The Equivalence of the Subsumption Theorem and the Refutation-completeness for Unconstrained Resolution

Shan-Hwei Nienhuys-Cheng cheng@cs.few.eur.nl
Ronald de Wolf bidewolf@cs.few.eur.nl
Department of Computer Science, H4-19
Erasmus University of Rotterdam
P.O. Box 1738, 3000 DR Rotterdam, the Netherlands

Subfield: Automated Reasoning, (Inductive) Logic Programming.

Abstract

The subsumption theorem is an important theorem concerning resolution. Essentially, it says that a set of clauses Σ logically implies a clause C, iff C is a tautology, or a clause D which subsumes C can be derived from Σ with resolution. It was originally proved in 1967 by Lee in [Lee67]. In Inductive Logic Programming, interest in this theorem is increasing since its independent rediscovery by Bain and Muggleton [BM92]. It provides a quite natural "bridge" between subsumption and logical implication. Unfortunately, a correct formulation and proof of the subsumption theorem are not available. It is not clear which forms of resolution are allowed. In fact, at least one of the current forms of this theorem is false. This causes a lot of confusion.

In this paper, we give a careful proof of the subsumption theorem for unconstrained resolution, and show that the well-known refutation-completeness of resolution is an immediate consequence of this theorem. On the other hand, we also show here that the subsumption theorem can be proved starting from the refutation-completeness. This establishes that these two results have equal strength.

Furthermore, we show that the subsumption theorem does not hold when only input resolution is used, not even in case Σ contains only one clause. Since [Mug92, Ide93a] assume the contrary, some results (for instance results on *nth roots* and *nth powers*) in these articles should perhaps be reconsidered.

1 Introduction

Inductive Logic Programming (ILP) investigates methods to learn theories from examples, within the framework of first-order logic. In ILP, the proof-method

that is most often used is resolution. A very important theorem concerning resolution is the *Subsumption Theorem*, which essentially states the following. Let Σ be a set of clauses and C a clause. Then $\Sigma \models C$, iff C is a tautology or there exists a clause D which subsumes C and which can be derived from Σ by resolution.

This theorem was first stated and proved by Lee in 1967 in [Lee67], his PhD-thesis. However, we have not been able to find a copy of his thesis. So it is unclear what precisely Lee stated, and how he proved his result. Surprisingly, nowhere in the standard literature concerning resolution (not even in Lee's own book [CL73]) the theorem is mentioned.[1]

One thing is clear, though: the subsumption theorem states that logical implication between clauses can be divided in two separate steps—a derivation by resolution, and then a subsumption. Hence the theorem provides a natural "bridge" between logical implication and subsumption. Subsumption is very popular in Inductive Logic Programming, since it is decidable and machine-implementable. However, subsumption is not "enough": if D subsumes C then $D \models C$, but not always the other way around. So it is desirable to make the step from subsumption to implication, and the subsumption theorem provides an excellent tool for those who want to make this step. It is used for instance in [Mug92, Ide93a][2] for inverse resolution. In [LN94b], the theorem is used to extend the result of [LN94a] that there does not exist an ideal refinement operator[3] in the set of clauses ordered by subsumption, to the result that there is no ideal refinement operator in the ordering induced by logical implication. In [NLT93], the theorem is related to several generality orderings.

The subsumption theorem is more natural than the better-known *refutation-completeness* of resolution, which states that an unsatisfiable set of clauses has a refutation (a derivation by resolution of the empty clause $\square$). For example, if one wants to prove $\Sigma \models C$ using the refutation-completeness, one must first normalize the set $\Sigma \cup \{\neg C\}$ to a set of clauses. Usually $\Sigma \cup \{\neg C\}$ is not a set of clauses, since negating the (universally quantified) clause C yields a formula which involves existential quantifiers. So if we want to prove $\Sigma \models C$ by refuting $\Sigma \cup \{\neg C\}$, we must first apply Skolemization to C. Deriving from Σ a clause D which subsumes C, is a much more "direct" way of proving $\Sigma \models C$.

Hence we—and perhaps many others—feel that the subsumption theorem deserves at least as much attention as the refutation-completeness. We can in fact prove that the latter is a direct consequence of the former, as given in Section 3. It is surprising that the subsumption theorem was so little known. Only after Bain and Muggleton rediscovered the theorem in [BM92], people have started paying attention to it.[4]

[1] Recently we received a copy of [Kow70] and a reference to [SCL69] from Stephen Muggleton. [Kow70] gives a proof of the subsumption theorem for unconstrained resolution, using semantic trees, which is very different from the proof we give here. [SCL69] proves a version of the subsumption theorem for semantic resolution.

[2] [Ide93a] is a PhD-thesis based upon articles such as [Ide93b, Ide93c, Ide93d].

[3] A refinement operator is a device to specialize clauses.

[4] From recent personal communication with Stephen Muggleton, we know Bain and Mug-

A proof of the subsumption theorem, based on the refutation-completeness, is given in the appendix of [BM92]. However, this proof seems unsatisfactory. For example, it does not take *factors* into account, whereas factors are necessary for completeness. Without factors one cannot derive the empty clause $\Box$ from the unsatisfiable set $\{(P(x) \vee P(y)), (\neg P(u) \vee \neg P(v))\}$ (see [GN87]). In fact, our counterexample in Section 5 also depends on factors. Furthermore, it is not always clear how the concepts that are used in the proof are defined, and how the skolemization works. Their proof is based on transforming a refutation-tree into a derivation-tree, but this transformation is not clearly defined and thus insufficient to prove that the transformation can always be performed.

Even though the proof in [BM92] is not quite satisfactory, it is often quoted—sometimes even incorrectly. The two main formulations we have found are the following:

S Let Σ be a set of clauses and C a clause which is not a tautology. Define $\mathcal{R}^0(\Sigma) = \Sigma$ and $\mathcal{R}^n(\Sigma) = \mathcal{R}^{n-1}(\Sigma) \cup \{C : C \text{ is a resolvent of } C_1, C_2 \in \mathcal{R}^{n-1}(\Sigma)\}$. Also define $\mathcal{R}^*(\Sigma) = \mathcal{R}^0(\Sigma) \cup \mathcal{R}^1(\Sigma) \cup \ldots$. Then the subsumption theorem is stated as follows (we assume the authors of [BM92] used '$\vdash$' for what we mean by '$\models$', i.e. logical implication):
$\Sigma \vdash C$ iff there exists a clause $D \in \mathcal{R}^*(\Sigma)$ such that D subsumes C.

S′ Let Σ be a set of clauses and C a clause which is not a tautology. Define $\mathcal{L}^1(\Sigma) = \Sigma$ and $\mathcal{L}^n(\Sigma) = \{C : C \text{ is a resolvent of } C_1 \in \mathcal{L}^{n-1}(\Sigma) \text{ and } C_2 \in \Sigma\}$. Also define $\mathcal{L}^*(\Sigma) = \mathcal{L}^1(\Sigma) \cup \mathcal{L}^2(\Sigma) \cup \ldots$. Then the subsumption theorem is stated as follows:
$\Sigma \models C$ iff there exists a clause $D \in \mathcal{L}^*(\Sigma)$ such that D subsumes C.

