# Induction of Stable Models

Ramón P. Otero

AI Lab. - Dept. of Computer Science
University of Corunna
15071 Corunna, Galicia, SPAIN
otero@dc.fi.udc.es

**Abstract.** In the line of previous work by S. Muggleton and C. Sakama, we extend the logical characterization of inductive logic programming, to normal logic programs under the stable models semantics. A logic program in this non-monotonic semantics can be contradictory or can have one or several models. We provide a complete characterization on the hypotheses solution to induction of this kind of programs.

## 1 Introduction

Consider the following motivating example.

*Example 1.* Given normal logic program $B$

$$p \leftarrow not\ q$$
$$q \leftarrow not\ p$$

assume we want an extension of this program, $B \cup H$, for the atoms of the set $E = \{p\}$ to be consequence of the extension, $B \cup H \models E$. Note that $B$ has two stable models $\{p\}$ and $\{q\}$. Thus there are not literals consequence of $B$. □

Consider the following solutions:

$$H_1 = \{p \leftarrow\},\ H_2 = \{p \leftarrow q\},\ H_3 = \{p \leftarrow not\ p\},\ H_4 = \{p \leftarrow not\ p, q\}.$$

Solution $H_1$ is directly a fact about the wanted atom and it can be induced by current ILP methods. Solutions $H_3$ and $H_4$ contain rules with negation as failure, thus only non-monotonic ILP could induce them.

But in fact $H_3$ and $H_4$ are not inducible using current NM-ILP methods for the following reasons.

• The program $B$ has several stable models, and, in particular, no atomic consequences. Thus the *enlarged bottom set* of E-IE [5], and the *expansion set* $M^+$ of NM-IE [8] are empty. Then there is no set from which to get possible candidates for body literals of the hypothesis $H$.

• The literal $p$ appears both in the body and in the head of $H$, that is not allowed in these methods, furthermore this makes $H_3$ alone inconsistent.

The most interesting case is solution $H_2$ because it is a positive rule, thus current ILP methods could be able to induce it. But it is not the case because

C. Rouveirol and M. Sebag (Eds.): ILP 2001, LNAI 2157, pp. 193–205, 2001.

IE is only defined for Horn logic programs and this is not the case of $B$ in the example. In other words, $q$ is not entailed by $B$ so it cannot be in the body of $H$. This last fact is also the reason why $H_2$ is not discovered by NM-IE (despite NM-IE is defined for non-Horn programs).

*Example 2.* Consider another normal logic program $B$

$$q \leftarrow not\ p$$
$$q \leftarrow not\ q$$

we want to learn $E = \{p\}$. Program $B$ has only one stable model, $\{q\}$. Thus the literals consequence of $B$ are $\{q, not\ p\}$. □

Consider the following tentative solutions:

$$H_1 = \{p \leftarrow\},\ H_2 = \{p \leftarrow q\},\ H_3 = \{p \leftarrow not\ p\},\ H_4 = \{p \leftarrow not\ p, q\}.$$

Hypothesis $H_1$ is not a solution, it makes the program $B \cup H_1$ contradictory, i.e. there is no stable model. In fact none of the four tentative hypotheses is a solution, because for any of them $B \cup H_i$ is contradictory.

These are all the hypotheses built from the literals consequence of $B$. Thus is there no solution? No, actually there are solutions. Consider $H_5 = \{p \leftarrow,\ q \leftarrow p\}$. The program $B \cup H_5 \models E$ and it is not contradictory, the unique stable model is $\{p, q\}$.

Inverse Entailment (IE) [4] and Enlarged Inverse Entailment (E-IE) by Muggleton [5], and Non-monotonic IE by Sakama [8], rely on the set of literals consequence of $B$ to define the hypothesis $H$ solution to induction.

When extending induction to nonmonotonic LP, the background knowledge $B$—containing negative literals in the rules—is no longer representable by the set of literals consequence of it. In Example 2, an alternative $B = \{q \leftarrow not\ p\}$, with the same set of consequences, $\{q, not\ p\}$, would accept $H_1$ as solution.

There is another extension made to the basic setting of ILP in these examples, namely, the predicate of the examples can be already present in the rules of $B$.

When this extension of $B$ is considered, the contribution of the background knowledge to the learning task is not only to provide facts about other predicates on which the induced rules can rely. But also to act as a constraint background knowledge that can forbid some solutions.

This fact is also present in basic ILP, e.g. the (extended) background knowledge can already entail one negative example, making induction impossible.

