Tidying up the Mess around the Subsumption Theorem in Inductive Logic Programming

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 [Lee67]. In Inductive Logic Programming, interest in this theorem is increasing since its 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 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 [Mug92, I-A93] 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. 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 first stated and proved by Lee in 1967 in his PhD-thesis [Lee67].

However, we have not been able to find a copy of his thesis.[1] 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.

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. 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. It is used in [BM92, I-A93, LN94b, Mug92, NLT93], for instance.

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 $\Box$). 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.

A reproof of the subsumption theorem is given in the appendix of [BM92]. Unfortunately this reproof is not fully correct. 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 4 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 correct, it is often quoted—sometimes even incorrectly. The two main formulations we have found are the following:

S Let T be a set of clauses and C a clause which is not a tautology. Define $\mathcal{R}^0(T) = T$ and $\mathcal{R}^n(T) = \mathcal{R}^{n-1}(T) \cup \{C : C \text{ is a resolvent of } C_1, C_2 \in \mathcal{R}^{n-1}(T)\}$. Also define $\mathcal{R}^*(T) = \mathcal{R}^0(T) \cup \mathcal{R}^1(T) \cup \ldots$. 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):
$T \vdash C$ iff there exists a clause $D \in \mathcal{R}^*(T)$ such that D subsumes C.

[1]One of the authors of this article actually sent a letter to dr. Lee, asking for details about the theorem and his proof. However, we have not received a reply.

S$'$ Let T be a set of clauses and C a clause which is not a tautology. Define $\mathcal{L}^1(T) = T$ and $\mathcal{L}^n(T) = \{C : C$ is a resolvent of $C_1 \in \mathcal{L}^{n-1}(T)$ and $C_2 \in T\}$. Also define $\mathcal{L}^*(T) = \mathcal{L}^1(T) \cup \mathcal{L}^2(T) \cup \ldots$. Then the subsumption theorem is stated as follows: $T \models C$ iff there exists a clause $D \in \mathcal{L}^*(T)$ 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 [I-A93, 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 $\mathcal{L}^*(T)$ is a member of T, so **S**$'$ is stronger than **S**. In fact, whereas **S** is true, **S**$'$ *is actually false!* An easy propositional counterexample is given on p. 99 of [GN87]. Another 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 $R(a) \vee S(a)$.

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 [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 no problem. Due to a lack of space we will not prove that here, but in another article. And if we examine [Mug92, I-A93] carefully, then we see that the results of these articles only depend on a special case of **S**$'$, namely the case where T 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 [Mug92, I-A93] which are consequences of this special case of **S**$'$ need to be reconsidered.

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. The main results that we have found are the following:

a. The subsumption theorem for unconstrained resolution.
b. The refutation-completeness of unconstrained resolution.
c. **S**$'$ is false, even when T (the set of premisses) contains only one clause.

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. Finally, in Section 4 we present our counterexample to the special case of **S**$'$.

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, and not as a *set* of literals. Hence we consider $P(a)$ and $P(a) \vee P(a)$ as different clauses. 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*.

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 two clauses 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. 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 variants 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 $\mathcal{R}^n(T)$ 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 $\mathcal{R}^3(\Sigma)$.

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.

$$\begin{tikzcd}
 & & & & & C_4 = P(x) \vee \neg Q(x) \arrow[dr] & & C'_1 = P(y) \vee Q(y) \vee R(y) \arrow[dl] \\
C_1 = P(x) \vee Q(x) \vee R(x) \arrow[dr] & & C_2 = \neg P(y) \vee Q(a) \arrow[dl] & & C_3 = \neg P(x) \vee \neg Q(x) \arrow[dr] & & C_5 = P(y) \vee P(y) \vee R(y) \arrow[dl] & \\
 & C_6 = Q(x) \vee R(x) \vee Q(a) \arrow[drr] & & & & C_7 = \neg Q(y) \vee R(y) \arrow[dll] & & \\
 & & & D = R(a) \vee R(a) & & & &
\end{tikzcd}$$