S is given in [BM92], **S′** is given in [Mug92]. In [Mug92], Muggleton does not prove **S′**, but refers instead to [BM92]. In other articles such as [Ide93a, LN94b, NLT93], the theorem is also given in the form of **S′**. These articles do not give a proof of **S′**, but refer instead to [BM92] or [Mug92]. That is, they refer to a proof of **S** assuming that this is also a proof of **S′**. But clearly that is not the case, because **S′** demands that at least one of the parent clauses of a clause in $\mathcal{L}^*(\Sigma)$ is a member of Σ, so **S′** is stronger than **S**. In fact, whereas **S** is true, **S′** *is actually false!* If **S′** were true, then input resolution would be refutation-complete (as we will show in Section 5), which it is not. An easy propositional counterexample for the refutation-completeness of input resolution is given on p. 99 of [GN87].

The confusion about **S′** is perhaps a consequence of the subtle distinction between linear resolution and input resolution. **S′** employs a form of *input* resolution, which is a special case of linear resolution. Linear resolution is complete[5], but input resolution is *not* complete. See [CL73] or [GN87].

However, the articles we mentioned do not use **S′** itself. [LN94b, NLT93] are restricted to Horn clauses. It can be shown that for Horn clauses there is

gleton discovered the theorem themselves, independently of [Lee67]. Only afterwards did they found out from references in other literature that their theorem was probably the same as the theorem in Lee's thesis.

[5] It is possible to prove that the subsumption theorem holds in case of linear resolution, but we will not do that here.

no problem. Due to a lack of space we will not prove that here, but in another article [NW95b]. If we examine [Mug92, Ide93a] carefully, then we see that the results of these articles only depend on a special case of $\mathbf{S}'$, namely the case where Σ consists of a single clause. Muggleton and others have used the definition of $\mathcal{L}^n(\{C\})$ to define *nth powers* and *nth roots*. Unfortunately, $\mathbf{S}'$ does not even hold in this special case. We give a counterexample in Section 5. This means that the results of [Mug92, Ide93a] which are consequences of this special case of $\mathbf{S}'$ need to be reconsidered.[6]

The confusion around the subsumption theorem made us investigate this theorem ourselves, which led to the discovery of the mixture of true and false results that we mentioned above. We investigated the subsumption theorem both in the case of unconstrained resolution, and in the case of SLD-resolution for Horn clauses. For the latter case we generalized the definition of SLD-resolution given in [Llo87], following an idea of [MP94]. The main results of both parts of our research can be summarized in the following sequence (where $\mathbf{a}\Rightarrow\mathbf{b}\Rightarrow\mathbf{c}\Rightarrow\mathbf{d}$).

- **a**. The subsumption theorem for unconstrained resolution.
- **b**. The refutation-completeness of unconstrained resolution.
- **c**. The refutation-completeness of SLD-resolution for Horn clauses.
- **d**. The subsumption theorem for SLD-resolution for Horn clauses.
- **e**. $\mathbf{S}'$ is false, even when Σ (the set of premisses) contains only one clause.

We defer **c** and **d** to the second part of our research [NW95b]. **c** is similar to Theorem 8.4 of [Llo87], but our proof of **c** is interesting in that it avoids partial orders and fixpoints, using only the basic definitions of unconstrained resolution and SLD-resolution.

In this paper we focus on **a**, **b** and **e**. In Section 2, we prove **a**. Our proof does not presuppose the refutation-completeness, contrary to the inadequate proof in [BM92]. **b** is well-known, but it is not well-known that **b** is a direct consequence of **a**, as we will show in Section 3 (this fact can also be used to establish the falsity of $\mathbf{S}'$). Thus **b** follows immediately from **a**. On the other hand, **a** can also be proved starting from **b**, as we show in Section 4. This establishes the equivalence of the subsumption theorem and the refutation-completeness: these results have equal power. Finally, in Section 5 we prove **e**, by presenting our counterexample to the special case of $\mathbf{S}'$ that we mentioned.

2 The Subsumption Theorem

In this section, we give a proof of the subsumption theorem. Before starting with our proof, we will first briefly define the main concepts we use. We treat a clause as a *disjunction* of literals, so we consider $P(a)$ and $P(a)\vee P(a)$ as different clauses. However, the results of our paper remain valid also for other notations, for instance if one treats a clause as a *set* of literals instead of a disjunction. For

[6] From recent personal communication with Peter Idestam-Almquist, we know he has adjusted his work from [Ide93a], based on our findings.

convenience, we use $C \subseteq D$ to denote that the set of literals appearing in the disjunction C is a subset of the set of literals in D.

Definition 1 Let C_1 and C_2 be clauses. If C_1 and C_2 have no variables in common, then they are said to be *standardized apart*.

Given C_1 and C_2, let $C_1' = L_1 \vee \ldots \vee L_i \vee \ldots \vee L_m$ and $C_2' = M_1 \vee \ldots \vee M_j \vee \ldots \vee M_n$ be variants of C_1 and C_2 respectively, which are standardized apart ($1 \leq i \leq m$ and $1 \leq j \leq n$). If the substitution θ is a most general unifier (mgu) of the set $\{L_i, \neg M_j\}$, then the clause

$$(L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_m \vee M_1 \vee \ldots \vee M_{j-1} \vee M_{j+1} \vee \ldots \vee M_n)\theta$$

is called a *binary resolvent* of C_1 and C_2. The literals L_i and M_j are said to be the literals *resolved upon*. $\diamond$

Definition 2 Let C be a clause, $L_1, \ldots, L_n$ ($n \geq 1$) unifiable literals from C, and θ an mgu of $\{L_1, \ldots, L_n\}$. Then the clause obtained by deleting $L_2\theta, \ldots, L_n\theta$ from $C\theta$ is called a *factor* of C.

A *resolvent* C of clauses C_1 and C_2 is a binary resolvent of a factor of C_1 and a factor of C_2, where the literals resolved upon are the literals unified by the respective factors. C_1 and C_2 are called the *parent clauses* of C. $\diamond$

Note that any non-empty clause C is a factor of itself, using the empty substitution ε as an mgu of a single literal in C.

Definition 3 Let Σ be a set of clauses and C a clause. A *derivation* of C from Σ is a finite sequence of clauses $R_1, \ldots, R_k = C$, such that each R_i is either in Σ, or a resolvent of two clauses in $\{R_1, \ldots, R_{i-1}\}$. If such a derivation exists, we write $\Sigma \vdash_r C$. A derivation of the empty clause $\Box$ from Σ is called a *refutation* of Σ. $\diamond$

Definition 4 Let C and D be clauses. We say D *subsumes* (or *θ-subsumes*) C if there exists a substitution θ such that $D\theta \subseteq C$.

