Least Generalizations under Implication

Shan-Hwei Nienhuys-Cheng and Ronald de Wolf

Erasmus University of Rotterdam, Department of Computer Science, H4-19,
P.O. Box 1738, 3000 DR Rotterdam, the Netherlands,
{cheng,bidewolf}@cs.few.eur.nl

Abstract. One of the most prominent approaches in Inductive Logic Programming is the use of *least generalizations* under subsumption of given clauses. However, subsumption is weaker than logical implication, and not very well suited for handling recursive clauses. Therefore an important open question in this area concerns the existence of least generalizations under implication (LGIs). Our main new result in this paper is the existence and computability of such an LGI for any finite set of clauses which contains at least one non-tautologous function-free clause. We can also define implication relative to background knowledge. In this case, least generalizations only exist in a very limited case.

1 Introduction

Inductive Logic Programming (ILP) is the intersection of Logic Programming and Machine Learning. It studies methods to induce clausal theories from given sets of positive and negative examples. An inductively inferred theory should imply all of the positive, and none of the negative examples. For instance, suppose we are given $P(0)$, $P(s^2(0))$, $P(s^4(0))$, $P(s^6(0))$ as positive examples, and $P(s(0)), P(s^3(0)), P(s^5(0))$ as negative examples. Then the set $\Sigma = \{P(0), (P(s^2(x)) \leftarrow P(x))\}$ is a solution: it implies all positive, and no negative examples. Note that this set can be seen as a description of the even integers. Thus induction of clausal theories is a form of learning from examples. For a more extensive introduction to ILP, we refer to [6, 10].

One of the most prominent approaches in ILP is the use of least generalizations under subsumption of given clauses, introduced by Plotkin [16, 17]. A clause C is a *least generalization under subsumption* (LGS) of a finite set S of clauses, if C subsumes every clause in S, and is subsumed by any other clause which also subsumes every clause in C. Plotkin's main result is that any finite set of clauses has an LGS. The construction of such a *least* generalization allows us to generalize the examples cautiously, avoiding over-generalization. Of course, we need not take the LGS of *all* positive examples, which would yield a theory consisting of only one clause. Instead, we might divide the positive examples into subsets, and take a separate LGS of each subset. That way we obtain a theory containing more than one clause.

However, subsumption is not fully satisfactory for such generalizations. For example, if S consists of $D_1 = P(f^2(a)) \leftarrow P(a)$ and $D_2 = P(f(b)) \leftarrow P(b)$, then $P(f(y)) \leftarrow P(x)$ is an LGS of S. The clause $P(f(x)) \leftarrow P(x)$, which seems more appropriate as a least generalization of S, cannot be found by Plotkin's approach, because it does not subsume D_1. As this example also shows, subsumption is particularly unsatisfactory for *recursive* clauses: clauses which can be resolved with themselves.

Because of the weakness of subsumption, it is desirable to consider least generalizations *under implication* (LGIs) instead. Accordingly, we want to find out whether Plotkin's positive result on the existence of LGSs holds for LGIs as well. Most ILP-researchers are inclined to believe that this question has a negative answer, due to the undecidability of logical implication between clauses [8]. If we restrict attention to Horn clauses (clauses with at most one positive literal), the question has indeed been answered negatively: there is no least Horn clause which implies both $P(f^2(x)) \leftarrow P(x)$ and $P(f^3(x)) \leftarrow P(x)$ [10]. However, Muggleton and Page [12] have shown that the *non-Horn* clause $P(f(x)) \vee P(f^2(y)) \leftarrow P(x)$ is an LGI of these two clauses. Therefore we investigate the existence of an LGI in the set of *general* (not necessarily Horn) clauses here.

No definive answer has as yet been given to this more general question, but some work has already been done. For instance, Idestam-Almquist [4] studies least generalizations under *T-implication* as an approximation to LGIs. Muggleton and Page [12] investigate *self-saturated* clauses. A clause is self-saturated if it is subsumed by any clause which implies it. A clause D is a self-saturation of C, if C and D are logically equivalent and D is self-saturated. As [12] states, if two clauses C_1 and C_2 have self-saturations D_1 and D_2, respectively, then an LGS of D_1 and D_2 is also an LGI of C_1 and C_2. This positively answers our question concerning the existence of LGIs for clauses which have a self-saturation. However, Muggleton and Page also show that there exist clauses which have no self-saturation. So the concept of self-saturation cannot solve the general question concerning the existence of LGIs.

In this paper, we prove the new result that if S is a finite set of clauses containing at least one non-tautologous function-free clause (among other clauses which may contain functions), then S has a computable LGI. Our proof is on the one hand based on the Subsumption Theorem for resolution [7, 5, 15], and on the other hand on a modification of some results of Idestam-Almquist [4] concerning T-implication. An immediate corollary of this result is the existence and computability of an LGI of any finite set of function-free clauses. This result does not solve the general question of the existence of LGIs, but it does provide a positive answer for a large class of cases: the presence of one non-tautologous function-free clause in a finite S already guarantees the existence and computability of an LGI of S. Because of the prominence of function-free clauses in ILP, this case may be of great practical signifcance.[1] Well-known ILP-systems such as FOIL

[1] Note that even for function-free clauses, the subsumption order is still not enough. Consider $D_1 = P(x, y, z) \leftarrow P(y, z, x)$ and $D_2 = P(x, y, z) \leftarrow P(z, x, y)$ (this example is adapted from Idestam-Almquist). D_1 is a resolvent of D_2 and D_2, and D_2 is a resolvent of D_1 and D_1. Hence D_1 and D_2 are logically equivalent. This means that D_1 is an LGI of the set $\{D_1, D_2\}$. However, the LGS of these two clauses is $P(x, y, z) \leftarrow P(u, v, w)$, which is clearly an over-generalization.

