Induction on Failure: Learning Connected Horn Theories

Tim Kimber, Krysia Broda, and Alessandra Russo

Imperial College London, UK
{timothy.kimber06,k.broda,a.russo}@imperial.ac.uk

Abstract. Several learning systems based on Inverse Entailment (IE) have been proposed, some that compute single clause hypotheses, exemplified by Progol, and others that produce multiple clauses in response to a single seed example. A common denominator of these systems is a restricted hypothesis search space, within which each clause must individually explain some example E, or some member of an abductive explanation for E. This paper proposes a new IE approach, called *Induction on Failure* (IoF), that generalises existing Horn clause learning systems by allowing the computation of hypotheses within a larger search space, namely that of *Connected Theories*. A proof procedure for IoF is proposed that generalises existing IE systems and also resolves Yamamoto's example. A prototype implementation is also described. Finally, a semantics is presented, called *Connected Theory Generalisation*, which is proved to extend Kernel Set Subsumption and to include hypotheses constructed within this new IoF approach.

Keywords: Inductive Logic Programming, Inverse Entailment, Abduction.

1 Introduction

Inductive Logic Programming (ILP) uses the expressive power of first-order logic and the sound theoretical foundations of logic programming, to create a branch of machine learning that seeks the construction of explanations for given examples relative to some background knowledge, in a form easily understood by the user. Among ILP systems that perform best on practical applications are those that use the *Inverse Entailment* (IE) approach to learn Horn theories [1,2,3,4]. These systems first construct a *most specific* hypothesis and then search through formulas that subsume it. The restricted search space that IE defines contributes to the efficiency of these systems. However, the assumptions made by current Horn theory IE systems, regarding the number of clauses necessary to explain a given example, mean that some hypotheses that correctly explain the observations are excluded.

The Progol systems [1,5] assume that every clause in a hypothesis H must individually explain at least one example E. This excludes hypotheses in which two

E. Erdem, F. Lin, and T. Schaub (Eds.): LPNMR 2009, LNCS 5753, pp. 169–181, 2009.

or more clauses combine to explain some E, a situation particularly, though not exclusively, relevant to *non-observational learning*. Consider, for example, the observation $E = \{p(a, b)\}$, background knowledge $B = \{p(X, Y) \leftarrow q(X), r(Y)\}$ and hypothesis $H = \{q(X)\} \cup \{r(X)\}$. This H is outside the Progol search space. A single clause that does explain some E may still be excluded, because the assumption is applied at the ground level first. For example, if the background clause above was in fact $p(X, Y) \leftarrow q(X), q(Y)$, the clause $H = q(X)$ explains E. However, the ground explanation for $p(a, b)$ is $\{q(a)\} \cup \{q(b)\}$ which contains two clauses, meaning it and its subsuming hypothesis $q(X)$ are not returned. The HAIL [3] and ALECTO [4] systems allow a multiple clause explanation of a single *seed* example, and do compute the hypotheses described so far. However, the single clause assumption is only weakened in these systems, not removed completely. Thus, for the previous example HAIL computes the ground unit explanation $\Delta = \{q(a), q(b)\}$ by abduction, and generates one, and only one, clause for each member of Δ. The next example shows that this approach also excludes some correct explanations.

Example 1.

$$E_1 = vehicle(focus)$$
$$B_1 = \{doors(focus, 5)\} \cup \{car(X) \leftarrow hatchback(X)\}$$
$$H_1 = \{vehicle(X) \leftarrow car(X)\} \cup \{hatchback(X) \leftarrow doors(X, 5)\}$$

♦

The abductive explanation $\Delta_1 = \{vehicle(focus)\}$, will constrain HAIL's search space to be the same as Progol's for this example, including only single clauses subsuming $H_1' = vehicle(X) \leftarrow doors(X, 5)$. The two-clause theory H_1 is outside this search, and cannot be computed by any of the ILP systems described.

Intuitively, one can see that H_1 is a structured explanation for E_1, and if the observation set includes data on cars and hatchbacks then it will be preferred to H_1' since it will cover examples with those predicates. This paper proposes a new IE approach, within the setting of Horn clauses, that enables the automatic computation of hypotheses such as H_1, as well as those considered by existing Horn ILP systems. This approach, called *Induction on Failure* (IoF), is based on the notion of a *Connected Theory*. A Connected Theory contains clauses that depend on one another, either directly or via clauses in the background knowledge. IoF is designed to find hypotheses as generalisations of a most specific Connected Theory, denoted $T_\perp$. The semantics of *Connected Theory Generalisation* is presented in Sect. 3, and is shown to define a set of hypotheses that extends those of Progol's Bottom Generalisation (BG) and HAIL's Kernel Set Subsumption (KSS).