Part of this study is centered in the identification of the constraint effect of the background.

Finally there is another effect when $B$ is a normal program: some consequences of $B$ may not be preserved after induction. Consider $p \leftarrow not\ q$ the consequences are $\{p\}$, we want to learn $\{q\}$ then $H = \{q \leftarrow\}$ is a solution, but now the consequences of $B \cup H$ are $\{q\}$, i.e. $p$ is no longer a consequence. These are 'nonmonotonic' consequences of $B$, i.e. consequences that relied on default assumptions about an atom being false.

In the next section we recall the definition of stable models. Then induction of stable models is characterized. We conclude discussing the results and commenting on related work.

## 2 Normal Logic Programs and Stable Models

A (ground) normal logic program is a set of rules of the form

$$A_0 \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n \tag{1}$$

where $n \geq m \geq 0$, and each $A_i$ is a ground atom. If a rule or a program does not contain the *not* operator it is called *positive*.

The stable model semantics of a normal logic program [2] is defined in two steps. First let $B$ be a positive program, then the *stable models* are the minimal sets of atoms $M$ that satisfy the condition: For each rule

$$A_0 \leftarrow A_1, \ldots, A_m$$

from $B$, if $A_i \in M$, for all $i : 1, \ldots, m$, then $A_0 \in M$.

Now let $B$ be a general ground program. For any set $M$ of atoms, let $B^M$ be the program obtained from $B$ by deleting

1. each rule that has a formula *not* $A$ in its body with $A \in M$, and
2. all formulas of the form *not* $A$ in the bodies of the remaining rules.

The program $B^M$ is positive; if $M$ is the stable model of this program then $M$ is a stable model of $B$.

Note that, by the definition, for positive programs the stable model is unique and coincides with the least Herbrand model of the program. This result also holds for Horn programs, as shown in [5], when the program is not contradictory; otherwise there is no least Herbrand model, nor stable model. Even for normal programs, when the least Herbrand model exists, the stable model coincides with it (and is unique), e.g. *stratified* normal programs. The difference is for the other normal programs, for which there can be no stable model, one or several stable models, e.g. $\{p \leftarrow notp, notq\}$ does not have stable model; $\{p \leftarrow notq,\ q \leftarrow notp\}$ has two stable models, $\{p\}$ and $\{q\}$.

A program is not contradictory iff it has one or more stable models. When there are several stable models, the atoms consequence of the program are the atoms common to all the stable models.

Stable models (and least Herbrand models) are minimal models; to further differentiate a stable model from (just) a model, i.e. not necessary minimal, we will call the latter a monotonic model of the program.

A set of atoms is a *monotonic model* of a rule (1) iff whenever $M$ satisfies the body, $A_i \in M$, for all $i : 1, \ldots, m$, and $A_j \notin M$, for each $j : m+1, \ldots, n$, then $M$ satisfies the head, $A_0 \in M$.

A set of atoms $M$ is a monotonic model of a program iff it is a monotonic model of each rule of the program.

It is easy to verify that every stable model is a monotonic model of the program.

## 3 Characterization of Induction in Stable Models

In this section we propose necessary and sufficient conditions for the existence of solution to induction of normal logic programs under stable models semantics. We do the characterization in three steps. First for induction from a complete set of examples, then induction in the usual ILP setting for which the set of examples is not complete. Finally induction from several sets of examples—a new ILP setting that is relevant in stable models programming.

### 3.1 Induction from Complete Sets

Consider the particular ILP setting in which the set of examples is complete, i.e. for every ground literal of the program there is either a positive example on it or a negative example. In this case, the set of examples corresponds to one model of the program.

In this setting, the task of finding an extension of $B$ that entails the set of examples is an application of the representation theorem for LP. Instead of finding a program that has a particular set of facts as consequence, it is to find an extension of a given program $B$ that has the particular set as consequence. (Without $B$ there is a simple solution, viz., a set of fact rules, one for each positive example.) But with $B$—as mentioned before—there can be no solution, e.g. when $B$ already entails one of the negative examples.

When $B$ is a normal program its behavior as a constraint on the solutions is stronger. As show in Example 2 in the introduction, even a program $B$ that does not entail negative examples, does not accept the simple solution of a set of facts on the positive examples.