[18], Linus [6], and Mobal [9], all use only function-free clauses.

Apart from "plain" subsumption, one can also define subsumption relative to background knowledge. The two best-known forms are Plotkin's *relative subsumption* [17], and Buntine's *generalized subsumption* [1]. Similarly, we can generalize implication to *relative* implication, which will be considered in Section 5.

The results of this paper, together with some other results on greatest specializations and the lattice-structure of sets of clauses ordered by subsumption or implication, are described in more detail in our article [14].

2 Preliminaries

In this section, we will define the main concepts we need. For the definitions of 'model', 'tautology', 'substitution', etc., we refer to [2]. A *positive literal* is an atom, a *negative literal* is the negation of an atom. A *clause* is a finite set of literals, treated as the universally quantified disjunction of those literals. If C is a clause, then C^+ denotes the set of positive literals in C, while C^- denotes the set of negative literals.

Definition 1. Let $\mathcal{A}$ be an alphabet of the first-order logic. Then the *clausal language* $\mathcal{C}$ *by* $\mathcal{A}$ is the set of all clauses which can be constructed from the symbols in $\mathcal{A}$.

Here we just presuppose some arbitrary alphabet $\mathcal{A}$, and consider the clausal language $\mathcal{C}$ based on this $\mathcal{A}$.

Definition 2. Let Γ be a set, and R be a binary relation on Γ.

1. R is *reflexive on* Γ, if xRx for every $x \in \Gamma$.
2. R is *transitive on* Γ, if for every $x, y, z \in \Gamma$, xRy and yRz implies xRz.
3. R is *symmetric on* Γ, if for every $x, y \in \Gamma$, xRy implies yRx.
4. R is *anti-symmetric on* Γ, if for every $x, y, z \in \Gamma$, xRy and yRx implies $x = y$.

If R is both reflexive and transitive on Γ, we say R is a *quasi-order* on Γ. If R is both reflexive, transitive, and anti-symmetric on Γ, we say R is a *partial order* on Γ. If R is reflexive, transitive and symmetric on Γ, R is an *equivalence relation* on Γ.

A quasi-order R on Γ induces an equivalence-relation $\sim$ on Γ, as follows: we say $x, y \in \Gamma$ are *equivalent* induced by R (denoted $x \sim y$) if both xRy and yRx. Using this equivalence relation, a quasi-order R on Γ induces a partial order R' on the set of equivalence classes in Γ, defined as follows: if $[x]$ denotes the equivalence class of x (i.e., $[x] = \{y \mid x \sim y\}$), then $[x]R'[y]$ iff xRy.

We first give a general definition of least generalizations for sets of clauses ordered by some quasi-order.

Definition 3. Let Γ be a set of clauses, $\geq$ be a quasi-order on Γ, $S \subseteq \Gamma$ be a finite set of clauses, and $C \in \Gamma$. If $C \geq D$ for every $D \in S$, then we say C is a *generalization* of S under $\geq$. Such a C is called a *least generalization (LG)* of S under $\geq$ in Γ, if $C' \geq C$ for every generalization $C' \in \Gamma$ of S under $\geq$.

It is easy to see that if some set S has an LG under $\geq$ in Γ, then this LG will be unique up to the equivalence induced by $\geq$ in Γ. That is, if C and D are both LGs of some set S, then we have $C \sim D$.

We will now define three increasingly strong quasi-orders on clauses: subsumption, implication, and relative implication.

Definition 4. Let C and D be clauses, and Σ be a set of clauses. C *subsumes* D, denoted as $C \succeq D$, if there exists a substitution θ such that $C\theta \subseteq D$. C and D are *subsume-equivalent* if $C \succeq D$ and $D \succeq C$.

Σ *(logically) implies* C, denoted as $\Sigma \models C$, if every model of Σ is also a model of C. C *(logically) implies* D, denoted as $C \models D$, if $\{C\} \models D$. C and D are *(logically) equivalent* if $C \models D$ and $D \models C$.

C *implies* D *relative to* Σ, denoted as $C \models_\Sigma D$, if $\Sigma \cup \{C\} \models D$. C and D are *equivalent relative to* Σ if $C \models_\Sigma D$ and $D \models_\Sigma C$.

If C does not subsume D, we write $C \not\succeq D$. Similarly we use $C \not\models D$ and $C \not\models_\Sigma D$. 'Least generalization under subsumption' will be abbreviated to LGS. Similarly, LGI is 'least generalization under implication', and LGR is 'least generalization under relative implication'.

If $C \succeq D$, then $C \models D$. The converse does not hold, as the examples in the Introduction showed. Similarly, if $C \models D$, then $C \models_\Sigma D$, and again the converse need not hold. Consider $C = P(a) \vee \neg P(b)$, $D = P(a)$, and $\Sigma = \{P(b)\}$: then $C \models_\Sigma D$, but $C \not\models D$.

