The Subsumption Theorem in Inductive Logic Programming: Facts and Fallacies

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

Abstract. The subsumption theorem is an important theorem concerning resolution. Essentially, it says that if a set of clauses Σ logically implies a clause C, then either 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 Inductive Logic Programming, interest in this theorem is increasing since its independent rediscovery by Bain and Muggleton. 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 just a special case of this theorem. We also show that the subsumption theorem does not hold when only input resolution is used, not even in case Σ contains only one clause. Since some recent articles assume the contrary, certain 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. Resolution is used for deduction, but also for induction in the form of inverse 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. If $\Sigma \models C$, then 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 probably first stated and proved by Lee in 1967 in his PhD-thesis [11]. 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, the theorem is mentioned nowhere in the standard reference-books concerning resolution, not even in Lee's own book [2].

We are aware of two other early papers concerning the subsumption theorem.[1] [8] gives a proof of the subsumption theorem for unconstrained resolution, which is rather

[1] We would like to thank Stephen Muggleton for sending us a copy of [8] and a reference to [20].

sketchy and presupposes knowledge of semantic trees. This makes that proof harder to understand than the proof we will give here, which only needs the elementary definitions of logic and resolution. [20] proves a version of the subsumption theorem for semantic resolution. However, these papers are rather unknown in the ILP-community, where different versions of the subsumption theorem are used, at least one of which is false. Our aim in this paper is to clarify this confusion.

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 generalizations in ILP, 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, since it states that implication = resolution + subsumption. It is used for instance in [4, 13][2] for inverse resolution. In [10], the theorem is used to extend the result of [9] that there does not exist an ideal refinement operator in subsumption, to the result that there is no ideal refinement operator in logical implication. In [15], 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 [1], people have started paying attention to it.

A reproof of the subsumption theorem is given in the appendix of [1]. However, we are not completely satisfied with their proof. For example, it does not take *factors* into account, whereas factors are necessary for completeness. Without factors one cannot derive the empty clause $\square$ from the unsatisfiable set $\{(P(x) \vee P(y)), (\neg P(u) \vee \neg P(v))\}$ (see [3]). 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 we feel this transformation is not clearly defined. Despite these points, Bain and Muggleton certainly deserve a lot of credit for their rediscovery of this theorem[3], which is very important in ILP. The proof in [1] is often quoted, and is sometimes also referred to as a proof for other kinds of resolution. 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 $\mathcal{R}^0(\Sigma) = \Sigma$

[2] [4] is a PhD-thesis based upon articles such as [5, 6, 7].

[3] From recent personal communication with Stephen Muggleton, we know Bain and Muggleton discovered the theorem themselves, independently of [11]. Only afterwards did they found out from references in other literature that their theorem was probably the same as Lee's.

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 [1] 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 [1], **S′** is given in [13]. In [13], Muggleton does not prove **S′**, but refers instead to [1]. In other articles such as [4, 10, 15], the theorem is also given in the form of **S′**. These articles do not give a proof of **S′**, but refer instead to [1] or [13]. 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!* A counterexample is the deduction from the set Σ that we use to illustrate our definitions in Section 2: no clause in $\mathcal{L}^*(\Sigma)$ subsumes $R(a) \vee S(a)$. Besides, if **S′** were true, then input resolution would be refutation-complete (as we will show in Section 3), which it is not. An easy propositional counterexample for the refutation-completeness of input resolution is given on p. 99 of [3].

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, but input resolution is not complete. See [2] or [3].

However, the articles we mentioned do not use **S′** itself. [10, 15] are restricted to Horn clauses. It can be shown that for Horn clauses there is no problem. Due to a lack of space we will not prove that here. If we examine [13, 4] carefully, then we see that the results of these articles only depend on a special case of **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, **S′** does not even hold in this special case. We give a counterexample in Section 4. This means that the results of [13, 4] which are consequences of this special case of **S′** need to be reconsidered.[4]

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. In this paper, we focus on three results of our research:

1. In Section 2, we prove **S**, the subsumption theorem for unconstrained resolution. Our proof does not presuppose the refutation-completeness, contrary to the proof in [1].
2. The refutation-completeness of unconstrained resolution is an immediate consequence of **S**, as we show in Section 3.
3. **S′** is false, even when Σ (the set of premisses) contains only one clause. In Section 4, we present our counterexample.

Our other results on the subsumption theorem are briefly described in the future work, Section 6.

[4] From recent personal communication with Peter Idestam-Almquist, we know he has adjusted his work from [4], incorporating our results.

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 convenience, we use $C \subseteq D$ to denote that the set of literals in 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 in 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. Factors are sometimes built into the resolution step itself—for instance in Robinson's original paper [19], where *sets* of literals from both parent clauses are unified—but we have chosen to seperate the definitions of a factor and a binary resolvent. The reason for this is that binary resolution without factors is sufficient in case of SLD-resolution for Horn clauses.

