# Abductive Concept Learning

Antonis C. Kakas

*Department of Computer Science, University of Cyprus*
*75 Kallipoleos str., CY-1678 Nicosia, Cyprus*

antonis@ucy.ac.cy

Fabrizio Riguzzi

*DEIS, Università di Bologna*
*Viale Risorgimento 2, 40136 Bologna, Italy*

friguzzi@deis.unibo.it

Received 21 September 1998

Revised 16 March 1999

***Abstract***

We investigate how abduction and induction can be integrated into a common learning framework. In particular, we consider an extension of Inductive Logic Programming (ILP) for the case in which both the background and the target theories are abductive logic programs and where an abductive notion of entailment is used as the basic coverage relation for learning. This extended learning framework has been called Abductive Concept Learning (ACL). In this framework, it is possible to learn with incomplete background information about the training examples by exploiting the hypothetical reasoning of abduction. We also study how the ACL framework can be used as a basis for multiple predicate learning.

An algorithm for ACL is developed by suitably extending the top-down ILP method: the deductive proof procedure of Logic Programming is replaced by an abductive proof procedure for Abductive Logic Programming. This algorithm also incorporates a phase for learning integrity constraints by suitably employing a system that learns from interpretations like ICL. The framework of ACL thus integrates the two ILP settings of explanatory (predictive) learning and confirmatory (descriptive) learning. The above algorithm has been implemented into a system also called ACL*[1] Several experiments have been performed that show the effectiveness of the ACL framework in learning from incomplete data and its appropriate use for multiple predicate learning.

**Keywords** Machine Learning, Inductive Logic Programming, Abductive Logic Programming, Non-Monotonic Reasoning.

## §1 Introduction and Motivation

The problem of integrating abduction and induction in Machine Learning systems has recently received renewed attention with several works on this topic [3, 2, 22, 10, 23, 30]. In [22] the notion of Abductive Concept Learning (ACL) was proposed as a learning framework based on an integration of Inductive Logic Programming (ILP) [57, 59] and Abductive Logic Programming (ALP) [40].

Abductive Concept Learning is an extension of ILP that allows us to learn abductive logic programs with abduction playing a central role in the covering relation of the learning problem. The abductive logic programs learned in ACL contain both rules for the concept(s) to be learned as well as general clauses called integrity constraints. These two parts are put together in a non-trivial way via the abductive reasoning of ALP which is then used as the basic covering relation for learning.

This paper presents the basic framework of ACL with its main characteristics and demonstrates its suitability for addressing several problems in ILP. The main motivation for developing ACL is to allow us to learn from incomplete information and to later be able to classify new cases that again could be incompletely specified. ACL provides a principled way to handle incomplete information in learning based on an underlying theory of abduction for knowledge representation. Indeed abduction is well-suited for representing problems with incomplete information (see e.g. [63, 43, 19, 34, 36, 41]) able to formulate a variety of such problems in Artificial Intelligence and other areas of Computer Science.

The central problem of learning abductive theories in ACL contains several useful and interesting subproblems that are of practical relevance. These

*1 The learning systems developed in this work together with sample experimental data can be found at the following address: `http://www-lia.deis.unibo.it/Software/ACL/`

problems include: (i) concept learning from incomplete background data where some of the background predicates are incompletely specified and (ii) concept learning from incomplete background data together with given integrity constraints that provide some information on the incompleteness of the data. In these cases, the treatment of incompleteness through abduction is integrated within the ILP learning process. This allows the possibility of learning more compact theories that can alleviate the problem of overfitting due to the incompleteness in the data. A specific subcase of these two problems and important third subproblem is that of (iii) multiple predicate learning, where each predicate is required to be learned from the incomplete data for the other predicates. Here the abductive reasoning can be used to suitably connect and integrate the learning of the different predicates. This can help to overcome some of the non-locality difficulties of multiple predicate learning, such as order-dependence and global consistency of the learned theory.

These subproblems of the full ACL task can be captured in a simpler subproblem of ACL, which we will call ACL1. Within ACL1 we learn only the rule part of an abductive theory but this in many cases is sufficiently general to allow us to address interesting problems as those described above. Apart from its practical relevance, the identification of the ACL1 subproblem is also useful in breaking the full ACL learning task into two separate but strongly inter-related phases of ACL1 and ACL2. ACL1 together with its rules also provides additional input, through abducible assumptions (which are related to the learned rules), to the second phase of ACL2 for learning integrity constraints that can (partly) confirm the correctness of these abducible assumptions. In this way, ACL synthesizes together the two main learning settings of ILP, namely those of explanatory (predictive) learning [58, 59] and confirmatory (descriptive) learning [16, 27].

An algorithm for ACL based on this separation into ACL1 and ACL2 is given. Within ACL1, this algorithm adapts the basic top-down method of ILP to deal with the incompleteness of information and to take into account the use of integrity constraints. It incorporates an abductive proof procedure and other abductive reasoning mechanisms from ALP that are suitably adapted for the context of learning. In the second phase of ACL2, the algorithm takes as input the output of ACL1 and calls on the ICL [18] learner to generate appropriate integrity constraints.

This algorithm has been implemented in a new ILP system also called

ACL and ACL1 for its subsystem. Based on these, a separated system for multiple predicate learning, called M-ACL, has been developed. Suitably adapted heuristics have been used that take into account the incompleteness of information. Several experiments have been carried out to test the ability of ACL to learn under incomplete information and to compare it with other systems such as FOIL, c4.5 and mFOIL that can learn in the face of missing information. These include experiments with data from the UCI repository and experiments with data from market research questionnaires where the available data can be incomplete for several reasons. We also present a number of experiments for multiple predicate learning with M-ACL and again compare with the MPL system of [17)]. The comparable and in some cases (marginally) better performance of ACL in these "proof of the principle" experiments demonstrate its ability to learn with incomplete information and its appropriate use for multiple predicate learning.

The motivating problem for ACL of learning under incomplete or missing information (from the background knowledge) in an ILP framework has received relatively little attention. Some exceptions to this include the recent works of ICL-Sat [15)] which learns from incomplete interpretations and [47)] which follows an approach similar to ours for learning the rules of an abductive theory. There are also several works, e.g. [24, 53)], that deal with the related problem of noise in the learning data but this is a different problem where the methods used can not always be applied as effectively to missing information. Another related problem is that of learning from incomplete or sparse training data particularly in the context of learning recursive definitions. The systems FORCE2 [11)], SKILit [39)], CHILLIN [74)] and FOIL-I [38)] were designed to handle this problem using intensional coverage techniques where the definition learned for the target predicate is used for evaluating recursive calls.

Most of the machine learning systems that deal with incomplete information are attribute-value learners. An ILP system for learning with incomplete information is LINUS [54)] but again this essentially relies on an attribute value representation. In general, these systems adopt different methods to first complete the missing information and then learn from the completed data. In contrast, in ACL the incomplete information is handled dynamically *within the learning process* in a principled way based on an underlying theory of abduction. In this way it combines in a non-trivial way the methods of abduction for dealing with incomplete information with methods of ILP learning.

The work of this paper develops further and completes preliminary work on the general topic of learning with abduction in [45]. It also extends and complements earlier work in [22, 26, 50, 51]. These previous works have addressed specific aspects of the general problem demonstrating through simple examples the potential of abduction in addressing various interesting problems. The current work provides a firm theoretical and algorithmic basis for the use of abduction in learning together with a thorough empirical study on non-trivial experimental data. It thus confirms and establishes the utility of abduction in ILP.

The rest of this paper is organized as follows. Section 2 presents a short review of ALP needed for the formulation and description of the main properties of ACL which are presented in section 3. Section 4 presents the basic algorithm for ACL and its properties for the single predicate case, while section 5 describes the application of ACL to multiple predicate learning. Section 6 presents our experiments with ACL, section 7 discusses related work and section 8 concludes the paper.

## §2 Abductive Logic Programming

In this section we briefly review some of the elements of Abductive Logic Programming (ALP) needed for the formulation of the learning framework of Abductive Concept Learning (ACL). For a more detailed presentation of ALP the reader is referred to the survey [40] (and its recent update [41]) and references therein.

Abductive Logic Programming is an extension of Logic Programming to support abductive reasoning with theories (logic programs) that incompletely describe their problem domain. In ALP this incomplete knowledge is captured (represented) by an abductive theory $T$. We will consider abductive theories of the following form.

**Definition 2.1 (Abductive theory)**
An **abductive theory T** in ALP is a triple $\langle P, A, I\rangle$, where $P$ is a definite logic program, $A$ is a set of predicates called **abducible predicates** (or simply **abducibles**), and $I$ is a set of range-restricted clauses called **integrity constraints**.

For simplicity of presentation we will assume that the logic program $P$ of an abductive theory is a definite Horn program with no negation (negation as failure) appearing in the body of the rules of $P$. However, this condition is not restrictive since negation as failure in a logic program can be treated through abduction in an associated abductive theory whose program is definite [25)].

As a knowledge representation framework, when we represent a problem in ALP via an abductive theory $T$, we generally assume that the abducible predicates in $A$ carry all the incompleteness of the program $P$ in modelling the external problem domain in the sense that if we (could) complete the abducible predicates in $P$ then $P$ would completely describe the problem domain.

An abductive theory can support abductive (or hypothetical) reasoning for several purposes such as diagnosis, planning or default reasoning. The central notion used for this is that of an *abductive explanation* for an observation or a given goal. To formalize this we need the notion of *generalized model* of an abductive theory introduced in [43)]. A generalized model (or generalized stable model as called in [43)]) is a minimal Herbrand model of the program but where the abducible predicates need not be minimized.

**Definition 2.2 (Generalized model)**
Let $T = \langle P, A, I \rangle$ be an abductive theory and $\Delta$ a set of ground abducible facts from $A$. $M(\Delta)$ is a **generalized model** of $T$ iff

- $M(\Delta)$ is the minimal Herbrand model of $P \cup \Delta$, and
- $M(\Delta)$ is a model of $I$, i.e., $M(\Delta) \models I$

We say that $\Delta$ is an **abductive extension** of $T$.

Here the semantics of integrity constraints is defined by the second condition in the definition. Their satisfaction requires that they are true statements in the computed model of the extension of the program with $\Delta$ for this extension to be allowed. In this case, we say that $\Delta$ is **consistent** with the constraints. We will assume that, for any abductive theory, the empty set of abducible assumptions is consistent.

An abductive theory is thus viewed as representing a collection of different allowed states given by the set of its generalized models.

**Definition 2.3 (Abductive explanation)**
Let $T = \langle P, A, I \rangle$ be an abductive theory and $\phi$ any formula[*2] called an observation (or a query). An **abductive explanation** for $\phi$ in $T$ is any set $\Delta$ of abducible facts from $A$ such that

[*2] In general, $\phi$ can be any formula but in many cases it suffices for $\phi$ to be a conjunction of ground facts.

- $M(\Delta)$ is a generalized model of $T$, and
- $M(\Delta) \models \phi$.

Based on this we define a *credulous* form of abductive entailment.

**Definition 2.4 (Abductive entailment)**
Let $T = \langle P, A, I \rangle$ be an abductive theory and $\phi$ any formula. Then, $\phi$ is **abductively entailed** by $T$, denoted by $T \models_A \phi$, iff there exists an abductive explanation of $\phi$ in $T$. If the explanation is $\Delta$, we also write $T \models_A \phi$ with $\Delta$.

Note that, although the integrity constraints reduce the number of possible explanations for an observation, it is still possible for several explanations that satisfy (do not violate) the integrity constraints to exist. To this end, additional criteria e.g. minimality (with respect to set inclusion) or some measure of cost of the abducible assumptions can help to discriminate between different explanations. The following example illustrates the above ideas.

**Example 2.1**
Consider the following abductive theory $\langle P, A, I \rangle$ with $P$ the logic program on family relations:

$$
\begin{aligned}
&father(X,Y) \leftarrow parent(X,Y), male(X)\\
&mother(X,Y) \leftarrow parent(X,Y), female(X)\\
&son(X,Y) \leftarrow parent(Y,X), male(X)\\
&daughter(X,Y) \leftarrow parent(Y,X), female(X)\\
&child(X,Y) \leftarrow son(X,Y)\\
&child(X,Y) \leftarrow daughter(X,Y)\\
&loves(X,Y) \leftarrow parent(X,Y)
\end{aligned}
$$

the integrity constraint $I = \{\leftarrow male(X), female(X)\}$, and abducible predicates $A = \{parent,\ male,\ female\}$.
Consider now the observation $O_1 = father(bob, jane)$ An abductive explanation for $O_1$ is the set $\Delta_1 = \{parent(bob, jane), male(bob)\}$. This is the unique minimal explanation. Let now $O_2 = child(john, mary)$ be another observation. This has two possible explanations $\Delta_2 = \{parent(john,\ mary), male(john)\}$ and $\Delta_2' = \{parent(john, mary), female(john)\}$. If we also knew that the fact $male(john)$ holds then $\Delta_2'$ would be rejected due to the violation of the integrity constraint. In fact, these two explanations are incompatible with each other.