The IoF approach computes the theory $T_\perp$ via a recursive inductive procedure. Where current Horn IE systems use a purely deductive *saturation* step to generate the body of clauses, IoF considers body literals which initially lack explanation, starting a new induction process for them. This recursive induction has as base case the standard saturation process adopted by existing systems. A full description of the IoF framework is given in Sect. 4.

In Sect. 5 it is also shown, using a prototype implementation called Imparo, that the proposed IoF approach is general enough to resolve Yamamoto's counter example to the completeness of IE [6], demonstrating its suitability for learning mutually recursive programs. Section 2 gives the necessary background information on ILP, and Sect. 6 describes related and future work.

2 Background

This section presents notation and terminologies used throughout the paper and briefly reviews the ILP task, the IE principle and the methods of Bottom Generalisation and Kernel Set Subsumption.

Notation and Terminology. All formulas are assumed to be constructed from a first-order logic signature. A first-order *clause* is a finite disjunction of zero or more literals $A_1 \vee \ldots \vee A_m \vee \neg A_{m+1} \vee \ldots \vee \neg A_n$, which is assumed to be universally quantified and will be written as the implication $A_1, \ldots, A_m \leftarrow A_{m+1}, \ldots, A_n$. A Horn clause is either the *empty clause* $\square$, or a *fact* A, or a *denial*, $\leftarrow A_1, \ldots, A_n$, or a *definite clause* $A_0 \leftarrow A_1, \ldots, A_n$, where A_0 is the *head* and $A_1, \ldots, A_n$ is the *body* of the clause. Capitalised identifiers denote meta variables. If C is a Horn clause, then C^+ is the set containing the head atom (if any) of C, and C^- is the set of body atoms of C. Moreover, given a set $S = \{C_1, \ldots, C_n\}$ of Horn clauses, S^+ denotes $C_1^+ \cup \ldots \cup C_n^+$ and S^- denotes $C_1^- \cup \ldots \cup C_n^-$. A *theory* is a set of clauses. The symbol $\models$ represents classical entailment. So, if S and T are theories and C is a clause, $S \models C$ means that C is satisfied in every model of S, and $S \models T$ means that $S \models D$ for every $D \in T$. The term *subsumption* and the symbol $\succeq$ refer to θ-subsumption. So, if C and D are clauses, $C \succeq D$ means that the set of literals in $C\theta$ is a (non-strict) subset of the set of literals in D for some substitution θ, and if S and T are theories, $S \succeq T$ means that every clause in T is subsumed by at least one clause in S. Throughout this paper the term *clause* will refer to a Horn clause, unless otherwise stated.

Inductive Logic Programming. Inductive Logic Programming (ILP) is concerned with the task of generalizing positive and negative examples with respect to a background theory. Formally, given background theory B, positive examples E_{pos} and negative examples E_{neg}, the task of ILP is to compute a theory H, called a *hypothesis*, that satisfies the conditions $B \cup H \models E_{pos}$ (*posterior sufficiency*) and $\forall x \in E_{neg}(B \cup H \not\models x)$ (*posterior satisfiability*). The ILP task of finding such an hypothesis H is called *inductive generalisation*. Standard preconditions of an ILP task are *prior necessity*, for which the given background theory must not already entail all the positive examples, $B \not\models E_{pos}$ and *prior satisfiability*, for which B must not entail any of the negative examples, $\forall x \in E_{neg}(B \not\models x)$.

Inverse Entailment. Various existing ILP systems build their approaches to the ILP task upon the principle of *Inverse Entailment* (IE), which states that the negation of every inductive hypothesis H for some example E may be deduced from B and $\neg E$. That is to say $B \cup \{\neg E\} \models \neg H$. This relationship is used to

derive a *most specific* hypothesis ϕ, as defined below, which allows the search space for the inductive generalisation to be limited to those H satisfying $H \models \phi$.

Definition 1 (Most Specific Hypothesis). *Let B be a set of clauses and E be a clause such that $B \not\models E$. Then a set of clauses H is a* most specific hypothesis *for E given B if and only if $B \cup H \models E$ and there does not exist a set H' of clauses such that $B \cup H' \models E$ and $H \models H'$ and $H' \not\models H$.*

Systems such as Progol and Aleph [2] employ IE within the technique of *Bottom Generalisation*. Bottom Generalisation computes a most specific hypothesis $\bot(B, E)$ called the *Bottom Clause* of B and E, as defined below. Since $\bot(B, E)$ is a single clause, its negation is a set of (possibly skolemised) ground literals each of which can be derived from B and $\neg E$. Any clause H which implies $\bot(B, E)$ is said to be derivable by BG, as formalised in Definition 3.