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

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). 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 $\Sigma \vdash_d C$.

$$\begin{tikzcd}
 & & \vdots & & & & \\
C_{m+1} = L_1 \vee \ldots \vee L_k \vee C' \arrow[dr] & & D_1 = \neg L_1 \vee D'_1 \arrow[dl] & \vdots & & & \\
 & L_2 \vee \ldots \vee L_k \vee C' \vee D'_1 \arrow[dr] & & D_2 = \neg L_2 \vee D'_2 \arrow[dl] & & & \\
 & & L_3 \vee \ldots \vee L_k \vee C' \vee D'_1 \vee D'_2 & & & & \\
 & & & \ddots & & \vdots & \\
 & & & & L_k \vee C' \vee D'_1 \vee \ldots \vee D'_{k-1} \arrow[dr] & & D_k = \neg L_k \vee D'_k \arrow[dl] \\
 & & & & & D = C' \vee D'_1 \vee \ldots \vee D'_k &
\end{tikzcd}$$

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

$\square$

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 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, [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 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 [CL73], 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 a simple induction on k, using the previous lemma. □

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 Σ be a set of clauses, and 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. Σ. If $\Sigma \vdash_r D$ and D subsumes $C\theta$, then D also subsumes C.*

Proof Since none of the constants $a_1, \ldots, a_n$ appears in Σ, none of these constants appears in D. 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. $\Box$

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 $C\theta$. Then by Lemma 4, D subsumes C. Hence $\Sigma \vdash_d C$. $\Box$

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)

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

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 $D = \Box$. $\Box$

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 T consists of only one clause. [Mug92, I-A93] 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 $T = \{C\}$, 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 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'.

$$\begin{tikzcd}
C_1 \arrow[dr] & & C_2 \arrow[dl] & C_3 \arrow[dr] & & C_4 \arrow[dl] \\
& D_1 \arrow[d, "\text{factor}"'] & & & D_2 \arrow[d, "\text{factor}"] & \\
& D'_1 \arrow[drr] & & & D'_2 \arrow[dl] & \\
& & & D & &
\end{tikzcd}$$

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

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

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 $\mathcal{L}^n(T)$ and $\mathcal{L}^*(T)$). This shows that input resolution is not complete, not even if T 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 $(P(x_1,x_2) \vee \neg P(x_4,x_1))\theta$.

$\square$

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. $\square$

5 Conclusion

In this paper, we discussed the importance of the subsumption theorem in ILP. No really rigorous proof of this theorem 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 was given by us. The refutation-completeness of resolution can then be considered as 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.

References

[BM92] M. Bain, and S. Muggleton. Non-monotonic Learning. In S. Muggleton (ed.), *Inductive Logic Programming*, pp. 145–153. APIC series 38, Academic Press, 1992.

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

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

[I-A93] P. Idestam-Almquist. *Generalization of Clauses*, PhD Thesis. Stockholm University, 1993.

[LN94a] P. van der Laag, and S.-H. Nienhuys-Cheng. Existence and Nonexistence of Complete Refinement Operators. In *Proc. of the European Conference on Machine Learning (ECML-94)*, Lecture Notes in Artificial Intelligence 784, pp. 307–322. Springer-Verlag, 1994.

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

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

[Mug92] S. Muggleton. Inverting Implication. In S. H. Muggleton, and K. Furukawa (eds.), *Proc. of the Second Int. Workshop on Inductive Logic Programming (ILP92).* ICOT Technical Memorandum TM-1182, 1992.

[NLT93] S.-H. Nienhuys-Cheng, P. van der Laag, and L. van der Torre. Constructing Refinement Operators by Deconstructing Logical Implication. In *Proc. of the Third Congress of the Italian Association for AI (AI*IA93)*, pp. 178–189. Lecture Notes in Artificial Intelligence 728, Springer-Verlag, 1993.