**Theorem 1. (Existence of solution, necessary condition)** *Given a normal logic program $B$, and a possible model $M$, there is no extension $H$ of $B$, such that $M$ is a stable model of $B \cup H$ if $M$ is not a monotonic model of $B$.*

*Proof.* Every stable model of a program is a monotonic model of it. The addition of more formulas $H$ to a given program $B$ only deletes monotonic models of $B$ (they have to satisfy $H$ also.) Thus if $M$ is not a monotonic model of $B$ the effect of adding more rules will not recover $M$ as monotonic model. □

Note that if we change monotonic model by stable model in the previous proof, the facts do not hold, we would be in a nonmonotonic formalism.

Given a complete set of examples $E = \{a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m\}$, it directly corresponds to one possible model $M$ denoted $\{a_1, \ldots, a_n\}$ of the program $B$. Thus if there is a solution $H$, $B \cup H \models E$, then the set $E$ considered as model, $M$, is a monotonic model of $B$.

Next we will show that the converse of Theorem 1 also holds, providing a complete characterization on the existence of solution to induction problems for a complete set of examples, under the stable model semantics.

**Theorem 2. (Existence of solution, sufficient condition)** *Given a normal logic program $B$, and a possible model $M$, there is an extension $H$ of $B$, such that $M$ is an stable model of $B \cup H$ if $M$ is a monotonic model of $B$.*

*Proof.* We will construct an $H$ that is solution. Consider $H = \{a_i \leftarrow | a_i \in M\}$ a set of fact rules corresponding to each of the positive atoms $a_i \in M$. Note that, by construction, $M$ is a monotonic model of $H$. (Each of the rules $a_i \leftarrow$ constructed from $M$ is satisfied by it.) As $M$ is a monotonic model of $B$, then it is a monotonic model of $B \cup H$.

Then we will verify that it is stable. Consider the reduct $(B \cup H)^M = B^M \cup H^M = B^M \cup H$ because $H$ is a positive program. $M$ is a stable model iff it is the minimal model of the positive program $B^M \cup H$. As $M$ is a monotonic model of $B$ it is a monotonic model of $B^M$ (the reduct $B^M$ has a subset of the rules of $B$, and for the remaining rules the negative literals deleted are satisfied by $M$). Assume $M$ is not minimal, then there is another (monotonic) model $M'$ of $B^M \cup H$ such that $M' \subset M$, then there is one atom, assume it is $a_k$, $a_k \in M$, $a_k \notin M'$. By construction of $H$ there is one fact rule $a_k \leftarrow$. Then $M'$ is not a monotonic model of this fact rule, thus it is not a monotonic model of $B^M \cup H$. □

In fact, there are other $H$ solutions when $M$ is a monotonic model of $B$. But the $H$ formed with facts is always a solution.

*Example 2 (cont.)* Theorem 2 may seem surprising if we recall Example 2 in the introduction. In that example it is shown that the simple $H$ formed with facts is not a solution, and, nevertheless, there are other solutions. The key observation is that $\{p\}$ in Example 2 is not a monotonic model of $B$. Thus this model, that entails the example set $E = \{p\}$ is not valid. The solution shown corresponds to the model $\{p, q\}$; and this do is a monotonic model of $B$.

There is a solution in Example 2 because $E$ is not considered a complete set (in the sense that *not* $q$ is not in the set). (If the case were that the example set $\{p\}$ is complete, i.e. $\{p, not\ q\}$, then by the previous result we would conclude that there is no solution.) □

The previous characterization of the existence of solution for induction of stable models, needs to detect whether a model is a monotonic model of a program or not.

Theoretically a model is a monotonic model of a program iff the model satisfies the program. Thus the model corresponding to the complete set of examples can be tested for satisfiability, constituting an implementation of induction in stable models.

Alternatively, the following result can be used to verify that a set is a monotonic model. We will use the same name to denote a set of atoms $M = \{a_1, \ldots, a_n\}$, and a set of fact rules on the atoms of the set, $M = \{a_1 \leftarrow , \ldots, a_n \leftarrow\}$.

**Proposition 1.** *Given a normal logic program $B$, $M$ is monotonic model of $B$ iff $M$ is a stable model of $B \cup M$.*

*Proof.* Consider the program $B \cup M$. If $M$ is a stable model of it then it is a monotonic model of $B \cup M$. Thus it has to be a monotonic model of both $B$ and $M$.

Proof in the other direction is similar to that of Theorem 2. □

Proposition 1 identifies monotonic models $M$ with those that verify they are stable models of $B \cup M$. In fact, in the conditions of Proposition 1, $M$ is the unique stable model of $B \cup M$. (Any other stable model $M'$ of $B \cup M$, will satisfy $M$, thus $M \subset M'$ and then $M'$ is not stable.)