Definition 2 (Bottom Clause [1]). *Let B be a Horn theory and E a Horn clause. Then $\bot(B, E)$, the* Bottom Clause of B and E, *is the disjunction of the ground literals in the set $\{L \mid B \cup \{\neg E\} \models \neg L\}$.*

Definition 3 (Bottom Generalisation [1]). *Let B be a Horn theory and E a Horn clause. A Horn clause H is said to be* derivable from B and E by Bottom Generalisation *if and only if $H \models \bot(B, E)$.*

The Progol system [1], implements Bottom Generalisation by means of Mode Directed Inverse Entailment (MDIE), which computes a definite bottom clause $A_0 \leftarrow A_1, \ldots, A_n$ based on $\bot(B, E)$. Ray et al. [3] showed that such a definite bottom clause could be defined as a head atom A_0 that, together with the background knowledge, explains the head of the example clause (i.e. $B \cup \{A_0\} \models E^+$), and a set of body atoms $\{A_1, \ldots, A_n\}$ that are *directly* derivable from the background knowledge and the body of E (i.e. $B \cup E^- \models A_i$ for each $1 \leq i \leq n$). *Kernel Set Subsumption* (defined below) extends this notion by defining a most specific hypothesis K, the Kernel Set, that is a *set* of ground Horn clauses where the set of head atoms of K explains the head of the example ($B \cup K^+ \models E^+$) and the body atoms of K are directly derivable from the background knowledge and E^- ($B \cup E^- \models K^-$).

Definition 4 (Kernel Set [3]). *Let B be a Horn theory and E be a Horn clause such that $B \not\models E$. Then a ground Horn theory $K = \{C_1, \ldots, C_k\}$ ($k \geq 1$) is a* Kernel Set of B and E *if and only if each clause $C_i, 1 \leq i \leq k$ is given by $A_0^i \leftarrow A_1^i, \ldots, A_{n_i}^i$, where $B \cup \{A_0^1, \ldots, A_0^k\} \models E^+$, and $B \cup E^- \models \{A_1^1, \ldots, A_{n_k}^k\}$.*

Definition 5 (Kernel Set Subsumption [3]). *Let B be a Horn theory and E be a Horn clause such that $B \not\models E$. A set of Horn clauses H is said to be* derivable from B and E by Kernel Set Subsumption, *denoted $B, E \vdash_{KSS} H$, if and only if there is a K such that K is a Kernel Set of B and E, and $H \succeq K$.*

Kernel Set Subsumption extends Bottom Generalisation since it has been shown that all clauses that can be derived by BG can also be derived by KSS, and that

some single clauses not found by BG are found by KSS [3]. Not only this, but KSS allows theories to be derived from a single seed example, whereas BG is limited to clauses. However, the nature of the Kernel Set imposes limits on the possible theories that KSS can compute. The next section describes Connected Theory Generalisation (CTG), a new semantic approach which further generalises KSS, and computes a "more complete" set of hypotheses.

3 Connected Theory Generalisation

This section proposes a new semantic approach to ILP, called *Connected Theory Generalisation* (CTG). This approach is based on the notion of a *Connected Theory*, defined below, and is proved to extend the semantics of Kernel Set Subsumption, which has been shown [3] to extend Bottom Generalisation [1]. The motivation is to extend the class of hypotheses learnable by Horn theory ILP systems.

The Kernel Set extends the notion of a Bottom Clause by replacing the single atom explanation of the given example, $\{A_0\}$, with a set of atoms $\{A_0^1, \ldots, A_0^k\}$. A Connected Theory extends the basic notion even further, not only generalising the head of $\bot(B, E)$ to be a set of atoms, but also extending the set of body atoms beyond those that are direct consequences of the background knowledge. Intuitively, all body atoms in $\bot(B, E)$, a single clause, must be implied by B for $B \cup \{\bot(B, E)\}$ to entail E. In a hypothesis which is a set of clauses this need not be true for a given member of the set. The Kernel Set retains this restriction, but, as explained below, it is relaxed in a Connected Theory leading to a wider class of most specific hypotheses, and thus a larger search space.

In general, for a set T of ground Horn clauses to be a hypothesis for an example E given background knowledge B, T must include a set T_1 of *root clauses* whose head atoms form an (abductive) explanation of the example E:

$$B \cup T_1^+ \models E \ .$$

If the body atoms of clauses in T_1 are all implied by B ($B \models T_1^-$), then T_1 is a hypothesis for E ($B \cup T_1 \models E$), and also a Kernel Set for B and E. On the other hand, if some body atoms in T_1 are not directly derivable from the background knowledge, T_1 is not an explanation for E. In this case, those body atoms in T_1 that are not consequences of B must themselves be explained. Such atoms are referred to as *secondary examples*, and require that T contains a set T_2 of *auxiliary clauses*, in addition to T_1. In the ground hypothesis shown for Example 1, the atom $car(focus)$ is a secondary example, and is explained by the auxiliary clause $hatchback(focus) \leftarrow doors(focus, 5)$. By analogy with T_1, the heads of the T_2 clauses must abductively explain the secondary examples:

$$B \cup T_2^+ \models T_1^- \ ,$$

and either $T_1 \cup T_2$ is now a hypothesis for E, or T_2 contains secondary examples, and T must include a further subset T_3, etc. A ground hypothesis such as T is called a *Connected Theory* for B and E, as formalised below.