We will now introduce the concept of *strong abductive explanation* and consider negative as well as positive observations. These extensions are useful for learning as we will see in the next section. They allow an incremental construction of an explanation for all the positive training examples that also accounts for the negative examples as negative observations. A strong abductive explanation contains extra assumptions, with respect to a minimal explanation, that ensure that any self-consistent addition of further assumptions to it would not result in the violation of the integrity constraints. In order to obtain this property of strong explanations, we need to be able to make explicitly negative abducible assumptions. This is obtained by considering, for each abducible predicate $abd(X)$, a new abducible predicate $not\_abd(X)$ that is related to $abd(X)$ by the constraint $\leftarrow abd(X), not\_abd(X)$. The addition of abducible predicates expressing falsity or absence of the positive assumption defines a new abductive semantics, called *three-valued generalized model semantics*, where abducible atoms can be true, false or undefined, differently from generalized models where all the abducible facts not in the model are considered to be false.

**Definition 2.5 (Three-valued Version of a Theory)**
Given an abductive theory $T = \langle P, A, I\rangle$, the **three-valued version** of $T$ is the theory $T^* = \langle P, A \cup A^*, I \cup I^*\rangle$ where, for each predicate $a \in A$, $A^*$ contains the new predicate symbol $not\_a$ and $I^*$ contains the denial $\leftarrow a(\vec{X}), not\_a(\vec{X})$.

We define the *complement* $\bar{l}$ of an abducible literal $l$ as

$$\bar{l} = \begin{cases} not\_a(\vec{X}) & \text{if } l = a(\vec{X}) \\ a(\vec{X}) & \text{if } l = not\_a(\vec{X}) \end{cases}$$

Given the three-valued version $T^* = \langle P, A \cup A^*, I \cup I^*\rangle$ of an abductive theory, a set of ground assumptions $\Delta^*$ with predicates from $A \cup A^*$ is called *self-consistent* if and only if it does not contain both a literal and its complement, i.e., iff $\Delta^* \models I^*$;

**Definition 2.6 (Three-valued generalized model)**
Let $T = \langle P, A, I\rangle$, be an abductive theory with $T^* = \langle P, A \cup A^*, I \cup I^*\rangle$ its three-valued version and $\Delta^*$ a set of ground abducible facts from $A \cup A^*$. $M(\Delta^*)$ is a **three-valued generalized model** of $T$ iff

- $\Delta^*$ is self-consistent;
- $M(\Delta^*)$ is the minimal Herbrand model of $P \cup \Delta^*$;
- $M(\Delta^*) \models I$.

The set $\Delta^*$ is an **abductive explanation** for a formula $\phi$ if and only if $M(\Delta^*)$ is a three-valued generalized model and $M(\Delta^*) \models \phi$.

In a three-valued generalized model $M(\Delta^*)$ of $T$, an abducible fact $a(\overline{c})$ is assumed true if $a(\overline{c}) \in \Delta^*$, is assumed false if $not\_a(\overline{c}) \in \Delta^*$ and is undefined otherwise. From this point onwards, unless otherwise specified, we will consider abductive theories in their three-valued version. Therefore, when we write $T_1 = \langle P_1, A_1, I_1 \rangle$, we mean the three-valued version of a theory $T = \langle P, A, I \rangle$, with $P_1 = P$, $A_1 = A \cup A^*$ and $I_1 = I \cup I^*$. Also when we refer to a generalized model we will mean a three-valued generalized model.

We can now define the notion of strong abductive explanation.

**Definition 2.7 (Strong abductive explanation)**
Let $T = \langle P, A, I \rangle$ be an abductive theory, $T^* = \langle P, A \cup A^*, I \cup I^* \rangle$ its three-valued version and $O$ a ground atomic fact called an observation (or a goal). A **strong abductive explanation** for $O$ in $T$ is any set $\Delta^*$ of abducible facts from $A \cup A^*$ such that

- $\Delta^*$ is an abductive explanation for $O$ and
- for any $\Delta^{'} \subseteq A \cup A^*$, if $M(\Delta^{'}) \models I$ and $\Delta^{'} \cup \Delta^*$ is self-consistent, then $M(\Delta^{'} \cup \Delta^*) \models I$.

The latter condition can be intuitively expressed in this way: $\Delta^*$ must be such that any other consistent abductive extension $\Delta'$ that is self-consistent with $\Delta^*$ can be added to $\Delta^*$ without violating the integrity constraints. We say that $M(\Delta^*)$ is a **strong generalized model** for $T$ and that $\Delta^*$ is a **strong abductive extension** of $T^{*3}$.

In the case of example 2.1, a strong abductive explanation for $O_1 = father(bob, jane)$ would be $\Delta_1^* = \{parent(bob, jane), male(bob), not\_female(bob)\}$. Then the assumption $\Delta' = \{female(bob)\}$, that, when added to $\Delta_1$, would violate the integrity constraints, can not now be self-consistently added to the strong explanation $\Delta_1^* \supseteq \Delta_1$.

We now give the definition of a strong abductive explanation for a negative observation $not\_O$. In this case we want an explanation that can not be extended in order to explain $O$. Note that the link between this definition and the general notion of strong abductive explanations described above comes through the canonical integrity constraint $\leftarrow not\_O, O$.

*3 Note that under our previous assumption the empty set is always consistent for any abductive theory; this means that it has always a strong abductive extension.

**Definition 2.8 (Strong abductive explanation of negative observations)** Consider an abductive theory $T = \langle P, A, I\rangle$ with $T^* = \langle P, A \cup A^*, I \cup I^*\rangle$ its three-valued version. Let a negative observation (or a goal), denoted by *not_O*, be given. A **strong abductive explanation** for *not_O* is any set $\Delta^*$ of abducible facts from $A \cup A^*$ such that

- $M(\Delta^*)$ is a strong generalized model of $T$ with $M(\Delta^*) \not\models O$, and
- for any $\Delta' \subset A \cup A^*$, if $\Delta'$ is an abductive explanation of $O$ then $\Delta' \cup \Delta^*$ is not self-consistent.

We say that *not_O* is abductively entailed by $T$ and denote this by $T \models_A$ *not_O* with $\Delta^*$.

Hence $O \notin M(\Delta^*)$ and more importantly $\Delta^*$ cannot be consistently extended to derive $O$. The strong abductive explanation is thus a set of sufficient assumptions which, when adopted, ensures that $O$ can not be abductively entailed in a way that would be self-consistent with these assumptions.

In order to illustrate this, consider again example 2.1 and consider the negative observation *not_father*$(jane, john)$. A strong abductive explanation for *not_father*$(jane,\ john)$ is $\Delta_1^* = \{$*not_male*$(jane)\}$ or $\Delta_2^* = \{$*not_parent*$(jane,\ john)\}$ since *father*$(jane,\ john)$ can not be derived by any self-consistent extension of either of these sets. In contrast, the empty explanation is an abductive explanation for *not_father*$(jane, john)$ since *father*$(jane, john) \notin M(\emptyset)$ but this explanation is not strong since it can be consistently extended with $\Delta' = \{parent(jane, john), male(jane)\}$ to derive *father*$(jane, john)$.

The strongness of an explanation $\Delta^*$ for a negative observation *not_e* means that it invalidates every possible explanation for $e$. This is expressed by the following property, that is a direct consequence of the definition of strong abductive explanation.

**Property 2.1**
Let $T$ be an abductive theory and $e$ an atom. Then

$$T \models_A not\_e \text{ with } \Delta^* \Rightarrow \forall \Delta^+ : T \models_A e \text{ with } \Delta^+, \quad \exists \bar{l} \in \Delta^* : l \in \Delta^+$$

In other words, a strong abductive explanation $\Delta^*$ for *not_e* contains the complement of (at least) one assumption from every explanation $\Delta^+$ of $e$.

It is easy to see that this property holds for each one of the strong abductive explanations $\Delta_1^* = \{$*not_parent*$(jane, john)\}$ or $\Delta_2^* = \{$*not_male*$(jane)\}$ for *not_father*$(jane,\ john)$ in example 2.1 above. Another example that illustrates this is as follows.

**Example 2.2**
Consider the following abductive theory $T = \langle P, A, I \rangle$

$$
\begin{aligned}
P = \{&sibling(X,Y) \leftarrow brother(X,Y),\\
&sibling(X,Y) \leftarrow sister(X,Y)\}\\
I = \{&\}\\
A = \{&brother, sister\}
\end{aligned}
$$

and the observation $O = sibling(bob, jane)$. The strong abductive explanations for $not\_O$ is $\Delta^* = \{not\_sister(bob, jane), not\_brother(bob, jane)\}$, while the explanations for $O$ are $\Delta_1^+ = \{sister(bob, jane)\}$ and $\Delta_2^+ = \{brother(bob, jane)\}$: $\Delta^*$ contains the complement of a literal from both $\Delta_1^+$ and $\Delta_2^+$.

The definition of strong abductive explanation can be generalized for a conjunction of positive and negative observations $C = O_1 \wedge \ldots \wedge O_m \wedge not\_O_1 \wedge \ldots \wedge not\_O_n$. A **strong abductive explanation** for the conjunction is any set $\Delta^*$ of abducible facts from $A \cup A^*$ such that $\Delta^*$ is a strong abductive explanation for every conjunct taken singularly.

The strong abductive explanation for the conjunction of two positive or negative observations can be obtained by taking the union of the strong abductive explanations for each observation separately, as stated by the following proposition (the proof of this is given in appendix 2). This is an important property for combining together explanations of different positive and negative training examples in learning.

**Proposition 2.1**
Let $T = \langle P, A, I \rangle$ be an abductive theory in its three-valued version and let $\Delta_1$ and $\Delta_2$ be two strong abductive explanations of, respectively, $G_1$ and $G_2$, where $G_1$ and $G_2$ can be either positive or negative goals. If $\Delta_1 \cup \Delta_2$ is self-consistent, then $\Delta_1 \cup \Delta_2$ is a strong abductive explanation for $G_1 \wedge G_2$.

As we will see in the next sections, in the Abductive Concept Learning framework, deductive entailment is replaced by abductive entailment as the coverage relation. Thus the deductive SLD (and SLDNF) proof procedures of Logic Programming are replaced by abductive proof procedures [25, 42, 44, 19, 69] of ALP. Any abductive procedure satisfying the following notion of abductive derivability is suitable.

**Definition 2.9 (Abductive derivability)**
Given an abductive theory $T = \langle P, A, I \rangle$ in its three-valued version, a goal $G$ and an initial strong abductive explanation $\Delta_i$, we say that a procedure *abductively derives* $G$ from $T$ if it returns a set of assumptions $\Delta_G$ such that $\Delta_G$ is a strong abductive explanation of $G$ and $\Delta_G \cup \Delta_i$ is consistent, i.e., $M(\Delta_G \cup \Delta_i) \models I$[*4]. In this case, we write $T \vdash_{\Delta_i}^{\Delta_G} G$.

For our study of Abductive Concept Learning we will employ an abductive proof procedure based on the one of [44] reported in Appendix 3 The proof procedure interleaves phases of *abductive* and *consistency derivations*. Intuitively, an abductive derivation is the standard Logic Programming derivation suitably extended in order to consider abducibles. When an abducible atom $\delta$ is encountered, it is added to the current set of assumptions (if it is not already there). The addition of $\delta$ must not result in a violation of the integrity constraints. To this purpose, a consistency derivation for $\delta$ is initiated to check this. Each integrity constraint is resolved against $\delta$ and it is verified that every resulting goal fails. In the consistency derivation, when a new abducible is encountered in one of these reduced goals, an abductive derivation for its complement is started in order to ensure the failure of this abducible. This subsidiary abductive derivation will often result in additional assumptions in the explanation set.

The version of the procedure which we will use is sound with respect to the notion of (strong) abductive derivability above for the case in which the integrity constraints are restricted to be denials with at least one (positive) abducible appearing explicitly in the body of the denial. This result follows directly from the soundness of the original procedure in [44] which computes strong explanations. The more general case of integrity constraints in the form of range restricted clauses, $A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m$, can be first transformed into the equivalent denial form $\leftarrow B_1 \wedge \ldots \wedge B_m \wedge \neg A_1 \wedge \ldots \wedge \neg A_k$ before they are processed by the abductive proof procedure.