System *smodels* by Niemela et al. [6] is a sound and complete implementation to find the stable models of a normal program.

The system smodels can be used to induce normal programs under the stable models semantics for the setting of complete sets of examples. Just consider the example set $E$ as a possible model $M$, try program $B \cup M$, if $M$ is a stable model of it, then there is a solution to induction, e.g. $H = M$ is a solution. If $M$ is not a stable model of it, then there is no solution.

*Example 1 and 2 (cont.)* Recall Example 1 in the introduction. Consider $B \cup E = \{p \leftarrow \mathit{not}\ q,\ q \leftarrow \mathit{not}\ p,\ p \leftarrow\}$ the stable model is $\{p\}$. Then $H = \{p \leftarrow\}$ is a solution.

For Example 2 consider $B \cup E = \{q \leftarrow \mathit{not}\ q,\ q \leftarrow \mathit{not}\ p,\ p \leftarrow\}$, there is no stable model. Then there is no solution.

Instead consider the examples set $E' = \{p, q\}$, $B \cup E'$ has $E'$ as stable model, then $H = \{p \leftarrow,\ q \leftarrow\}$ is a solution. □

### 3.2 Induction from Non-complete Sets

Consider that the set of examples is not complete—the usual ILP setting. The definition of solution to an induction problem is as follows.

Given the two parts of the set of examples $E = E^+ \cup E^-$, i.e. the positive examples $E^+$ and the negative examples $E^-$, there is a solution $H$, in the presence of background knowledge $B$, iff $B \cup H \models E^+$, $B \cup H \not\models E^-$, and $B \cup H \not\models \bot$.

**Definition 1 (Complete extension)** *Given a set of examples $E = E^+ \cup E^-$, an interpretation $M$ (of $B \cup E$) is a complete extension of $E$ iff $E^+ \subset M$ and $M \cap E^- = \emptyset$.* □

The following result identifies the existence of solution using the results for the case of complete set of examples.

**Theorem 3. (Existence of solution)** *Given a normal logic program $B$, and a set of examples $E = E^+ \cup E^-$ there is a solution $H$ to induction iff there is (at least) one complete extension $M$ of $E$ that is a monotonic model of $B$.*

*Proof.* If there is a solution then the stable model $M$ of $B \cup H$ exists (because $B \cup H \not\models \bot$), thus it is a monotonic model of $B$. Furthermore, $M$ is a complete extension of $E$, because $B \cup H \models E^+$ thus $M \models E^+$ ($E^+ \subset M$), and $B \cup H \not\models E^-$,

thus $M \not\models E^-$ ($M \cap E^- = \emptyset$). (Note that the last relation means that $M \not\models e_j^-$ for every $e_j^- \in E^-$.)

If there is a monotonic model $M$ of $B$ that is a complete extension of $E$, then by Theorem 2 there is an extension $H = M$ such that $M$ is stable model of $B \cup H$. Then $B \cup H \not\models \bot$. As $M$ is a complete extension of $E$, $M \models E^+$ and $M \not\models E^-$. Thus $B \cup H \models E^+$ and $B \cup H \not\models E^-$. (Recall that, in fact, $M$ is the unique stable model of $B \cup M$.) □

For an alternative characterization as we did for the complete set case, we can recall also those results here. Then from the previous theorem and Proposition 1, we get the following.

**Corollary 1.** *Given a normal logic program $B$, and a set of examples $E = E^+ \cup E^-$ there is a solution $H$ to induction iff there is (at least) one complete extension $M$ of $E$ that is a stable model of $B \cup M$.* □

¿From an implementation point of view now it is needed to search the extensions of the set of examples for a complete set that is stable of itself added to $B$.

The direct implementation is to call several times the system smodels with $B$ and one of the complete extensions until one of them has itself as stable model. (If none of them are its own stable model then there is no solution.)

Note that in this setting we have a choice on the possible solutions. Several extensions of the set of examples can have solution. This is the usual choice in induction from the most specific solution to the most general solution.

In this setting, the search can be reduced using the following result.