Definition 6 (Connected Theory). *Let B be a Horn theory and E be a ground Horn clause such that $B \not\models E$. Let $T_1, \ldots, T_n$ be n sets of ground Horn clauses ($n \geqslant 1$), and let $T = T_1 \cup \ldots \cup T_n$. T is a* Connected Theory *for B and E if and only if (i) $B \cup T_1^+ \models E^+$, (ii) $B \cup E^- \cup T_{i+1}^+ \models T_i^-$ $(1 \leqslant i < n)$, (iii) $B \cup E^- \models T_n^-$, and (iv) $B \cup T \not\models \square$.*

A set of Horn clauses which entails a Connected Theory is said to be derivable by Connected Theory Generalisation, as formalised in Definition 7 below. Theorem 1 shows such a set of clauses to be a correct hypothesis.

Definition 7 (Connected Theory Generalisation). *Let B be a Horn theory and E be a ground Horn clause such that $B \not\models E$. A set H of Horn clauses is said to be* derivable from B and E by Connected Theory Generalisation*, denoted $B, E \vdash_{CTG} H$, if and only if there is a T such that T is a Connected Theory for B and E and $H \models T$.*

Theorem 1 (Soundness of Connected Theory Generalisation). *Let B and H be Horn theories and E be a ground Horn clause such that $B \not\models E$. If $B, E \vdash_{CTG} H$ then $B \cup H \models E$.*

Proof. By Definitions 6 and 7, H entails some Connected Theory T for B and E, and T comprises n subsets $(n > 0)$ $T_1 \ldots T_n$, such that $B \cup T_1^+ \models E^+$, and $B \cup E^- \cup T_{i+1}^+ \models T_i^-$ $(1 \leqslant i < n)$, and $B \cup E^- \models T_n^-$. Since $B \cup E^- \models T_n^-$, and $T \cup T_n^- \models T_n^+$, then $B \cup E^- \cup T \models T_n^+$ by transitivity of $\models$. Also, since $B \cup E^- \cup T_{i+1}^+ \models T_i^-$ and $T \cup T_i^- \models T_i^+$ $(1 \leqslant i < n)$, then $B \cup E^- \cup T \models T_i^+$ $(1 \leqslant i < n)$, and since $B \cup T_1^+ \models E^+$, then $B \cup E^- \cup T \models E^+$ or $B \cup T \models E$. Finally, since $H \models T$, $B \cup H \models E$. □

Thus Connected Theory Generalisation is a sound inductive learning method for Horn theories. Theorem 2, below, shows that CTG extends Kernel Set Subsumption, which itself has been shown to extend Bottom Generalisation [3].

Theorem 2 (CTG extends Kernel Set Subsumption). *Let B be a Horn theory and E be a Horn clause such that $B \not\models E$. The set of hypotheses derivable from B and E by Connected Theory Generalisation strictly includes the set of hypotheses derivable from B and E by Kernel Set Subsumption.*

Proof. The proof is in two parts. First it is shown that all hypotheses derivable by KSS are derivable by CTG. Then it is shown, by means of a counter example, that some hypotheses derivable by CTG are not derivable by KSS. Part (i). Assume $B, E \vdash_{KSS} H$ where H is a set of Horn clauses. Then H subsumes some Kernel Set K of B and E. By Definition 4, K is a set of clauses $\{A_0^j \leftarrow A_1^j, \ldots, A_{n_j}^j, 1 \leq j \leq k\}$, for some k, where $B \cup \{A_0^1, \ldots, A_0^k\} \models E^+$ and $B \cup E^- \models A_i^j$, for $1 \leq i \leq n_j$, and $1 \leq j \leq k$. Therefore $B, E \vdash_{CTG} H$ with the Connected Theory $T = \{A_0^j \leftarrow A_1^j, \ldots, A_{n_j}^j, 1 \leq j \leq k\}$. Hence $B, E \vdash_{KSS} H$ only if $B, E \vdash_{CTG} H$. Part (ii). Let $p/1$ and $q/1$ be predicates, let a and b be constants, let $B = \{q(a) \leftarrow p(b)\} \cup \{q(b)\}$ and let $E = p(a)$. The hypothesis $H = \{p(X) \leftarrow q(X)\}$

is *not* derivable by KSS since it does not θ-subsume $K = \{p(a) \leftarrow q(b)\}$, but H *is* derivable by CTG as it subsumes $T = \{p(a) \leftarrow q(a), q(b)\} \cup \{p(b) \leftarrow q(b)\}$. Hence $B, E \vdash_{CTG} H$ does not imply $B, E \vdash_{KSS} H$. □