The completeness of this type of abductive procedures is in general difficult to achieve. A main source of incompleteness stems from the fact that the procedures can not select a non ground abductive goal. One way to address this is by a suitable restriction on the class of programs in our abductive theories, analogous to the range-restrictedness [55] of normal logic programs for avoiding floundering on negative conditions, which ensures that abducible conditions can

[*4] Since the theory is in its three-valued version, this means that $\Delta_G \cup \Delta_i$ is also self-consistent

always be grounded before selection. Another source of incompleteness is the difficulty in capturing the failure of goals that involve positive loops within a consistency derivation. For example, if a denial integrity constraint contains the condition $p$ and this is defined in the program by the rule $p \leftarrow p$ then such a constraint is always satisfied within the generalized model semantics. The abductive procedures will however fall into a loop in the corresponding consistency derivation when checking the satisfaction of such a constraint. Again to tackle this form of incompleteness we can restrict the class of programs to be stratified or acyclic [4)].

In practice, the abductive theories learned within the Abductive Concept Learning (ACL) framework will often satisfy these restrictions (this can be partially achieved by a suitable bias in the learning phase) and hence the abductive procedure is effectively complete on these theories. Similarly, although the procedures as defined in [44)] will not always guarantee that the first explanation found is minimal (this depends on the order in which the clauses are tried) the type of theories learned by ACL are such that the first explanation is indeed minimal or if not then a minimal explanation can be found on backtracking.

## §3 Learning with Abduction

Abductive Concept Learning (ACL) differs from ILP with both the background knowledge and the learned theory being abductive theories. The language of the hypotheses and of the background knowledge is that of abductive theories of Abductive Logic Programming with the following restrictions[*5].

- The background knowledge $T = \langle P, A, I \rangle$ does not contain any target predicate(s) neither in the program $P$ nor in the integrity constraints $I$. The empty set of abducible assumptions is a consistent abductive extension of $T$.
- The integrity constraints are range-restricted clauses $A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m$, with at least one of $B_1, \ldots, B_m$ abducible. Also, for each $A_j$ in the head of the clause, its definition in the program $P$ of the background theory does not depend on abducibles, namely $A_j$ is not abducible and recursively none of the conditions in the rules of $P$ for $A_j$ is abducible.

The language of the examples is simply that of atomic ground facts on the

[*5] These language restrictions are not necessary for the definition of the ACL problem but rather are needed for the development of the algorithms to solve this problem.

target predicate(s). We note here that the ACL framework does not allow the acquisition of a new (unknown in the background theory) abducible thus limiting its scope of learning e.g. for predicate invention. The possibility to lift this restriction is discussed briefly below in the section on related work.

**Definition 3.1 (Abductive Concept Learning)**

**Given**

- a set of positive examples $E^+$,
- a set of negative examples $E^-$,
- an abductive theory $T = \langle P, A, I \rangle$ as background theory,
- an hypothesis space $\mathcal{T} = \langle \mathcal{P}, \mathcal{I} \rangle$ consisting of a space of possible programs $\mathcal{P}$ and a space of possible constraints $\mathcal{I}$ satisfying the language restrictions given above except that now a possible program can contain the target predicate(s).

**Find**

A set of rules $P' \in \mathcal{P}$ and a set of constraints $I' \in \mathcal{I}$ such that the new abductive theory $T' = \langle P \cup P', A, I \cup I' \rangle$ satisfies the following conditions

- $T' \models_A E^+$,
- $\forall e^- \in E^-, T' \not\models_A e^-$.

where $E^+$ stands for the conjunction of all positive examples.

We say that an individual example $e$ is *covered* by a theory $T'$ iff $T' \models_A e$.

In effect, we have replaced the deductive entailment in the ILP problem with abductive entailment to define the ACL learning problem.

The fact that the conjunction of positive examples must be entailed means that, for every positive example, there must exist an abductive explanation and the explanations for all the positive examples must be consistent with each other. For negative examples, it is required that no abductive explanation exists for any of them. Abductive concept learning can be illustrated as follows.

**Example 3.1**
Suppose we want to learn the concept $father$. Let the background theory be $T = \langle P, A, \emptyset \rangle$ where:

$P = \{parent(john, mary), male(john),$
$parent(david, steve),$
$parent(kathy, ellen), female(kathy)\}$

$A = \{male, female\}$.

Let the training examples be:

$E^+ = \{father(john, mary), father(david, steve)\}$

$E^- = \{father(kathy, ellen), father(john, steve)\}$

In this case, a possible hypotheses $T' = \langle P \cup P', A, I' \rangle$ learned by ACL would consist of

$P' = \{father(X, Y) \leftarrow parent(X, Y), male(X).\}$

$I' = \{\leftarrow male(X), female(X).\}$

This hypothesis satisfies the definition of ACL because:

1. $T' \models_A father(john, mary), father(david, steve)$ with $\Delta = \{male(david)\}$,
2. $T' \not\models_A father(kathy, ellen)$, as the only possible explanation for this goal, namely $\{male(kathy)\}$ is made inconsistent by the learned integrity constraint in $I'$.
3. $T' \not\models_A father(john, steve)$, as this has no possible abductive explanations.

Hence, despite the fact that the background theory is incomplete (in its abducible predicates), ACL can find an appropriate solution to the learning problem by suitably extending the background theory with abducible assumptions. Note that the learned theory without the integrity constraint would not satisfy the definition of ACL, because there would exist an abductive explanation for the negative example $father(kathy, ellen)$, namely $\Delta^- = \{male(kathy)\}$. This explanation is prohibited in the complete theory by the learned constraint together with the fact $female(kathy)$. Note that if the predicate $parent$ is also incompletely specified and hence also declared as abducible, then this hypothesis $T'$ would not constitute a valid ACL solution to the learning problem as the negative example $father(john, steve)$ will have the abductive explanation $\{parent(john, steve)\}$. Extra integrity constraints will be needed in $I'$ to render this assumption inconsistent.

It is important to note that the treatment of positive and negative examples in ACL is asymmetric with respect to the existence of abductive explanations. For positive examples, it is sufficient that there exists one explanation for the conjunction of all positive examples that is consistent with the constraints, whereas, for each negative example, all possible explanations must be made inconsistent by the constraints.

We note here that an alternative definition of ACL could require the weaker condition that each negative example can not be abductively entailed by $(P \cup P') \cup \Delta$ where $\Delta$ are the assumptions required to explain all the positive examples. This will be used below as a subproblem of ACL that is useful for the development of an algorithm to solve the full ACL problem. An even weaker condition would be that $(P \cup P') \cup \Delta$ does not deductively entail any of the negative examples. But this requirement is in general too weak as any rule that contains an abducible condition in the body would easily fail to deductively conclude its head because of the incompleteness of the abducible. Instead, we want to be sure that the theory can not be completed so that the negative examples can be derived.

In order to achieve this requirement for the negative examples, we require the existence of a **strong** abductive explanation for (the complement of) each negative examples. Adding these strong explanations to the background theory then ensures that no negative example can be abductively explained. In the example above, the negative example $father(kathy, ellen)$ can be uncovered by adding the strong abductive explanation $\Delta^* = \{not\_male(kathy)\}$ for $not\_father(kathy, ellen)$ to the theory. This is sufficient to ensure that this negative example can no longer be abductively entailed even in the absence of any integrity constraints in $I'$. Moreover, these strong abductive explanations can suggest what new integrity constraints can be learned in $I'$ so that the negative examples can not be covered.

This observation suggests a natural way in which the full ACL problem can be split into two subproblems: (1) learning the rules together with appropriate strong explanations and (2) learning integrity constraints. We will see that the solutions of the two subproblems can be combined to obtain a solution for the original problem.

The first subproblem, called ACL1, has the following definition.

**Definition 3.2 (ACL1)**

**Given**

- a set of positive examples $E^+$,
- a set of negative examples $E^-$,
- an abductive theory $T = \langle P, A, I \rangle$ as background theory,
- a hypothesis space of possible programs $\mathcal{P}$.

**Find**

A set of rules $P' \in \mathcal{P}$ such that the new abductive theory $T_{ACL1} = \langle P \cup P', A, I \rangle$ satisfies the following conditions

- $T_{ACL1} \models_A E^+$ with $\Delta^+$,
- $T_{ACL1} \models_A not\_E^-$ with $\Delta^-$,
- $\Delta^+ \cup \Delta^-$ is self-consistent.

where $not\_E^-$ stands for the conjunction of the complement of every negative example.

We say that a theory $T$ **ACL1-covers** an individual positive example $e^+$ iff $T \models_A e^+$ and that $T$ **does not ACL1-cover** $e^+$ iff $T \not\models_A e^+$.

If $T \models_A e^+$ with $\Delta = \emptyset$, then we say that $e^+$ is **ACL1-covered without abduction**, otherwise we say that it is **ACL1-covered with abduction**.

For negative examples, we say that a theory $T$ **ACL1-uncovers** an individual negative example $e^-$ iff $T \models_A not\_e^-$ and that $T$ **does not ACL1-uncover** $e^-$ iff $T \not\models_A not\_e^-$.

If $T \models_A not\_e^-$ with $\Delta = \emptyset$, then we say that $e^-$ is **ACL1-uncovered without abduction**, otherwise we say that it is **ACL1-uncovered with abduction**.

In effect, in ACL1 we require the existence of a strong abductive explanation for the negation of each negative example from the learned theory extended with $\Delta^+$. This is weaker than the condition required by the full ACL problem which is that no negative example has an abductive explanation from the learned theory $T_{ACL1}$ i.e. that every negative example is false in all the abductive extensions of the $T_{ACL1}$. However, the existence of a strong abductive explanation means that it is possible to satisfy this stronger requirement.

Indeed, the information generated by ACL1 through the strong abductive explanations for negative examples can be used to provide a solution of the full ACL problem through a second learning phase. From the output of ACL1, i.e. its set of rules and the sets of assumptions $\Delta^+$ and $\Delta^-$ for covering positive examples and uncovering negative ones, a solution to ACL can be found by learning constraints that are consistent with $\Delta^+$ and inconsistent with the complement of every abducible in $\Delta^-$. In fact, the strong abductive explanation $\Delta^-$ will contain, for every negative example $e^-$, a strong abductive explanation $\Delta_{e^-}$ for $not\_e^-$. This explanation, according to property 2.1, contains assumptions that would invalidate directly any possible abductive explanation of $e^-$. Hence by making all the complements of assumptions in $\Delta^-$ inconsistent we make all possible explanations of every $e^-$ inconsistent.

Thus the definition of the second subproblem, called ACL2, can be given as follows.

**Definition 3.3 (ACL2)**

**Given**

- a solution of ACL1
  - $T_{ACL1} = \langle P \cup P', A, I \rangle$,
  - $\Delta^+$,
  - $\Delta^-$,
- a hypothesis space of possible constraints $\mathcal{I}$ satisfying the same requirements as in ACL.

**Find**

A set of constraints $I' \in \mathcal{I}$ such that the new abductive theory $T' = \langle P \cup P', A, I \cup I' \rangle$ satisfies the following condition

- $M_{P \cup P'}(\Delta^+) \models I'$,
- $\forall l \in \Delta^-$, $M_{P \cup P'}(\{\bar{l}\}) \not\models I'$.

Note that the third condition of ACL1 requiring $\Delta^+ \cup \Delta^-$ to be self-consistent helps to avoid the case of posing an empty ACL2 problem. If this cannot be satisfied, i.e. $\Delta^+ \cup \Delta^-$ is not self-consistent, then the corresponding ACL2 problem cannot have any solutions.

The theory $T' = \langle P \cup P', A, I \cup I' \rangle$, obtained by combining the solutions of the two subproblems, gives a solution to the full ACL problem.

**Theorem 3.1**
Let $T_{ACL1} = \langle P \cup P', A, I \rangle$, $\Delta^+$ and $\Delta^-$ be the solution of ACL1 given training sets $E^+$ and $E^-$, background theory $T = \langle P, A, I \rangle$ and space of possible programs $\mathcal{P}$. Moreover, let $T' = \langle P \cup P', A, I \cup I' \rangle$ be the solution to ACL2 given the previous solution of ACL1 and hypothesis space $\mathcal{I}$. Then $T'$ is a solution to the ACL problem that has $E^+$ and $E^-$ as training sets, $T$ as background theory and $\mathcal{P}$ and $\mathcal{I}$ as spaces of possible programs and constraints.

