leads to the actual deletion of the subsumed clause, the resulting rule is still more powerful than the usual subsumption rule in standard clause logic. The subsumption concept obtained from restricting c-dissubsumption to actual clause deletion in standard clause logic is the so-called *H-subsumption* (a term coined in [FL 96] in order to emphasize, that this subsumption concept is parameterized by a specific Herbrand universe H), i.e.: Let $\mathcal{C}$ be a set of clauses over the Herbrand universe H and D be a clause over H. Then H-subsumption is defined as follows: $\mathcal{C} \leq_{ss}^{H} D$ $\Leftrightarrow$ "$\mathcal{C}$ H-subsumes D" $\Leftrightarrow$ there exists a finite subset $\mathcal{C}'$ of $\mathcal{C}$ s.t. for all ground substitutions τ based on H, there is a clause $C \in \mathcal{C}'$ s.t. $C \leq_{ss} D\tau$.

Analogously to the combination of resolution refinements with the usual subsumption rule, resolution refinements can be combined with H-subsumption. However, the usual proof found in the literature for the completeness of A-ordering resolution and semantic clash resolution together with ordinary subsumption cannot be simply extended to H-subsumption. This is due to the fact that the following lemma, which plays a crucial role in the usual completeness proof, no longer works, if H-subsumption is used.

Lemma 4.2 *Let Res_x denote the set of clauses derivable either by A-ordering resolution or by semantic clash resolution. Furthermore let $\mathcal{C}' \leq_{ss} \mathcal{C}$ and $D \in Res_x(\mathcal{C})$. Then $\mathcal{C}' \cup Res_x(\mathcal{C}') \leq_{ss} D$, i.e.: either $\mathcal{C}' \leq_{ss} D$ or there is an R_x-resolvent $D' \in Res_x(\mathcal{C}')$ s.t. $D' \leq_{ss} D$.*

Proof: cf. [Lei 97], theorem 4.2.1 and lemma 4.2.3

The following counter-example shows, that the above lemma does not hold for H-subsumption, i.e.: Even though $\mathcal{C}' \leq_{ss}^{H} \mathcal{C}$ and $D \in Res_x(\mathcal{C})$, then D is not necessarily H-subsumed by $\mathcal{C}' \cup Res_x(\mathcal{C}')$.

Example 4.3 *Let $\mathcal{C}' = \{C_1, C_2, C_3, C_4, C_5\}$ and $\mathcal{C} = \{C_6, C_7\}$ be clause sets over the Herbrand universe $H(\Sigma)$ with $\Sigma = \{a^0, f^1, P^1, Q^1, R^1\}$, and let the C_i's be defined as follows:*

$$\begin{array}{lll} C_1 = P(f(x)) & C_4 = R(a) & C_6 = P(x) \vee Q(x) \\ C_2 = Q(a) & C_5 = R(f^2(x)) & C_7 = \neg Q(x) \vee R(y) \\ C_3 = \neg Q(x) \vee R(f(a)) & & \end{array}$$

Let $<_d$ be the well-known A-ordering defined on the term depth and the maximal depth of variable occurrences (cf. [Lei 97], example 3.3.1) and let the interpretation $\mathcal{M}$ be given through the atom representation $\mathcal{M} := \{P(x), Q(x), R(f(a))\}$.

Then $C_8 = P(x) \vee R(y)$ is an $R_{<_d}$-resolvent and an $R_{\mathcal{M}}$-resolvent of C_6 and C_7, but C_8 is neither H-subsumed by $\mathcal{C}' \cup Res_{<_d}(\mathcal{C}')$ nor by $\mathcal{C}' \cup Res_{\mathcal{M}}(\mathcal{C}')$. Note that this is due to the fact that the (unrefined) resolvent $R(f(a)) \in Res(C_2, C_3)$ is neither an $R_{<_d}$-resolvent nor an $R_{\mathcal{M}}$-resolvent.

Nevertheless, with theorem 3.5 at our disposal, the completeness of A-ordering resolution and semantic clash resolution with H-subsumption is a simple corollary.

5 Ordering-based Redundancy Criteria

Bachmair and Ganzinger introduced a general framework for proving the completeness of an inference system (cf. [BG 94]). Within this framework, they also provide an abstract redundancy criterion, which allows the deletion of clauses while preserving the refutational completeness of the calculus. Their ideas were further developed by Nieuwenhuis and Rubio (cf. [NR 95]). In particular, they modified the redundancy criterion to allow for slightly more powerful deletion rules and to provide a general pattern for completeness proofs.

