The Subsumption Theorem for Several Forms of Resolution

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 the following completeness result for resolution: if Σ is a set of clauses and C is a clause, then Σ logically implies C iff C is a tautology, or there exists a clause D which subsumes C, and which can be derived from Σ by some form of resolution. Different versions of this theorem exist, depending on the kind of resolution we use. It provides a more "direct" form of completeness than the better known refutation-completeness, which often makes the Subsumption Theorem better suited for theoretical research.

In this paper we investigate for which forms of resolution the theorem holds, and for which it does not. We collect results earlier obtained by others, and contribute some results of our own. The main results of the paper are as follows. For "unconstrained" resolution, the Subsumption Theorem holds, and is equivalent to the refutation-completeness: the one can be proved from the other. The same is true for linear resolution. For input resolution, the theorem is false, even in the special case where Σ contains only one clause. In case of SLD-resolution for Horn clauses, the Subsumption Theorem again holds, and is equivalent to the refutation-completeness of SLD-resolution.

1 Introduction

The *Subsumption Theorem* is the following statement:

> If Σ is a set of clauses and C is a clause, then Σ logically implies C ($\Sigma \models C$) iff C is a tautology, or there exists a clause D which subsumes C, and which can be derived from Σ by some form of resolution.

Different versions of the theorem exist, depending on the instantiation of "some form of resolution." We could allow arbitrary binary trees of resolution steps ("unconstrained resolution") as derivation, or we could allow only linear derivations, etc. This is similar to the *refutation-completeness* for proof by contradiction: here we have the refutation-completeness of unconstrained resolution, the refutation-completeness of linear resolution, etc.

The refutation-completeness is a form of completeness that is much better known than the Subsumption Theorem. It states that a set of clauses is unsatisfiable iff the set has a refutation (a derivation of the empty clause $\Box$, which represents a contradiction). It can be used to prove any case of logical implication between clauses. For if we have a set Σ, a clause C, θ is a Skolem substitution for C w.r.t. Σ, and $C\theta = L_1 \vee \ldots \vee L_n$, then $\Sigma \models C$ iff $\Sigma \cup \{\neg L_1, \ldots, \neg L_n\}$ is unsatisfiable iff $\Sigma \cup \{\neg L_1, \ldots, \neg L_n\}$ has a refutation.

However, the Subsumption Theorem is a more "direct" form of completeness than the refutation-completeness. By the Subsumption Theorem, we can straightforwardly prove C from Σ by taking a number of resolution steps starting from clauses in Σ, and then taking a subsumption step leading to C. There is no need to use the detour which first negates C, and then applies proof by refutation. In a derivation of a clause which subsumes C, the relation between on one hand the premisses in Σ, and on the other hand the conclusion C, is much easier to see than in a proof by refutation. Accordingly, the Subsumption Theorem gives us a more clear view of the structure of logical implication than the refutation-completeness. For this reason, the Subsumption Theorem is sometimes a more convenenient result than the refutation-completeness, perhaps not for efficient deduction, but certainly for theoretical analysis.

Examples of such theoretical analysis are the various ways the theorem is applied in Inductive Logic Programming (ILP). ILP, the intersection of Logic Programming and Machine Learning, is concerned with learning clausal theories from given sets of positive and negative examples. For a general introduction into ILP, we refer to [LD94, MDR94]. The use of subsumption is very popular in ILP, since it is decidable and machine-implementable. However, subsumption is weaker than implication: if C subsumes D then $C \models D$, 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 "bridge" for those who want to make this step, since it states that implication = resolution + subsumption. It is used for instance in [Mug92, IA93, IA95, NCLT93, LNC94a, LNC94b].

As our survey later on in this section will show, in automated theorem proving the Subsumption Theorem received most attention around 1970. In ILP, the Subsumption Theorem was first rediscovered by Bain and Muggleton [BM92].[1] A proof of the Subsumption Theorem for unconstrained resolution, based on the refutation-completeness, is given in the appendix of [BM92]. However, this proof seems 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]). Furthermore, it is not always clear how the concepts that are used in the proof are defined, and how the Skolemization works.

[1] From personal communication with Stephen Muggleton, we know Bain and Muggleton discovered the theorem themselves, independently of [Lee67]. Only afterwards did they found out from references in other literature that their theorem was roughly the same as the theorem in Lee's thesis.

Even though the proof in [BM92] is not quite correct, it is often quoted—sometimes even incorrectly. The two main formulations of the Subsumption Theorem that we have found in ILP-literature, 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 $\mathcal{R}^n(\Sigma) = \mathcal{R}^{n-1}(\Sigma) \cup \{C \mid C \text{ is a resolvent of } C_1, C_2 \in \mathcal{R}^{n-1}(\Sigma)\}$. Also define $\mathcal{R}^*(\Sigma) = \mathcal{R}^0(\Sigma) \cup \mathcal{R}^1(\Sigma) \cup \ldots$. Then the Subsumption Theorem is stated as follows:
$\Sigma \models C$ 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 $\mathcal{L}^n(\Sigma) = \{C \mid C \text{ is a resolvent of } C_1 \in \mathcal{L}^{n-1}(\Sigma) \text{ and } C_2 \in \Sigma\}$. 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]. [Mug92] does not include a proof of **S′**, but refers instead to [BM92]. In other work such as [IA93, NCLT93, LNC94b, MDR94, BG96], the theorem is also given in the form of **S′**. These texts 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 which it is not, as we will see in Section 6.

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, as is well-known, but input resolution is not.

However, the articles we mentioned do not always use **S′** itself. [LNC94b, NCLT93] are restricted to Horn clauses. In Section 7 we show that SLD-resolution for Horn clauses has its own Subsumption Theorem, so for Horn clauses there is no problem. If we examine [Mug92, IA93, MDR94, BG96], carefully, then we see that the results of these articles (which are also about non-Horn clauses) only depend on a special case of **S′**, namely the case where Σ consists of a single clause. Unfortunately, **S′** does not even hold in this special case. We give a counterexample in Section 6. This means that results which are consequences of this special case of **S′** need to be reconsidered.[2] These particularly include results on *nth powers* and *nth roots*. If $D \in \mathcal{L}^n(\{C\})$, then D is called an nth power of C, and C is called an nth root of D. Clearly, the falsity of **S′** renders false the completeness of powers and roots reported in [Mug92, MDR94, BG96].

This confusion in the ILP-community about various forms of the Subsumption Theorem provided the motivation for the research we present in this paper. Our aim was to find out for which versions of resolution the Subsumption Theorem holds, and for which it does not.

[2] Idestam-Almquist has adjusted his results from [IA93] in [IA95], incorporating our findings as published in [NCW96].

Let us first see what results have already been proved in the literature. Surprisingly, the Subsumption Theorem is mentioned nowhere in the standard reference books on resolution, such as [CL73, Lov78, Llo87]. Hence we have to rely on journals, conference proceedings, and theses. A weak form of the Subsumption Theorem was first proved by Lee in 1967 in his PhD thesis [Lee67], only 2 years after Robinson's introduction of the resolution principle in [Rob65]. His result is the following: $\Sigma \models C$ iff C is a tautology, or there exists a clause D which *implies* C (and thus not necessarily *subsumes* C), and which can be derived from Σ by unconstrained resolution.

The "real" Subsumption Theorem—i.e., where D subsumes C rather than only implying it—appears to have been first proved in [SCL69]. Here the result is proved for several forms of *semantic* resolution. Since semantic resolution is a constrained form of resolution, their results immediately imply the Subsumption Theorem for unconstrained resolution.[3] Kowalski [Kow70] explicitly proved the result for unconstrained resolution, but his proof is rather sketchy and presupposes knowledge of semantic trees. Minicozzi and Reiter [MR72] proved the Subsumption Theorem for linear resolution. After that, interest in the Subsumption Theorem seems to have faded somewhat. However, recently Inoue [Ino92] has developed SOL-resolution (Skip Ordered Linear resolution), and proved a version of the Subsumption Theorem for it. He also gave an overview of the results of [Lee67, SCL69, MR72].

In this paper, we consider four kinds of resolution: "unconstrained" resolution, linear resolution, input resolution, and SLD-resolution, the latter only for Horn clauses. We collect some of the results mentioned in the last paragraph, and contribute some results of our own.

The paper is organized as follows. In the next section we start out with our main definitions, and state some results that will be needed in the proofs later on. In Section 3, we give a new, direct proof of the Subsumption Theorem for unconstrained resolution. In our opinion, this proof is easier to understand than earlier proofs of the same result [Kow70, BM92], which presuppose the refutation-completeness of resolution. In Section 4, we then show that the refutation-completeness of unconstrained resolution is an immediate corollary of the Subsumption Theorem. Conversely, in the same section we also give a second proof of the Subsumption Theorem, this time starting from the refutation-completeness. Thus these two completeness results are actually equivalent: the one can be proved from the other [NCW95a].