The proof of this theorem is reported in Appendix 1. Once decomposed into its two subproblems, it becomes clear that ACL combines the two ILP settings of explanatory (predictive) learning and confirmatory (descriptive) learning. In fact, ACL1 can be seen as a problem of learning from entailment, while ACL2 as a problem of learning from interpretations.

The algorithm we present in the next section solves the ACL problem by first solving ACL1 and then ACL2. In example 3.1, the solution of ACL1 consists of the rule $father(X,Y) \leftarrow parent(X,Y),\ male(X)$, together with $\Delta^+ = \{male(david), not\_female(david)\}$ and $\Delta^- = \{not\_male(kathy)\}$. Given this intermediate solution, we can now apply a second phase where integrity constraints are learned from the background knowledge and the explanations obtained in the first phase. We want to make $male(kathy)$ inconsistent while keeping $\Delta^+$ consistent: $\leftarrow male(X), female(X)$ is a constraint that satisfies these conditions.

We note that in many cases, ACL1 can be useful on its own merit, e.g. when we have sufficient information in the integrity constraints of the background theory or for problems where indeed this weaker requirement on negative examples is sufficient. We will see examples of such cases in the following sections 5 and 6.

## 3.1 Monotonicity and Generality

Abductive Logic Programs are inherently non-monotonic. Given two abductive theories $T_1 = \langle P_1, A, I_1 \rangle$ and $T_2 = \langle P_2, A, I_2 \rangle$ that each entail a goal, their union $T = \langle P_1 \cup P_2, A, I_1 \cup I_2 \rangle$ does not necessarily entail this goal. Non-monotonicity poses problems in learning as algorithms based on the covering approach can not be used. In general, we can not learn a theory by iteratively adding a clause to a partial hypothesis because the addition of a clause can reduce the number of positive examples covered by the hypothesis since this addition can render some of the abductive assumptions inconsistent.

By splitting the ACL problem into the two phases of ACL1 and ACL2, we can recover a form of restricted monotonicity. In the first phase of ACL1 where the integrity constraints remain fixed we have two cases to consider: (i) monotonicity under the addition of a new clause in the program $P$ of the current hypothesis and (ii) monotonicity under the addition of new abductive assumptions as we move from one training example to another. The second case can be dealt with by employing a suitable abductive proof procedure for ALP based on strong abductive explanations, as discussed in the previous section, carrying the explanation of the previous examples when testing the abductive coverage (or uncoverage if the example is negative) of the next example. The first case is in general more difficult but in the particular case of interest since the new (learned) clauses can only affect the extension of the target(s) predicates, we can satisfy this monotonicity requirement by restricting (as we have) the language of the integrity constraints and the program of the background theory to be independent of the target predicate(s).

In the second phase of ACL2, where the program of the abductive hypothesis is fixed and we vary the integrity constraints, monotonicity in ensured by the specific definition of the ACL2 problem that we have adopted where by construction the new learned integrity constraints must be consistent with the abductive assumptions $\Delta^+$ required for the coverage of the positive examples. Hence these examples will continue to be abductively entailed by the theory after the addition of the new integrity constraints generated by ACL2.

The non-monotonic nature of the hypothesis space of abductive theories introduces another difficulty in the task of solving the ACL problem. It makes it difficult to have a generality[*6] structure on this space that can be useful in the search for solutions to our learning problem. In general, there is no natural generality structure on the full space of abductive theories but again the separation of the problem into its two phases of ACL1 and ACL2 allows us to define generality relations separately on the rule part $P$ and integrity constraints $I$ of the abductive theories. These separate generality relations can be defined via classical deductive entailment in the same way as in ILP. For the rule part we can use the same generality relation as in ILP, namely that: $P_1$ is more general that $P_2$ iff $P_1 \models P_2$.

For the integrity constraints we can define their generality relation as follows.

**Definition 3.4**
Given two sets on integrity constraints $I_1$ and $I_2$, $I_1$ is more general than $I_2$ iff $I_2 \models I_1$.

The generality of integrity constraints is defined in this dual way as their role in the abductive entailment (and hence in the coverage relation) is to restrict the possibility of making abductive assumptions by requiring that these assumptions define (through the program) a model of the constraints (see definition 2.2). In this way, the generality of integrity constraints increases as the number of abductive explanations that they allow increases.

[*6] As usual, a theory $T_2$ is more general that another theory $T_1$ iff $Covers(T_1) \subset Covers(T_2)$ where $Covers(T)$ denotes the set of all examples that are covered by the theory $T$.

The use of these usual generality relations on the separate parts of an abductive theory means, as we shall see in the next section, that we can adapt standard ILP techniques, e.g. generalization and specialization operators based on $\theta$-subsumption [61, 62], in developing algorithms for the separate phases of ACL1 and ACL2.

# §4 An Algorithm for ACL

The ACL problem can be solved by the following algorithm, also called ACL. The algorithm is composed of two steps, one for each of the subproblems of the full ACL problem.

**Algorithm ACL**:

1. **Learn rules** (ACL1): find a set of rules $P'$ and two sets of assumptions $\Delta^+$ and $\Delta^-$ such that
   - $\langle P \cup P', A, I\rangle \vdash_{\emptyset}^{\Delta^+} E^+$,
   - $\langle P \cup P', A, I\rangle \vdash_{\Delta^+}^{\Delta^-} not\_E^-$

   where $\vdash_{\Delta}^{\Delta'}$ denotes an abductive derivability relation satisfying definition 2.9.
2. **Learn constraints** (ACL2): find a set of integrity constraints $I'$ such that
   - $M(\Delta^+) \models I'$,
   - $\forall l \in \Delta^-$, $M(\{\bar{l}\}) \not\models I'$

   where $M(\Delta^+)$ and $M(\{\bar{l}\})$ denote the minimal Herbrand model of $P \cup P' \cup \Delta^+$ and $P \cup P' \cup \{\bar{l}\}$.

ACL1 is solved by an algorithm also called ACL1 that will be presented in section 4.1. Note that this algorithm uses strong abductive explanations for the positive examples $E^+$ (as well as the negative examples) thus exploiting the property of proposition 2.1 for combining incrementally separate explanations and in particular for ensuring that the union $\Delta^+ \cup \Delta^-$ of the computed assumptions is consistent with the learned theory. ACL2 can be solved by employing a framework that performs discriminant learning from interpretations, such as ICL [18]. We will explain in more detail how ICL can be applied in section 4.2.

## 4.1 An Algorithm for ACL1

The algorithm for ACL1 is based on the generic top-down ILP algorithm

**procedure** ACL1(
  **inputs :** $E^+, E^-$ : training sets,
    $T = \langle P, A, I\rangle$ : background abductive theory,
  **outputs :** $H$ : learned theory, $\Delta^+, \Delta^-$ : abduced literals)

$H := \emptyset$
$\Delta^+ := \emptyset$
$\Delta^- := \emptyset$
**repeat**
  Specialize($T, H, E^+, E^-, \Delta^+, \Delta^-; Rule, E^+_{Rule}, \Delta^+_{Rule}, \Delta^-_{Rule}$)
  $E^+ := E^+ \setminus E^+_{Rule}$
  $H := H \cup \{Rule\}$
  $\Delta^+ := \Delta^+ \cup \Delta^+_{Rule}$
  $\Delta^- := \Delta^- \cup \Delta^-_{Rule}$
**until** $E^+ = \emptyset$ (*sufficiency stopping criterion*)
**output** $H, \Delta$

**Fig. 1** ACL, the covering loop

(see e.g. [52]) and extends the algorithm in [26]. In this section, we consider only a single predicate learning task. We will discuss in section 5 the problem of learning multiple predicates. The top level covering and specialization loops of the algorithm are shown in Fig. 1 and Fig. 2 respectively.

The generic top-down algorithm has been extended in several ways to take into account the abductive coverage relation of ACL1. New clauses are generated by beam search, initialized to a clause with an empty body for the target predicate, using a specially defined heuristic evaluation function. This is adapted from the usual accuracy function to allow for the possibility of missing information on some of the background predicates.

The evaluation of a clause is done by starting an abductive derivation for each positive example and for the complement of each negative example. The derivation is performed using a procedure based on the abductive procedure outlined in appendix 3. For each example $e$ a call AbductiveDerivation($e, \langle P \cup H \cup \{Rule\}, A, I\rangle, \Delta_{in}; \Delta_e$) returns a strong abductive explanation $\Delta_e$ for the goal $e$ (which is either of the form $e^+$ or $not\_e^-$) starting from an initial set of

**procedure** Specialize(
    **inputs :** $T$ : background theory,
        $H$ : current hypothesis, $E^+, E^-$ : training sets,
        $\Delta^+, \Delta^-$ : current set of abduced literals
    **outputs :** $Best$ : rule, $E^+_{Best}$ : examples covered by $Best$,
    $\Delta^+_{Best}, \Delta^-_{Best}$ : literals abduced when testing $Best$)

$Beam$ := { $\langle p(X) \leftarrow true., Value\rangle$, where $p$ is a target predicate,
    $Value$ is the value of the heuristic function for the rule}
Select and remove the best rule $Best$ from $Beam$
**repeat**
    $BestRefinements$ := set of refinements of $Best$ allowed
        by the language bias
    **for all** $Rule \in BestRefinements$ **do**
        $Value$ := Evaluate($Rule, T, H, E^+, E^-, \Delta^+, \Delta^-$)
        **if** $Rule$ covers at least one pos. ex. **then**
            add $\langle Rule, Value\rangle$ to $Beam$
    **endfor**
    Remove rules in $Beam$ exceeding the beam size
    Select and remove the best rule $Best$ from $Beam$
**until** $Best$ ACL1-uncovers every $e^- \in E^-$ (*necessity stopping criterion*)
Test the coverage of $Best$ obtaining:
    $E^+_{Best}$ the set of positive examples covered by $Best$
    $\Delta^+_{Best}$ and $\Delta^-_{Best}$ the sets of literals abduced during
    the derivation of $e^+$ and $not\_e^-$ ($e^+ \in E^+_{Best}, e^- \in E^-$)
**output** $Best, E^+_{Best}, \Delta^+_{Best}, \Delta^-_{Best}$

**Fig. 2** ACL, the specialization loop

assumptions $\Delta_{in}$, i.e. $\langle P \cup H \cup \{Rule\}, A, I\rangle \vdash^{\Delta_e}_{\Delta_{in}} e$. $\Delta_{in}$ consists of the set of assumptions abduced for earlier examples thus ensuring that the assumptions made during the derivation of the current example are consistent with the ones made before. Note that $\Delta_e$ contains all the assumptions needed to explain $e$, even those that are already contained in $\Delta_{in}$. This is needed for the evaluation of the heuristic value of the clause as well as for the second phase (ACL2) of the ACL algorithm, where we learn the constraints, as the learned constraints must make inconsistent all the assumptions in the explanations $\Delta_{not\_e^-}$ of any negative example $e^-$. The procedure Evaluate is shown in Fig. 3. When using abduction for the coverage of a set of examples, two types of cost must be taken into account: the first cost is given by the number of examples in the set that are covered (or uncovered) using abduction with respect to those covered without abduction, i.e. are deductively covered (or uncovered), while the second cost is the cost of each assumption in the explanations for individual examples. These costs are taken into account by the heuristics, which weight examples differently depending on whether they are covered with abduction or without abduction, with a relative weight depending on some estimate of the probability of the assumptions.

The heuristic function is thus calculated based on the number of covered positive (and uncovered negative examples), distinguishing between examples covered (uncovered) with or without abduction. Note that the use of strong explanations affects this distinction of non coverage of negative examples as strong explanations would include negative abducibles for examples that can not be uncovered by the failure of a non-abducible condition in the body of the clause. For a clause (or rule) $c$ the heuristic function takes the form of an *expected classification accuracy* [52]:

$$A(c) = p(\oplus|c)$$

where $p(\oplus|c)$ is the probability that an example covered by clause $c$ is positive. In defining the probability we need to give different strength to positive examples covered (negative examples uncovered) with assumptions (i.e. $T \vdash^{\Delta}_{\Delta_{in}} e$ with $\Delta \neq \emptyset$, where $e$ is either $e^+$ or $not\_e^-$) or without assumptions (i.e. $\Delta = \emptyset$).