As with both the Bottom Clause and Kernel Set, restricting a Connected Theory to Horn clause logic results in several alternative most specific hypotheses. If the Bottom Clause and the Kernel (the non-Horn formula underpinning the Kernel Set) are represented thus:

$$\bot(B, E) = A_0^1 \vee \ldots \vee A_0^m \leftarrow A_1 \wedge \ldots \wedge A_n \ ;$$
$$Ker = \Delta_1 \vee \ldots \vee \Delta_m \leftarrow A_1 \wedge \ldots \wedge A_n \ ;$$

then alternative definite Bottom Clauses $\bot_i$ (resp. Kernel Sets K) can be seen to comprise one of the atoms (resp. conjunctions of atoms) on the left of the implication, together with all the atoms to the right. All alternative most specific Connected Theories are captured by the equivalent formula

$$T_\bot = S_1 \vee \ldots \vee S_m \leftarrow A_1 \wedge \ldots \wedge A_n \ ,$$

where each S_i is an alternative *set of Horn clauses* that, together with the background knowledge, explains E^+ ($B \cup S_i \models E^+$, $1 \leq i \leq m$) and where each A_i is entailed by the background knowledge and E^- ($B \cup E^- \models A_i$, $1 \leq i \leq n$). This reflects the fact that the alternative most specific Connected Theories differ in their body atoms as well as their heads, as illustrated by Example 2.

Example 2. Find a Horn theory that explains E_2, given background knowledge B_2. Hypothesis clauses may have x, u, v or w in the head, and p, a or b in the body.

$$E_2 = x \qquad B_2 = \{a \leftarrow u, v\} \cup \{b \leftarrow v, w\} \cup \{p\}$$ ♦

There are five alternative most specific hypotheses $T_\bot$ for B_2 and E_2 as shown in Table 1. The simplest, Theory 1, is the Kernel Set. Each of the other theories in the table is also a most specific hypothesis by virtue of the inclusion of different secondary examples, and auxiliary clauses to explain them.

Since none of the different $T_\bot$ imply one another, each is the bottom element of a separate implication lattice that, taken together, form the full search space for the inductive problem. Depending on the amount of underlying structure in the data (in the case of the example, whether u, v or w have also been observed), generalisation of any of the alternative $T_\bot$ could yield the preferred hypothesis.

Table 1. All possible most specific connected theories for B_2 and E_2

Theory	1	2	3	4	5
$\boldsymbol{T_\bot}$	$x \leftarrow p$	$x \leftarrow p, a$ $u \leftarrow p$ $v \leftarrow p$	$x \leftarrow p, b$ $v \leftarrow p$ $w \leftarrow p$	$x \leftarrow p, a, b$ $u \leftarrow p, b$ $v \leftarrow p$ $w \leftarrow p$	$x \leftarrow p, a, b$ $u \leftarrow p$ $v \leftarrow p$ $w \leftarrow p, a$

4 Induction on Failure

This section describes the IoF proof procedure for computing Connected Theories. The key features of the procedure are the inclusion of secondary examples in the saturation phase, and their recursive inductive explanation.

The top level covering loop is shown in Fig. 1. The initial set of examples is divided into positive examples, E_{pos}, which should be provable from the theory at the end of $Cover$, and negative examples, E_{neg} which should not. The hypothesis language is defined by sets of head (M_h) and body (M_b) mode declarations as defined in [1] and briefly explained below. M_h, which is translated into a set A of abducible atoms, M_b and E_{neg} are constant for a given application, and so global to all procedures. The cover loop proceeds by selecting a seed example $E \in E_{pos}$, and generating a hypothesis H to explain it. H is added to B and all members of E_{pos} implied, or "covered", by the new program are removed from the set. The loop continues until E_{pos} is empty.

The goal $\leftarrow E$ is queried against the background program in *Abduce*, based on the Kakas and Mancarella (KM) abductive procedure [7], collecting any ground atoms that are needed to refute it into the set Δ, which is thus an abductive explanation for E. The members of Δ will form the heads of the root clauses. These head atoms are "saturated" by adding body atoms to them, forming $T_\perp$, a most specific Connected Theory for B and E. The *Saturate* algorithm is shown in Fig. 2. As explained in Sect. 3, in general there are multiple $T_\perp$, each alternative being computed using a restricted set of abducibles $A_{aux} \subseteq A$. The final step in computing H is to search the lattice of sets of clauses which subsume $T_\perp$. The *Search* procedure returns the preferred H according to criteria appropriate to the particular learning task. A common criterion is to select the most *compressive* hypothesis, the H with the highest ratio of positive examples covered to number of literals in the hypothesis, which does not cover negative examples. The final H chosen for a given seed example E is the one with the best measure across the entire search space defined by all $T_\perp$.