We now proceed to define a proof procedure for logical implication between clauses, using resolution and subsumption.

Definition 5. Let C_1 and C_2 be clauses. If C_1 and C_2 have no variables in common, then they are said to be *standardized apart*.

Given clauses $C_1 = L_1 \vee \ldots \vee L_i \vee \ldots \vee L_m$ and $C_2 = M_1 \vee \ldots \vee M_j \vee \ldots \vee M_n$ which are standardized apart. If the substitution θ is a most general unifier (mgu) of the set $\{L_i, \neg M_j\}$, then the clause $((C_1 - L_i) \cup (C_2 - M_j))\theta$ is a *binary resolvent* of C_1 and C_2. L_i and M_j are said to be the literals *resolved upon*.

If C_1 and C_2 are not standardized apart, we can take a variant C_2' of C_2, such that C_1 and C_2' are standardized apart. For simplicity, a binary resolvent of C_1 and C_2' is also called a binary resolvent of C_1 and C_2 itself.

Definition 6. Let C be a clause, and θ an mgu of $\{L_1, \ldots, L_n\} \subseteq C$ $(n \geq 1)$. Then the clause $C\theta$ is called a *factor* of C.

Definition 7. A *resolvent* C of clauses C_1 and C_2 is a binary resolvent of a factor of C_1 and a factor of C_2, where the literals resolved upon are the literals unified in the respective factors. C_1 and C_2 are the *parent clauses* of C.

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 $\Sigma \vdash_r C$.

Definition 9. 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 $D \succeq C$.

The next result, proved in [15], generalizes Herbrand's Theorem:

Theorem 10. *Let Σ be a set of clauses, and C a ground clause. If $\Sigma \models C$, then there is a finite set Σ_g of ground instances of clauses in Σ, such that $\Sigma_g \models C$.*

The following Subsumption Theorem gives a precise characterization of implication between clauses in terms of resolution and subsumption. It was first proved in [7, 5], and reproved in [15].

Theorem 11 (Subsumption Theorem). *Let Σ be a set of clauses, and C be a clause. Then $\Sigma \models C$ iff $\Sigma \vdash_d C$.*

The next lemma was first proved by Gottlob [3]. Actually, it is an immediate corollary of the Subsumption Theorem:

Lemma 12 (Gottlob). *Let C and D be non-tautologous clauses. If $C \models D$, then $C^+ \succeq D^+$ and $C^- \succeq D^-$.*

Proof. Since $C^+ \succeq C$, if $C \models D$, then we have $C^+ \models D$. Since C^+ cannot be resolved with itself, it follows from the Subsumption Theorem that $C^+ \succeq D$. But then C^+ must subsume the positive literals in D, hence $C^+ \succeq D^+$. Similarly $C^- \succeq D^-$. □

An important consequence of this lemma concerns the *depth* of clauses:

Definition 13. Let t be a term. If t is a variable or constant, then the *depth* of t is 1. If $t = f(t_1, \ldots, t_n)$, $n \geq 1$, then the depth of t is 1 plus the depth of the t_i with largest depth. The *depth* of a clause C is the depth of the term with largest depth in C.

Suppose $C \models D$, and D is not a tautology. By Gottlob's Lemma, we must have $C^+ \succeq D^+$ and $C^- \succeq D^-$. Since applying a substitution cannot decrease the depth of a clause, it follows that $depth(C) \leq depth(D)$. Hence in case $depth(C) > depth(D)$ and D is not a tautology, we know C cannot imply D. For instance, take $D = P(x, f(x, g(y))) \leftarrow P(g(a), b)$, which has depth 3. Then a clause C containing a term $f(x, g^2(y))$ (depth 4) cannot imply D.

Definition 14. Let S and S' be finite sets of clauses, $x_1, \ldots, x_n$ all distinct variables appearing in S, and $a_1, \ldots, a_n$ distinct constants not appearing in S or S'. Then $\sigma = \{x_1/a_1, \ldots, x_n/a_n\}$ is called a *Skolem substitution* for S w.r.t. S'. If S' is empty, we just say that σ is a Skolem substitution for S.

Lemma 15. *Let Σ be a set of clauses, C be a clause, and σ be a Skolem substitution for C w.r.t. Σ. Then $\Sigma \models C$ iff $\Sigma \models C\sigma$.*

Proof.

$\Rightarrow$: Obvious.

$\Leftarrow$: Suppose C is not a tautology, and let $\sigma = \{x_1/a_1, \ldots, x_n/a_n\}$. If $\Sigma \models C\sigma$, it follows from the Subsumption Theorem that there is a D such that $\Sigma \vdash_r D$, and $D \succeq C\sigma$. Thus there is a θ, such that $D\theta \subseteq C\sigma$. Note that since $\Sigma \vdash_r D$ and none of the constants $a_1, \ldots, a_n$ appears in Σ, none of these constants appears in D. Now let θ' be obtained by replacing in θ all occurrences of a_i by x_i, for every $1 \leq i \leq n$. Then $D\theta' \subseteq C$, hence $D \succeq C$. Therefore $\Sigma \vdash_d C$, and hence $\Sigma \models C$. □

3 Least Generalizations under Implication

In this section, we show that any finite set of clauses which contains at least one non-tautologous function-free clause, has an LGI in $\mathcal{C}$. An immediate corollary is the existence of an LGI of any finite set of function-free clauses. In our usage of the word, a 'function-free' clause may contain constants, even though constants are sometimes seen as functions of arity 0. Note that a clause is function-free iff it has depth 1.