The heuristic function used is

$$A(c) = \frac{n^{\oplus} + k^{\oplus} \times n^{\oplus}_A}{n^{\oplus} + n^{\ominus} + k^{\oplus} \times n^{\oplus}_A + k^{\ominus} \times n^{\ominus}_A}$$

where, for any given clause $c$, $n^{\oplus}, n^{\oplus}_A, n^{\ominus}, n^{\ominus}_A$ are defined as in procedure Evaluate

```
function Evaluate(
    inputs : Rule: rule, T = ⟨P, A, I⟩ : background theory,
        H : current hypothesis, E⁺, E⁻ : training sets,
        Δ⁺, Δ⁻ : current sets of abduced literals)
    returns the value of the heuristic function for Rule

n⊕ := 0, number of pos. ex. ACL1-covered by Rule without abduction
n⊕_A := 0, number of pos. ex. ACL1-covered by Rule with abduction
n⊖ := 0, number of neg. ex. not ACL1-uncovered by Rule
n⊖_A := 0, number of neg. ex. ACL1-uncovered by Rule with abduction
Δ_in := Δ⁺ ∪ Δ⁻
for each e⁺ ∈ E⁺ do
    if AbductiveDerivation(e⁺, ⟨P ∪ H ∪ {Rule}, A, I⟩, Δ_in; Δ_e⁺)
        succeeds then
        if Δ_e⁺ = ∅ then
            increment n⊕
        else
            increment n⊕_A
        endif
        Δ_in := Δ_in ∪ Δ_e⁺
    endif
endfor
for each e⁻ ∈ E⁻ do
    if AbductiveDerivation(not_e⁻, ⟨P ∪ H ∪ {Rule}, A, I⟩, Δ_in; Δ_e⁻)
        succeeds then
        if Δ_e⁻ ≠ ∅ then
            increment n⊖_A
        endif
        Δ_in := Δ_in ∪ Δ_e⁻
    else
        increment n⊖
    endif
endfor
return Heuristic(n⊕, n⊕_A, n⊖, n⊖_A)
```

**Fig. 3** ACL, evaluation of a clause

according to the abductive coverage of positive and negative examples by $c$.

The coefficients $k^{\oplus}$ and $k^{\ominus}$ are introduced in order to take into account the degree of confidence in the assumptions made, respectively, for positive and negative examples. They are an estimate of the fraction of assumptions made that are correct. For example, consider a clause $c$ of the form:

$p(X) \leftarrow Body(X)$

where $Body(X)$ is a conjunction of literals not containing an abducible. Suppose we want to evaluate the refinement $c'$ obtained by adding to $c$ the abducible literal $abd(X)$. Clause $c$ covers $n^{\oplus}(c)$ positive examples without abduction: out of these, $c'$ will cover $n^{\oplus}(c')$ positive examples without abduction (for which a fact of the form $abd(\vec{t})$ is in the background program), $n^{\oplus}_A(c')$ with abduction (a fact of the form $abd(\vec{t})$ is abduced) and it will not cover $n^{\oplus}(c) - n^{\oplus}(c') - n^{\oplus}_A(c')$ examples ($abd(\vec{t})$ could not be abduced because of constraints).

As an example, consider the following training set

$E^+ = \{p(a), p(b), p(c), p(d), p(e), p(f)\}$

Suppose that $Body(X)$ is true for $X$ equal to all the constants $a, b, c, d, e, f$, therefore $n^{\oplus}(c) = 6$. Moreover, suppose the background knowledge contains the facts $abd(a)$ and $abd(b)$ and the constraint $\leftarrow abd(X), q(X)$ together with the facts $q(e)$ and $q(f)$. Therefore, $p(a)$ and $p(b)$ will be covered by $c'$ without abduction ($n^{\oplus}(c') = 2$), $p(c)$ and $p(d)$ will be covered with abduction ($n^{\oplus}_A(c') = 2$) and $p(e)$ and $p(f)$ will not be covered because the assumptions $abd(e)$ and $abd(f)$ are inconsistent with the integrity constraint $\leftarrow abd(X), q(X)$.

$k^{\oplus}(c')$ expresses an estimate of the fraction of the $abd(\vec{t})$ assumptions that are correct in the sense that, if the knowledge were complete, $abd(\vec{t})$ would be known to be true. This percentage is estimated by assuming that the ratio of true facts over the total number of facts for the unknown atoms is the same for the known atoms. Therefore $k^{\oplus}(c')$ is given by the following formula

$$k^{\oplus}(c') = \frac{\text{\# of true atoms}}{\text{\# of known atoms}} = \frac{n^{\oplus}(c')}{n^{\oplus}(c) - n^{\oplus}_A(c')}$$

The true atoms are the facts (in the background program) of the form $abd(\vec{t})$ that corresponds to examples covered by $c'$, therefore their number is $n^{\oplus}(c')$. The false atoms are the ones for which the constraints inhibited the assumption of a fact of the form $abd(\vec{t})$. The unknown atoms are the ones for which it was possible to make an assumption of the form $abd(\vec{t})$, hence their number is $n^{\oplus}_A(c')$. The number of known atoms is given by the total number of atoms in the sample universe (i.e. of the examples covered by $c$) minus the number of unknown atoms. In the example above, the number of true atoms for $abd(X)$ is 2 ($abd(a)$ and $abd(b)$) and the number of unknown atoms is 2 ($abd(c)$ and $abd(d)$), so the number of known atoms is 6-2=4 and $k^{\oplus}(c')$ has the value 2/4=0.5.

In the case in which no constraints are available, $n^{\oplus}(c')+n^{\oplus}_{A}(c') = n^{\oplus}(c)$ and $k^{\oplus}(c')$ is always 1. In this case, we use following more conservative estimate

$$k^{\oplus}(c') = \frac{n^{\oplus}(c')}{n^{\oplus}(c)}$$

with a lower bound, set by default to 0.1, so that $k^{\oplus}(c')$ can not drop below this threshold. This estimate turned out often to be more realistic also when constraints are available, due to the fact that much more positive information (represented by facts of the programs) is usually available rather than negative information (represented by constraints). Therefore, this more conservative estimate was used in most of the experiments.

Finally, we must consider the case in which some abducibles were already present in $Body(X)$. We will assume that all the examples covered by $c'$ with abduction are covered with abduction as well by $c$. $k^{\oplus}(c')$ must then express the probability that both the current assumptions and those made before are true at the same time. Therefore:

$$k^{\oplus}(c') = k^{\oplus}(c) \times \frac{n^{\oplus}(c')}{n^{\oplus}(c)}$$

The formula for $k^{\ominus}(c')$ can be derived with a similar reasoning:

$$k^{\ominus}(c') = k^{\ominus}(c) \times \frac{n^{\ominus}(c')}{n^{\ominus}(c)}$$

As mentioned above, these heuristics act as a method for assigning a cost on the abductive assumptions required for the appropriate coverage of the positive and negative examples. This has two effects. First by assigning a cost factor ($k^{\oplus}$ or $k^{\ominus}$) to coverage which is not deductive but needs some assumptions it gives preference to learned definitions that would not need abduction if this is indeed possible. Secondly, if this is not possible, then the varying cost of the abducible assumptions (depending on their relative frequency in the background theory) gives a preference amongst the different possible abductive explanations (and hence coverage) of the examples. In this way the heuristics help to select simple definitions for the concepts avoiding possibilities which require a large number of unlikely abductive assumptions. Finally, we note that it is possible to use, together with the above heuristics, other cost functions on the abducibles, possibly specific to the particular learning problem at hand, without any essential change to the ACL algorithm described above.

## 4.2 Learning Integrity Constraints

The second subproblem ACL2 of learning integrity constraints can be seen as a problem of learning from interpretations where we have to discriminate between allowed interpretations (explanations for positive examples) and forbidden interpretations (explanations for negative examples). The framework of ICL [18] solves exactly this problem and we can therefore use it to solve ACL2. We recall here the definition of the ICL problem.

**Definition 4.1 (ICL Problem)**

**Given**

- a definite clause background theory $B$,
- a set of positive interpretations $P$,
- a set of negative interpretations $N$.

**Find** a clausal theory $H$ such that

- for all $p \in P$, $M(B \cup p)$ is a true interpretation of $H$, i.e. $M(B \cup p) \models H$ (Completeness);
- for all $n \in N$, $M(B \cup n)$ is a false interpretation of $H$, i.e. $M(B \cup n) \not\models H$ (Consistency);

In our case, we have to learn integrity constraints on abducibles by using the information contained in the sets $\Delta^+$ and $\Delta^-$ generated from ACL1. ICL can be used to solve the ACL2 problem with the following inputs:

- the program $P \cup P'$ as the background knowledge $B$,
- one positive interpretation $p = \Delta^+$;
- one negative interpretation $n_i = \{\overline{l_i}\}$ for each $l_i \in \Delta^-$.

Learned constraints will be true in the model $M(\Delta^+)$ and will be false in each model $M(\{\overline{l_i}\})$. Therefore, when the integrity constraints are added to the final abductive theory, they will not allow any of the abductive assumptions $\overline{l}_i$ with $l_i \in \Delta^-$. This in turn means (see theorem 3.1) that negative examples cannot be abductively entailed as required for the full ACL problem.

We mention here that another possibility of integrating the two subproblems of ACL1 and ACL2 is to record in ACL1 all possible explanations $\Delta_e$-

for each negative example $e^-$ in its positive form and to give to ICL each one of these explanations $\Delta_{e^-}$ as negative interpretations. In this way, we do not decide a priori in ACL1 how (i.e. on which assumption) each of the explanations for negative examples must be made inconsistent later by the constraints produced by ACL2. This decision is taken a-posteriori by ACL2 itself when it produces the constraints. Hence ICL has the freedom to make $\Delta_{e^-}$ inconsistent on any of the abducibles in it. Learning constraints is now easier because ICL can choose which abducible to make inconsistent in each explanation $\Delta_{e^-}$. However, this alternative way of splitting the ACL problem is only appropriate when assumptions for positive examples cannot contradict those for negative examples. Otherwise, such an inconsistency will not be detected until the end of the second phase requiring the (costly) return to the first phase.

## 4.3 Properties of the Algorithm

In this section, we show the soundness of the ACL algorithm given in the previous section and discuss its (lack of) completeness.

Let us first define formally the properties of soundness and completeness of an inductive algorithm for the problem of ACL. Given an algorithm, $\mathcal{A}$, for ACL we shall write $\mathcal{A}(\langle \mathcal{P}, \mathcal{I} \rangle, E^+, E^-, T) = T'$ to indicate that, given the hypothesis space $\langle \mathcal{P}, \mathcal{I} \rangle$, the positive and negative examples $E^+$ and $E^-$, and the background knowledge $T$, the algorithm outputs a program $T'$. With respect to the ACL problem definition of section 3, soundness and completeness are defined as follows.

**Definition 4.2 (Soundness)**
An algorithm $\mathcal{A}$ is sound if whenever $\mathcal{A}(\langle \mathcal{P}, \mathcal{I} \rangle, E^+, E^-, \langle P, A, I \rangle) = T'$, then $T' = \langle P \cup P', A, I \cup I' \rangle$ satisfies the conditions of definition 3.1, i.e. $P' \in \mathcal{P}$, $I' \in \mathcal{I}$ and

- $T' \models_A E^+$,
- $\forall e^- \in E^-$, $T' \not\models_A e^-$.

**Definition 4.3 (Completeness)**
An algorithm $\mathcal{A}$ is complete if whenever there is a $T'$ such that $T' = \langle P \cup P', A, I \cup I' \rangle$ satisfies the conditions of definition 3.1, i.e. $P' \in \mathcal{P}$, $I' \in \mathcal{I}$ and

- $T' \models_A E^+$,
- $\forall e^- \in E^-$, $T' \not\models_A e^-$.

then $\mathcal{A}(\langle \mathcal{P}, \mathcal{I} \rangle, E^+, E^-, \langle P, A, I \rangle) = T'$.

The algorithm for ACL presented above in this section 4 in terms of the ACL1 and ACL2 algorithms is sound but not complete.

**Theorem 4.1 (Soundness)**
The algorithm ACL is sound.

The proof of this theorem is given in appendix 2. The ACL algorithm is incomplete because the search space of ACL1 is not completely explored. In particular, there are two choice points which are not considered in order to reduce the computational complexity of the algorithm. The first choice point is related to the greedy search in the space of possible programs as in most ILP systems. When no new clause can be added by the specialization loop, no backtracking is performed on previous clauses added. This can prevent the system from finding a solution when it is learning a recursive predicate because of the interaction among clauses: an overgeneral clause may make inconsistent a correct clause still to be learned that calls it.

The second choice point concerns the different abductive explanations that may be available for each example: the choice of an explanation for an example can affect the coverage of future examples. The algorithm does not perform backtracking on example explanations, it just selects one according to the order of the learned clauses which in turn depends on the heuristics that is used and then commits to it.