The Subsumption Theorem also holds for linear resolution. In Section 5 we give a proof of this result which is similar to the proof given in [MR72]. Moreover, we also show that the Subsumption Theorem for linear resolution is equivalent to the refutation-completeness of linear resolution. In Section 6, we show that the Subsumption Theorem does not hold for input resolution, not even in case Σ contains only one clause. This is a new result, which we first proved in [NCW96].

[3] The name "completeness theorem for consequence finding" is also sometimes used. As far as we know, the name "Subsumption Theorem" was introduced in [Kow70].

Finally, in Section 7 we discuss SLD-resolution for Horn clauses. We first give a proof of the well-known refutation-completeness of SLD-resolution. This proof is easier to understand than the one given in [Llo87], since our proof does not require fixed-point theory. We then proceed to prove the Subsumption Theorem for SLD-resolution. This new result [NCW95b] generalizes Theorem 18 of [MP94], which gives the result for the case where Σ contains only one clause. Moreover, as in the cases of unconstrained and linear resolution, we show that the Subsumption Theorem is equivalent to the refutation-completeness also in case of SLD-resolution.[4]

2 Preliminaries

In this section we define our main concepts, and state some results that will be needed in later proofs. For the definitions of 'atom', 'model', 'substitution', '(most general) unifier' etc., we refer to [CL73, Llo87].

Definition 1 A *literal* is an atom or the negation of an atom. A *positive literal* is an atom, a *negative literal* is the negation of an atom. $\diamond$

Here we adopt the convention that the negation of a negative literal $\neg A$ is A. So for instance, if $L = \neg Q(x, y)$, then we use $\neg L$ to denote $Q(x, y)$.

Definition 2 A *clause* is a universally quantified disjunction of zero or more literals. $\diamond$

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, in particular if one treats a clause as a *set* of literals instead of a disjunction. Set-notation would require a few small changes in the proofs below.

For convenience, we use $L \in C$ to denote that L is a literal in C, and $C \subseteq D$ to denote that the set of literals appearing in the disjunction C is a subset of the set of literals in D. We will not write out the universal quantifiers in front of a clause. The empty clause (a disjunction of zero literals) is denoted by $\Box$. This represents a contradiction.

One important subset of the set of all clauses, is the set of *Horn* clauses:

Definition 3 A *definite program clause* is a clause containing one positive, and zero or more negative literals. A *(definite) goal* is a clause containing only negative literals. A *Horn clause* is either a definite program clause or a definite goal. $\diamond$

If a definite program clause consists of the positive literal A and the negative literals $\neg B_1, \ldots \neg B_n$, then such a clause can be written as the following implication:

$$A \leftarrow B_1, \ldots, B_n.$$

[4]Though this equivalence holds for each of the forms of resolution that we discuss here, it does not holds for every conceivable kind of resolution. [MR72] discusses m.c.l.-resolution. This is refutation-complete, but the Subsumption Theorem does not hold for it.

A is called the *head* of the clause, $B_1, \ldots, B_n$ is called the *body* of the clause. In case of an atom A (that is, if $n = 0$), we can omit the '$\leftarrow$'-symbol. A definite goal can be written as

$$\leftarrow B_1, \ldots, B_n.$$

The empty clause $\Box$ is also considered to be a goal.

Below we give our definitions for resolution:

Definition 4 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.* $\diamond$

Definition 5 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. If the substitution θ is an mgu (most general unifier) 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 6 Let C be a clause, $L_1, \ldots, L_n$ $(n \geq 1)$ some unifiable literals from C, and θ an mgu of the set $\{L_1, \ldots, L_n\}$. Then the clause obtained by deleting $L_2\theta, \ldots, L_n\theta$ from $C\theta$ is called a *factor* of C. $\diamond$

Note that every non-empty clause C is a factor of C itself, using the identity substitution ε as mgu of one literal in C. Factors are sometimes built into the resolution step itself—for instance in Robinson's original paper [Rob65], 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 7 Let C_1 and C_2 be two clauses. A *resolvent* R of 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 R. $\diamond$

It is easy to show that resolution is sound: if R is a resolvent of C_1 and C_2, then $\{C_1, C_2\} \models R$.

Below we define a *derivation.* In later sections, we will put some constraints on this concept, yielding, respectively, linear, input, and SLD-derivations. The kind of derivation defined here, will sometimes be referred to as "unconstrained" resolution.

Definition 8 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. $\diamond$

A derivation of a clause C from a set Σ can be represented as a binary tree of resolution steps, with clauses from Σ as leaves and C as root.

If we add a subsumption step to a derivation, we get a *deduction*.

Definition 9 Let C and D be clauses. We say C *subsumes* D if there exists a substitution θ such that $C\theta \subseteq D$. $\diamond$

If C subsumes D, then $C \models D$. Subsumption is also sometimes called θ-subsumption.

Example 1 $C = P(x) \vee Q(x,y)$ subsumes $D = P(a) \vee Q(a,y) \vee R(x)$. $\triangleleft$

Definition 10 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. If $\Sigma \vdash_d C$, we say C can be *deduced* from Σ. $\diamond$

Example 2 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 $\mathcal{R}^3(\Sigma)$.

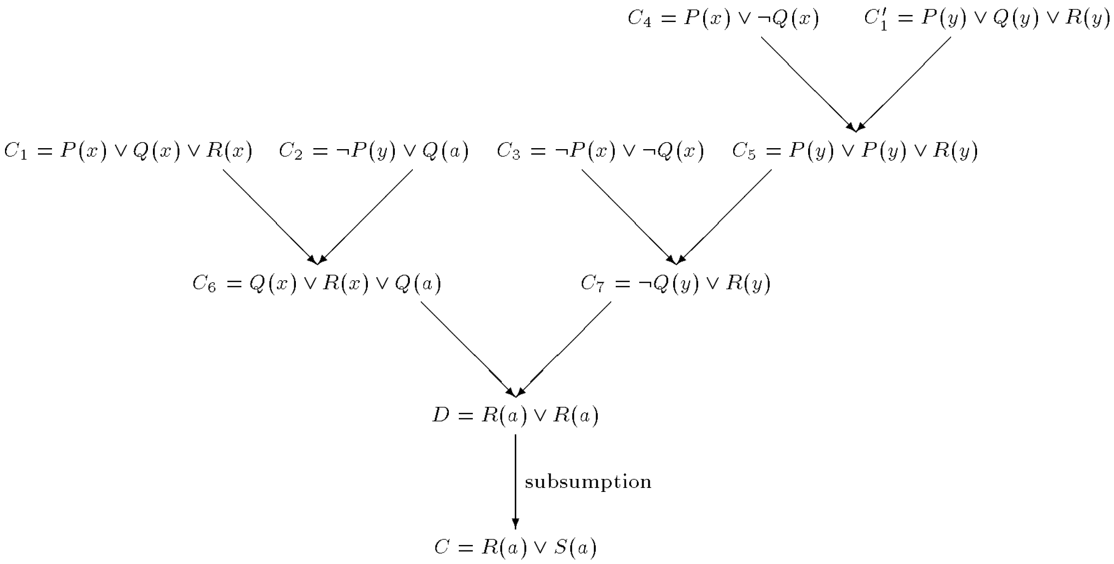

Figure 1: A deduction of C from Σ

$\triangleleft$

We will sometimes need a *Skolem substitution*, defined as follows:

Definition 11 Let Σ be a set of clauses, and C be 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 the substitution $\{x_1/a_1, \ldots, x_n/a_n\}$ is called a *Skolem substitution* for C w.r.t. Σ. $\diamond$

Example 3 Let $\Sigma = \{(P(x) \vee \neg Q(y, f(a))), \neg P(b)\}$ and $C = Q(z, y) \vee P(b)$. Then $\{z/c, y/d\}$ is a Skolem substitution for C w.r.t. Σ. $\triangleleft$

We end this section by listing a number of results that will be needed later on. Some of these apply to arbitrary first-order formulas, so also to clauses. The first two are easy to prove.

Theorem 1 (Deduction Theorem) *Let Σ be a set of formulas, and ϕ and ψ be formulas. Then $\Sigma \cup \{\phi\} \models \psi$ iff $\Sigma \models (\phi \rightarrow \psi)$.*

Proposition 1 *Let Σ be a set of formulas and ϕ a formula. Then $\Sigma \models \phi$ iff $\Sigma \cup \{\neg\phi\}$ is unsatisfiable.*

A proof of the following *Compactness Theorem* can be found in [BJ89].