Let Σ be a set of clauses and C a clause. We say there exists a *deduction* of C from Σ, written as $\Sigma \vdash_d C$, if C is a tautology, or if there exists a clause D such that $\Sigma \vdash_r D$ and D subsumes C. $\diamond$

To illustrate these definitions, we will give an example of a deduction of the clause $C = R(a) \vee S(a)$ from the set $\Sigma = \{(P(x) \vee Q(x) \vee R(x)), (\neg P(x) \vee Q(a)), (\neg P(x) \vee \neg Q(x)), (P(x) \vee \neg Q(x))\}$. Figure 1 shows a derivation of the clause $D = R(a) \vee R(a)$ from Σ. Note that we use the factor $Q(a) \vee R(a)$ of the parent clause $C_6 = Q(x) \vee R(x) \vee Q(a)$ in the last step of the derivation, and also the factor $P(y) \vee R(y)$ of $C_5 = P(y) \vee P(y) \vee R(y)$ in the step leading to C_7. Since D subsumes C, we have $\Sigma \vdash_d C$.

It is not very difficult to see the equivalence between our definition of a derivation, and the definition of $\mathcal{R}^n(\Sigma)$ we gave in Section 1. For instance, in figure 1, C_1, C_2, C_3, C_4, C_1' are variants of clauses in $\mathcal{R}^0(\Sigma)$ (C_1 and C_1' are

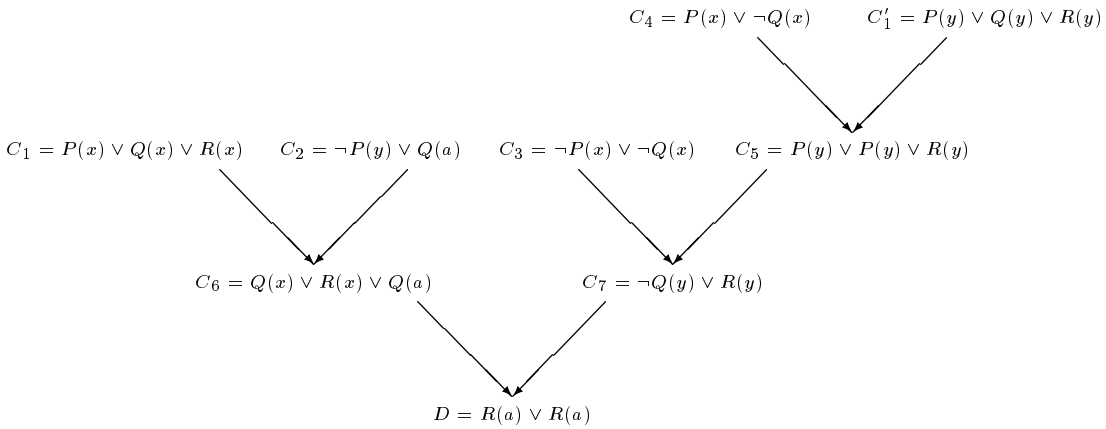

Figure 1: The tree for the derivation of D from Σ

variants of the same clause). C_5, C_6 are in $\mathcal{R}^1(\Sigma)$, C_7 is in $\mathcal{R}^2(\Sigma)$, and D is in $\mathcal{R}^3(\Sigma)$.

The subsumption theorem states that $\Sigma \models C$ iff $\Sigma \vdash_d C$. The 'if'-part of this result follows immediately from the soundness of resolution. We prove the 'only-if' part in a number of successive steps in the following subsections. First we prove the theorem in case both Σ and C are ground, then we prove it in case Σ consists of arbitrary clauses but C is ground, and finally we prove the theorem when neither Σ nor C need to be ground.

2.1 The Subsumption Theorem for Ground Σ and C

First we prove our result for the case when both Σ and C are restricted to ground clauses.

Theorem 1 *Let Σ be a set of ground clauses, and C be a ground clause. If $\Sigma \models C$, then $\Sigma \vdash_d C$.*

Proof If C is a tautology, the theorem is obvious. Assume C is not a tautology. Then we need to find a clause D, such that $\Sigma \vdash_r D$ and $D \subseteq C$ (note that for ground clauses D and C, D subsumes C iff $D \subseteq C$). The proof is by induction on the number of clauses in Σ.

1. Suppose $\Sigma = \{C_1\}$. We will show that $C_1 \subseteq C$. Suppose $C_1 \not\subseteq C$. Then there exists a literal L, such that $L \in C_1$ but $L \notin C$. Let I be an interpretation which makes L true, and all literals in C false (such an I exists, since C is not a tautology). Then I is a model of C_1, but not of C. But that contradicts $\Sigma \models C$. So $C_1 \subseteq C$, and $\Sigma \vdash_d C$.
2. Suppose the theorem holds if $|\Sigma| \leq m$. We will prove that this implies that the theorem also holds if $|\Sigma| = m + 1$. Let $\Sigma = \{C_1, \ldots, C_{m+1}\}$, and $\Sigma' = \{C_1, \ldots, C_m\}$. If C_{m+1} subsumes C or $\Sigma' \models C$, then the theorem holds. So assume C_{m+1} does not subsume C and $\Sigma' \not\models C$.

The idea is to derive, using the induction hypothesis, a number of clauses from which a derivation of a subset of C can be constructed (see figure 2). First note that since $\Sigma \models C$, we have $\Sigma' \models C \vee \neg C_{m+1}$ (using the Deduction Theorem[7]). Let $L_1, \ldots, L_k$ be all the literals in C_{m+1} which are not in C ($k \geq 1$ since C_{m+1} does not subsume C). Then we can write $C_{m+1} = L_1 \vee \ldots \vee L_k \vee C'$, where $C' \subseteq C$ (the order of literals in a clause is not important). Since C does not contain L_i ($1 \leq i \leq k$), the clause $C \vee \neg L_i$ is not a tautology. Also, since $\Sigma' \models C \vee \neg C_{m+1}$ and C_{m+1} is ground, we have that $\Sigma' \models C \vee \neg L_i$, for each i. Then by the induction hypothesis, there exists for each i a ground clause D_i such that $\Sigma' \vdash_r D_i$ and $D_i \subseteq (C \vee \neg L_i)$. We will use C_{m+1} and the derivations from Σ' of these D_i to construct a derivation of a subset of C from Σ. $\neg L_i \in D_i$, otherwise $D_i \subseteq C$ and $\Sigma' \models C$. So we can write each D_i as $\neg L_i \vee D'_i$, and $D'_i \subseteq C$. The case where some D_i contains $\neg L_i$ more than once can be solved by taking a factor of D_i.

Now we can construct a derivation of the ground clause defined as $D = C' \vee D'_1 \vee \ldots \vee D'_k$ from Σ, using C_{m+1} and the derivations of $D_1, \ldots, D_k$ from Σ'. See figure 2. In this tree, the derivations of $D_1, \ldots, D_k$ are indicated by the vertical dots. So we have that $\Sigma \vdash_r D$. Since $C' \subseteq C$, and $D'_i \subseteq C$ for each i, we have that $D \subseteq C$. Hence $\Sigma \vdash_d C$.