Definition 16. A clause is *function-free* if it does not contain function symbols of arity 1 or more.

Definition 17. Let C be a clause, $x_1, \ldots, x_n$ all distinct variables in C, and K a set of terms. Then the *instance set* of C w.r.t. K is $\mathcal{I}(C, K) = \{C\theta \mid \theta = \{x_1/t_1, \ldots, x_n/t_n\}$, where $t_i \in K$, for every $1 \leq i \leq n\}$. If $\Sigma = \{C_1, \ldots, C_k\}$ is a set of clauses, then the *instance set* of Σ w.r.t. K is $\mathcal{I}(\Sigma, K) = \mathcal{I}(C_1, K) \cup \ldots \cup \mathcal{I}(C_k, K)$.

For example, if $C = P(x) \vee Q(y)$ and $T = \{a, f(z)\}$, then $\mathcal{I}(C, T) = \{(P(a) \vee Q(a)), (P(a) \vee Q(f(z))), (P(f(z)) \vee Q(a)), (P(f(z)) \vee Q(f(z)))\}$.

Definition 18. Let S be a finite set of clauses, and σ a Skolem substitution for S. The *term set* of S by σ is the set of all terms (including subterms) occurring in $S\sigma$.

A term set of S by some σ is a finite set of ground terms. For instance, the term set of $D = P(f^2(x), y, z) \leftarrow P(y, z, f^2(x))$ by $\sigma = \{x/a, y/b, z/c\}$ is $T = \{a, f(a), f^2(a), b, c\}$.

Consider $C = P(x, y, z) \leftarrow P(z, x, y)$, and D, σ and T as above. Then $C \models D$, and also $\mathcal{I}(C, T) \models D\sigma$, since $D\sigma$ is a resolvent of $P(f^2(a), b, c) \leftarrow P(c, f^2(a), b)$ and $P(c, f^2(a), b) \leftarrow P(b, c, f^2(a))$, which are in $\mathcal{I}(C, T)$. As we will show in the next lemma, this holds in general: if $C \models D$ and C is function-free, then we can restrict attention to the ground instances of C instantiated to terms in the term set of D by some σ.

The proof of Lemma 19 uses the following idea. Consider a derivation of a clause E from a set Σ of ground clauses. Suppose some of the clauses in Σ contain terms not appearing in E. Then any literals containing these terms in Σ must be resolved away in the derivation. This means that if we replace all the terms in the derivation that are not in E, by some other term t, then the result will be another derivation of E. For example, the left of figure 1 shows a derivation of length 1 of E. The term $f^2(b)$ in the parent clauses does not appear in E. If we replace this term by the constant a, the result is another derivation of E (right of the figure).

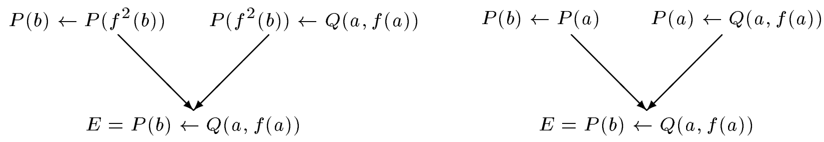

Fig. 1. Transforming the left derivation yields the right derivation

Lemma 19. *Let C be a function-free clause, D be a clause, σ be a Skolem substitution for D w.r.t. $\{C\}$, and T be the term set of D by σ. Then $C \models D$ iff $\mathcal{I}(C, T) \models D\sigma$.*

Proof.
$\Leftarrow$: Since $C \models \mathcal{I}(C, T)$ and $\mathcal{I}(C, T) \models D\sigma$, we have $C \models D\sigma$. Now $C \models D$ by Lemma 15.

$\Rightarrow$: If D is a tautology, then $D\sigma$ is a tautology, so this case is obvious. Suppose D is not a tautology, then $D\sigma$ is not a tautology. Since $C \models D\sigma$, it follows from Theorem 10 that there exists a finite set Σ of ground instances of C, such that $\Sigma \models D\sigma$. By the Subsumption Theorem, there exists a derivation from Σ of a clause E, such that $E \succeq D\sigma$. Since Σ is ground, E must also be ground, so we have $E \subseteq D\sigma$. This implies that E only contains terms from T.

Let t be an arbitrary term in T, and let Σ' be obtained from Σ by replacing every term in clauses in Σ which is not in T, by t. Note that since each clause in Σ is a ground instance of the function-free clause C, every clause in Σ' is also a ground instance of C. Now it is easy to see that the same replacement of terms in the derivation of E from Σ results in a derivation of E from Σ': (1) each resolution step in the derivation from Σ can also be carried out in the derivation from Σ', since the same terms in Σ are replaced by the same terms in Σ', and (2) the terms in Σ that are not in T (and hence are replaced by t), do not appear in the conclusion E of the derivation.