Theorem 2 (Compactness) *If Σ is an infinite, unsatisfiable set of formulas, then there exists a finite, unsatisfiable subset of Σ.*

The Compactness Theorem has the following consequence:

Theorem 3 *Let Σ be an infinite set of formulas, and ϕ be a formula. If $\Sigma \models \phi$, then there is a finite subset Σ' of Σ, such that $\Sigma' \models \phi$.*

Proof If $\Sigma \models \phi$, then by Proposition 1, $\Sigma \cup \{\neg\phi\}$ is unsatisfiable. By the Compactness Theorem, there is a finite unsatisfiable set $\Gamma \subseteq \Sigma \cup \{\neg\phi\}$. Put $\Sigma' = \Gamma \backslash \{\neg\phi\}$. Then $\Sigma' \subseteq \Sigma$, and since $\Sigma' \cup \{\neg\phi\}$ is unsatisfiable, we have $\Sigma' \models \phi$ by Proposition 1. $\square$

The next result is Proposition 3.3 of [Llo87] and Theorem 4.2 of [CL73].

Proposition 2 *A set of clauses Σ is unsatisfiable iff Σ has no Herbrand models.*

3 The Subsumption Theorem

In this section, we prove the Subsumption Theorem for (unconstrained) resolution: $\Sigma \models C$ iff $\Sigma \vdash_d C$. Thus any clause which is a logical consequence of Σ, can be deduced from Σ. We prove this in a number of successive steps in the following subsections. First we prove the result 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 be ground.

3.1 The Subsumption Theorem for ground Σ and C

Lemma 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 By Theorem 3, we can assume Σ is finite. Assume C is not a tautology. Then we need to find a clause D such that $\Sigma \vdash_r D$ and $D \subseteq C$ (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. (See figure 2 for illustration of this case). 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 $\Sigma' \cup \{C_{m+1}\} \models C$, it follows from Theorem 1 that $\Sigma' \models (C_{m+1} \rightarrow C)$, hence $\Sigma' \models C \vee \neg C_{m+1}$.
   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. For each i, $\neg L_i \in D_i$, 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 for a schematic representation of this derivation. 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 $\Sigma \vdash_d C$.

$\square$

3.2 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 $\Sigma_g \models C$,

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

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

where Σ_g is a set of ground instances of clauses of Σ. Then by Lemma 1 there is a clause D such that $\Sigma_g \vdash_r D$, and D subsumes C. Finally, we "lift" this derivation to a derivation from Σ. The next two results show that logical implication between clauses can be translated to logical implication between ground clauses. The first of these is Herbrand's Theorem.

Theorem 4 (Herbrand) *A set of clauses Σ is unsatisfiable iff there exists a finite unsatisfiable set Σ_g of ground instances of clauses in Σ.*

Proof

$\Leftarrow$: Σ_g is a finite set of ground instances of clauses in Σ, so $\Sigma \models \Sigma_g$. Hence if Σ_g is unsatisfiable, then Σ is unsatisfiable.

$\Rightarrow$: Let Σ' be the (possibly infinite) set of all ground instances of clauses in Σ. Let I be an Herbrand interpretation. It is not very difficult to see that I is an Herbrand model of a clause C iff I is an Herbrand model of the set of all ground instances of C. Therefore I is a model of Σ iff I is a model of Σ'. Now we have the following:

Σ is unsatisfiable iff (by Proposition 2)
Σ has no Herbrand models iff
Σ' has no Herbrand models iff (by Proposition 2)
Σ' is unsatisfiable.

Finally, by the Compactness Theorem (Theorem 2) there is a *finite* unsatisfiable subset Σ_g of Σ'. $\square$

Theorem 5 *Let Σ be a set of clauses, and C be a ground clause. If $\Sigma \models C$, then there exists a finite set Σ_g of ground instances of clauses 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 4, 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 Proposition 1)
$\Sigma \cup \{\neg C\}$ is unsatisfiable iff
$\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ is unsatisfiable iff (by Theorem 4)
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 Proposition 1)
$\Sigma_g \models (L_{i_1} \vee \ldots \vee L_{i_j})$.

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

Example 4 Let $\Sigma = \{(P(f(x)) \vee \neg P(x)), P(x)\}$ and $C = P(f(f(a)))$. Then $\Sigma \models C$. $\Sigma_g = \{(P(f(f(a))) \vee \neg P(f(a))), (P(f(a)) \vee \neg P(a)), P(a)\}$ is a set of ground instances of clauses of Σ, and $\Sigma_g \models C$. $\triangleleft$

The following two lemmas are sufficient to "lift" a derivation, that is, to turn a derivation from instances of certain clauses into a derivation from those clauses themselves.

Lemma 2 *Let C_1 and C_2 be two clauses, and C_1' and C_2' instances of C_1 and C_2, respectively. If R' is a resolvent of C_1' and C_2', then there exists a resolvent R of C_1 and C_2, such that R' is an instance of R.*

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 $C_2' = C_2\sigma_2$. Suppose R' is a resolvent of C_1' and C_2'. Then R' is a binary resolvent of a factor of C_1' and a factor of C_2'. See the figure for illustration.

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

Let γ_1 be an mgu of $L_a \vee \ldots \vee L_m$. Then $(L_1 \vee \ldots \vee L_a)\gamma_1$ is a factor of C_1. Note that $\sigma_1\theta_1$ is a unifier of $L_a, \ldots, L_m$. Since γ_1 is an mgu of $L_a, \ldots, L_m$, there exists a substitution δ_1 such that $\sigma_1\theta_1 = \gamma_1\delta_1$. Similarly, $(M_1 \vee \ldots \vee M_b)\gamma_2$

is a factor of C_2, with γ_2 as mgu of $M_b \vee \ldots \vee M_n$, 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_i is more general than $\sigma_i\theta_i$ $(i = 1, 2)$, $L_i\gamma_1$ 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 $R = (D_1\gamma_1 \vee D_2\gamma_2)\theta$, which can be written as $R = (D_1 \vee D_2)\gamma_1\gamma_2\theta$. Since R 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.

$$
\begin{tikzcd}
C_1 \arrow[dr, "\gamma_1"] \arrow[dd, "\sigma_1"'] & & & & C_2 \arrow[dl, "\gamma_2"'] \arrow[dd, "\sigma_2"] \\
& \text{factor} \arrow[dr, "\theta"] \arrow[dd, "\delta_1"'] & & \text{factor} \arrow[dl, "\theta"'] \arrow[dd, "\delta_2"] & \\
C_1' \arrow[dr, "\theta_1"] & & R \arrow[dd, "\delta"'] & & C_2' \arrow[dl, "\theta_2"'] \\
& \text{factor} \arrow[dr, "\mu"] & & \text{factor} \arrow[dl, "\mu"'] & \\
& & R' & &
\end{tikzcd}
$$

It remains to show that R' is an instance of R. 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 $R' = (D_1 \vee D_2)\sigma_1\theta_1\sigma_2\theta_2\mu = (D_1 \vee D_2)\gamma_1\delta_1\gamma_2\delta_2\mu = (D_1 \vee D_2)\gamma_1\gamma_2\delta_1\delta_2\mu = (D_1 \vee D_2)\gamma_1\gamma_2\theta\delta = R\delta$. Hence R' is an instance of R. □

Lemma 3 (Derivation lifting) *Let Σ be a set of clauses, and Σ' 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. Suppose $k = 1$. $R_1' \in \Sigma'$, so there exists a clause $R_1 \in \Sigma$ such that R_1' is an instance of R_1.
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 two clauses C_1' and C_2' in $\{R_1', \ldots, R_m'\}$. Then there exist two clauses C_1 and C_2 in $\{R_1, \ldots, R_m\}$ such that C_1' is an instance of C_1, and C_2' is an instance of C_2. It follows from Lemma 2 that there is a resolvent R_{m+1} of C_1 and C_2, such that R_{m+1}' is an instance of R_{m+1}. So the lemma holds for $k = m + 1$.

□

The previous lemmas are sufficient to prove the Subsumption Theorem for the case where C is ground.

Lemma 4 *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 Theorem 5, 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 from Lemma 1 there exists a clause D' such that $\Sigma_g \vdash_r D'$, and D' subsumes 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 D' is an instance of R_k. Let $D = R_k$. Then $\Sigma \vdash_r D$ and D subsumes C (since D' subsumes C). $\square$

3.3 The Subsumption Theorem (general case)

Finally we prove the Subsumption Theorem for arbitrary Σ and C. The following lemma shows that if we have derived some clause D from Σ which subsumes $C\theta$—where θ is a Skolem substitution for C w.r.t. Σ—then D also subsumes C. For instance, suppose $D = P(x)$, $C = P(y) \vee Q(z)$ and $\theta = \{y/a, z/b\}$. D subsumes $C\theta$, but since θ replaces each variable by a constant that does not appear in Σ, C or D, D also subsumes C itself.

Lemma 5 *Let C and D be clauses. Let $\theta = \{x_1/a_1, \ldots, x_n/a_n\}$ be a Skolem 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. $\square$

Finally we can prove the general case of the Subsumption Theorem:

Theorem 6 (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 subsumption.