**Proposition 2.** *Given a normal logic program $B$, $M'$ is monotonic model of $B$ if $M'$ is a stable model of $B \cup M$ and $M \subseteq M'$.*

*Proof.* Consider we add some atoms of $M'$, $M \subset M'$ to $B$, and $M'$ is a stable model of $B \cup M$. Then $M'$ is a monotonic model of $B \cup M$, thus it is also a monotonic model of $B$. □

Consider a set of examples $E = E^+ \cup E^-$. We extract from $E$ the subset of positive examples $M = E^+$. Sometimes $B \cup M$ does not have $M$ as stable model but some superset $M'$ as stable model. Then if $M'$ is a complete extension of $E$, there is solution to induction.

Even when $B \cup M$ does not have stable model at all, there can be a solution to induction. The situation in nonmonotonic induction is that $B \cup E^+$ can be contradictory and still an extension of $E^+$ provide a consistent extension for $B$. Thus we have to search for a consistent extension of $B$, and among all these extensions we can choose following particular generalization criteria.

Note that these results not only characterize the existence of solution to induction, but every $H$ solution to induction. There is a particular $H$ solution iff the model of $B \cup H$ is a monotonic model of $B$ and a complete extension of $E$.

In summary, there are three kinds of solutions in this setting of normal logic programs.

- *Minimally extended.* The $H$ that are facts on the positive examples *minimally extended* to avoid contradiction with $B$.
- *Generalizations.* The $H$ that are *generalizations* of the minimal ones, thus implying more atoms. These solutions can be constructed by just adding more fact rules to the minimal ones, or—as usual in ILP—by first-order generalization. But not every extension is solution, these extended $H$ have to verify the conditions of the characterization, in particular, the monotonic model condition (appart from the usual condition on complete extension).
- *Nonmonotonic.* There is a new kind of $H$ solution in this setting (also present in NM-IE [8]), viz., $H$ that use negation as failure, let us call them *nonmonotonic* hypotheses.

The nonmonotonic hypotheses do not really constitute solutions more specific than the minimally extended ones.

Furthermore, these nonmonotonic solutions have a property that is usually non-intended: the examples learned are not necessarily preserved after further induction (induction in several steps, or multiple predicate learning). The nonmonotonic behavior of $B \cup H$ is stronger than the one with minimally extended or generalized hypotheses, because part of the examples entailed by $B \cup H$ might rely on default assumptions.

Consider that we want to further extend $B \cup H$ with $H'$ to cover additional examples. If the task is performed by considering $B' = B \cup H$ and applying the basic procedure to arrive to $B' \cup H'$, then some of the previous examples covered by $H$ can become uncovered after the addition of $H'$. (The coverage of the previous examples has been done nonmonotonically, thus they are not necessarily entailed after any addition of more rules to the program.) On the other hand, any $H$ composed of fact rules, entails the examples monotonically, thus this situation cannot arise. (What can happen is the alternative situation, that some of the negative examples are covered after the addition of $H'$, but this is a well known fact already in ILP for Horn theories.)

**Induction from Non-complete Sets under Background Horn Theories.** So far we have shown that solutions composed of a collection of fact rules characterize the existence of solution in ILP for normal logic programs.

The extension made here only points out that precisely the set of facts from the positive examples does not need to be such a solution. Nevertheless, there is a restriction that precisely characterizes the existence of solution with the set of facts from the positive examples.

Consider that the background knowledge is a Horn theory, i.e. definite clauses and goal clauses (constraints), thus no clause contains the *not* operator.

Then the following result holds.

**Theorem 4. (Existence of solution)** *Given a Horn logic program $B$, and a (consistent) set of examples $E = E^+ \cup E^-$ there is a solution $H$ to induction iff $H = E^+$ is a solution.*

*Proof.* We only need to prove one direction. Obviously if $H = E^+$ is a solution, there is solution.

Assume there is a solution $H'$. Then we will verify that $H = E^+$ is also a solution. We will use the monotonic properties of Horn logic programs.

Consider that the stable model of $B \cup E^+$ exists, thus $B \cup E^+ \not\models \bot$. (Furthermore it is unique ( [5], Lemma 1) and coincides with the Least Herbrand model.) Now recall that Horn programs verify monotonic properties, thus $B \cup E^+ \models E^+$, simply because $E^+ \models E^+$. Finally $B \cup E^+ \not\models E^-$ because there is a solution $H'$ such that $B \cup H' \not\models E^-$ and $B \cup H' \models E^+$, thus we can add the consequences $B \cup H' \cup E^+ \not\models E^-$, and remove the hypothesis $B \cup E^+ \not\models E^-$.