Begin Cover
given B, E_{pos}
let $A = Initialise(M_h)$
while $E_{pos} \neq \emptyset$
select a seed example $E \in E_{pos}$
let $H = \{E\}$
for each $\Delta = Abduce(E, B, A)$
for each $T_\perp = Saturate(\Delta, \emptyset, \{E\}, E_{pos} \cup \Delta, B, A_{aux})$, where $A_{aux} \subseteq A$
$H = Search(T_\perp, B, E_{pos}, H)$
$B = B \cup H$
$E_{pos} = E_{pos} - \{ex \in E_{pos} \mid B \models ex\}$
End Cover

Fig. 1. The Cover loop algorithm

Begin Saturate
1 given $\Delta, Sat, E_{sec}, Abd, B, A_{aux}$
2 let $T_\perp = \emptyset$
3 for each $\alpha \in \Delta$ and $\alpha \notin Sat$:
4 let $R = \alpha$
5 $A'_{aux} = A_{aux} - \{\alpha\}$
6 for each $\beta \in M_b$:
7 replace all placeholders in β with input terms or fresh variables
8 for each $(\theta, \Delta_\beta) = C\text{-}Abduce(\beta, B, A'_{aux}, Abd)$ and $\beta\theta \notin E_{sec}$
10 $R = R \vee \neg\beta\theta$
11 if $\Delta_\beta \neq \emptyset$
12 $E'_{sec} = E_{sec} \cup \{\beta\theta\}$
13 $Abd = Abd \cup \Delta_\beta$
14 $T_\perp = T_\perp \cup Saturate(\Delta_\beta, Sat, E'_{sec}, Abd, B, A'_{aux})$
15 $T_\perp = T_\perp \cup \{R\}$
16 $Sat = Sat \cup \{\alpha\}$
17 return $T_\perp$
End Saturate

Fig. 2. The Saturate algorithm

The inputs to a call to *Saturate* are the set Δ of ground atoms to be saturated, the set Sat of ground atoms already saturated, the set E_{sec} of secondary examples in the current branch of the computation, the set Abd of atoms abduced so far, B and A_{aux}. The output is a most specific Connected Theory $T_\perp$.

Figure 3 shows a tree depiction of the *Saturate* computation which generates Theory 4 in Example 2:

$$T_\perp = \{x \leftarrow a, b, p\} \cup \{u \leftarrow b, p\} \cup \{v \leftarrow p\} \cup \{w \leftarrow p\} \ .$$

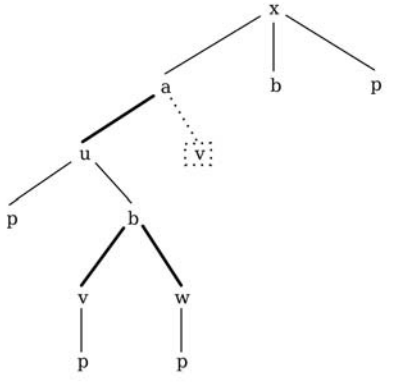

Fig. 3. A tree depiction of the saturation of $\Delta = \{x\}$ in Example 2. The branch containing the node v, and connected to a with a dotted edge, is pruned and does not form part of $T_\perp$.

Thin edges connect clause heads (parent nodes) to their body atoms (children). Bold edges connect secondary examples (parent) to clauses which explain them (subtrees). In this example the abductive explanation for the seed, x, is $\Delta = \{x\}$. Each atom $\alpha \in \Delta$ becomes the head of a new root clause R (line 4), unless it has already been saturated (line 3) in some previous iteration. *Saturate* proceeds to look for body atoms for each α that are compatible with M_b. In a first-order case each body mode declaration β is instantiated with relevant input terms (line 7) to produce a ground (if all placeholders are inputs), or partially ground atom. This atom is now queried against the background program by *C-Abduce*, which extends the KM procedure by checking "global consistency" with E_{neg} and *Abd*. If successful *C-Abduce* instantiates β with substitution θ, and returns a (possibly empty) set Δ_β of abduced atoms.