$\Rightarrow$: Assume C is not a tautology. Let θ be a Skolem 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 Lemma 4 there is a clause D such that $\Sigma \vdash_r D$ and D subsumes $C\theta$. Since D is derived from Σ, D does not contain any of the constants in θ. Therefore θ is also a Skolem substitution for C w.r.t. D. Then by Lemma 5, D subsumes C. Hence $\Sigma \vdash_d C$. $\square$

4 The refutation-completeness

4.1 From Subsumption Theorem to refutation-completeness

The Subsumption Theorem actually tells us that resolution and subsumption form a complete set of derivation-rules for clauses. Though the resolution rule by itself is not complete for clauses in general, for instance, $P(x) \models P(a)$ but $P(x) \not\vdash_r P(a)$, resolution is complete w.r.t. unsatisfiable sets of clauses. This refutation-completeness is an easy consequence of the Subsumption Theorem:

Theorem 7 (Refutation-completeness) *Let Σ be a set of clauses. Then Σ is unsatisfiable iff $\Sigma \vdash_r \Box$.*

Proof

$\Leftarrow$: By the soundness of resolution.

$\Rightarrow$: Suppose Σ is unsatisfiable. Then $\Sigma \models \Box$. So by Theorem 6 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 $D = \Box$. $\Box$

4.2 From refutation-completeness to Subsumption Theorem

In the previous subsection, we showed that the refutation-completeness is a direct consequence of the Subsumption Theorem. Here 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: the one can be proved from the other.

To prove the Subsumption Theorem from the refutation-completeness, we will first show how to turn a refutation of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ into a deduction of $L_1 \vee \ldots \vee L_k$ from Σ. Thus our proof is constructive, and somewhat similar to the approach in [BM92]. We start with an example. Suppose $\Sigma = \{(P(x) \vee \neg R(f(f(b)))), (R(f(x)) \vee \neg R(x))\}$, and $C = P(x) \vee Q(x) \vee \neg R(b)$. First we note that $\theta = \{x/a\}$ is a Skolem substitution for C w.r.t. Σ. Now $\neg C\theta \Leftrightarrow \{\neg P(a), \neg Q(a), R(b)\}$. Figure 3 shows a refutation of $\Sigma \cup \{\neg P(a), \neg Q(a), R(b)\}$.

Now by omitting the leaves of the refutation-tree which come from $\neg C\theta$ (the framed literals) and by making appropriate changes in the tree, we get a derivation of the clause $D = P(x) \vee \neg R(b)$ (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 6 *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.

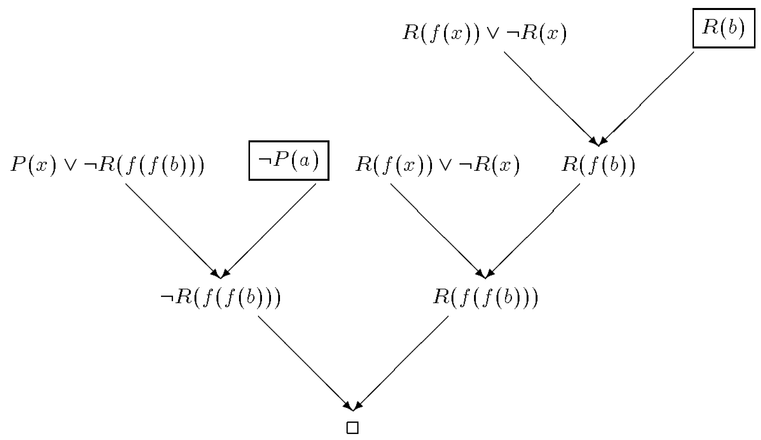

Figure 3: A refutation of $\Sigma \cup \{\neg P(a), \neg Q(a), R(b)\}$

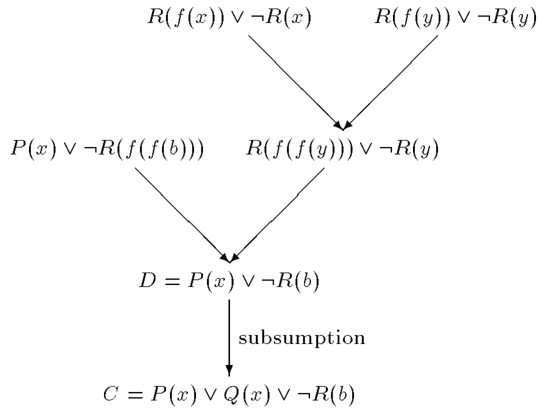

Figure 4: A deduction of C from Σ, obtained from the previous figure

1. Suppose $r = 0$. Then $R_n = \square \in \Sigma$, so 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, since 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 3) 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 6) once more, this time starting from Theorem 7.

Theorem 6 (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 subsumption.

$\Rightarrow$: If C is a tautology, the theorem is obvious. Assume C is not a tautology. Let θ be a Skolem substitution for C w.r.t. Σ. Suppose $C\theta = L_1 \vee \ldots \vee L_k$. Since C is not a tautology, $C\theta$ is not a tautology. $C\theta$ is ground and $\Sigma \models C\theta$, so by Proposition 1 the set of clauses $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ is unsatisfiable. Then it follows from Theorem 7 that $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_r \Box$. Therefore by Lemma 6, there exists a clause D such that $\Sigma \vdash_r D$, and D subsumes $C\theta$. Finally, from Lemma 5, 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 8 *For unconstrained resolution, the Subsumption Theorem and the refutation-completeness are equivalent.*

5 Linear resolution

Linear resolution is characterized by the linear shape of its derivations. It is more efficient than unconstrained resolution, because the number of possible derivations is significantly decreased by the linear constraint on the shape of a derivation. It was independently introduced by Loveland [Lov70] and Luckham [Luc70]. An important further restriction called *SL-resolution* (Linear resolution with a Selection function) was introduced and shown to be refutation-complete by Kowalski and Kuehner [KK71]. Minicozzi and Reiter proved the Subsumption Theorem for linear resolution in [MR72]. More recently, Inoue [Ino92] developed SOL-resolution (Skip Ordered Linear resolution) and proved a version of the Subsumption Theorem for it.

For the sake of transparency, we will define a very simple form of linear resolution here. Many features and restrictions could be added on to improve efficiency (see the references given above). We will prove the Subsumption Theorem and the refutation-completeness for this form of linear resolution.

After that, we will define a further restriction of linear resolution called *input* resolution, and show that this is *not* complete for general clauses, not even when the set of premisses contains only one clause.

5.1 Definitions

Definition 12 Let Σ be a set of clauses and C be a clause. A *linear derivation* of C from Σ is a finite sequence of clauses $R_0, \ldots, R_k = C$, such that $R_0 \in \Sigma$ and each R_i with $1 \leq i \leq k$ is a resolvent of R_{i-1} and a clause $C_i \in \Sigma \cup \{R_0, \ldots, R_{i-2}\}$.

R_0 is called the *top clause*, $R_0, \ldots, R_k$ the *center clauses*, and $C_1, \ldots, C_k$ are called the *side clauses* of this linear derivation. If a linear derivation of C from Σ exists, we write $\Sigma \vdash_{lr} C$.

A linear derivation of $\square$ from Σ is called a *linear refutation* of Σ. $\diamond$

Linear derivations are characterized by the "linear" shape of their corresponding derivation-trees. See figure 5. Such a tree can be turned into a derivation-tree for unconstrained resolution by adding the derivations of each side clause C_i which is not in Σ.

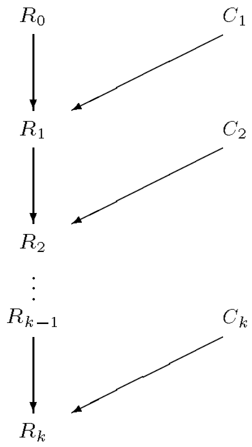

Figure 5: The characteristic shape of a linear derivation

Linear deductions are defined as follows:

Definition 13 Let Σ be a set of clauses and C a clause. There exists a *linear deduction* of C from Σ, written as $\Sigma \vdash_{ld} C$, if C is a tautology, or if there exists a clause D such that $\Sigma \vdash_{lr} D$ and D subsumes C. $\diamond$

5.2 The refutation-completeness

A proof of the refutation-completeness of a form of linear resolution called *OL-resolution* (Ordered Linear resolution), is given in Theorem 7.2 of [CL73]. However, this proof contains an error. In fact, OL-resolution is not refutation-complete, as described on pp. 324–325 of [Ino92]. Nevertheless, we can adapt the proof of [CL73] to yield a correct proof for our own definition of linear resolution. First we prove the case for ground clauses, which is then lifted. The proof of the following lifting lemma is similar to Lemma 3.

Lemma 7 (Linear derivation lifting) *Let Σ be a set of clauses, and Σ' be a set of instances of clauses in Σ. Suppose $R'_0, \ldots, R'_k$ is a linear derivation of the clause R'_k from Σ'. Then there exists a linear derivation $R_0, \ldots, R_k$ of the clause R_k from Σ, such that R'_i is an instance of R_i, for each i.*

The next lemma is the refutation-completeness of linear resolution for ground clauses.