Consider that there is no stable model of $B \cup E^+$. But if there is another solution, $B \cup H' \models E^+$ and $B \cup H' \not\models \bot$. Applying the monotonic properties of Horn programs, $B \cup H' \cup E^+ \models E^+$, thus $B \cup E^+ \not\models \bot$. Then there is a stable model of $B \cup E^+$ if there is any solution $H'$. □

### 3.3 Induction from Several Sets of Examples

Under stable models semantics, normal logic programs constitute a new declarative programming paradigm. The idea relies on the fact that logic programs can have no stable model, one or several stable models.

Each stable model is associated with one (alternative) solution to the problem described by the program. Thus when there are several stable models, the problem has several solutions; and when there is no stable model, the problem does not have a solution.

Typical problems of this kind are combinatorial problems, e.g. finding the different ways the nodes of a graph can be colored verifying that no adjacent nodes have the same color. Other typical examples are planning problems, i.e. finding the sequence of actions that lead to a given goal state from a given initial state of the domain.

For these kind of applications of stable models programming, induction would be welcomed. To this end the usual setting of ILP has to be extended. The direct extension is to consider several sets of examples. Each one corresponding to an intended solution to the problem.

**Definition 2 (Induction of (several) stable models)** *Given a logic program $B$, and several sets of examples $E_1, \ldots, E_n$ (each one composed of two parts, $E_i = E_i^+ \cup E_i^-$) there is a solution program $H$ to induction iff for each set $E_i$, $i : 1, \ldots, n$ there is a stable model, $M_i$ of $B \cup H$, such that $M_i \models E_i^+$ and $M_i \not\models E_i^-$ ($M_i \cap E_i^- = \emptyset$).* □

Note that the usual definition of induction in ILP is the particular case for a unique set of examples.

The previous results still worth to characterize induction of several stable models. Before we will need the concept of antichain and a result about it (similar to one by V. Marek and M. Truszczynski in [3]).

A collection of sets of atoms, $M_1, \ldots, M_n$ form an *antichain* iff whenever $M_i \subseteq M_j$ then $M_i = M_j$, for every $i, j : 1, \ldots, n$. Thus no set is subset of another set in the collection.

**Proposition 3.** *Given a normal logic program $B$, and a collection of monotonic models $\{M_i, i : 1, \ldots, n\}$ of $B$ that form an antichain, then there is an extension $H$ such that $B \cup H$ has the models $M_i, i : 1, \ldots, n$ as stable models (simultaneously).*

*Proof.* We have to propose a set of rules $H$ that when added to $B$ make all the $M_i, i : 1, \ldots, n$ stable.

One possibility is to add an $H_i$ to make each $M_i$ stable. But each of these $H_i$ has to be carefully chosen not to forbid the other intended stable $M_j$. This will be achieved if $H_i$ is able to make $M_i$ stable, while keeping all the monotonic models of $B$ (unless subsets of $M_i$). Then we have to add to $B$ rules that are not satisfied only by subsets of $M_i$ but any other set (not subset of $M_i$) will satisfy the rules in $H_i$.

Consider $H_i = \{a_l \leftarrow NB \mid a_l \in M_i\}$ where $NB = not\ b_1, \ldots, not\ b_m$ for all $b_j \notin M_i$. Each $H_i$ is a set of rules, one for each positive atom in $M_i$ as head, and the same body for all of them, the conjunction of the negative literals for the atoms not in $M_i$. Then every subset of $M_i$ does not have any $b_j$ and also does not have some $a_k$ positive in $M_i$. Thus it does not verify the rule in $H_i$ corresponding to $a_k$.

For any other model that is not a subset of $M_i$ then there is an $a_r$ that is not in $M_i$ thus the model does not verify $not\ a_r$ that is in every body of the rules of $H_i$, thus the model satisfies all the rules of $H_i$. Then $H_i$ does not delete any other model—unless subsets of $M_i$.

We show that $H_i$ makes $M_i$ stable. Consider the reduct $(B \cup H_i)^{M_i}$. It is equal to $B^{M_i} \cup M_i$, because $H_i^{M_i} = M_i$ (all the $b_j$ are not in $M_i$ thus all those literals are deleted in the reduct $H_i^{M_i}$, but the rules are kept as fact rules.) (From this point the proof of Theorem 2 can be directly followed.) As $M_i$ is a monotonic model of $B$, it is a monotonic model of $B^{M_i}$. By Proposition 1 it is a stable model of $B^{M_i} \cup M_i$. Thus $M_i$ is stable of $B \cup H_i$.