Finally, we comment that with respect to the generality relations defined in section 3.1 for the separate parts of the hypothesis space, the solution found by the ACL algorithm combines a most general program with a most specific set of integrity constraints. Finding most specific integrity constraints means that these will restrict as much as possible the number of abductive extensions that are allowed by the learned theory. This is desirable since initially, with no constraints, any set of assumptions is allowed: with the learned constraints we want to maximize the information gained from them by maximizing the collection of assumption sets that they exclude.

## §5 ACL for Multiple Predicate Learning

ACL finds a natural application in the problem of Multiple Predicate Learning (MPL) in ILP. In MPL we have a learning situation which is similar to the problem of learning with incompleteness in the background data, since each predicate to be learned forms part of the background theory for the other predicates and the available definitions for the target predicates are incomplete during learning. Multiple predicate learning is a task that poses a number of problems to most ILP systems. These problems and difficulties have been exposed in [17]. In this section we will discuss these problems and show how they can be addressed within the ACL framework.

In multiple predicate learning it is necessary to distinguish between two types of consistency of a learned clause: *local* and *global consistency* of a new clause with respect to the theory learned so far (*current hypothesis*). The following definitions extend those given in [17] by relating the consistency of a clause to the current partial hypothesis. Intuitively, a clause is locally consistent if it does not cover any negative example for its head predicate when it is added to a consistent partial hypothesis. On the other hand, a clause is globally consistent if the theory obtained by adding it to the current partial hypothesis no negative example for any target predicate is covered when this is added to the hypothesis.

**Definition 5.1 (Local consistency)**
Let $H$ be a consistent hypothesis and $c$ a clause for the predicate $p_i$. Then $c$ is **locally consistent with respect to** $H$ if and only if $covers(B, H \cup \{c\}, E^-_{p_i}) = \emptyset$, where $E^-_{p_i}$ are the given negative examples on $p_i$.

**Definition 5.2 (Global consistency)**
Let $H$ be a consistent hypothesis and $c$ a clause for any target predicate. Then $c$ is **globally consistent with respect to** $H$ if and only if $covers(B, H \cup \{c\}, E^-) = \emptyset$, where $E^-$ is the set of negative examples on all target predicates.

By repeating several times a single predicate learning task, we repeatedly add locally consistent clauses to the current partial hypothesis. However, when learning multiple predicates, adding a locally consistent clause to a consistent hypothesis can produce a globally inconsistent hypothesis as it is shown in the next example adapted from [17].

**Example 5.1**
Suppose we want to learn the definitions of $ancestor$ and $father$ from the knowledge base:

$$B = \{parent(a,b), parent(d,b), parent(b,c), male(a), female(b)\}$$

and the training sets:

$$E^+ = \{ancestor(a,b), ancestor(d,c), father(a,b)\}$$
$$E^- = \{ancestor(b,a), ancestor(a,d), father(b,c), father(a,c)\}$$

Suppose that the system has first generated the rules:

$ancestor(X,Y) \leftarrow parent(X,Y).$

$father(X,Y) \leftarrow ancestor(X,Y), male(X).$

The second rule is incorrect but the system has no means of discovering it at this stage, since it is locally and globally consistent with respect to the partial definition for *ancestor*.

Then the system learns the recursive rule for *ancestor*:

$ancestor(X,Y) \leftarrow parent(X,Z), ancestor(Z,Y).$

This clause is locally consistent with respect to the current hypothesis because none of the negative examples for *ancestor* will be covered, but is globally inconsistent because the negative example $father(a,c)$ will be covered.

The system MPL of [17] uses *intentional coverage* and addresses the problem of maintaining the global consistency of the current hypothesis by re-testing the negative examples for all predicates after the addition of a clause and by performing backtracking on clause addition to the theory.

Another problem that can arise in multiple predicate learning concerns the case when scarce training examples, particularly negative examples, are available for a subsidiary predicate. In this case, a system could learn an overgeneral definition for the subsidiary predicate and this may prevent the system from finding a consistent definition for other predicates.

**Example 5.2**

Suppose we want to learn grandfather and father. Let the background theory be:

$P = \{parent(john, mary), male(john),$

$parent(david, steve), male(david), male(steve),$

$parent(steve, jim), male(jim),$

$parent(mary, ellen), female(mary), female(ellen)$

$parent(ellen, sue), female(sue)\}$

and let the training data for both concepts be:

$E^+ = \{grandfather(john, ellen), grandfather(david, jim),$

$father(john, mary)\}$

$E^- = \{grandfather(mary, sue), grandfather(mary, john),$

$father(john, ellen), father(david, jim), father(jim, david)\}$

A system that learns first the rule for father, may learn the overgeneral rule

$father(X,Y) \leftarrow parent(X,Y)$

since it is consistent with the negative examples for $father$. Then, it would not be able to accept the correct rule for grandfather, since

$$grandfather(X,Y) \leftarrow father(X,Z), parent(Z,Y).$$

would cover as well the negative example $grandfather(mary, sue)$.
On the other hand, if the system learns first the above correct rule for grandfather it again needs to recognize that this implies additional negative examples for $father$ in order to avoid the same overgeneral rule for $father$.

## 5.1 M-ACL: a Multiple Predicate Learning framework

The basic idea of performing multiple predicate learning through ACL is to set the target predicates to be learned as abducible predicates and use the abductive information that ACL1 generates on these to link the learning of the different predicates. This information can be used in two inter-related ways. Firstly, it acts as extra training examples for the target predicates. After the generation of each clause by ACL1, the associated assumptions $\Delta^+$ and $\Delta^-$ about other target predicates are added to the training set according to their sign. In effect, training information for one predicate is transformed into training information for other predicates. At the same time, this abductive information generated by ACL1 is used to give us an extra mechanism for ensuring global consistency in the hypothesis in a way similar to abductive truth maintenance systems [44, 32]. The MPL algorithm and system is obtained from ACL1 by encompassing this in a process that uses the abductive information, produced by ACL1, to detect and restore consistency.

The M-ACL algorithm is therefore based on a dynamic set of training examples $E$ for the target predicates that contains the given training examples together with those generated through abduction. It rests on the important observation that, for definite logic programs, we can verify the **global consistency** property of a clause (definition 5.2) by testing only the negative training examples for its head predicate in the **abductively extended** training set of examples.

The M-ACL algorithm shown in Fig. 4 extends that of ACL1 in several ways. An extension of the ACL1 Specialize procedure is used, denoted by Specialize$_M$. This uses *extensional coverage* and tries to generate a new clause $r$ that is correct with respect to the current **extended** set of training examples $E_c$. If this is possible, then the generated clause, $r$, will cover a set of positive examples $E_r^+$ and no negative example ($E_r^- = \emptyset$) with the assumptions $\Delta_r$.

**procedure** M-ACL(
  **inputs :** $E^+, E^-$ : training sets,
    $P$ : background theory,
  **outputs :** $H$ : learned theory, $E_A$ : abduced examples)

$H := \emptyset$
$\Delta := \emptyset$
$E_c := E^+ \cup not\_E^-$
**repeat**
  Specialize$_M(P, H, E_c, \Delta;\ r, E_r^+, E_r^-, \Delta_r)$
  $E_c := E_c \setminus E_r^+$
  $H := H \cup \{r\}$
  Test$(H,\ \Delta_r^-; \Delta_f^-)$
  **while** $\Delta_f^-$ is non-empty:
    Choose$(\Delta_f^-;$ Ab$)$
    Refine$(H,$Ab$,\Delta, E_c;\ H, E_c, \Delta)$
    $\Delta_f^- := \Delta_f^- \setminus \{$Ab$\}$
  **endwhile**
  If $E_r^- \neq \emptyset$ then
    RetractClauses$(H, \Delta, E_r^-, E_c; H, \Delta, E_c)$
  Update$(E_c, \Delta_r; E_c)$
  $\Delta := \Delta \cup \Delta_r$
**until** $E_c^+ = \emptyset$ (*covering loop*)
$E_A := E_c \setminus (E^+ \cup E^-)$
**output** $H, E_A$

**Fig. 4** The M-ACL algorithm

If no rule consistent with the current set of negative examples can be found, then Specialize$_M$ looks for a clause that is consistent only with the original set of examples but covers the subset $E_r^-$ of negative examples generated by abduction. If no such clause can be found, then Specialize$_M$ fails and M-ACL also fails.

We then check if the generated clause, that was found extensionally consistent, is also intensionally consistent. To this purpose, the set of negative assumptions $\Delta_r^- \subseteq \Delta_r$ generated by Specialize$_M$ is tested against the current hypothesis: the assumptions are considered as negative examples that must not be covered. If some of these assumptions are violated ($\Delta_f^-$ denotes this set of violated assumptions), we try to remove these violations by iteratively choosing some assumption(s) $Ab$ from $\Delta_f^-$ and refining the current hypothesis. The refinement consists in specializing (or retracting and re-learning) the existing rules that currently define the target predicate of the assumption(s) $Ab$ and are causing the violation with $Ab$.

If $E_r^-$ is not empty, then the clause is locally but not globally consistent and we backtrack on the clauses that generated the covered examples. These rules are deleted from the current hypothesis, positive examples covered by them are re-added to the training set and assumptions and examples generated by them are removed from $\Delta$ and from the training set. In order to support the backtracking required at this step, the abductive procedure employed by ACL1 is extended to record, for every assumption, the clause responsible for generating it.

These two tests on $\Delta_r^-$ and $E_r^-$, when they are successful (i.e. when both $\Delta_f^-$ and $E_r^-$ are empty), ensure that the next candidate hypothesis, i.e. $H \cup \{r\}$, is globally (intensionally) consistent. We point out that the generation in M-ACL of the candidate clauses by Specialize$_M$ (using extensional coverage) allows the interleaving of learning clauses for different target predicates. M-ACL does not require a given order in which to learn the target predicates: the specialization loop in Specialize$_M$ is initialized with an empty body clause for each target predicate and the same heuristic function is used in order to select the next clause to refine and therefore the next predicate to learn.

Let us now examine how this algorithm and the M-ACL system that is based on it, behaves in the cases of examples 5.1 and 5.2. In example 5.1, suppose the system has generated in the current hypothesis the clauses

$$ancestor(X,Y) \leftarrow parent(X,Y).$$

$father(X,Y) \leftarrow ancestor(X,Y), male(X).$

M-ACL will produce, together with the above clause for $father$, the assumption $\{not\_ancestor(a,c)\}$ due to the negative example $father(a,c)$. These assumptions then become additional negative examples for $ancestor$. Their test does not produce a violation and so at this point the system tries to find a clause covering the remaining positive examples for $ancestor$. The correct solution

$ancestor(X,Y) \leftarrow parent(X,Z), ancestor(Z,Y).$

is not globally consistent, since it covers the new negative example $ancestor(a,c)$ generated from the rule for $father$. However, this clause is locally consistent, since it does not cover any of the negative examples in the original training set of $ancestor$. It is therefore added to the current hypothesis and the system backtracks to the clause that has generated this violating assumption, namely to the clause for $father$. This clause, together with the assumptions that it has generated, are retracted and the examples for $father$ are re-added to the training set for this concept to be learned again. At this point, the system is able to learn the correct rule for $father$

$father(X,Y) \leftarrow parent(X,Y), male(X).$

This example shows one way in which the M-ACL system uses the dynamically generated abductive information on the target predicates to have a focussed mechanism of detection and repair of global inconsistencies allowing us to recover from an incorrect rule. In example 5.2, M-ACL learns first the rule for $grandfather$ because more information is available about it and the heuristic function prefers it to any of the rules for $father$. When M-ACL generates the rule

$grandfather(X,Y) \leftarrow parent(Z,Y), father(X,Z)$

it uses the examples for $father$ as background knowledge making also assumptions about it when this is needed. The above rule will be learned by M-ACL by making the assumptions $\{not\_father(mary, ellen),\ father(david, steve)\}$ that become additional training examples for $father$. From this new training set, the system is then able to generate the correct rule for $father$. Note that, without the new negative example $father(mary, ellen)$, it would have been impossible to generate the correct rule for $father$ and the overgeneral rule $father(X,Y) \leftarrow parent(X,Y)$ would have been learned. Thus M-ACL avoids (in this case) the problem of overgeneralization.

M-ACL does not overgeneralize even if the system first generates the overgeneral rule for $father$. In this case, extensional coverage still allows Specialize$_M$

to generate the correct rule for $grandfather$ and to generate the same negative assumption on $father$ as above. At this stage the M-ACL system will recognize that we have a violation on the assumption Ab=$\{not\_father(mary, ellen)\}$ and the Refine procedure will lead the system to specialize (or re-learn) the rule for $father$ thus producing at the end the same correct and complete hypothesis as above. Hence, independently of the order of learning, the same extra assumptions are generated and used to produce the same final result.