Lemma 8 *If Σ is an unsatisfiable set of ground clauses, and $C \in \Sigma$ such that $\Sigma \backslash \{C\}$ is satisfiable, then there is a linear refutation of Σ with C as top clause.*

Proof By the Compactness Theorem (Theorem 2), we can assume Σ is finite. Let n be the number of distinct ground atoms appearing in literals in clauses in Σ. We prove the lemma by induction on n.

1. If $n = 0$, then $\Sigma = \{\Box\}$. Since $\Sigma \backslash \{C\}$ is satisfiable, $C = \Box$
2. Suppose the lemma holds for $n \leq m$, and suppose $m + 1$ distinct atoms appear in Σ. We distinguish two cases.
   Case 1: Suppose $C = L$, where L is a literal. We first delete all clauses from Σ which contain the literal L (so we also delete C itself from Σ). Then we replace clauses which contain the literal $\neg L$ by clauses constructed by deleting these $\neg L$ (so for example, $L_1 \vee \neg L \vee L_2$ will be replaced by $L_1 \vee L_2$). Call the finite set obtained in this way Γ. Note that the literal L, nor its negation, appears in clauses in Γ. If M were a Herbrand model of Γ, then $M \cup \{L\}$ would be a Herbrand model of Σ. Thus since Σ is unsatisfiable, Γ must be unsatisfiable.
   Now let Σ' be an unsatisfiable subset of Γ, such that every proper subset of Σ' is satisfiable. Σ' must contain a clause D' obtained from a member of Σ which contained $\neg L$, for otherwise the unsatisfiable set Σ' would be a subset of $\Sigma \backslash \{C\}$, contradicting the assumption that $\Sigma \backslash \{C\}$ is satisfiable. By construction of Σ', we have that $\Sigma' \backslash \{D'\}$ is satisfiable. Furthermore, Σ' contains at most m distinct atoms, so by the induction hypothesis there exists a linear refutation of Σ' with top clause D'. See the left of figure 6 for illustration.
   The side clauses in this refutation that are not previous center clauses, are either members of Σ or obtained from members of Σ by the deletion of $\neg L$. In the latter kind of side clauses, put back the deleted $\neg L$ literals, and add these $\neg L$ to all later center clauses. Note that afterwards, these center clauses may contain multiple copies of $\neg L$. In particular, the last center clause changes from $\Box$ to $\neg L \vee \ldots \vee \neg L$. Since D' is a resolvent of C and $D = \neg L \vee D' \in \Sigma$, we can add C and D as parent clauses on top of the previous top clause D'. That way, we get a linear derivation of $\neg L \vee \ldots \vee \neg L$ from Σ, with top clause C. Finally, the literals in $\neg L \vee \ldots \vee \neg L$ can be resolved away using the top clause $C = L$ as side clause. This yields a linear refutation of Σ with top clause C (see the right of figure 6).

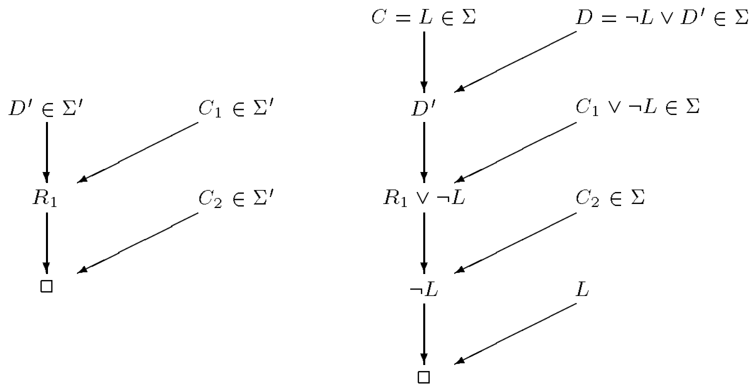

Figure 6: Case 1 of the proof

Case 2: Suppose $C = L \vee C'$, where C' is a non-empty clause. C' cannot contain $\neg L$, for otherwise C would be a tautology, contradicting the assumption that Σ is unsatisfiable while $\Sigma\backslash\{C\}$ is satisfiable.
Obtain Σ' from Σ by deleting clauses containing $\neg L$, and by removing the literal L from the remaining clauses. Note that $C' \in \Sigma'$. If M were an Herbrand model of Σ', then $M \cup \{\neg L\}$ would be an Herbrand model of Σ. Thus since Σ is unsatisfiable, Σ' is unsatisfiable.
Furthermore, because $\Sigma\backslash\{C\}$ is satisfiable, by Proposition 2 there is an Herbrand model M' of $\Sigma\backslash\{C\}$. Since Σ is unsatisfiable, M' is not a model of C. L is a literal in C, hence L must be false under M'. Every clause in $\Sigma'\backslash\{C'\}$ is obtained from a clause in $\Sigma\backslash\{C\}$ by deleting L from it. Since M' is a model of every clause in $\Sigma\backslash\{C\}$ and L is false under M', every clause in $\Sigma'\backslash\{C'\}$ is true under M'. Therefore M' is a model of $\Sigma'\backslash\{C'\}$, which shows that $\Sigma'\backslash\{C'\}$ is satisfiable.
Then by the induction hypothesis, there exists a linear refutation of Σ' with top clause C'. Now similar to case 1, put back previously deleted L literals to the top and side clauses, and to the appropriate center clauses. This gives a linear derivation of $L \vee \ldots \vee L$ from Σ with top clause C.
Note that $\{L\} \cup (\Sigma\backslash\{C\})$ is unsatisfiable, because L is false in any Herbrand model of $\Sigma\backslash\{C\}$, as shown above. On the other hand, $\Sigma\backslash\{C\}$ is satisfiable. Thus by case 1 of this proof, there exists a linear refutation of $\{L\} \cup (\Sigma\backslash\{C\})$ with top clause L. Since L is a factor of $L \vee \ldots \vee L$, we can put our linear derivation of $L \vee \ldots \vee L$ "on top" of this linear refutation of $\{L\} \cup (\Sigma\backslash\{C\})$ with top clause L, thus obtaining a linear refutation of Σ with top clause C.

$\square$

Theorem 9 (Refutation-completeness of linear resolution) *Let Σ be a set of clauses. Then Σ is unsatisfiable iff $\Sigma \vdash_{lr} \square$.*

Proof

$\Leftarrow$: By the soundness of resolution.

$\Rightarrow$: Suppose Σ is unsatisfiable. Then by Theorem 4, there is a finite unsatisfiable set Σ_g of ground instances of clauses in Σ'. Let Σ'_g be an unsatisfiable subset of Σ_g, and $C \in \Sigma'_g$ such that $\Sigma'_g \backslash \{C\}$ is satisfiable. From Lemma 8, we have $\Sigma'_g \vdash_{lr} \Box$. Hence $\Sigma \vdash_{lr} \Box$ by Lemma 7. $\Box$

5.3 The Subsumption Theorem

Starting from the refutation-completeness, it is now possible to prove also the Subsumption Theorem for linear resolution. Our proof is similar to the one given in [MR72]. We use the refutation-completeness, and then turn a linear refutation into a linear deduction, using the following lemma:

Lemma 9 *Let Σ be a set of clauses, and $C = L_1 \vee \ldots \vee L_k$ be a non-tautologous ground clause. If $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_{lr} \Box$, then $\Sigma \vdash_{ld} C$.*

Proof Suppose $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_{lr} \Box$. Then there exists a linear refutation $R_0, \ldots, R_n = \Box$ of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$. Notice that the top clause and the first side clause in this linear refutation cannot both be members of $\{\neg L_1, \ldots, \neg L_k\}$, because C is not a tautology. Thus we can assume $R_0 \in \Sigma$. It is then possible to prove by induction on n that this linear refutation can be transformed into a linear deduction of C from Σ with top clause R_0:

1. If $n = 0$, then $R_0 = \Box \in \Sigma$.
2. Suppose the lemma holds for $n \leq m$. Let $R_0, \ldots, R_{m+1} = \Box$ be a linear refutation of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$. Then $R_1, \ldots, R_{m+1}$ is a linear refutation of $\Sigma \cup \{R_1\} \cup \{\neg L_1, \ldots, \neg L_k\}$. By the induction hypothesis, there is a linear derivation of a clause D from $\Sigma \cup \{R_1\}$, with top clause R_1, such that D subsumes C.
   Suppose R_1 is itself a resolvent of two members of Σ. Then we also have $\Sigma \vdash_{lr} D$, so the lemma holds in this case.
   The only remaining case we have to check, is where R_1 is a resolvent of $R_0 \in \Sigma$ and some $\neg L_s$ $(1 \leq s \leq k)$. Let $R_0 = M_1 \vee \ldots \vee M_j \vee \ldots \vee M_h$. Suppose R_1 is a binary resolvent of $(M_1 \vee \ldots \vee M_j)\sigma$ (a factor of R_0, 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 $R_1 = (M_1 \vee \ldots \vee M_{j-1})\sigma\theta$ and $R_0\sigma\theta = R_1 \vee L_s \vee \ldots \vee L_s$ $(h - j + 1$ copies of $L_s)$, since $M_j, \ldots, M_h$ are all unified to L_s by $\sigma\theta$.
   Now replace each time R_1 appears as leaf (i.e., top or side clause) in the derivation-tree of D, by $R_0\sigma\theta = R_1 \vee L_s \vee \ldots \vee L_s$, and add $L_s \vee \ldots \vee L_s$ to all decendants of such an R_1-leaf. This gives a new derivation, in which each resolvent is the corresponding resolvent in the old derivation of D plus some extra copies of L_s. Thus we obtain a linear derivation of $D \vee L_s \vee \ldots \vee L_s$ from $\Sigma \cup \{R_0\sigma\theta\}$. Since $R_0\sigma\theta$ is an instance of a clause from Σ, we can lift (by Lemma 7) 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 $\Sigma \vdash_{ld} C$.