In Fig. 3 the first body atom for x to be proved by *C-Abduce* is a, with $\Delta_a = \{u, v\}$. Since Δ_a is not empty, a is a secondary example, requiring its own $T_\perp$ explanation. This new auxiliary hypothesis is induced via a recursive call to *Saturate* with input $\Delta = \{u, v\}$. This is the fundamental feature of the Induction on Failure procedure. Standard saturation procedures, using deduction rather than abduction to generate the body of clauses, would fail to prove atoms such as a and terminate. Induction on Failure instead induces new rules to explain a.

The tree also demonstrates how the procedure efficiently produces a theory which is consistent with the background knowledge and example. Two pruning steps are used. Firstly, the condition $\beta\theta \notin E_{sec}$ (line 8) prevents a secondary example appearing twice in the same branch of the tree. This, for instance allowing a to be a child of u, v or w in the figure, would correspond to the secondary example forming part of its own explanation, meaning $T_\perp$ would no longer be a correct hypothesis. The check eliminates such cyclic definitions. If the set of possible body atoms is finite, this check also guarantees that all branches of the tree terminate. Secondly, the condition $\alpha \notin Sat$ (line 3) prevents clauses being generated twice in different parts of the tree. So, even though v is part of the explanation for a in the example, the figure shows that the computation proceeds to generate the tree in a depth first manner, and a v clause is first added under b. The loop check prevents this clause being regenerated when completing the explanation of a and the second v node, shown with a dotted line, is pruned. Consequently, an abduced (head) atom can appear only once in the entire tree.

5 Imparo

Imparo, a prototype implementation of the IoF procedure, has been written in Sictus Prolog. A general-to-specific search of the hypothesis space bounded by the theory consisting of the empty clause, $\{\square\}$ and the ground $T_\perp$ theories is carried out using a branch and bound algorithm, similar to that of Aleph [2]. Coverage of each theory searched is tested using OLDT (tabled) resolution [8] to ensure that queries with a finite set of solutions will terminate.

As an illustration of the execution of the system, the result of applying Imparo to Yamamoto's main example in [6] is presented. This example is used in [6] to highlight the incompleteness of Bottom Generalisation for finding single clause hypotheses, since the clause $odd(s(X)) \leftarrow even(X)$ cannot be derived by BG.

Example 3 (Yamamoto). [1]

$$B_3 = \left\{ \begin{array}{l} even(s(X)) \text{ :- } odd(X) \\ even(0) \end{array} \right\} \cup \left\{ \begin{array}{l} nat(0) \\ nat(s(X)) \text{ :- } nat(X) \end{array} \right\}$$

$$E_3 = odd(s^3(0)), E_{pos} = \{odd(s^5(0))\}, E_{neg} = \{odd(0), odd(s^2(0))\}$$

$$M_3 = \{modeh(*, odd(+nat)), modeb(*, even(+nat)), modeb(*, +nat = s(-nat)))\}$$

♦

The $s/1$ function is handled by adding the $+nat = s(-nat)$ declaration to M_b, which generates equality body literals. Imparo returns the following output:

```
The seed example is odd(s(s(s(0))))
New most specific explanation:
---------------------------------
odd(A):-A=s(B),even(B),B=s(C),C=s(D),even(D)
odd(A):-A=s(B),even(B)
---------------------------------
Searching generalisations...
...
The most successful hypothesis is:
---------------------------------
odd(A):-A=s(B),even(B)
---------------------------------
19 nodes searched. Time taken: 0.21 seconds.
```

The most specific hypothesis generated by Imparo contains two ground clauses:

$$\left\{ \begin{array}{l} odd(s^3(0)) \leftarrow s^3(0) = s(s^2(0)), even(s^2(0)), s^2(0) = s(s^1(0)), s^1(0) = s(0), even(0) \\ odd(s^1(0)) \leftarrow s^1(0) = s(0), even(0) \end{array} \right\}$$

computed from the seed example $odd(s^3(0))$. Saturation of the abductive explanation $\Delta_1 = \{odd(s^3(0))\}$ yields five body literals including $even(0)$ which can be proved directly from the background knowledge, and $even(s^2(0))$ which is a secondary example. The other possible $even/1$ literals, $even(s^3(0))$ and $even(s(0))$ are inconsistent with the negative examples. The explanation of $even(s^2(0))$ is $\Delta_2 = \{odd(s(0))\}$. Saturation of Δ_2 produces no secondary examples and so computation of $T_\perp$ terminates. Distinct variables replace each ground term in $T_\perp$ in the program output. The single clause theory $\{odd(A) \leftarrow A = s(B), even(B)\}$

[1] E_{pos} and E_{neg} do not form part of Yamamoto's formulation of the example. They are added here only to guide the search towards choosing $odd(s(X)) \leftarrow even(X)$ as the preferred hypothesis (the simplest clause with full coverage of the examples).

is returned by the search, which is equivalent to $H_3 = odd(s(X)) \leftarrow even(X)$, in which the equality has been "flattened".