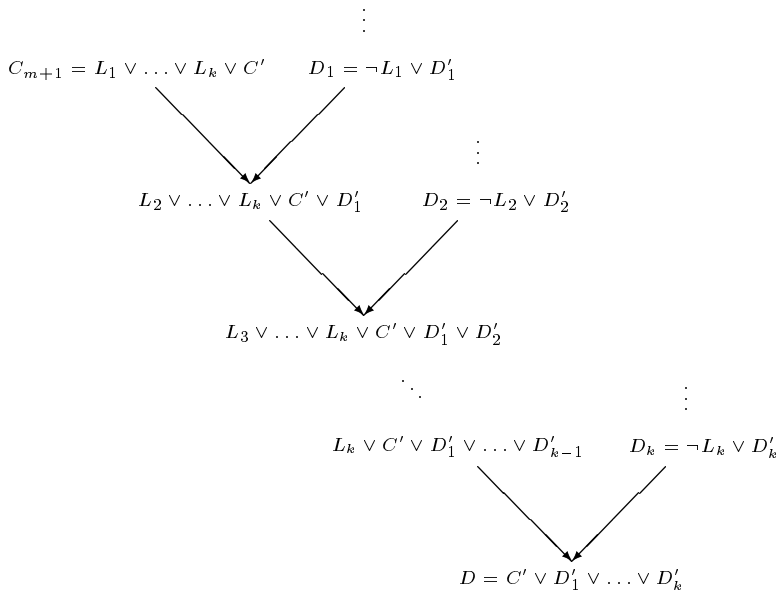

Figure 2: The tree for the derivation of D from Σ

□

[7] $\Sigma \cup \{C\} \models D$ iff $\Sigma \models (C \rightarrow D)$.

2.2 The Subsumption Theorem when C is Ground

In this section, we will prove the theorem in case C is ground and Σ is a set of arbitrary clauses. The idea is to "translate" $\Sigma \models C$ to $\Sigma_g \models C$, where Σ_g is a set of ground instances of clauses of Σ. Then by Theorem 1, there is a clause D such that $\Sigma_g \vdash_r D$, and D subsumes C. Afterwards, we can "lift" this to a deduction of C from Σ.

Theorem 2 (Herbrand, [CL73]) *A set Σ of clauses is unsatisfiable iff there is a finite unsatisfiable set Σ' of ground instances of clauses of Σ.*

Lemma 1 *Let Σ be a set of clauses, and C be a ground clause. If $\Sigma \models C$, then there exists a finite set of clauses Σ_g, where each clause in Σ_g is a ground instance of a clause in Σ, such that $\Sigma_g \models C$.*

Proof Let $C = L_1 \vee \ldots \vee L_k$ $(k \geq 0)$. If Σ is unsatisfiable then the lemma follows immediately from Theorem 2, so suppose Σ is satisfiable. Note that since C is ground, $\neg C$ is equivalent to $\neg L_1 \wedge \ldots \wedge \neg L_k$. Then:

> $\Sigma \models C$ iff (by the Deduction Theorem)
> $\Sigma \cup \{\neg C\}$ is unsatisfiable iff
> $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ is unsatisfiable iff (by Theorem 2)
> there exists a finite unsatisfiable set Σ', consisting of ground instances
> of clauses from $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$.

Since Σ is satisfiable, the unsatisfiable set Σ' must contain one or more members of the set $\{\neg L_1, \ldots, \neg L_k\}$, i.e. $\Sigma' = \Sigma_g \cup \{\neg L_{i_1}, \ldots, \neg L_{i_j}\}$, where Σ_g is a finite non-empty set of ground instances of clauses in Σ. So:

> Σ' is unsatisfiable iff
> $\Sigma_g \cup \{\neg L_{i_1}, \ldots, \neg L_{i_j}\}$ is unsatisfiable iff
> $\Sigma_g \cup \{\neg(L_{i_1} \vee \ldots \vee L_{i_j})\}$ is unsatisfiable iff (by the Deduction Theorem)
> $\Sigma_g \models (L_{i_1} \vee \ldots \vee L_{i_j})$.

Since $\{L_{i_1}, \ldots, L_{i_j}\} \subseteq C$, it follows that $\Sigma_g \models C$. □

The next lemma shows that if a set Σ' consists of instances of clauses in Σ, then a derivation from Σ' can be "lifted" to a derivation from Σ. Similar lifting-lemmas are proved in [CL73, GN87]. We prove our own lifting-lemma, because our definition of resolution slightly differs from the definitions used in those books (we treat a clause as a disjunction, rather than as a set of literals). Because of its rather technical nature, we have deferred the proof to Appendix A.

Lemma 2 (Derivation Lifting) *Let Σ be a set of clauses, and Σ' be a set of instances of clauses in Σ. Suppose $R'_1, \ldots, R'_k$ is a derivation of the clause R'_k from Σ'. Then there exists a derivation $R_1, \ldots, R_k$ of the clause R_k from Σ, such that R'_i is an instance of R_i, for each i.*

Theorem 3 *Let Σ be a set of clauses, and C be a ground clause. If $\Sigma \models C$, then $\Sigma \vdash_d C$.*

Proof If C is a tautology, the theorem is obvious. Assume C is not a tautology. We want to find a clause D such that $\Sigma \vdash_r D$ and D subsumes C. From $\Sigma \models C$ and Lemma 1, there exists a finite set Σ_g such that each clause in Σ_g is a ground instance of a clause in Σ, and $\Sigma_g \models C$. Then by Theorem 1, there exists a ground clause D' such that $\Sigma_g \vdash_r D'$, and $D' \subseteq C$. Let $R'_1, \ldots, R'_k = D'$ be a derivation of D' from Σ_g. From Lemma 2, we can "lift" this to a derivation $R_1, \ldots, R_k$ of R_k from Σ, where $R_k\theta = D'$ for some θ. Let $D = R_k$. Then $D\theta = D' \subseteq C$. Hence D subsumes C. □

2.3 The Subsumption Theorem (General Case)

In this subsection, we will prove the subsumption theorem for arbitrary Σ and C. In the proof, we will use a *Skolemizing substitution*.

Definition 5 Let Σ be a set of clauses, and C a clause. Let $x_1, \ldots, x_n$ be all the variables appearing in C and $a_1, \ldots, a_n$ be distinct constants not appearing in Σ or C. Then $\{x_1/a_1, \ldots, x_n/a_n\}$ is called a *Skolemizing substitution* for C w.r.t. Σ. ◇

Lemma 3 *Let C and D be clauses. Let $\theta = \{x_1/a_1, \ldots, x_n/a_n\}$ be a Skolemizing substitution for C w.r.t. D. If D subsumes $C\theta$, then D also subsumes C.*

Proof Since D subsumes $C\theta$, there exists a substitution σ such that $D\sigma \subseteq C\theta$. Let σ be the substitution $\{y_1/t_1, \ldots, y_m/t_m\}$. Let σ' be the substitution obtained from σ by replacing each a_i by x_i in every t_j. Note that $\sigma = \sigma'\theta$. Since θ only replaces each x_i by a_i $(1 \leq i \leq n)$, it follows that $D\sigma' \subseteq C$, so D subsumes C. □

Theorem 4 (Subsumption Theorem) *Let Σ be a set of clauses, and C be a clause. Then $\Sigma \models C$ iff $\Sigma \vdash_d C$.*

Proof

$\Leftarrow$: Follows immediately from the soundness of resolution, and the fact that if D subsumes C, then $D \models C$.