$\Box$

Theorem 10 (Subsumption Theorem for linear resolution) *Let Σ be a set of clauses, and C be a clause. Then $\Sigma \models C$ iff $\Sigma \vdash_{ld} C$.*

Proof

$\Leftarrow$: By the soundness of resolution and subsumption.

$\Rightarrow$: If C is a tautology, the theorem is obvious. Assume C is not a tautology. Let θ be a Skolem 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 by Proposition 1. Then it follows from Theorem 9 that $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_{lr} \Box$. Therefore by Lemma 9, there exists a clause D such that $\Sigma \vdash_{lr} D$, and D subsumes $C\theta$. From Lemma 5, D also subsumes C itself. Hence $\Sigma \vdash_{ld} C$. $\Box$

We have now proved the Subsumption Theorem of linear resolution starting from the refutation-completeness of linear resolution. Conversely, the latter also follows immediately from the former, in the same way as Theorem 7 followed from Theorem 6 in the previous section. Hence also for linear resolution we have the equivalence between these two completeness results.

Theorem 11 *For linear resolution, the Subsumption Theorem and the refutation-completeness are equivalent.*

6 Input resolution

Linear resolution is a restriction of unconstrained resolution. Linear resolution can itself be further restricted to *input* resolution, by stipulating that each side clause should be a member of Σ. Contrary to linear resolution, deduction using input resolution is not complete, not even when the set of premisses Σ contains only one clause. Before we give our counterexample, we will first formally define input resolution:

Definition 14 Let Σ be a set of clauses and C be a clause. An *input derivation* of C from Σ is a linear derivation in which each side clause C_i is a member of Σ. The side clauses $C_1, \ldots, C_k$ in an input derivation are also called *input clauses*. If an input derivation of C from Σ exists, we write $\Sigma \vdash_{ir} C$.

An input derivation of $\Box$ from Σ is called an *input refutation* of Σ. $\diamond$

Definition 15 Let Σ be a set of clauses and C a clause. There exists an *input deduction* of C from Σ, written as $\Sigma \vdash_{id} C$, if C is a tautology, or if there exists a clause D such that $\Sigma \vdash_{ir} D$ and D subsumes C. $\diamond$

It is well-known that input resolution is not refutation-complete. A simple propositional example suffices to show this. Let $\Sigma = \{(P \vee Q), (P \vee \neg Q), (\neg P \vee Q), (\neg P \vee \neg Q)\}$. Figure 7 shows a refutation by unconstrained resolution of Σ. This proves that Σ is unsatisfiable.

Unfortunately, there does not exist an *input* refutation of Σ. It is easy to see the reason for this. To reach the empty clause $\Box$, the last input clause in an

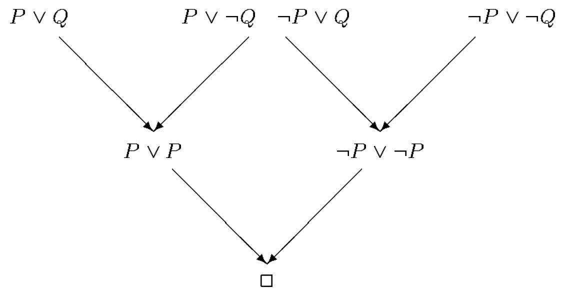

Figure 7: An unconstrained refutation of Σ

input refutation of Σ should contain only one literal, or have a factor containing only one literal. However, each clause in Σ contains two distinct literals. Hence there is no input refutation of Σ.

So input resolution is not refutation-complete. This implies also that the Subsumption Theorem does not hold either for input resolution, since the refutation-completeness would be a direct consequence of it. We can in fact prove a stronger negative result, namely that the Subsumption Theorem for input resolution is not even true in the simple case where Σ contains only a single clause. In our counterexample we let $\Sigma = \{C\}$, where C is the following clause:

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

Figure 8 shows that clause D (see below) can be derived from C by unconstrained resolution. This also shows that $C \models D$.

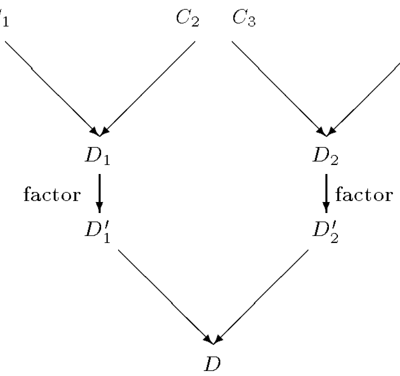

Figure 8: The derivation of D from C by unconstrained resolution

Figure 8 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'.

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

Thus 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 3. This shows that input resolution is not complete, not even if Σ contains only one clause.

The following lemma shows that each clause which can be derived from C by input resolution contains an instance of $P(x_1, x_2) \vee \neg P(x_4, x_1)$ or an instance of $Q(x_2, x_3) \vee \neg Q(x_3, x_4)$.

Lemma 10 *Let C be as defined above. If $C \vdash_{ir} E$, 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 Let $R_0, \ldots, R_k = E$ be an input derivation of E from C. We prove the lemma by induction on k:

1. $R_0 = C$, so the lemma is obvious if $k = 0$.
2. Suppose the lemma holds for $k \leq n$. Let $R_0, \ldots, R_{n+1} = E$ be an input derivation of E from C. Note that the only factor of C is C itself. Therefore E is a binary resolvent of C and a factor of R_m. 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 $(P(x_1, x_2) \vee \neg P(x_4, x_1))\theta$. Hence the lemma also holds for $k = n + 1$.

□

Proposition 3 *Let C and D be as defined above. Then $C \not\vdash_{id} D$.*

Proof Suppose $C \vdash_{id} D$. Then since D is not a tautology, there exists a clause E such that $C \vdash_{ir} E$ and E subsumes D. From Lemma 10 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. But then E does not subsume D, so we found a contradiction. Hence $C \not\vdash_{id} D$. □

So we see that input deduction is not complete: $C \models D$, but $C \not\vdash_{id} D$. This is unfortunate, since input resolution is more efficient than unconstrained resolution or linear resolution. However, if we restrict ourselves to *Horn clauses*, a special case of input resolution called *SLD-resolution* can be shown to be complete. This will be the topic of the next section.

7 SLD-resolution

SLD-resolution for Horn clauses was introduced by Kowalski [Kow74]. It is simpler than the unconstrained or linear resolution that we need for general clauses.

Definition 16 Let Σ be a set of Horn clauses, and C be a Horn-clause. An *SLD-derivation* of C from Σ is a finite sequence of Horn clauses $R_0, \ldots, R_k = C$, such that $R_0 \in \Sigma$ and each R_i $(1 \leq i \leq k)$ is a binary resolvent of R_{i-1} and a definite program clause $C_i \in \Sigma$, using the head of C_i and a *selected atom* in the body of R_{i-1} as the literals resolved upon.

R_0 is called the *top clause*, and the C_i are the *input clauses* of this SLD-derivation. If an SLD-derivation of C from Σ exists, we write $\Sigma \vdash_{sr} C$. An SLD-derivation of $\Box$ from Σ is called an *SLD-refutation* of Σ. $\diamond$

Note that either each R_i in an SLD-derivation is a goal, or each R_i is a definite program clause. Also note that each resolvent in an SLD-derivation is a *binary* resolvent, so no factors are used here. The selected atom can be selected by a so-called *computation rule*, and it can be shown that the refutation-completeness of SLD-resolution is independent of the computation rule that is used. We will not go into that here (see [Llo87]).

Definition 17 Let Σ be a set of Horn clauses and C a Horn clause. There exists an *SLD-deduction* of C from Σ, written as $\Sigma \vdash_{sd} C$, if C is a tautology, or if there is a Horn clause D, such that $\Sigma \vdash_{sr} D$ and D subsumes C. $\diamond$

Example 5 Consider $\Sigma = \{P(0, x, x), (P(s(x), y, s(z)) \leftarrow P(x, y, z))\}$, a set of clauses which formalizes addition. Let us see how we can prove $C = P(s^2(0), s(0), s^3(0))$ (that is, $2+1=3$) from this set by SLD-resolution. Figure 9 shows an SLD-derivation of $R_2 = P(s^2(0), y, s^2(y))$ from Σ. Here the selected atoms are underlined. Since R_2 subsumes C, we have $\Sigma \vdash_{sd} C$.