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 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 $\mathcal{R}^3(\Sigma)$.

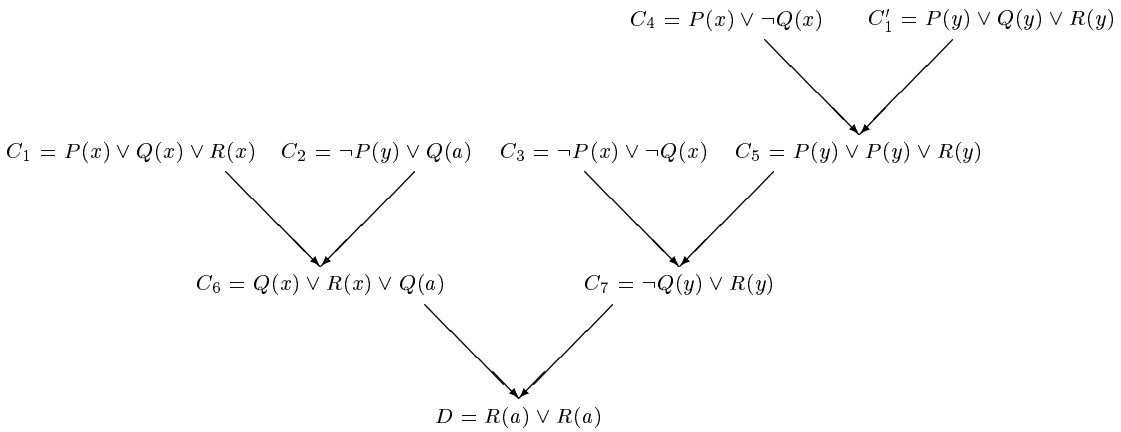

Figure 1: The tree for the derivation of D from Σ

The subsumption theorem states that if $\Sigma \models C$, then $\Sigma \vdash_d C$. We prove this 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 the subsumption theorem 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 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. First note that since

$\Sigma \models C$, we have $\Sigma' \models C \vee \neg C_{m+1}$ (using the Deduction Theorem[5]). 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 $C' \subseteq C$. 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 $\Sigma \vdash_d C$.

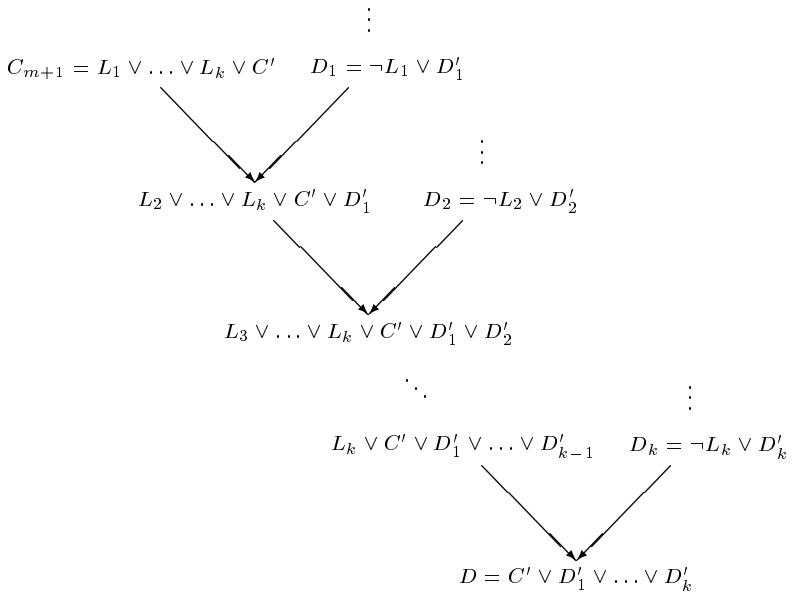

Figure 2: The tree for the derivation of D from Σ

□

Notice that the previous proof in fact contains an algorithm, a procedure to construct the deduction of C from Σ. First the algorithm checks if C is a tautology, which is the case iff C contains a complementary pair of literals. If not, then the construction proceeds by induction on $|\Sigma|$. If $\Sigma = \{C_1\}$, then $C_1 \subseteq C$. If $\Sigma = \{C_1, \ldots, C_m, C_{m+1}\}$, then we can write $C_{m+1} = L_1 \vee \ldots \vee L_k \vee C'$, where $C' \subseteq C$. As our proof has shown, we can then find deductions of $\neg L_i \vee C$ from $\Sigma' = \{C_1, \ldots, C_m\}$ ($i = 1, \ldots, k$). C_{m+1} can be combined with these deductions of $\neg L_i \vee C$ to form a deduction of C from Σ, as shown in figure 2.

2.2 The subsumption theorem when C is ground

In this section, we will prove the subsumption 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

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

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. We can "lift" this to a deduction of C from Σ.