$\Rightarrow$: If C is a tautology, the theorem is obvious. Assume C is not a tautology. Let θ be a Skolemizing substitution for C w.r.t. Σ. Then $C\theta$ is a ground clause which is not a tautology, and $\Sigma \models C\theta$. So by Theorem 3, there is a clause D such that $\Sigma \vdash_r D$ and D subsumes $C\theta$. Since θ is a Skolemizing substitution for C w.r.t. Σ, and D can only contain constants appearing in Σ, θ is also a Skolemizing substitution for C w.r.t. D. Then by Lemma 3, D also subsumes C. Thus we have $\Sigma \vdash_d C$. □

3 The Refutation-Completeness of Resolution

The subsumption theorem actually tells us that resolution and subsumption form a complete set of proof-rules for clauses. A form of completeness that is usually stated in the literature on resolution is the refutation-completeness. This is an easy consequence of the subsumption theorem.

Theorem 5 (Refutation-Completeness of Resolution) *Let Σ be a set of clauses. Then Σ is unsatisfiable iff $\Sigma \vdash_r \Box$.*

Proof

$\Leftarrow$: Follows immediately from the soundness of resolution.

$\Rightarrow$: Suppose Σ is unsatisfiable. Then $\Sigma \models \Box$. So by Theorem 4, there exists a clause D, such that $\Sigma \vdash_r D$ and D subsumes the empty clause $\Box$. But $\Box$ is the only clause which subsumes $\Box$, so $D = \Box$. $\Box$

4 The Other Way Around

In Section 3, we showed that the refutation-completeness is a direct consequence of the subsumption theorem. In this section, we will show the converse: that we can obtain the subsumption theorem from the refutation-completeness. This establishes the equivalence of the subsumption theorem and the refutation-completeness (i.e., the one can be proved from the other).

To prove the subsumption theorem from the refutation-completeness, we will show how to turn a refutation of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ into a deduction of $L_1 \vee \ldots \vee L_k$. Thus our proof, which has some similarities with the unsatisfactory proof of the subsumption theorem in [BM92], is constructive. We start with an example. Suppose $\Sigma = \{(P(x) \vee \neg R(f(f(b)))), (R(f(x)) \vee \neg R(x))\}$, and $C = P(x) \vee Q(x) \vee \neg R(b)$. It is not difficult to see that $\Sigma \models C$. We would like to produce a deduction of C from Σ. First we note that $\theta = \{x/a\}$ is a Skolemizing substitution for C w.r.t. Σ. Since $\Sigma \models C\theta$, we know that $\Sigma \cup \{\neg P(a), \neg Q(a), R(b)\}$ is unsatisfiable, and hence (by the refutation-completeness) has a refutation. Figure 3 shows such a refutation.

Now by omitting the leaves of the refutation-tree which come from $C\theta$ (the framed literals) and by making appropriate changes in the tree, we get a derivation of the clause $D = P(x) \vee \neg R(b)$. See figure 4. D subsumes C, so we have turned the refutation of figure 3 into a deduction of C from Σ.
This approach also works in the general case. The following lemma does most of the work.

Lemma 4 *Let Σ be a set of clauses, and $C = L_1 \vee \ldots \vee L_k$ be a non-tautologous ground clause. If $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_r \Box$, then $\Sigma \vdash_d C$.*

Proof Suppose $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_r \Box$. Then there exists a refutation $R_1, \ldots, R_n = \Box$ of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$. Let r be the number of resolvents in this sequence ($r = n-$the number of members of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ in $R_1, \ldots, R_n$). We prove the lemma by induction on r.

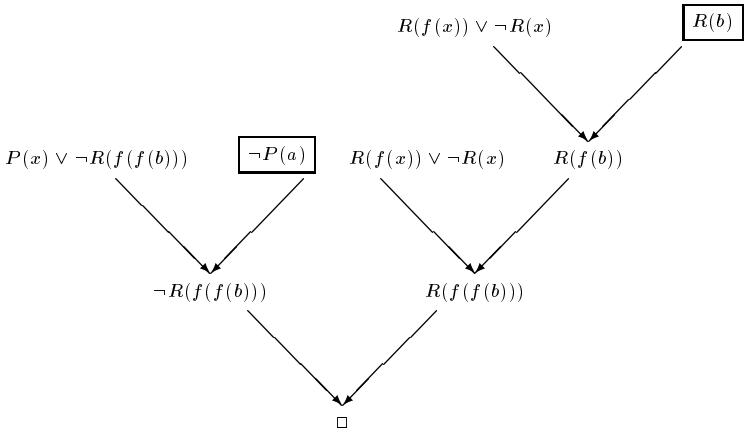

Figure 3: A refutation of $\Sigma \cup \{\neg P(a), \neg Q(a), R(b)\}$

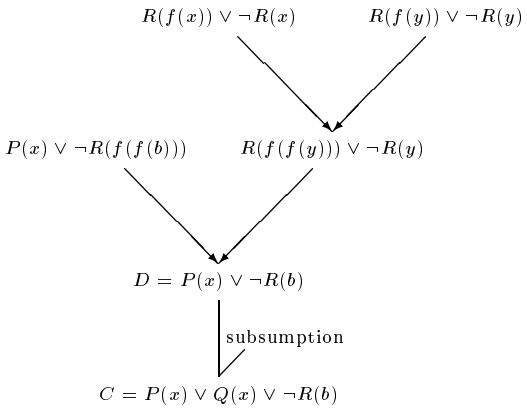

Figure 4: A deduction of C from Σ, obtained by transforming the previous figure

1. Suppose $r = 0$. Then we must have $R_n = \square \in \Sigma$, so obviously the lemma holds.
2. Suppose the lemma holds for $r \leq m$. We will prove that this implies that the lemma also holds for $r = m + 1$. Let $R_1, \ldots, R_n = \square$ be a refutation of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ containing $m + 1$ resolvents. Let R_i be the first resolvent. Then $R_1, \ldots, R_n = \square$ is a refutation of $\Sigma \cup \{R_i\} \cup \{\neg L_1, \ldots, \neg L_k\}$ containing only m resolvents, since R_i is now one of the original premisses. Hence by the induction hypothesis, there is a clause D, such that $\Sigma \cup \{R_i\} \vdash_r D$ and D subsumes C.
   Suppose R_i is itself a resolvent of two members of Σ. Then we also have $\Sigma \vdash_r D$, so the lemma holds in this case. Note that R_i cannot be a resolvent of two members of $\{\neg L_1, \ldots, \neg L_k\}$ because this set does not contain a complementary pair, due to the fact that C is not a tautology.
   The only remaining case we have to check, is where R_i is a resolvent of