Finally the addition of the other $H_j$ to $B$ (for the other stable) do not interfere with each other if the $\{M_i, i : 1, \ldots, n\}$ collection form an antichain.

Recall—as mentioned above—that any other model $M_j$ that is not a subset of a given $M_i$, satisfy the rules in $H_i$ because there is one $a_r \in M_j$ that is present in the body of the rules of $H_i$ as $not\ a_r$. Thus $H_i^{M_j} = \emptyset$ for every $i, j : 1, \ldots, n$. Then the reduct $(B \cup H)^{M_i} = B^{M_i} \cup H_1^{M_i} \cup \ldots \cup H_n^{M_i} = B^{M_i} \cup H_i^{M_i}$. Thus every $M_i$ is a stable model of $B \cup H$. □

**Theorem 5. (Existence of solution)** *Given a normal logic program $B$, and several sets of examples $E_1, \ldots, E_n$ there is a solution $H$ to induction iff*

*i) for each set $E_i$ there is (at least) one complete extension $M_i$ of $E_i$ that is a monotonic model of $B$, and*

*ii) the set of complete extensions* $\{M_i, i : 1, \ldots, n\}$ *form an antichain.*

*Proof.* The antichain condition is needed because the collection of stable models of a program always form an antichain. (Recall that if a model is stable, no subset of it is (simultaneously) stable.)

The proof the this theorem follows that of Theorem 3.

If there is a solution $H$ then $B \cup H$ has a collection of stable models $\{M_i, i : 1, \ldots, n\}$, thus they form an antichain. Furthermore each $M_i$ is a monotonic model of $B$. Finally, for each $E_i$ there is an $M_i$ that is a complete extension of it, because $M_i \models E_i^+$ $(E_i^+ \subset M_i)$, and $M_i \not\models E_i^-$ $(M_i \cap E_i^- = \emptyset)$.

If there is a collection of monotonic models $\{M_i, i : 1, \ldots, n\}$ of $B$ that form an antichain, then by Proposition 3 there is an extension $H$ such that $B \cup H$ has the models $\{M_i, i : 1, \ldots, n\}$ as stable models (simultaneously).

As each $E_i$ has one $M_i$ that is a complete extension of it, $M_i \models E_i^+$ and $M_i \not\models E_i^-$. □

Note that the solution $H$ with only facts is not, in general, a solution for several sets of examples, as it was in the other settings.

This result shows that nonmonotonic hypotheses (i.e. with negation as failure in the body) are only truly needed when there are several sets of examples.

As in the other settings, the existence of solution does not mean that $H$ has to be just in the form we used for the proofs. Other solutions can exists, as we showed before, but recall that only when we are in the conditions of the result presented. In this sense, the collection of $\{H_i, i : 1, \ldots, n\}$ proposed can be thought of the *most specific* solution to the problem, and also the *most conservative* solution (in the sense that it keeps as many monotonic models of the extended program as possible).

For an implementation point of view, we can use the results on the other settings. Notice that induction from several sets can be made separately for each set, thus as a case of induction from a non-complete set. The only difference is that instead of using $H = M$ a set of facts, we have to test with the $H_i$ rules, $H_i = \{a_l \leftarrow not\ b_1, \ldots, not\ b_m \mid a_l \in M_i\}$, where $b_1, \ldots, b_m$ are all the $b_j \notin M_i$.

*Example 3.* Given normal logic program $B$

$$p \leftarrow not\ q$$

Assume we want an extension of this program, $B \cup H$, for the atoms of the sets $E_1 = \{p\}$ and $E_2 = \{q\}$ to be consequence of corresponding stable models of the extension. Note that $B$ has one stable model $\{p\}$.

Consider the complete extensions $M_1 = E_1^+$ and $M_2 = E_2^+$. They form an antichain collection. They are monotonic models of $B$. Thus there is solution. Build $H_1 = \{p \leftarrow not\ q\}$ and $H_2 = \{q \leftarrow not\ p\}$. Then $B \cup H_1 \cup H_2 = \{p \leftarrow not\ q,\ q \leftarrow not\ p\}$ indeed has $M_1$ and $M_2$ as stable models. □

*Example 4.* Consider a simple graph with two nodes $n(1)$, $n(2)$, connected by an arc $a(1, 2)$. We want to find the different ways the nodes of a graph can be

colored with two colors, we will represent $w(X)$ as white, being black all the other nodes for which $w(X)$ is false.