H_3 cannot be derived by BG or KSS as it does not imply the bottom clause (and kernel set) $odd(s^3(0)) \leftarrow even(0)$. However, H_3 does imply both clauses in $T_\perp$, and can be learned by Imparo. This example demonstrates the extended search space of IoF, and its suitability to mutually recursive learning tasks.

6 Related Work and Conclusion

This section relates the IoF approach to other existing IE approaches, focusing mainly on those that generalise the search space beyond that of Bottom Generalization, and summarises the contribution of this paper.

The HAIL (Hybrid Abductive Inductive Learning) proof procedure [3] is a multiple clause IE learning approach that implements the KSS semantics described in Sects. 2 and 3. CTG extends KSS as shown by Theorem 2.

The method of *CF-Induction* described by Inoue in [9] is a full clausal consequence finding approach, with a semantics based on *characteristic clauses*. The characteristic clauses, $Carc(S, \mathcal{P})$, of a program S are those that are implied by S and not subsumed by any other such clause. $\mathcal{P}$ is a so-called *production field* and specifies language bias. A hypothesis H is derivable by CF-Induction if $Carc(B \wedge \neg E, \mathcal{P}) \models CC(B, E)$ and $H \models \neg CC(B, E)$. So, the bridge formula $CC(B, E)$ is equivalent to a (negated) most specific hypothesis, and is in fact a set of ground instances of clauses in $Carc(B \wedge \neg E, \mathcal{P})$. CF-Induction is complete for full clause hypotheses. Yamamoto and Fronhöfer also report a complete multiple clause IE method in full clausal logic [10]. This approach is based on *residue hypotheses*. A residue hypothesis $Res(T)$ for a ground theory T is obtained by deleting all tautological clauses from $\overline{T}$. The procedure described in [10] generates a most specific hypothesis by taking some subset S of the ground instances of $B \cup \{\neg E\}$ and computing $Res(S)$. Since CF-Induction and the Residue method are complete for full clausal theories they are also complete for Horn theories, and so are capable of learning Connected Theories, if appropriate selection of their bridge formulas is made. In IoF, abduction is used to guide selection of the connected clauses. It is also the authors' conjecture that the IoF procedure is complete for Horn theories, with respect to the CTG semantics. Further investigation is required to formally relate the properties and performance of IoF and these full clausal systems.

In summary, this paper has proposed the notion of a Connected Theory as a new type of bridge formula for inverse entailment in Horn programs. A proof procedure, Induction on Failure, has been described which computes most specific Connected Theories via a recursive abductive algorithm. The semantics of Connected Theory Generalisation has been shown to be more complete for single clause hypotheses than the Bottom Generalisation of Progol, and more complete for multiple clause hypotheses than the Kernel Set Subsumption of HAIL. Further work on Induction on Failure will include characterisation of the completeness of the procedure, and the extension of the approach to compute normal program hypotheses.

References

1. Muggleton, S.H.: Inverse Entailment and Progol. New Generat. Comput. 13, 245–286 (1995)
2. Srinivasan, A.: The Aleph Manual, version 4 (2003), `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/index.html`
3. Ray, O., Broda, K., Russo, A.: Hybrid Abductive Inductive Learning: A Generalisation of Progol. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 311–328. Springer, Heidelberg (2003)
4. Moyle, S.A.: An Investigation into Theory Completion Techniques in Inductive Logic Programming. PhD thesis, University of Oxford (2000)
5. Muggleton, S.H., Bryant, C.H.: Theory Completion Using Inverse Entailment. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 130–146. Springer, Heidelberg (2000)
6. Yamamoto, A.: Which Hypotheses Can Be Found with Inverse Entailment? In: Džeroski, S., Lavrač, N. (eds.) ILP 1997. LNCS, vol. 1297, pp. 296–308. Springer, Heidelberg (1997)
7. Kakas, A.C., Mancarella, P.: Database Updates through Abduction. In: 16th International Conference on Very Large Databases, pp. 650–661. Morgan Kaufmann, San Francisco (1990)
8. Tamaki, H., Sato, T.: OLD Resolution with Tabulation. In: Shapiro, E. (ed.) ICLP 1986. LNCS, vol. 225, pp. 84–98. Springer, Heidelberg (1986)
9. Inoue, K.: Induction as Consequence Finding. Mach. Learn. 55, 109–135 (2004)
10. Yamamoto, A., Fronhöfer, B.: Hypotheses Finding via Residue Hypotheses with the Resolution Principle. In: Arimura, H., Sharma, A.K., Jain, S. (eds.) ALT 2000. LNCS (LNAI), vol. 1968, pp. 156–165. Springer, Heidelberg (2000)