Both the criteria of Bachmair/Ganzinger and of Nieuwenhuis/Rubio aim at the deletion of the whole clause, rather than just the redundant parts. Hence, as example 4.1 in chapter 4 already illustrated, the power of deletion rules based on these criteria naturally cannot compare with the power of the deletion rules given in definition 2.1.

However, the question arises as to whether the deletion rules from section 2.1 are still more powerful, if we restrict their application to those cases where they lead to the actual deletion of the whole clause. In particular, the cDsub-rule then collapses to the H-subsumption discussed in chapter 4. We shall show in this chapter, that the redundancy criterion from [BG 94] is incomparable with H-subsumption. A comparison of the redundancy criterion from [NR 95] with H-subsumption can be found in [Pic 98b].

In [BG 94], the following concrete redundancy criterion (based on an abstract criterion not discussed here) is defined: A ground clause C is redundant w.r.t. a clause set $\mathcal{C}$, if there exist ground instances $C_1, \ldots, C_n$ of clauses in $\mathcal{C}$ s.t.: $C_1, \ldots, C_n \models C$ and $C_i < C$ for all i. A non-ground clause C is redundant w.r.t. a clause set $\mathcal{C}$, if all its ground instances are redundant.

In order to deal with subsumption, ground instances of clauses are considered as pairs consisting of the original clause and the instantiating substitution. An ordering on clauses covering subsumption can then be defined as follows:

$$(C, \sigma) < (D, \tau) :\Leftrightarrow \begin{array}{ll} 1. & C\sigma < D\tau \text{ or} \\ 2. & C\sigma = D\tau \text{ and } C \text{ properly subsumes } D. \end{array}$$

The following two propositions show, that the redundancy criterion of Bachmair/Ganzinger is incomparable with H-subsumption:

Proposition 5.1 (H-subsumed clause) *There is a clause set $\mathcal{C}$ and a clause $C \in \mathcal{C}$, s.t. C may be deleted by H-subsumption but C is not redundant by the criterion of Bachmair/Ganzinger.*

Proof (sketch): Consider the clause set $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ over the Herbrand universe $H(\Sigma)$ with $\Sigma = \{a^0, f^1, P^2\}$, where the C_i's are defined as follows:

$$C_1 = P(x, f(y)) \quad C_3 = P(a, f(x))$$
$$C_2 = P(f(x), y) \quad C_4 = P(f(x), a)$$

Then both relations $\{C_1, C_4\} \leq_{ss}^{H} C_2$ and $\{C_2, C_3\} \leq_{ss}^{H} C_1$ hold.

Note that C_1 and C_2 have the common ground instance $P(f(a), f(a))$, i.e.: $C_1\sigma_1 = C_2\sigma_2$, where $\sigma_1 = \{x \leftarrow f(a), y \leftarrow a\}$ and $\sigma_2 = \{x \leftarrow a, y \leftarrow f(a)\}$. By the transitivity and irreflexivity of $<$, it is impossible that both inequalities $(C_1, \sigma_1) < (C_2, \sigma_2)$ and $(C_2, \sigma_2) < (C_1, \sigma_1)$ are true. Hence, either C_1 or C_2 is not redundant in the sense of [BG 94], although this clause may be deleted by H-subsumption. ◇

Proposition 5.2 (redundant clause) *There is a clause set $\mathcal{C}$ and a clause $C \in \mathcal{C}$, s.t. C is redundant by the criterion of Bachmair/Ganzinger but C may not be deleted by H-subsumption.*

Proof (sketch): Consider the clause set $\mathcal{C} = \{C_1, C_2, C_3\}$ over the Herbrand universe $H(\Sigma)$ with $\Sigma = \{a^0, f^1, P^1, Q^1, R^1, S^1\}$, where the C_i's are defined as follows:

$$C_1 = P(a) \vee Q(f(a)) \qquad C_3 = Q(f(a)) \vee R(f(a)) \vee S(f(a))$$
$$C_2 = \neg P(a) \vee R(f(a))$$

Then C_3 is clearly implied by C_1 and C_2, since $Q(f(a)) \vee R(f(a)) \models C_3$ and $Q(f(a)) \vee R(f(a))$ is an unrefined resolvent of C_1 and C_2. Hence, C_3 is redundant by the criterion of Bachmair/Ganzinger. On the other hand, C_3 is not H-subsumed by $\{C_1, C_2\}$. ◇

6 Concluding Remarks and Future Work

First, the completeness of the calculus of Caferra et al. (including the redundancy elimination rules of c-dissubsumption and c-distautology) has been proven. This completeness result has then been carried over to H-subsumption in standard clause logic. The comparison with other redundancy concepts has basically shown that H-subsumption is a very strong redundancy criterion. In particular, the example from proposition 5.2 of a clause that is not H-subsumed but redundant in the sense of Bachmair/Ganzinger and Nieuwenhuis/Rubio should not be overestimated: Actually, the "concrete" redundancy criterion of clause implication given in [BG 94] and [NR 95] is not really concrete, since it is not decidable on the non-ground level. Hence, even in cases where the criteria of [BG 94] and [NR 95] are stronger than H-subsumption, the latter criterion still seems to be about the strongest genuine concretisation of the abstract redundancy criteria that we can possibly expect. Future work should, therefore, concentrate on a thorough complexity analysis and on the design of an efficient algorithm for H-subsumption (and/or c-dissubsumption) rather than on the search for stronger concrete redundancy criteria.