$\triangleleft$

7.1 The refutation-completeness

In this subsection, we will prove the well-know result that SLD-resolution is refutation-complete: a set of Horn clauses is unsatisfiable iff it has an SLD-refutation. Our proof is similar to the proof for the refutation-completeness of linear resolution that we gave in Section 5. It is different from the proof given in [Llo87], since our proof does not require fixed-point theory. Instead, it only uses the basic definitions of resolution. First we establish the refutation-completeness for ground Horn clauses:

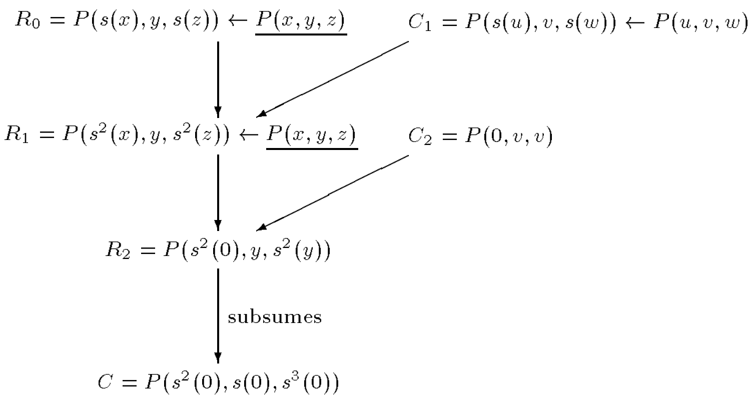

Figure 9: An SLD-deduction of C from Σ

Lemma 11 *If Σ is a finite unsatisfiable set of ground Horn clauses, then $\Sigma \vdash_{sr} \Box$.*

Proof Let n be the number of atomic clauses (clauses which consist of a single positive literal) in Σ. The proof is by induction on n.

1. If $n = 0$, then $\Box \in \Sigma$, for otherwise the empty set would be an Herbrand model of Σ.
2. Suppose the lemma holds for $0 \leq n \leq m$. Suppose Σ contains $m + 1$ atomic clauses. If $\Box \in \Sigma$ the lemma is obvious, so suppose $\Box \notin \Sigma$.
   Let A be an atomic clause in Σ. We first delete all clauses from Σ which have A as head (so we also delete the atomic clause A from Σ). Then we replace clauses which have A in their body by clauses constructed by deleting these atoms A from the body (so for example, $B \leftarrow A, B_1, \ldots, B_k$ will be replaced by $B \leftarrow B_1, \ldots, B_k$). Call the set obtained in this way Σ'.
   If M were a Herbrand model of Σ', then $M \cup \{A\}$ would be a Herbrand model of Σ. Thus since Σ is unsatisfiable, Σ' must be unsatisfiable. Σ' only contains m atomic clauses, so by the induction hypothesis, there is an SLD-refutation of Σ'. If this refutation only uses clauses from Σ' which were also in Σ, then this is also an SLD-refutation of Σ, so then we are done.
   Otherwise, if C is the top clause or an input clause in this refutation and $C \notin \Sigma$, then C was obtained from some $C' \in \Sigma$ by deleting all atoms A from the body of C'. For all such C, do the following: restore the previously deleted copies of A to the body of C (which turns C into C' again), and add these atoms A to all later resolvents. This way, we can turn the SLD-refutation of Σ' into an SLD-derivation of $\leftarrow A, \ldots, A$ from Σ. (See figure 10 for illustration, where we add previously deleted atoms A to the bodies of R_0 and C_2.) Since also $A \in \Sigma$, we can construct an SLD-refutation of Σ, using A a number of times as input clause to resolve away all members of the goal $\leftarrow A, \ldots, A$.

$\Box$

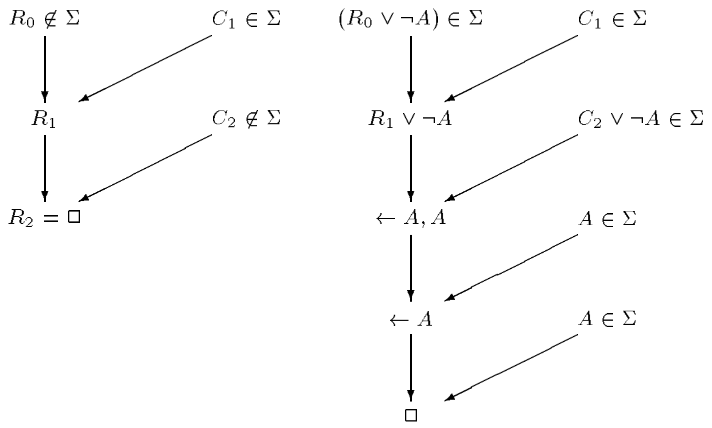

Figure 10: The SLD-refutations of Σ' (left) and Σ (right)

The proof of the lifting lemma for SLD-resolution is similar to Lemma 3.

Lemma 12 (SLD-derivation lifting) *Let Σ be a set of Horn clauses, and Σ' be a set of instances of clauses in Σ. Suppose $R'_0, \ldots, R'_k$ is an SLD-derivation of the clause R'_k from Σ'. Then there exists an SLD-derivation $R_0, \ldots, R_k$ of the clause R_k from Σ, such that R'_i is an instance of R_i, for each i.*

The previous lemmas allow us to prove the refutation-completeness of SLD-resolution:

Theorem 12 (Refutation-completeness of SLD-resolution) *Let Σ be a set of Horn clauses. Then Σ is unsatisfiable iff $\Sigma \vdash_{sr} \Box$.*

Proof

$\Leftarrow$: By the soundness of resolution.

$\Rightarrow$: Suppose Σ is unsatisfiable. By Theorem 4, there is a finite unsatisfiable set Σ' of ground instances of clauses in Σ. From Lemma 11, we have $\Sigma' \vdash_{sr} \Box$. Using Lemma 12, we can lift this to $\Sigma \vdash_{sr} \Box$. $\Box$

7.2 The Subsumption Theorem

Here we will prove the Subsumption Theorem for SLD-resolution. As in the case of linear resolution, we establish this result by translating a refutation to a deduction, using the following lemma:

Lemma 13 *Let Σ be a set of Horn clauses, and $C = L_1 \vee \ldots \vee L_k$ be a non-tautologous ground Horn clause. If $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_{sr} \Box$, then $\Sigma \vdash_{sd} C$.*

Proof Suppose $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_{sr} \Box$, that is, there exists an SLD-refutation $R_0, \ldots, R_n = \Box$ of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$. By induction on n:

1. If $n = 0$, then $R_0 = \Box \in \Sigma$, so then the lemma is obvious.

2. Suppose the lemma holds for $n \leq m$. Let $R_0, \ldots, R_{m+1} = \Box$ be an SLD-refutation of $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$. Then $R_1, \ldots, R_{m+1}$ is an SLD-refutation of $\Sigma \cup \{R_1\} \cup \{\neg L_1, \ldots, \neg L_k\}$. By the induction hypothesis, there is an SLD-derivation $R'_1, R'_2, \ldots, R'_l$ from $\Sigma \cup \{R_1\}$, where R'_l subsumes C. Note that R_1 must be a definite goal, so R_1 can only be used as top clause in this derivation.
If $R'_1 \neq R_1$, then $R'_1 \in \Sigma$. Moreover, in that case R_1 is used nowhere in the SLD-derivation of R'_l, so then this is an SLD-derivation of R'_l from Σ, and hence $\Sigma \vdash_{sd} C$. In case $R'_1 = R_1$, we distinguish three possibilities:
   1. R_1 is a binary resolvent of a goal $G \in \Sigma$ and a definite clause $C_1 \in \Sigma$. Then $G, R'_1, R'_2, \ldots, R'_l$, with C_1 as first input clause, is an SLD-derivation from Σ. R'_l subsumes C, so then $\Sigma \vdash_{sd} C$.
   2. R_1 is a binary resolvent of a negative literal $\neg L \in \{\neg L_1, \ldots, \neg L_k\}$, and a definite clause $C_1 \in \Sigma$ (note that this means that C is a definite program clause, with L as head). Let θ be the mgu used in this resolution-step, so $C_1\theta = L \vee R_1$. Then $C_1\theta, L \vee R'_2, \ldots, L \vee R'_l$ is an SLD-derivation of $L \vee R'_l$ from $\Sigma \cup \{C_1\theta\}$. (See Figure 11 for illustration.) $C_1\theta$ is an instance of a clause in Σ, so by Lemma 12, we can find an SLD-derivation from Σ of a clause D, of which $L \vee R'_l$ is an instance. Since R'_l subsumes C and $L \in C$, $L \vee R'_l$ subsumes C, and hence D also subsumes C. Therefore $\Sigma \vdash_{sd} C$.