Since there is a derivation of E from Σ, we have $\Sigma' \models E$, and hence $\Sigma' \models D\sigma$. Σ' is a set of ground instances of C and all terms in Σ' are terms in T, so $\Sigma' \subseteq \mathcal{I}(C, T)$. Hence $\mathcal{I}(C, T) \models D\sigma$. $\square$

Lemma 19 cannot be generalized to the case where C contains function symbols of arity ≥ 1, take $C = P(f(x), y) \leftarrow P(z, x)$ and $D = P(f(a), a) \leftarrow P(a, f(a))$.

Then $T = \{a, f(a)\}$ is the term set of D, and we have $C \models D$, yet it can be seen that $\mathcal{I}(C, T) \not\models D$. The argument used in the previous lemma does not work here, because different terms in some ground instance need not relate to different variables. For example, in the ground instance $P(f^2(a), a) \leftarrow P(a, f(a))$ of C, we cannot just replace $f^2(a)$ by some other term, for then the resulting clause would not be an instance of C.

On the other hand, Lemma 19 can be generalized to a *set* of clauses instead of a single clause. If Σ is a finite set of function-free clauses, C is an arbitrary clause, and σ is a Skolem substitution for C w.r.t. Σ, then we have that $\Sigma \models C$ iff $\mathcal{I}(\Sigma, T) \models C\sigma$. The proof is almost literally the same as above.

This result implies that $\Sigma \models C$ is reducible to an implication $\mathcal{I}(\Sigma, T) \models C\sigma$ between ground clauses. Since, by the next lemma, implication between ground clauses is decidable, it follows that $\Sigma \models C$ is decidable in case Σ is function-free.

Lemma 20. *The problem whether $\Sigma \models C$, where Σ is a finite set of ground clauses and C is a ground clause, is decidable.*

Proof. Let $C = L_1 \vee \ldots \vee L_n$, and $\mathcal{A}$ be the set of all ground atoms occurring in Σ and C. Now $\Sigma \models C$ iff
$\Sigma \cup \{\neg L_1, \ldots, \neg L_n\}$ is unsatisfiable iff (by Theorem 4.2 of [2])
$\Sigma \cup \{\neg L_1, \ldots, \neg L_n\}$ has no Herbrand model iff
no subset of $\mathcal{A}$ is an Herbrand model of $\Sigma \cup \{\neg L_1, \ldots, \neg L_n\}$.
Since $\mathcal{A}$ is finite, the last statement is decidable. □

Corollary 21. *The problem whether $\Sigma \models C$, where Σ is a finite set of function-free clauses and C is a clause, is decidable.*

The following sequence of lemmas is adapted, with modifications, from Idestam-Almquist [4], where they are given for T-implication.

Lemma 22. *Let S be a finite set of non-tautologous clauses, $V = \{x_1, \ldots, x_m\}$ be a set of variables, and let $G = \{C_1, C_2, \ldots\}$ be a (possibly infinite) set of generalizations of S under implication. Then the set $G' = \mathcal{I}(C_1, V) \cup \mathcal{I}(C_2, V) \cup \ldots$ is a finite set of clauses.*

Proof. Let d be the maximal depth of the terms in clauses in S. It follows from Lemma 12 that G (and hence also G') cannot contain terms of depth greater than d, nor predicates, functions or constants other than those in S. The set of literals which can be constructed from predicates in S, and from terms of depth at most d consisting of functions and constants in S and variables in V, is finite. Hence the set of clauses which can be constructed from those literals is also finite. G' is a subset of this set, so G' is a finite set of clauses. □

Lemma 23. *Let D be a clause, C be a function-free clause such that $C \models D$, $T = \{t_1, \ldots, t_n\}$ be the term set of D by σ, $V = \{x_1, \ldots, x_m\}$ be a set of variables, and $m \geq n$. If E is an LGS of $\mathcal{I}(C, V)$, then $E \models D$.*

Proof. Let $\gamma = \{x_1/t_1, \ldots, x_n/t_n, x_{n+1}/t_n, \ldots, x_m/t_n\}$ (it does not matter to which terms the variables $x_{n+1}, \ldots, x_m$ are mapped by γ, as long as they are mapped to terms in T). Suppose $\mathcal{I}(C,V) = \{C\rho_1, \ldots, C\rho_k\}$. Then $\mathcal{I}(C,T) = \{C\rho_1\gamma, \ldots, C\rho_k\gamma\}$. Let E be an LGS of $\mathcal{I}(C,V)$ (note that E must be function-free). Then for every $1 \leq i \leq k$, there are θ_i such that $E\theta_i \subseteq C\rho_i$. This means that $E\theta_i\gamma \subseteq C\rho_i\gamma$ and hence $E\theta_i\gamma \models C\rho_i\gamma$, for every $1 \leq i \leq k$. Therefore $E \models \mathcal{I}(C,T)$.