Summarizing, we point out that in effect the M-ACL system uses a hybrid of extensional and intensional coverage: extensional coverage in the generation of candidate clauses using examples of other target predicates as background facts together with an intensional test of the theory on the generated negative assumptions. Its test for global consistency is performed only on a "narrow" subset of the negative examples, testing only the negative abduced examples for the head predicate of the clause under test. In contrast, the system MPL [17)] checks the negative examples for all target predicate after the addition of a clause. In this way M-ACL performs a smaller number of tests with respect to MPL. In section 6.2 some experiments on learning multiple predicates with M-ACL are described and the results obtained are compared with MPL: however, since the available implementation of MPL is only prototypical and uses a semi-automated covering step, no efficiency comparison was possible.

# §6 Experiments

Two series of experiments have been performed in order to show the ability of ACL to (1) learn from incomplete background knowledge and (2) to perform multiple predicate learning. These experiments have been carried out with an implementation of the ACL systems in Prolog. Their main purpose is to show that the theoretical framework of ACL can be realized into a practical system and to confirm that the framework can indeed address problems of learning under incomplete information. Comparisons with other existing systems are carried out more on the level of accuracy of the learned theory rather than computational efficiency (the implementation of ACL is a first implementation with little if any code optimization). These experiments are thus "proof of the principle" experiments to test the basic principles of abductive concept learning and to demonstrate its appropriateness for these types of problems.

## 6.1 Learning from Incomplete Background Knowledge

The main purpose of these experiments was to test how well ACL could learn under incomplete information and to investigate its behaviour under different forms and degrees of incompleteness. The following three datasets are presented here: (1) a database of family relations with varying degree of incompleteness, (2) (real-life) market research questionnaires which is incomplete due to unanswered questions or "don't care" answers and (3) the congressional voting records database from the UCI repository [7].

In these experiments, the results of ACL have been compared with those of FOIL [64], mFOIL [24] and when the data can be represented in attributed-value form with c4.5 [66]. As might be expected, the results of ACL were better in every experiment, with respect to accuracy and compactness of the learned theory, than those of FOIL which does not contain any special facility for missing information. For this reason the specific details of the FOIL results are not presented below showing only comparisons with mFOIL and c4.5 where appropriate[*7].

The mFOIL and c4.5 systems have special techniques for handling imperfect data that can be either noisy or incomplete. The approach of mFOIL for dealing with incomplete data consists in relaxing the completeness requirement for the sufficiency stopping criterion: mFOIL stops adding a clause to the theory when too few positive examples remain for a clause to be *significant* or when no significant clause can be found with expected accuracy greater than the default. The significance test is based on the *likelihood ratio statistic* [46]: a clause is deemed significant if its likelihood ratio is higher than a certain significance threshold. The default value for the significance threshold is 6.64. Unless otherwise specified, mFOIL was run in all the experiments with the parameters set in the following way: the heuristic function is the m-estimate with m=2, the beam size is 5, no negation as failure literals are allowed in the language bias, the minimum number of examples that each rule must cover is 1 and the significance threshold is 6.64.

To deal with incomplete information c4.5 adopts a probabilistic approach as follows. When c4.5 chooses a test on an attribute $A$ and splits the (current) training examples $T$ into subsets $T_j$ according to the outcome of this test, if the attribute $A$ is known for a training example $e$ with value $O_i$ then $e$ is added into the corresponding set of training examples $T_i$ indicating a probability 1.

[*7] The ACL system was also compared with the recent system of ICL-Sat [15] for learning from partial interpretations with favourable results.

Otherwise, if $A$ is unknown for a case $e$, then $e$ is associated with each subset $T_i$ with a weight representing the probability of the case belonging to that subset estimated by the (weighted) relative frequency of an example having a particular value for $A$ amongst all examples for which $A$ is known.

The major part of these experiments has concentrated on investigating the behaviour of the ACL1 subsystem of ACL. This was done to facilitate an equal comparison with existing systems that learn only rules (or only constraints but again used as classification rules). In most cases though full abductive theories have also been learned that include integrity constraints which support the abductive rules generated by ACL1 thus solving the full ACL problem.

### [ 1 ] Learning Family Relations

In this experiment the problem of learning family predicates is considered, e.g. that of learning the concept of father, from a database of family relations [6] containing facts about several other predicates such as parent, son, daughter, grandfather, male and female etc. We performed several experiments with different degree of incompleteness of the background knowledge and compared the results of ACL1 with those of mFOIL.

The complete background knowledge contains, amongst its 740 facts, 72 facts about parent, 31 facts about male and 24 facts about female. The training set contains 36 positive examples of father taken from the family database and 200 negative examples of father that were generated randomly. Experiments were performed from datasets containing 100%, 90%, 80%, 70%, 60%, 50% and 40% of the facts. The incomplete datasets were generated by randomly taking out facts from the background knowledge. The experiments were performed using 5-fold cross validation: the examples were divided into 5 blocks and, for each level of incompleteness, 5 experiments were performed using 4 blocks as the training set and leaving the other one for testing.

The experiments with ACL1 were performed first by considering a background knowledge with no constraints[*8] and then by adding the following integrity constraints:

$\leftarrow male(X), female(X).$
$\leftarrow son(X,Y), female(X).$
$\leftarrow daughter(X,Y), male(X).$

[*8] For this a version of ACL1 was used specific for the case where the integrity constraints have a simple form.

**Table 1** Performance on the family data

| Data | Accuracy | | | Run Times (seconds) | | |
|---|---|---|---|---|---|---|
| | ACL1 | mFOIL | ACL1+IC | ACL1 | mFOIL | ACL1+IC |
| 100% | 1 | 1 | 1 | 38 | 62 | 60 |
| 90% | 1 | 0.992 | 1 | 42 | 877 | 62 |
| 80% | 0.996 | 0.984 | 0.996 | 305 | 926 | 9831 |
| 70% | 1 | 0.975 | 1 | 251 | 676 | 11434 |
| 60% | 1 | 0.984 | 1 | 158 | 555 | 1134 |
| 50% | 0.972 | 0.984 | 1 | 201 | 1091 | 1063 |
| 40% | 0.976 | 0.951 | 0.988 | 171 | 1329 | 745 |

ACL1 learned theories that are simpler, i.e. contain less rules with shorter bodies, than those learned by mFOIL. Table 1 shows the accuracies and run times of the experiment to learn the concept $father$ for the theories generated by ACL1, mFOIL and ACL1 plus integrity constraints for all levels of incompleteness. The values shown are the average values over the 5 folds.

ACL1 was able to learn theories that are more accurate than mFOIL for all levels of incompleteness apart from 50 %, with run times that are significantly lower than mFOIL ones. When constraints are added to the theory, the accuracy of ACL1 improves further for incompleteness 50 % and 40 % but with runtimes that are significantly higher. The slower performance of ACL is due to the computational overhead of finding and maintaining a consistent set of assumptions under which the coverage of positive examples and non-coverage of the negative examples is ensured. This is an additional output that other ILP systems do not provide which is useful for the further learning of constraints in teh second phase of ACL. The cost of consistency checking can potentially be very high and therefore the experimenter should try to control this by minimizing the number and complexity of the constraints used.

Moreover, for these experiments, we have applied the ICL system [18] to solve also the ACL2 problem of learning integrity constraints from the associated assumptions generated in the first phase by ACL1 and thus solving the full ACL problem. In some cases, this generated the constraints that we usually expect from this domain such as

$\leftarrow male(X), female(X)$, or

$\leftarrow parent(X,Y), parent(Y,X)$

In other cases, instead, the generated constraints are more specific, such as

$parent(X, Y) \leftarrow male(X), son(Y, X).$

$parent(X, Y); mother(X, Y) \leftarrow son(Y, X).$

This is due to the fact that the purpose of the generated constraints in ACL2 is just to support the assumptions of the ACL1 part, without considering other data from the background theory, and therefore ICL selects any set of constraints that is sufficient to achieve this specific task.

### [ 2 ] Marketing Research Data

ACL has been used on several sets of real world data from market research questionnaires aiming to understand the possible success or failure of selling a new product. In this subsection we report on one such experiment.

This case concerns a market research on a new soft drink brand. The research was conducted by asking 100 people to taste the drink and to fill a questionnaire regarding the characteristics of the drink and their personal tastes. The concept we want to learn is $buy(X)$ that expresses whether the person would buy the drink or not. Out of the 100 people interviewed, 52 answered that they would buy the product, 32 would not and 16 don't know. Therefore we have 52 positive examples and 32 negative ones.

There are 24 background predicates representing the answers to the questionnaire. Some questions require an answer chosen among a number of values: for example, the question about the aroma of the drink can be answered with "low", "right" or "high". These values have been represented using the predicates $lowaroma(X)$, $rightaroma(X)$ and $higharoma(X)$. Instead, questions requiring a yes-no answer have been represented using a single predicate: for example, whether the person likes natural things is encoded with the predicate $likenatural(X)$.

Some questions are unanswered or have don't care answers and these have been treated as incomplete information in the background. Out of 24 background predicates, 8 are incomplete with degree of incompleteness from 37% (i.e. 37 people out of 100 have not answered or have answered don't care) up to 89%. The incomplete background predicates have been considered as abducibles and integrity constraints have been introduced in order to avoid the abduction of two different answers for the same question. For example, for the question of "overall flavour" we have the following constraint on the abducible predicates that record answers to this question:

**Table 2** Performance on the drinks questionnaire data

| Accuracy | | | Run Times (seconds) | | |
|---|---|---|---|---|---|
| ACL1 | c4.5 | mFOIL | ACL1 | c4.5 | mFOIL |
| 0.8525 | 0.812 | 0.78 | 6.77 | 1.92 | 11.13 |

$\leftarrow goodflavouroverall(X), poorflavouroverall(X).$

ACL1, mFOIL, c4.5 (and FOIL) were run on this data. The performance of ACL1, mFOIL and c4.5 were compared by means of a 5-fold cross validation. The average results for accuracy and runtimes are shown in table 2. ACL1 has found theories that are, on average, more accurate than c4.5 and mFOIL with run times higher than c4.5 but lower than mFOIL. In general, the dominant rules found by ACL1 (and the other systems) were judged to be meaningful by the experts.

The second phase of ACL was also run on this data to find constraints which support the abductive rules and assumptions of ACL1. For example, one of the constraints found was

$\leftarrow goodflavouroverall(X), higharoma(X)$

which (partially) complements the available knowledge on $goodflavouroverall(X)$. On average, the constraints found where again judged to be significant by experts.

## [ 3 ] Congressional Voting Records

This dataset was taken from the UCI repository of machine learning databases [7]. The dataset includes votes for each of the U.S. House of Representatives Congressmen on 16 issues. The aim is to classify a Congressman as a democrat or a republican according to the way he voted on these 16 issues. There are nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

There are 435 examples in this dataset (267 democrats, 168 republicans) and 16 attributes, one for each voted issue. 14 out of 16 attributes have unknown values, for a number of cases that varies from a minimum of 7 to a maximum of 48. The experiments were performed using 10-fold cross validation and the

**Table 3** Performance on the congressional voting data

| Accuracy | | | Run Times (seconds) | | |
|---|---|---|---|---|---|
| ACL1 | c4.5 | mFOIL | ACL1 | c4.5 | mFOIL |
| 0.931 | 0.964 | 0.942 | 306 | 4 | 76 |

results of ACL1 were compared with those of c4.5 and mFOIL. The average accuracies and run times are shown in table 3.

The accuracy of ACL1 is comparable to c4.5 and mFOIl with the one of c4.5 marginally better. The computational efficiency of ACL1 was inferior to both c4.5 and mFOIL but this is to be expected as the ACL1 system used is a first simple implementation with no code optimization. As in the previous experiments its main overhead stems from the extra computation that it performs in order to find necessary assumptions under which the negative examples can not be covered and the consistency checking for these assumptions.

## 6.2 Multiple Predicate Learning

In this section we present some experiments that have been performed with the M-ACL system: (1) learning a definite clause grammar for simple sentences, (2) learning the definitions of the mutually recursive predicates *even* and *odd* and (3) learning multiple family relations. As with the previous experiments their main aim is to demonstrate the basic principles of ACL in its application to multiple predicate learning and to show that M-ACL exhibits desirable properties for this type of learning e.g. that it is independent of the order of the predicates, it maintains global consistency, it can backtrack from a wrong clause and it can avoid overgeneralization.