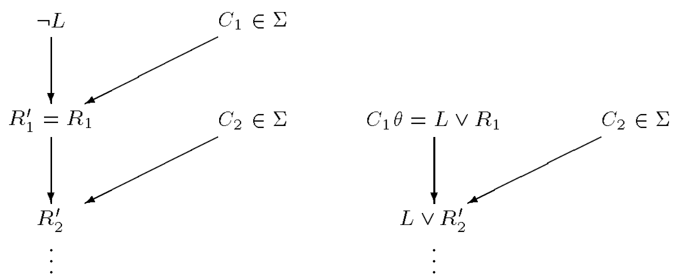

Figure 11: Illustration of case 2 of the proof

   3. R_1 is a binary resolvent of a goal $G \in \Sigma$, and a positive literal $L \in \{\neg L_1, \ldots, \neg L_k\}$. Let θ be the mgu used in this resolution step, so $G\theta = \neg L \vee R_1$. Then $G\theta = \neg L \vee R'_1, \neg L \vee R'_2, \ldots, \neg L \vee R'_l$ is an SLD-derivation of $\neg L \vee R'_l$ from $\Sigma \cup \{G\theta\}$. $G\theta$ is an instance of a clause in Σ, so by Lemma 12, we can find an SLD-derivation from Σ of a clause D, of which $\neg L \vee R'_l$ is an instance. Since R'_l subsumes C and $\neg L \in C$, $\neg L \vee R'_l$ subsumes C, and hence D also subsumes C. Therefore $\Sigma \vdash_{sd} C$.

$\Box$

Now we can prove the Subsumption Theorem for SLD-resolution. This result generalizes Theorem 18 of [MP94], where the theorem is proved for the case where Σ contains only one clause (though ignoring that C may be a tautology).

Theorem 13 (Subsumption Theorem for SLD-resolution) *Let Σ be a set of Horn clauses, and C be a Horn clause. Then $\Sigma \models C$ iff $\Sigma \vdash_{sd} C$.*

Proof

$\Leftarrow$: By the soundness of resolution and subsumption.

$\Rightarrow$: If C is a tautology, the theorem is obvious. Assume C is not a tautology. Let θ be a Skolem 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 by Proposition 1 the set of clauses $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\}$ is unsatisfiable. Then it follows from Theorem 12 that $\Sigma \cup \{\neg L_1, \ldots, \neg L_k\} \vdash_{sr} \Box$. Therefore by Lemma 13, there exists a clause D such that $\Sigma \vdash_{sr} D$, and D subsumes $C\theta$. From Lemma 5, D also subsumes C itself. Hence $\Sigma \vdash_{sd} C$. $\Box$

Note the following special case of this result: if Σ is a set of definite program clauses and A is an atom such that $\Sigma \models A$, then there exists an atom B such that $\Sigma \vdash_{sr} B$ and A is an instance of B.

Furthermore, analogous to the case of linear resolution, we also have the following equivalence:

Theorem 14 *For SLD-resolution, the refutation-completeness and the Subsumption Theorem are equivalent.*

8 Summary

The Subsumption Theorem is the following statement:

> If Σ is a set of clauses and C is 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 some form of resolution.

This theorem is a more direct form of completeness than the better-known refutation-completeness of resolution, and hence sometimes more useful, particularly for theoretical analysis.

Different versions of the theorem exist, depending on the instantiation of "some form of resolution." We have proved here that the Subsumption Theorem holds for unconstrained resolution and linear resolution for general clauses. Moreover, for each of these two forms of resolution, the Subsumption Theorem is equivalent to the refutation-completeness of that form of resolution: the one can be proved from the other. On the other hand, the Subsumption Theorem does not hold for input resolution, not even in the simple case where Σ contains only one clause. For SLD-resolution for Horn clauses, the Subsumption Theorem does hold, and is again equivalent to the refutation-completeness of SLD-resolution.

Acknowledgment

We would like to thank Stephen Muggleton for sending us a copy of [Kow70] and a reference to [SCL69], and Katsumi Inoue for drawing our attention to his paper and some other papers.

References

[BG96] Francesco Bergadano and Daniele Gunetti. *Inductive Logic Programming: From Machine Learning to Software Engineering.* The MIT Press, 1996.

[BJ89] George S. Boolos and Richard C. Jeffrey. *Computability and Logic.* Cambridge University Press, Cambridge (UK), third edition, 1989.

[BM92] Michael Bain and Stephen Muggleton. Non-monotonic learning. In Stephen Muggleton, editor, *Inductive Logic Programming*, volume 38 of *APIC Series*, pages 145–153. Academic Press, 1992.

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, San Diego, 1973.

[GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, Palo Alto (CA), 1987.

[IA93] Peter Idestam-Almquist. *Generalization of Clauses.* PhD thesis, Stockholm University, 1993.

[IA95] Peter Idestam-Almquist. Generalization of clauses under implication. *Journal of Artificial Intelligence Research*, 3:467–489, 1995.

[Ino92] Katsumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56:301–353, 1992.

[KK71] Robert Kowalski and Donald Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[Kow70] Robert A. Kowalski. The case for using equality axioms in automatic demonstration. In *Proceedings of the Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 112–127. Springer-Verlag, 1970.

[Kow74] Robert A. Kowalski. Predicate logic as a programming language. *Information Processing*, 74:569–574, 1974.

[LD94] Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.

[Lee67] Richard Char-Tung Lee. *A Completeness Theorem and a Computer Program for Finding Theorems Derivable from Given Axioms.* PhD thesis, University of California, Berkeley, 1967.

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

[LNC94a] Patrick R. J. van der Laag and Shan-Hwei Nienhuys-Cheng. Existence and nonexistence of complete refinement operators. In

F. Bergadano and L. De Raedt, editors, *Proceedings of the 7th European Conference on Machine Learning (ECML-94)*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 307–322. Springer-Verlag, 1994.

[LNC94b] Patrick R. J. van der Laag and Shan-Hwei Nienhuys-Cheng. A note on ideal refinement operators in inductive logic programming. In Wrobel [Wro94], pages 247–262.

[Lov70] Donald W. Loveland. A linear format for resolution. In *Proceedings of the IRIA Symposium on Automatic Demonstration, Versailles, France, 1968*, pages 147–162. Springer-Verlag, 1970.

[Lov78] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis*. North Holland, New York, 1978.

[Luc70] D. Luckham. Refinements in resolution theory. In *Proceedings of the IRIA Symposium on Automatic Demonstration, Versailles, France, 1968*, pages 163–190. Springer-Verlag, 1970.

[MDR94] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19:629–679, 1994.

[MP94] Stephen Muggleton and C. David Page. Self-saturation of definite clauses. In Wrobel [Wro94], pages 161–174.

[MR72] Eliana Minicozzi and Raymond Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence*, 3:175–180, 1972.

[Mug92] Stephen Muggleton. Inverting implication. In S. Muggleton and K. Furukawa, editors, *Proceedings of the 2nd International Workshop on Inductive Logic Programming (ILP-92)*, Tokyo, 1992. ICOT Research Center. ICOT Technical Memorandum TM-1182.

[NCLT93] Shan-Hwei Nienhuys-Cheng, Patrick R. J. van der Laag, and Leon van der Torre. Constructing refinement operators by deconstructing logical implication. In Pietro Torasso, editor, *Proceedings of the 3rd Conference of the Italian Association for Artificial Intelligence (AI*IA-93)*, volume 728 of *Lecture Notes in Artificial Intelligence*, pages 178–189. Springer-Verlag, 1993.

[NCW95a] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. The equivalence of the subsumption theorem and the refutation-completeness for unconstrained resolution. In K. Kanchanasut and J.-J. Lévy, editors, *Proceedings of the Asean Computer Science Conference (ACSC-95)*, volume 1023 of *Lecture Notes in Computer Science*, pages 269–285. Springer-Verlag, 1995.

[NCW95b] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. The subsumption theorem revisited: Restricted to SLD-resolution. In J. C. van Vliet, editor, *Proceedings of Computing Science in the Netherlands (CSN-95)*, pages 143–154. SION, 1995.

[NCW96] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. The subsumption theorem in inductive logic programming: Facts and fallacies. In Luc De Raedt, editor, *Advances in Inductive Logic Programming*, pages 265–276. IOS Press, Amsterdam, 1996.

[Rob65] J. Alan Robinson. A machine oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[SCL69] J. R. Slagle, C. L. Chang, and R. C. T. Lee. Completeness theorems for semantic resolution in consequence-finding. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence (IJCAI-69)*, pages 281–285, 1969.

[Wro94] Stefan Wrobel, editor. *Proceedings of the 4th International Workshop on Inductive Logic Programming (ILP-94)*, volume 237 of *GMD-Studien*, Bad Honnef/Bonn, 1994. Gesellschaft für Mathematik und Datenverarbeitung.