The completeness proof in chapter 3 was only carried out for c-clauses without equality. Likewise, the completeness of H-subsumption was only concluded for standard clauses without equality. In [HR 91], the semantic tree method is extended to transfinite semantic trees so as to cover clauses with equality. Furthermore, some new definitions and a new proof strategy (namely, transfinite induction) are introduced to prove the completeness of several calculi using paramodulation and some refinements thereof via transfinite semantic trees. For details, [HR 91] has to be referred to. However, analogously to theorem 3.2,

it can be shown that the "maximum consistent tree" does not grow when one of the non-refutational rules from Caferra et al. is applied. But then an extension of theorem 3.4 to an analogous completeness result for an inference system based on a c-resolution refinement and a c-paramodulation refinement should not be too difficult.

From a theoretical point of view, this extension to clauses with equality is necessary to arrive at a final judgement in the comparison between the power of H-subsumption and the redundancy concepts of [BG 94] and [NR 95]. A more practical motivation for this kind of extension is the search for an appropriate definition of a "fair" rule application strategy also in case of clauses with equality. In chapter 3, a natural definition of fairness of a rule application strategy came as a by-product of the completeness proof. Note that the concept of "fair" theorem proving derivations from [BG 94] and [NR 95] is rather abstract. In particular, the set $\mathcal{C}_\infty$ of persisting clauses is not available during the deduction process itself. Again, our definition of fairness from definition 3.3 can be seen as some kind of concretisation of the corresponding concepts from [BG 94] and [NR 95].

References

[BCP 94] Ch. Bourely, R.Caferra, N.Peltier: A Method for Building Models automatically. Experiments with an Extension of Otter. Proceedings of CADE-12, LNAI 814, pp. 72-86 Springer (1994).

[BG 94] L.Bachmair, H.Ganzinger: Rewrite-based Equational Theorem Proving with Selection and Simplification, Journal of Logic and Computation, Vol 4 No 3, pp. 217-247 (1994).

[CL 89] H. Comon, P. Lescanne: Equational Problems and Disunification, Journal of Symbolic Computation, Vol 7, pp. 371-425 (1989).

[CP 95] R.Caferra, N.Peltier: Extending semantic Resolution via automated Model Building: applications, Proceedings of IJCAI'95, Morgan Kaufmann (1995)

[CP 96] R.Caferra, N.Peltier: Decision Procedures using Model Building Techniques, Proceedings of CSL'95, LNCS 1092, pp.130-144, Springer (1996).

[CZ 91] R.Caferra, N.Zabel: Extending Resolution for Model Construction, Proceedings of Logics in AI - JELIA '90, LNAI 478, pp. 153-169, Springer (1991).

[FL 96] C.Fermüller, A.Leitsch: Hyperresolution and Automated Model Building, Journal of Logic and Computation, Vol 6 No 2, pp.173-230 (1996).

[HR 91] J.Hsiang, M. Rusinowitch: Proving Refutational Completeness of Theorem-Proving Strategies: The Transfinite Semantic Tree Method, Journal of the ACM, Vol 38 No 3, pp. 559 - 587 (1991).

[KH 69] R.Kowalski, P.J.Hayes: Semantic Trees in Automated Theorem Proving, Machine Intelligence 4, pp. 87-101, Edinburgh University Press (1969).

[Lei 97] A.Leitsch: The Resolution Calculus, Texts in Theoretical Computer Science, Springer (1997).

[NR 95] R.Nieuwenhuis, A.Rubio: Theorem Proving with ordering and equality constrained clauses, Journal of Symbolic Computation, Vol 11, pp. 1-32 (1995).

[Pic 98a] R.Pichler: Extending Decidable Clause Classes via Constraints, in Proceedings of FTP'98 (International Workshop, First-Order Theorem Proving), Vienna (1998).

[Pic 98b] R.Pichler: Completeness and Redundancy in Constrained Clause Logic (full version), technical report TR-CS-RP-98-3 of the Technical University of Vienna, available as `ftp://ftp.logic.at/pub/reini/redu.ps` (1998).