$C' \in \Sigma$ and some $\neg L_s$ $(1 \leq s \leq k)$. Let $C' = M_1 \vee \ldots \vee M_j \vee \ldots \vee M_h$. Suppose R_i is a binary resolvent of $(M_1 \vee \ldots \vee M_j)\sigma$ (a factor of C', using σ as an mgu of $\{M_j, \ldots, M_h\}$) and $\neg L_s$, with θ as mgu of $M_j\sigma$ and L_s. Then $R_i = (M_1 \vee \ldots \vee M_{j-1})\sigma\theta$ and $C'\sigma\theta = R_i \vee L_s \vee \ldots \vee L_s$ ($h - j + 1$ copies of L_s), since $M_j, \ldots, M_h$ are all unified to L_s by $\sigma\theta$.
Now replace each time R_i appears as leaf in the derivation-tree of D, by $C'\sigma\theta = R_i \vee L_s \vee \ldots \vee L_s$, and add $L_s \vee \ldots \vee L_s$ to all decendants of such an R_i-leaf. Then we obtain a derivation of $D \vee L_s \vee \ldots \vee L_s$ from $\Sigma \cup \{C'\sigma\theta\}$. Since $C'\sigma\theta$ is an instance of a clause from Σ, we can lift (by Lemma 2) this derivation to a derivation from Σ of a clause D', which has $D \vee L_s \vee \ldots \vee L_s$ as an instance. Since D subsumes C, D' also subsumes C. Hence $\Sigma \vdash_d C$.

□

Now we can prove the subsumption theorem (Theorem 4) once more, this time starting from Theorem 5.

Theorem 4 (Subsumption Theorem) Let Σ be a set of clauses, and C be a clause. Then $\Sigma \models C$ iff $\Sigma \vdash_d C$.

Proof

$\Leftarrow$: By the soundness of resolution, and the fact that if D subsumes C, then $D \models C$.

$\Rightarrow$: If C is a tautology, the theorem is obvious. Assume C is not a tautology. Let θ be a Skolemizing substitution for C w.r.t. Σ. Let $C\theta$ be the clause $L_1 \vee \ldots \vee L_k$. Since C is not a tautology, $C\theta$ is not a tautology. $C\theta$ is ground and $\Sigma \models C\theta$, so the set of clauses $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ is unsatisfiable. Then it follows from Theorem 5 that $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_r \Box$. Therefore by Lemma 4, there exists a clause D such that $\Sigma \vdash_r D$, and D subsumes $C\theta$. From Lemma 3, D also subsumes C itself. Hence $\Sigma \vdash_d C$. □

Now that we have shown that the subsumption theorem can be proved from the refutation-completeness, and vice versa, we also have the following:

Theorem 6 *For unconstrained resolution, the subsumption theorem and the refutation-completeness are equivalent.*

5 The Incompleteness of Input Resolution

Note that if **S′** (the subsumption theorem for input resolution) that we mentioned in Section 1 were true, then it would follow along the same lines as Theorem 5 that input resolution is refutation-complete. However, since it is well-known that input resolution is *not* refutation-complete [CL73, GN87], this again shows that **S′** cannot be true. Since [Mug92, Ide93a] only use the special case of **S′** where Σ contains only one clause, we investigate this here. We will show that **S′** is not even true in this special case. Hence the counterexample we give here

is relevant for [Mug92, Ide93a], and also for other results based on $\mathbf{S}'$. In our counterexample we let $\Sigma = \{C\}$, with C:

$$C = P(x_1, x_2) \vee Q(x_2, x_3) \vee \neg Q(x_3, x_4) \vee \neg P(x_4, x_1).$$

Figure 5 shows that clause D (see below) can be derived from C by unconstrained resolution. This also shows that $C \models D$. Figure 5 makes use of the clauses listed below. C_1, C_2, C_3, C_4 are variants of C. D_1 is a binary resolvent of C_1 and C_2, D_2 is a binary resolvent of C_3 and C_4 (the underlined literals are the literals resolved upon). D'_1 is a factor of D_1, using the subtitution $\{x_5/x_1, x_6/x_2\}$. D'_2 is a factor of D_2, using $\{x_{11}/x_{12}, x_{13}/x_9\}$. Finally, D is a binary resolvent of D'_1 and D'_2.

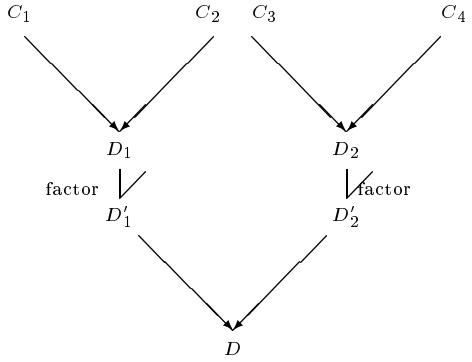

Figure 5: The derivation of D from C by unconstrained resolution

$$\begin{array}{rcl}
C_1 & = & P(x_1,x_2) \vee \underline{Q(x_2,x_3)} \vee \neg Q(x_3,x_4) \vee \neg P(x_4,x_1) \\
C_2 & = & P(x_5,x_6) \vee \overline{Q(x_6,x_7)} \vee \underline{\neg Q(x_7,x_8)} \vee \neg P(x_8,x_5) \\
C_3 & = & P(x_9,x_{10}) \vee \underline{Q(x_{10},x_{11})} \vee \neg Q(x_{11},x_{12}) \vee \neg P(x_{12},x_9) \\
C_4 & = & P(x_{13},x_{14}) \vee \overline{Q(x_{14},x_{15})} \vee \underline{\neg Q(x_{15},x_{16})} \vee \neg P(x_{16},x_{13}) \\
D_1 & = & P(x_1,x_2) \vee \neg Q(x_3,x_4) \vee \underline{\neg P(x_4,x_1)} \vee \overline{P(x_5,x_6)} \vee Q(x_6,x_2) \vee \\
 & & \neg P(x_3,x_5) \\
D_2 & = & P(x_9,x_{10}) \vee \neg Q(x_{11},x_{12}) \vee \neg P(x_{12},x_9) \vee P(x_{13},x_{14}) \vee \\
 & & Q(x_{14},x_{10}) \vee \neg P(x_{11},x_{13}) \\
D'_1 & = & \underline{P(x_1,x_2)} \vee \neg Q(x_3,x_4) \vee \neg P(x_4,x_1) \vee Q(x_2,x_2) \vee \neg P(x_3,x_1) \\
D'_2 & = & \overline{P(x_9,x_{10})} \vee \neg Q(x_{12},x_{12}) \vee \underline{\neg P(x_{12},x_9)} \vee P(x_9,x_{14}) \vee Q(x_{14},x_{10}) \\
D & = & \neg Q(x_3,x_4) \vee \neg P(x_4,x_1) \vee \overline{Q(x_2,x_2)} \vee \neg P(x_3,x_1) \vee P(x_2,x_{10}) \vee \\
 & & \neg Q(x_1,x_1) \vee P(x_2,x_{14}) \vee Q(x_{14},x_{10})
\end{array}$$

So D can be derived from C using unconstrained resolution. However, neither D nor a clause which subsumes D can be derived from C using only *input* resolution. We prove this in Proposition 1 (see the Introduction of this paper for the definition of $\mathcal{L}^n(\Sigma)$ and $\mathcal{L}^*(\Sigma)$). This shows that the subsumption theorem does not hold for input resolution, not even if Σ contains only one clause.