Since $C \models D$, we know from Lemma 12 that constants appearing in C must also appear in D. This means that σ is a Skolem substitution for D w.r.t. $\{C\}$. Then from Lemma 19 we know $\mathcal{I}(C,T) \models D\sigma$, hence $E \models D\sigma$. Furthermore, since E is an LGS of $\mathcal{I}(C,V)$, all constants in E also appear in C, hence all constants in E must appear in D, so σ is a Skolem substitution for D w.r.t. $\{E\}$. Then $E \models D$ by Lemma 15. $\square$

Consider $C = P(x,y,z) \leftarrow P(y,z,x)$ and $D = \leftarrow Q(w)$. Both C and D imply the clause $E = P(x,y,z) \leftarrow P(z,x,y), Q(b)$. Now note that $C \cup D = P(x,y,z) \leftarrow P(y,z,x), Q(w)$ also implies E. This holds for clauses in general:

Lemma 24. *Let C, D, and E be clauses such that C and D are standardized apart. If $C \models E$ and $D \models E$, then $C \cup D \models E$.*

Proof. Suppose $C \models E$ and $D \models E$, and M be a model of the clause $C \cup D$. Since C and D are standardized apart, the clause $C \cup D$ is equivalent to the formula $\forall(C) \vee \forall(D)$ (where $\forall(C)$ denotes the universally quantified clause C). This means that M is a model of C or a model of D. Then it follows from $C \models E$ and $D \models E$ that M is a model of E. Therefore $C \cup D \models E$. $\square$

Now we can prove the existence of an LGI of any finite set S of clauses which contains at least one non-tautologous and function-free clause. In fact we can prove something stronger, namely that this LGI is a *special* LGI, which is not only implied, but actually *subsumed* by any other generalization of S:

Definition 25. Let $\mathcal{C}$ be a clausal language, and S be a finite subset of $\mathcal{C}$. An LGI C of S in $\mathcal{C}$ is called a *special* LGI of S in $\mathcal{C}$, if $C' \succeq C$ for every generalization $C' \in \mathcal{C}$ of S under implication.

Note that if D is an LGI of a set containing at least one non-tautologous function-free clause, then by Lemma 12 D is itself function-free, because it should imply the function-free clause(s) in S. For instance, $C = P(x,y,z) \leftarrow P(y,z,x), Q(w)$ is an LGI of $D_1 = P(x,y,z) \leftarrow P(y,z,x), Q(f(a))$ and $D_2 = P(x,y,z) \leftarrow P(z,x,y), Q(b)$. Note that this LGI is properly subsumed by the LGS of $\{D_1, D_2\}$, which is $P(x,y,z) \leftarrow P(x',y',z'), Q(w)$. An LGI may sometimes be the empty clause $\square$, for example if $S = \{P(a), Q(a)\}$.

Theorem 26 (Existence of special LGI in $\mathcal{C}$). *Let $\mathcal{C}$ be a clausal language. If S is a finite set of clauses from $\mathcal{C}$, and S contains at least one non-tautologous function-free clause, then there exists a special LGI of S in $\mathcal{C}$.*

Proof. Let $S = \{D_1, \ldots, D_n\}$ be a finite set of clauses from $\mathcal{C}$, such that S contains at least one non-tautologous function-free clause. We can assume without loss of generality that S contains no tautologies. Let σ be a Skolem substitution for S, $T = \{t_1, \ldots, t_m\}$ be the term set of S by σ, $V = \{x_1, \ldots, x_m\}$ be a set of variables, and $G = \{C_1, C_2, \ldots\}$ be the set of all generalizations of S under implication in $\mathcal{C}$. Note that $\Box \in G$, so G is not empty. Since each clause in G must imply the function-free clause(s) in S, it follows from Lemma 12 that all members of G are function-free. By Lemma 22, the set $G' = \mathcal{I}(C_1, V) \cup \mathcal{I}(C_2, V) \cup \ldots$ is a finite set of clauses. Since G' is finite, the set of $\mathcal{I}(C_i, V)$s is also finite. For simplicity, let $\{\mathcal{I}(C_1, V), \ldots, \mathcal{I}(C_k, V)\}$ be the set of all distinct $\mathcal{I}(C_i, V)$s.

Let E_i be an LGS of $\mathcal{I}(C_i, V)$, for every $1 \leq i \leq k$, such that $E_1, \ldots, E_k$ are standardized apart. For every $1 \leq j \leq n$, the term set of D_j by σ is some set $\{t_{j_1}, \ldots, t_{j_s}\} \subseteq T$, such that $m \geq j_s$. ¿From Lemma 23, we have that $E_i \models D_j$, for every $1 \leq i \leq k$ and $1 \leq j \leq n$, hence $E_i \models S$. Now let $F = E_1 \cup \ldots \cup E_k$, then we have $F \models S$ from Lemma 24.