Theorem 2 (Herbrand, [2]) *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 two lemmas show 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. We omit the proof of the first lemma, which is fairly straightforward. It is similar to Lemma 5.1 of [2], taking into account that we use a slightly different definition of a resolvent.

Lemma 2 *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.*

Lemma 3 (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 2 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}.

□

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 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. It follows from Lemma 3 that 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 4 *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. If $\Sigma \models C$, then $\Sigma \vdash_d C$.*

Proof 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 4, D subsumes C. Hence $\Sigma \vdash_d C$. □

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 (note that the converses of Theorems 4 and 5 follow immediately from the soundness of resolution and subsumption).

Theorem 5 (Refutation-completeness of Resolution) *Let Σ be a set of clauses. If Σ is unsatisfiable, then $\Sigma \vdash_r \square$.*

Proof 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 $D = \Box$. $\Box$

Note that if the $\mathbf{S}'$ that we mentioned in Section 1 were true, then it would follow along the same lines as the theorem above that input resolution is refutation-complete. However, since we know that input resolution is *not* refutation-complete, this again shows that $\mathbf{S}'$ cannot be true.

4 The incompleteness of input resolution

In this section, we show that the subsumption theorem for input resolution ($\mathbf{S}'$) is not true, not even in the special case where the set of premises Σ consists of only one clause. [13, 4] state $\mathbf{S}'$ without proof, and apply this special case of $\mathbf{S}'$. Hence the counterexample we give here is relevant for those articles, 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 3 shows that clause D (see below) can be derived from C by unconstrained resolution. This also shows that $C \models D$. Figure 3 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 substitution $\{x_5/x_1, x_6/x_2\}$. 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'.

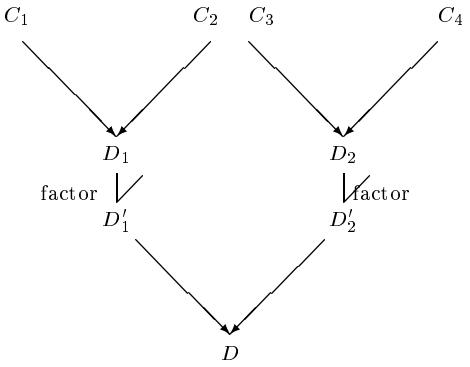

Figure 3: The derivation of D from C by unconstrained resolution

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

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. □

5 Conclusion

In this paper, we discussed the importance of the subsumption theorem in ILP. No really rigorous proof based on the elementary definitions of resolution 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 is then an easy corollary of this theorem. 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.

6 Future work

In this section, we will briefly describe some other results of our research concerning the subsumption theorem. These will be published elsewhere (see [16, 17]).

Firstly, we have been able to prove the subsumption theorem for unconstrained resolution starting from the refutation-completeness [17]. This proof seems to share the same general idea with the proof of the subsumption theorem given in [1], so our proof can be seen as an improved version of their proof. Since, conversely, the refutation-completeness is also an immediate consequence of the subsumption theorem (as shown

in Theorem 5), this establishes the equivalence of the subsumption theorem and the refutation-completeness for unconstrained resolution.

Secondly, we have been able to prove the subsumption theorem for *linear* resolution. So linear resolution, which is more efficient than unconstrained resolution, is complete for general clauses. In addition, it can be shown also for this case that the subsumption theorem and the refutation-completeness can be proved from one another. Hence we have the equivalence of these two completeness-results also for the case of linear resolution.

Thirdly, we have proved the subsumption theorem for SLD-resolution, restricted to Horn clauses [16]. For this we generalized the definition of SLD-resolution given in [12], following an idea of [14]. Since SLD-resolution is a special case of input resolution, this implies that version $\mathbf{S}'$ of the subsumption theorem does hold in the restricted case of Horn clauses. We have also been able to find a new proof of the refutation-completeness of SLD-resolution, which has the advantage of avoiding partial orders and fixpoint-theory. This makes our proof of this important completeness result—on which, for instance, PROLOG is based—easier to understand for people in ILP than the proof of [12]. Finally, also in the case of SLD-resolution the equivalence of the subsumption theorem and the refutation-completeness can be established.

All these results will be collected in the workreport [18].

References

[1] 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.

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

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

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

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Nienhuys-Cheng, S.-H., and de Wolf, R., 'The Equivalence of the Subsumption Theorem and the Refutation-Completeness for Unconstrained Resolution', to appear in: *Proc. of the Asian Computer Science Conference (ACSC-95)*, Lecture Notes in Computer Science, Springer-Verlag, December 1995.

[18] Nienhuys-Cheng, S.-H., and de Wolf, R., 'The Subsumption Theorem for Several Forms of Resolution', forthcoming workreport, Erasmus University Rotterdam, Computer Science Department.

[19] Robinson, J. A., 'A Machine Oriented Logic Based on the Resolution Principle', in: *Journal of the ACM*, 12, 1965, pp. 23–41.

[20] 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.