Lemma 5 *Let C be as defined above. Then for each $n \geq 1$: if $E \in \mathcal{L}^n(\{C\})$, then E contains an instance of $P(x_1, x_2) \vee \neg P(x_4, x_1)$ or an instance of $Q(x_2, x_3) \vee \neg Q(x_3, x_4)$.*

Proof By induction on n:

1. $\mathcal{L}^1(\{C\}) = \{C\}$, so the lemma is obvious for $n = 1$.
2. Suppose the lemma holds for $n \leq m$. Let $E \in \mathcal{L}^{m+1}(\{C\})$. Note that the only factor of C is C itself. Therefore E is a binary resolvent of C and a factor of a clause in $\mathcal{L}^m(\{C\})$. Let θ be the mgu used in obtaining this binary resolvent. If $P(x_1, x_2)$ or $\neg P(x_4, x_1)$ is the literal resolved upon in C, then E must contain $(Q(x_2, x_3) \vee \neg Q(x_3, x_4))\theta$. Otherwise $Q(x_2, x_3)$ or $\neg Q(x_3, x_4)$ is the literal resolved upon in C, so then E contains $(P(x_1, x_2) \vee \neg P(x_4, x_1))\theta$.

□

Proposition 1 *Let C and D be as defined above. Then $\mathcal{L}^*(\{C\})$ does not contain a clause which subsumes D.*

Proof Suppose $E \in \mathcal{L}^*(\{C\})$. From Lemma 5 and the definition of $\mathcal{L}^*(\{C\})$, we know that E contains an instance of $P(x_1, x_2) \vee \neg P(x_4, x_1)$ or an instance of $Q(x_2, x_3) \vee \neg Q(x_3, x_4)$. It is easy to see that neither $P(x_1, x_2) \vee \neg P(x_4, x_1)$ nor $Q(x_2, x_3) \vee \neg Q(x_3, x_4)$ subsumes D. Then E does not subsume D. □

6 Conclusion

This paper forms the first part of our research concerning the subsumption theorem. This part pertains to unconstrained resolution. The second part of our research is concerned with SLD-resolution and Horn clauses, and is described in [NW95b]. There we show that the subsumption theorem for SLD-resolution is equivalent with the refutation-completeness of SLD-resolution.

In this paper, we discussed the importance of the subsumption theorem in ILP. No really rigorous proof of this theorem for unconstrained resolution was until now available, and applications of the theorem in the literature often use the incorrect version $\mathbf{S}'$. A proof of the subsumption theorem for unconstrained resolution was given by us. The refutation-completeness of unconstrained resolution was then shown to be an easy corollary of this theorem. Since the subsumption theorem in turn also follows from the refutation-completeness (as proved in Section 4), we have in fact proved that the subsumption theorem and the refutation-completeness are equivalent.

Finally we showed that $\mathbf{S}'$ is not even true when the set of premisses consists of only one clause. This means that results based on $\mathbf{S}'$ or its special case, among which are results on nth powers and nth roots, need to be reconsidered.

References

[BM92] Bain, M., and Muggleton, S., 'Non-monotonic Learning', in: Muggleton, S. (ed.), *Inductive Logic Programming*, APIC series, no. 38, Academic Press, 1992, pp. 145–153.

[CL73] Chang, C. L., and Lee, R. C. T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, San Diego, 1973.

[GN87] Genesereth, M. R., and Nilsson, N. J., *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Palo Alto, 1987.

[Ide93a] Idestam-Almquist, P., *Generalization of Clauses*, PhD Thesis, Stockholm University, 1993.

[Ide93b] Idestam-Almquist, P., 'Generalization under Implication by Recursive Anti-Unification', in: *Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 1993.

[Ide93c] Idestam-Almquist, P., 'Generalization under Implication by Using Or-Introduction', in: *Proceedings of the European Conference on Machine Learning-93*, Springer Verlag, 1993.

[Ide93d] Idestam-Almquist, P., 'Generalization under Implication: Expansion of Clauses for Indirect Roots', in: *Scandinavian Conference on Artificial Intelligence-93*, IOS Press, Amsterdam, Netherlands, 1993.

[Kow70] Kowalski, R., 'The Case for Using Equality Axioms in Automatic Demonstration', in: *Proc. of the Symposium on Automatic Demonstration*, Lecture Notes in Mathematics 125, Springer Verlag, 1970, pp. 112–127.

[LN94a] van der Laag, P., and Nienhuys-Cheng, S.-H., 'Existence and Nonexistence of Complete Refinement Operators', in: *Proc. the European Conference on Machine Learning (ECML-94)*, Lecture Notes in Artificial Intelligence 784, Springer-Verlag, pp. 307–322.

[LN94b] van der Laag, P., and Nienhuys-Cheng, S.-H., 'A Note on Ideal Refinement Operators in Inductive Logic Programming', in: Wrobel, S. (ed.), *Proc. of the Fourth Int. Workshop on Inductive Logic Programming (ILP-94)*, Bad Honnef, Germany, 1994, pp. 247–262.

[Lee67] Lee, R. C. T., *A Completeness Theorem and a Computer Program for Finding Theorems Derivable from Given Axioms*, PhD Thesis, University of California, Berkeley, 1967.

[Llo87] Lloyd, J. W., *Foundations of Logic Programming*, Second edition, Springer-Verlag, Berlin, 1987.

[Mug92] Muggleton, S., 'Inverting Implication', in: Muggleton, S. H., and Furukawa, K. (eds.), *Proc. of the Second Int. Workshop on Inductive Logic Programming (ILP-92)*, ICOT Technical Memorandum TM-1182, 1992.

[MP94] Muggleton, S., and Page, C. D., 'Self-Saturation of Definite Clauses', in: Wrobel, S. (ed.), *Proc. of the Fourth Int. Workshop on Inductive Logic Programming (ILP-94)*, Bad Honnef, Germany, 1994, pp. 161–174.

[NLT93] Nienhuys-Cheng, S.-H., van der Laag, P., and van der Torre, L., 'Constructing Refinement Operators by Deconstructing Logical Implication', in: *Proc. of the Third Congress of the Italian Association for Artificial Intelligence (AI*IA93)*, Lecture Notes in Artificial Intelligence 728, Springer-Verlag, pp. 178–189.

[NW95a] Nienhuys-Cheng, S.-H., and de Wolf, R., 'The Subsumption Theorem in Inductive Logic Programming: Facts and Fallacies', to appear in: *Proc. of the Fifth Workshop on Inductive Logic Programming (ILP-95, workreport)*, Leuven, September 1995.