Background knowledge $B$ is the graph, and the undirected condition of the graph

$$n(1) \leftarrow$$
$$n(2) \leftarrow$$
$$a(1,2) \leftarrow$$
$$a(X,Y) \leftarrow a(Y,X), n(X), n(Y)$$

The sets of examples inform on possible solutions, $E_1 = \{w(1), not\, w(2)\}$ and $E_2 = \{w(2)\}$. Notice that the union $E_1 \cup E_2$ is contradictory. And also that $B$ has a unique stable model $M_B = \{n(1), n(2), a(1,2), a(2,1)\}$.

Consider the complete extensions $M_1 = E_1^+ = \{w(1)\}$ and $M_2 = E_2^+ = \{w(2)\}$. They form an antichain collection. But they are not monotonic models of $B$. Consider the complete extensions $M_1 = E_1^+ \cup M_B = \{w(1), n(1), n(2), a(1,2), a(2,1)\}$ and $M_2 = E_2^+ \cup M_B = \{w(2), n(1), n(2), a(1,2), a(2,1)\}$. They form an antichain collection and they are monotonic models of $B$. Thus there is solution.

Build $H_1 = \{w(1) \leftarrow not\ w(2)\}$ and $H_2 = \{w(2) \leftarrow not\ w(1)\}$. (We are considering here only the $H$ with head the predicate of the examples, the other can also be added to $B$.) Then $B \cup H_1 \cup H_2$ indeed has $M_1$ and $M_2$ as stable models.

There are other solutions. Consider for instance $H_1' = \{w(1) \leftarrow not\ w(2), a(1,2)\}$ and $H_2' = \{w(2) \leftarrow not\ w(1), a(2,1)\}$. (Both can be generalized to $H'' = \{w(X) \leftarrow not\ w(Y), a(X,Y)\}$.) They correspond to the same monotonic models as $H_1$ and $H_2$. We just added one of the monotonic consequences of $B$ to the body of the hypotheses, an addition that can always be made without affecting the stable models of a program. □

## 4 Discussion and Related Work

The results shown do actually apply to other ILP settings, as far as they use LP semantics for which stable models is a conservative extension.

This characterization can be understood as a basis on which alternative techniques for induction can be defined. For example, it would be interesting to find more efficient characterizations, in the line E-IE [5] or NM-IE [8] work on other settings, to reduce the search for solutions. Besides most of the work on ILP to identify the most general solution, or other criteria for preferred solution, will be worth in this new domain.

This characterization extends the proposal of NM-IE [8], characterizing induction, in general, for normal logic programs, including, e.g. from contradictory background knowledge, contradictory hypothesis. It also clarifies some of the results in [7] and [1].

Furthermore we identify necessary and sufficient conditions for the existence of solution to induction in normal programs. Recall for instance that the conditions of NM-IE hold on Example 2, $B$ has a unique stable, $H$ has a unique

stable, including the $H$ solution. But there is no $H$ by the theoretical method (neither by the algorithm) in NM-IE because all of them lead to contradictory $B \cup H$. (Consider $B \cup \{not\ L\} = B \cup \{not\ p \leftarrow\}$ there is only one stable (the same as for $B$), $\{q\}$. The rules this stable is counter-model are the four tentative solutions in that example.)

Finally, induction from several sets of examples is defined and characterized.

**Acknowledgements**

This research is partially supported by Government of Spain grant PB97-0228.

## References

1. M. Bain and S. Muggleton. Nonmonotonic learning. In S. Muggleton, editor, *Inductive Logic Programming*, pages 145–161. Academic Press, 1992.
2. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.
3. W. Marek and M. Truszczynski. *Nonmonotonic Logic – Context-Dependent Reasoning.* Series Artificial Intelligence, Springer-Verlag, 1993.
4. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
5. S. Muggleton. Completing inverse entailment. In *Proc. of the 8th International Workshop on Inductive Logic Programming, ILP 98, LNAI 1446*, pages 245–249, 1998.
6. Ilkka Niemelä and Patrick Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proc. of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 97, LNAI 1265*, pages 420–429, 1997.
7. C. Sakama. Some properties of inverse resolution in normal logic programs. In *Proc. of the 9th International Workshop on Inductive Logic Programming, ILP 99, LNAI 1634*, pages 279–290, 1999.
8. C. Sakama. Inverse entailment in nonmonotonic logic programs. In *Proc. of the 10th International Conference on Inductive Logic Programming, ILP 00, LNAI 1866*, pages 209–224, 2000.