To prove that F is a special LGI of S, it remains to show that $C_j \succeq F$, for every $j \geq 1$. For every $j \geq 1$, there is an i $(1 \leq i \leq k)$, such that $\mathcal{I}(C_j, V) = \mathcal{I}(C_i, V)$. So for this i, E_i is an LGS of $\mathcal{I}(C_j, V)$. C_j is itself also a generalization of $\mathcal{I}(C_j, V)$ under subsumption, hence $C_j \succeq E_i$. Then finally $C_j \succeq F$, since $E_i \subseteq F$. $\Box$

Corollary 27. *Let $\mathcal{C}$ be a clausal language. Then for every finite set of function-free clauses $S \subseteq \mathcal{C}$, there exists an LGI of S in $\mathcal{C}$.*[2]

Proof. Let S be a finite set of function-free clauses in $\mathcal{C}$. If S only contains tautologies, any tautology will be an LGI of S. Otherwise, let S' be obtained by deleting all tautologies from S. By the previous theorem, there is a special LGI of S'. Clearly, this is also a special LGI of S itself in $\mathcal{C}$. $\Box$

4 The LGI is Computable

In the previous section we proved the *existence* of an LGI in $\mathcal{C}$ of every finite set S of clauses containing at least one non-tautologous function-free clause. In this section, we will establish the *computability* of such an LGI. The next algorithm, extracted from the proof of the previous section, computes this LGI:

LGI-Algorithm
Input: A finite set S of clauses, at least one of which is non-tautologous and function-free.
Output: An LGI of S in $\mathcal{C}$.

[2] Niblett [13, p. 135] claims that it is simple to show that LGIs exist in a language with only a finite number of constants and no function symbols. Such a result would imply our corollary. However, Niblett has not provided a proof, and neither has anyone else, as far as we know. We would be rather surprised if a proof exists which is actually much simpler than the proof we have given here.

1. Remove all tautologies from S (a clause is a tautology iff it contains literals A and $\neg A$), call the remaining set S'.
2. Let m be the number of distinct terms in S', let $V = \{x_1, \ldots, x_m\}$. (Notice that this m is the same number as the number of terms in the term set T used in the proof of Theorem 26.)
3. Let G be the (finite) set of all clauses which can be constructed from predicates and constants in S' and variables in V.
4. Let $\{U_1, \ldots, U_n\}$ be the set of all subsets of G.
5. Let H_i be an LGS of U_i, for every $1 \leq i \leq n$. These H_i can be computed by Plotkin's algorithm [16].
6. Remove from $\{H_1, \ldots, H_n\}$ all clauses which do not imply S' (since each H_i is function-free, by Corollary 21 this implication is decidable), and standardize the remaining clauses $\{H_1, \ldots, H_q\}$ apart.
7. Return the clause $H = H_1 \cup \ldots \cup H_q$.

The correctness of this algorithm follows from the proof of Theorem 26. First notice that $H \models S$ by Lemma 24. Furthermore, note that all $\mathcal{I}(C_i, V)$s mentioned in the proof of Theorem 26, are elements of the set $\{U_1, \ldots, U_n\}$. This means that for every E_i in the set $\{E_1, \ldots, E_k\}$ mentioned in that proof, there is a clause H_j in $\{H_1, \ldots, H_q\}$ such that E_i and H_j are subsume-equivalent. Then it follows that the LGI $F = E_1 \cup \ldots \cup E_k$ of that proof subsumes the clause $H = H_1 \cup \ldots \cup H_q$ that our algorithm returns. On the other hand, F is a special LGI, so F and H must be subsume-equivalent.

Suppose the number of distinct constants in S' is c, and the number of distinct variables in step 2 of the algorithm is m. Furthermore, suppose there are p distinct predicate symbols in S', with respective arities $a_1, \ldots, a_p$. Then the number of distinct atoms that can be formed from these constants, variables and predicates, is $l = \sum_{i=1}^{p} (c + m)^{a_i}$, and the number of distinct literals that can be formed, is $2 \cdot l$. The set G of distinct clauses which can be formed from these literals is the power set of this set of literals, so $|G| = 2^{2 \cdot l}$. Then the set $\{U_1, \ldots, U_n\}$ of all subsets of G contains $2^{|G|} = 2^{2^{2 \cdot l}}$ members.

Thus the algorithm outlined above is not very efficient (to say the least). A more efficient algorithm may exist, but since implication is harder than subsumption and the computation of an LGS is already quite expensive, we should not put our hopes too high. Nevertheless, the existence of the LGI-algorithm does establish the theoretical point that the LGI of any finite set of clauses containing at least one non-tautologous function-free clause, is computable.

Theorem 28 (Computability of LGI). *Let $\mathcal{C}$ be a clausal language. If S is a finite set of clauses from $\mathcal{C}$, and S contains at least one non-tautologous function-free clause, then the LGI of S in $\mathcal{C}$ is computable.*

5 Least Generalizations under Relative Implication

Implication is stronger than subsumption, but implication relative to background knowledge is even more powerful, since background knowledge can be used to model all sorts of useful properties and relations. Here we will discuss least generalizations under implication relative to some given background knowledge Σ (LGRs).

We will show that even if S and Σ are both finite sets of *function-free* clauses, an LGR of S relative to Σ need not exist. Let $D_1 = P(a)$, $D_2 = P(b)$, $S = \{D_1, D_2\}$, and $\Sigma = \{(P(a) \vee \neg Q(x)), (P(b) \vee \neg Q(x))\}$. This S has no LGR relative to Σ in $\mathcal{C}$.