[NW95b] Nienhuys-Cheng, S.-H., and de Wolf, R., 'The Subsumption Theorem Revisited: Restricted to SLD-resolution', to appear in: *Proc. of Computing Science in the Netherlands (CSN-95)*, Utrecht, November 1995.

[SCL69] Slagle, J. R., Chang, C. L., and Lee, R. C. T., 'Completeness Theorems for Semantic Resolution in Consequence-finding', in: *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI-69)*, 1969, pp. 281–285.

A A Proof of the Lifting Lemma

In this appendix we give the rather technical proof of the lifting lemma.

Lemma 6 *If C_1' and C_2' are instances of C_1 and C_2, respectively, and if C' is a resolvent of C_1' and C_2', then there is a resolvent C of C_1 and C_2, such that C' is an instance of C.*

Proof We assume without loss of generality that C_1 and C_2, and C_1' and C_2' are standardized apart. Let $C_1 = L_1 \vee \ldots \vee L_m$, $C_2 = M_1 \vee \ldots \vee M_n$, $C_1' = C_1\sigma_1$, and $C_2' = C_2\sigma_2$ (here we can assume σ_k only acts on variables in C_k, $k = 1, 2$). Suppose C' is a resolvent of C_1' and C_2'. Then C' is a binary resolvent of a factor of C_1' and a factor of C_2'.

For notational convenience, we assume without loss of generality that the factor of C_1' is $(L_1 \vee \ldots \vee L_i)\sigma_1\theta_1$, where θ_1 is an mgu of $\{L_i\sigma_1, \ldots, L_m\sigma_1\}$. Similarly, the factor of C_2' that is used, is $(M_1 \vee \ldots \vee M_j)\sigma_2\theta_2$, where θ_2 is an mgu of $\{M_j\sigma_2, \ldots, M_n\sigma_2\}$. $L_i\sigma_1\theta_1$ and $M_j\sigma_2\theta_2$ are the literals resolved upon, say with mgu μ. Abbreviate $L_1 \vee \ldots \vee L_{i-1}$ to D_1, and $M_1 \vee \ldots \vee M_{j-1}$ to D_2. Then $C' = (D_1\sigma_1\theta_1 \vee D_2\sigma_2\theta_2)\mu$. By our assumption of standardizing apart, this can be written as $C' = (D_1 \vee D_2)\sigma_1\theta_1\sigma_2\theta_2\mu$.

Let γ_1 be an mgu of $\{L_i, \ldots, L_m\}$. Then $(L_1 \vee \ldots \vee L_i)\gamma_1$ is a factor of C_1. Note that $\sigma_1\theta_1$ is a unifier of $L_i, \ldots, L_m$. Since γ_1 is an mgu of $\{L_i, \ldots, L_m\}$, there exists a substitution δ_1 such that $\sigma_1\theta_1 = \gamma_1\delta_1$. Similarly, $(M_1 \vee \ldots \vee M_j)\gamma_2$ is a factor of C_2, with γ_2 as mgu of $\{M_j, \ldots, M_n\}$, and there is a δ_2 such that $\sigma_2\theta_2 = \gamma_2\delta_2$.

Since $L_i\sigma_1\theta_1$ and $\neg M_j\sigma_2\theta_2$ can be unified (they have μ as mgu) and γ_k is more general than $\sigma_k\theta_k$ $(k = 1, 2)$, $L_i\gamma_1$ and $\neg M_j\gamma_2$ can be unified. Let θ be an mgu of $L_i\gamma_1$ and $\neg M_j\gamma_2$. Define $C = (D_1\gamma_1 \vee D_2\gamma_2)\theta$, which can be written as $C = (D_1 \vee D_2)\gamma_1\gamma_2\theta$. Since C is a binary resolvent of the above-mentioned factors of C_1 and C_2, it is a resolvent of C_1 and C_2 (see figure 6 for illustration).

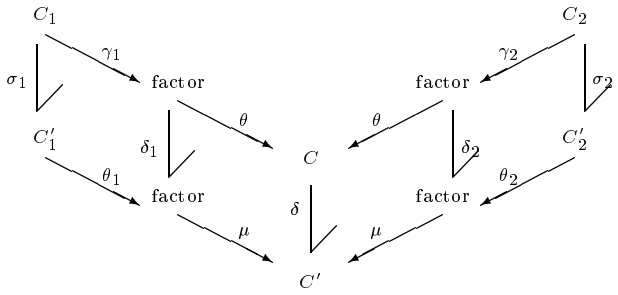

Figure 6: Lifting a resolvent

It remains to show that C' is an instance of C. Since $L_i\gamma_1\delta_1\delta_2\mu = L_i\sigma_1\theta_1\delta_2\mu = L_i\sigma_1\theta_1\mu = \neg M_j\sigma_2\theta_2\mu = \neg M_j\gamma_2\delta_2\mu = \neg M_j\gamma_2\delta_1\delta_2\mu$, the substitution $\delta_1\delta_2\mu$ is a unifier of $L_i\gamma_1$ and $\neg M_j\gamma_2$. θ is an mgu of $L_i\gamma_1$ and $\neg M_j\gamma_2$, so there exists a substitution δ such that $\delta_1\delta_2\mu = \theta\delta$. Therefore $C' = (D_1 \vee D_2)\sigma_1\theta_1\sigma_2\theta_2\mu = (D_1 \vee D_2)\gamma_1\delta_1\gamma_2\delta_2\mu = (D_1 \vee D_2)\gamma_1\gamma_2\delta_1\delta_2\mu = (D_1 \vee D_2)\gamma_1\gamma_2\theta\delta = C\delta$. Hence C' is an instance of C. □

Lemma 2 (Derivation Lifting) Let Σ be a set of clauses, and Σ' be a set of instances of clauses in Σ. Suppose $R_1', \ldots, R_k'$ is a derivation of the clause R_k' from Σ'. Then there exists a derivation $R_1, \ldots, R_k$ of the clause R_k from Σ, such that R_i' is an instance of R_i, for each i.

Proof The proof is by induction on k.

1. If $k = 1$, then $R_1' \in \Sigma'$, so there is a clause $R_1 \in \Sigma$ of which R_1' is an instance.
2. Suppose the lemma holds if $k \leq m$. Let $R_1', \ldots, R_m', R_{m+1}'$ be a derivation of R_{m+1}' from Σ'. By the induction hypothesis, there exists a derivation $R_1, \ldots, R_m$ of R_m from Σ, such that R_i' is an instance of R_i, for all i $1 \leq i \leq m$. If $R_{m+1}' \in \Sigma'$, the lemma is obvious. Otherwise, R_{m+1}' is a resolvent of clauses $R_i', R_j' \in \{R_1', \ldots, R_m'\}$. It follows from Lemma 6 that there exists a resolvent R_{m+1} of R_i and R_j such that R_{m+1}' is an instance of R_{m+1}.

□