Suppose C is an LGR of S relative to Σ. Note that if C contains the literal $P(a)$, then the Herbrand interpretation which makes $P(a)$ true, and which makes all other ground literals false, would be a model of $\Sigma \cup \{C\}$ but not of D_2, so then we would have $C \not\models_\Sigma D_2$. Similarly, if C contains $P(b)$ then $C \not\models_\Sigma D_1$. Hence C cannot contain $P(a)$ or $P(b)$ as literals. Now let d be a constant not appearing in C. Let $D = P(x) \vee Q(d)$, then $D \models_\Sigma S$. By the definition of an LGR, we should have $D \models_\Sigma C$. Then by the Subsumption Theorem, there must be a derivation from $\Sigma \cup \{D\}$ of a clause E, which subsumes C. The set of all clauses which can be derived (in 0 or more resolution-steps) from $\Sigma \cup \{D\}$ is $\Sigma \cup \{D\} \cup \{(P(a) \vee P(x)), (P(b) \vee P(x))\}$. But none of these clauses subsumes C, because C does not contain the constant d, nor the literals $P(a)$ or $P(b)$. Hence $D \not\models_\Sigma C$, contradicting the assumption that C is an LGR of S relative to Σ in $\mathcal{C}$.

However, we can identify a special case in which the LGR *does* exist. Here $\Sigma = \{L_1, \ldots, L_k\}$ should be a set of function-free ground literals. A notational remark: if C is a clause, we use $C \cup \overline{\Sigma}$ to denote the clause $C \cup \{\neg L_1, \ldots, \neg L_k\}$. Note that $\{C\} \cup \Sigma$ is a *set* of clauses, while $C \cup \overline{\Sigma}$ is a single clause (a set of literals).

Theorem 29 (Existence of LGR in $\mathcal{C}$). *Let $\mathcal{C}$ be a clausal language and $\Sigma \subseteq \mathcal{C}$ be a finite set of function-free ground literals. If $S \subseteq \mathcal{C}$ is a finite set of clauses, containing at least one D for which $D \cup \overline{\Sigma}$ is non-tautologous and function-free, then S has an LGR in $\mathcal{C}$ relative to Σ.*

Proof. Let $S = \{D_1, \ldots, D_n\}$. It can be seen that since Σ is a finite set of ground literals, for any clauses C and D we have $C \models_\Sigma D$ (i.e., $\Sigma \cup \{C\} \models D$) iff $C \models (D \cup \overline{\Sigma})$. Hence an LGI in $\mathcal{C}$ of $T = \{(D_1 \cup \overline{\Sigma}), \ldots, (D_n \cup \overline{\Sigma})\}$ is also an LGR of S in $\mathcal{C}$. The existence of such an LGI of T follows from Theorem 26. □

It is interesting to compare this result with relative *subsumption*. Plotkin [17] proved that any finite set of clauses has a least generalization under relative subsumption, if the background knowledge Σ is a set of ground literals. This result forms the basis of GOLEM [11], one of the most prominent ILP systems. Under relative *implication*, the background knowledge should not only be ground, but function-free as well. Moreover, the set S to be generalized should contain at least one D such that $D \cup \overline{\Sigma}$ is non-tautologous and function-free. Thus on the one hand, relative implication is a more powerful order than relative subsumption, but on the other hand, the existence of least generalizations can only be guaranteed in a much more restricted case.

6 Conclusion

Implication is more appropriate for least generalizations than subsumption. We have proved here that any finite set of clauses containing at least one non-tautologous function-free clause has a computable LGI. For sets of clauses which all contain functions, the existence of an LGI remains an open question. In case of implication relative to background knowledge, least generalizations need not exist, except for very restricted cases.

References

1. W. Buntine. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence*, 36:149–176, 1988.
2. C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, San Diego, 1973.
3. G. Gottlob. Subsumption and implication. *Inf. Process. Lett.*, 24(2):109–111, 1987.
4. P. Idestam-Almquist. Generalization of clauses under implication. *Journal of Artificial Intelligence Research*, 3:467–489, 1995.
5. R. 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.
6. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.
7. 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.
8. J. Marcinkowski and L. Pacholski. Undecidability of the horn-clause implication problem. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 354–362, Pittsburg, 1992.
9. K. Morik, S. Wrobel, J.-U. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications.* Academic Press, London, 1993.
10. S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and methods. *Journal of Logic Programming*, 19–20:629–679, 1994.
11. S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Inductive Logic Programming*, volume 38 of *APIC Series*, pages 281–298. Academic Press, 1992.
12. S. Muggleton and C. D. Page. Self-saturation of definite clauses. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming (ILP-94)*, volume 237 of *GMD-Studien*, pages 161–174, Bad Honnef/Bonn, 1994. Gesellschaft für Mathematik und Datenverarbeitung.
13. T. Niblett. A study of generalisation in logic programs. In D. Sleeman, editor, *Proceedings of the 3rd European Working Sessions on Learning (EWSL-88)*, pages 131–138, 1988.
14. S.-H. Nienhuys-Cheng and R. de Wolf. Least generalizations and greatest specializations of sets of clauses. *Journal of Artificial Intelligence Research*, 4:341–363, 1996.

15. S.-H. Nienhuys-Cheng and R. de Wolf. The subsumption theorem in Inductive Logic Programming: Facts and fallacies. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 265–276. IOS Press, Amsterdam, 1996.
16. G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
17. G. D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6:101–124, 1971.
18. J. R. Quinlan and R. M. Cameron-Jones. Foil: A midterm report. In P. B. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning (ECML-93)*, volume 667 of *Lecture Notes in Artificial Intelligence*, pages 3–20. Springer-Verlag, 1993.