The same experiments were also tried using the MPL system of [17] in order to compare the learned theories. However, the available implementation of MPL is only prototypical with its covering step semi-automated. It also differs from the theoretical algorithm since it adopts a hill-climbing search instead of beam search. Due to these limitations of the prototype it was difficult to perform any significant comparison either in terms of the quality of the theory learned or the runtimes of the two systems. In fact, the hill-climbing search of the MPL system prevented it from finding solutions for the grammar data and family relations data, while on the *even* and *odd* data MPL was not able to learn the correct theory due to its specific intensional test of examples.

### [ 1 ] Grammar

The data for this experiment is taken from [15]. The aim is to learn the following definite clause grammar for parsing very simple English sentences:

(1) $sent(A, B) \leftarrow np(A, C), vp(C, B).$

(2) $np(A, B) \leftarrow det(A, C), noun(C, B).$

(3) $vp(A, B) \leftarrow verb(A, B).$

(4) $vp(A, B) \leftarrow verb(A, C), np(C, B).$

In [15] Claudien-Sat is used to solve this task starting from different input interpretations.

The first interpretation corresponds to a complete syntactic analysis of the sentence "the dog eats the cat". Therefore the data set contains all the positive and negative facts mentioning the following lists: [the,dog,eats,the,cat], [dog,eats,the,cat], [eats,the,cat], [the,cat], [cat] and []. Another interpretation contains some ungrammatical sentences and corresponds to several attempts to analyze "the cat the cat". It includes all positive and negative facts mentioning the following lists: [the,cat,the,cat], [cat,the,cat], [cat,cat], [the,cat], [cat], [cat,the] and []. Similarly, another interpretation contains all positive and negative facts mentioning the lists [the,cat,eats], [cat,eats], [cat,sings], [the,cat,sings], [dog,cat], [sings], [eats],[the] and [].

M-ACL has learned the above rules in the following order: (2), (3), (1), (4). Note that the definition for *sent* was learned at a point where the definition for *vp* was not complete. This was possible because the system used the examples for *vp* to complete its definition, by exploiting the hybrid form of coverage. In this case the training set was such that no assumption has been necessary for covering the positive examples for *sent*, while some negative assumptions about *np* were necessary in order to avoid the coverage of negative examples for *sent*.

### [ 2 ] Mutually Recursive Predicates

The task consists in learning the mutually recursive definition for the predicates $even(X)$ and $odd(X)$

(1) $even(X) \leftarrow zero(X).$

(2) $odd(X) \leftarrow succ(X, Y), even(Y).$

(3) $even(X) \leftarrow succ(X, Y), odd(Y).$

The background knowledge contains the fact $zero(0)$ and the definition of the predicate $succ(X, Y)$ whose meaning is "$X$ is the successor of $Y$". The training set is obtained from a complete training set containing facts for all the numbers

from 0 to 9 by removing some of these. For example, we may remove the positive examples $odd(1), odd(5), odd(7), even(2), even(6)$ and the negative examples $even(3), even(7), even(9), odd(6), odd(8)$. The training set is therefore given by:
$E^+ = \{odd(3), odd(9), even(0), even(4), even(8)\}$
$E^- = \{even(1), even(5), odd(0), odd(2), odd(4)\}$ M-ACL generated the following output:

```
/* Execution time 0.440000 seconds. Generated rules */

rule(even(A),[zero(A)],c2)
GC: yes, LC: yes
Covered positive examples: [even(0)]
Covered positive abduced examples: []
Covered negative abduced examples: []
Abduced literals: []


rule(even(A),[succ(A,B),odd(B)],c13)
GC: yes, LC: yes
Covered positive examples: [even(8),even(4)]
Covered positive abduced examples: []
Covered negative abduced examples: []
Abduced literals: [[odd(7),c13]]


rule(odd(A),[succ(A,B),even(B)],c21)
GC: yes, LC: yes
Covered positive examples: [odd(9),odd(3)]
Covered positive abduced examples: [odd(7)]
Covered negative abduced examples: []
Abduced literals: [[not(even(3)),c21]]
```

Every rule learned is represented in the form `rule(Head,Body,Identifier)` where `Identifier` is a constant that identifies the rule. The attributes `GC` and `LC` that appear after each rule express whether the rule is, respectively, globally or locally consistent.

M-ACL has learned clause (3) by exploiting the examples for *odd* as background knowledge and by abducing the missing example $odd(7)$. This example is then added to the training set and is covered by clause (2). For clause (3), no negative assumption was necessary for not covering the two negative examples $even(1)$ and $even(5)$ because $odd(0)$ and $odd(4)$ are in the negative training set. Instead, for clause (2), the negative assumption $not\_even(3)$ was necessary to uncover the negative example $odd(4)$. This experiment shows the ability to learn mutually recursive predicates, exploiting both extensional coverage and abduction.

MPL was not able to find a solution in this case due to the intensional coverage test it adopts. MPL first learns clause (1) and then it is not able to learn any of the recursive clauses because, by using intensional coverage, clause (2) would cover only the example $odd(1)$ which is not in the training set, while clause (3) does not cover any example.

### [ 3 ] Multiple Family Relations

Several experiments to learn multiple family relations were carried out. In one such experiment the task is to learn the predicates brother and sibling from a background theory about parent, male and female. The bias allowed the body of the rules for brother to be any subset of

$$\{parent(X,Y), parent(Y,X), sibling(X,Y), sibling(Y,X),\\ male(X), male(Y), female(X), female(Y)\}$$

while the body of the rules for sibling can be any subset of

$$\{parent(X,Y), parent(Y,X), parent(X,Z),\\ parent(Z,X), parent(Z,Y), parent(Y,Z),\\ male(X), male(Y), male(Z), female(X), female(Y), female(Z)\}$$

Therefore, the rules we are looking for are

$$brother(X,Y) \leftarrow sibling(X,Y), male(X).\\ sibling(X,Y) \leftarrow parent(Z,X), parent(Z,Y).$$

The family database considered for these experiments, taken from [17], contains 16 facts about brother, 38 about sibling, 22 about parent, 9 about male and 10 about female. The background knowledge was obtained from this database by considering all the facts about male and female and only 50 % of the facts about parent (selected randomly). The training set contains all the facts about brother and 50 % of the facts about sibling (also selected randomly). Negative examples were generated by making the Closed World Assumption and taking a random sample of the false atoms: 36 negative examples for sibling and 37 for brother. For this problem the abducible predicates are the target predicates *brother* and *sibling* plus the background predicate *parent*.

From this data, M-ACL has constructed first the rule

$$brother(X,Y) \leftarrow sibling(Y,X), male(X).$$

It has exploited both the positive examples of sibling to cover positive examples of brother and negative examples of sibling to avoid covering negative examples for brother. This rule was constructed first because the heuristics preferred it to the rules for sibling, as more information was available for the predicate sibling rather than for parent. When learning this rule, ACL1 has made a number of assumptions on sibling: it has abduced 3 positive facts (that become positive examples for sibling) and 33 negative facts (that become negative examples for sibling). Then, M-ACL constructs the rule for sibling

$$sibling(X, Y) \leftarrow parent(Z, X), parent(Z, Y).$$

using this new training set and making assumptions on parent.

This experiment shows again how M-ACL is able to learn multiple predicates exploiting the information available and generating new data for one predicate while learning another.

## §7 Related Work

The work of this paper builds on earlier proposals in [22] and [26, 50, 51] for learning simpler forms of abductive theories. In [22] the basic definition of Abductive Concept Learning was introduced and various relations among induction and abduction were investigated. Its study of ACL was primarily at the abstract level indicating, through simple examples, the properties that such learning would have with no algorithmic study of the problem. In the current paper, after modifying slightly the definition, the problem of ACL is studied in detail at the theoretical, algorithmic and empirical level establishing firmly its theoretical and practical properties.

The previous works in [26, 50, 51] have studied specific aspects of the general problem of learning with abduction and ACL. Again this was done more at a descriptive level on examples studying simple algorithms for such learning and demonstrating the potential of abduction in addressing some interesting problems in learning. Specifically, in [26] an algorithm was presented that is able to learn theories with exceptions by introducing new abducibles into rules.

In [51] and [50], the problem of ACL is studied but this study is restricted only to the first part of the problem, namely to ACL1. As above, the learning algorithm used is simple (with a depth first search), appropriate only for toy examples of the problem. Moreover, again the emphasis of this work was put on the application to learning recursive predicates and on the problem of learning exceptions by concentrating on the specific case of abduction for negation as failure. The current work has more emphasis on the application to the problems of learning from incomplete background information and multiple predicate learning.

In general, the work in the present paper extends and complements these previous works in several ways by providing a firmer theoretical framework for abduction in learning, a method for learning constraints in the second phase of ACL2, an effective algorithm based on a beam search with an appropriate heuristic function, a practical system for ACL and a thorough empirical study on non-trivial experimental data. In this way the current work establishes firmly the utility of abduction in learning advocated by the previous work.

The use of abduction in learning, either in an implicit or explicit form, has recently been examined by several works [1, 2, 3, 10, 13, 56, 37, 47, 68]. The abductive assumptions generated during learning are then used in different ways depending on the kind of learning task the system is performing. In many cases abduction is used as a useful mechanism that can support some of the activities of the learning system. For example, in theory revision, abduction is used as one of the basic revision operators for the overall learning process [2, 13, 56, 68, 67]. For each individual positive example that is not entailed by the theory, abduction is applied to determine the set of assumptions that would allow it to be proved. These assumptions are then used to suggest where the current theory should be revised. In [13, 2] the assumptions are either added as facts to the theory or new clauses are learned for covering them. Theory revision thus uses induction over abduced examples in order to achieve its task. In addition, some of these systems use abductive assumptions for revising overspecific rules by removing from them the literal(s) that generated the assumption [56]. This type of integration of abduction and induction has been studied in a principled way in [3] where an integrated framework that combines Abductive and Inductive Logic Programming is proposed. The generation of abduced examples in theory revision (refinement) is similar to one of the uses of abduction in ACL where abductive assumptions are generated and used as extra training information either for learning integrity constraints or for learning other predicates in the M-ACL framework for multiple predicate learning. However, in ACL abduction plays an additional and more central role in the definition of the basic learning covering relation.

Abduction is used as well when performing predicate invention [72, 31, 70, 49]: if no literal in the language bias can be found that can make a clause consistent, a literal for a new predicate is generated and added to the clause. Then, examples for the new predicate are generated by means of abduction from the positive and negative examples for the target predicate. Recently, this type of invention of new abducibles has been studied in [35] when learning non-monotonic logic programs. The ACL framework as defined in section 3 of this paper does not allow for the use of new (unknown to the background theory) abducible predicates. This though can be extended to accommodate this possibility. The extension of the theoretical framework is straightforward simply by the introduction in the language of such abducible predicates. In practice though the extension of the ACL algorithm and system in order to accommodate abducible invention is more complicated needing to take into account semantical and practical considerations relating to the incompleteness of the learning data. A first attempt would extend the ACL algorithm so that if no literal in the bias can make the current clause consistent (thus reflecting the incompleteness in the available data), a literal for a new abducible predicate may be added to the clause and the abductive proof procedure used to generate training examples for the new abducible predicate. We can also use similar methods as in M-ACL to exploit this new predicate when learning other predicates thus evaluating the utility of this form of predicate invention in repeated learning as proposed in the recent work of [48]. Further investigation of this problem of abducible invention is required.

Abduction is used also as a suitable mechanism for extending Explanation Based Learning [10, 60] in cases where the given domain theory is incomplete in the description of some of its predicates. These predicates are then treated as abducible and proofs can be completed by abduction before they are generalized.

The work of [71] proposes an approach where abduction is used as the basic covering relation for learning in a different way with respect to us. Abduction is carried out on the concept to be learned rather than on the (incomplete) background predicates. A system, called LAB, is presented that uses a simple, propositional form of abduction in the context of a particular application of learning theories for a diagnostic reasoning model. In this reasoning model, theories are composed of rules of the form $symptom \Leftarrow disorder$ and the task of abduction is to find a (minimum) set of disorders that explains all the symptoms. LAB is given as input a set of training cases each consisting of a set of symptoms together with their correct diagnosis (set of disorders) and it produces a theory such that the correct diagnosis for each training example is a (minimum) abductive explanation. In LAB, therefore, the explanations themselves are considered as the output of the target theory (the target predicates are the abducible *disorder* predicates) requiring that the learned theory respects the input-output couples given in the training cases.

Recently, the deeper relationship between abduction and induction has been the topic of study of two workshops [28, 29] where various (preliminary) works on the integration of abduction in learning have been proposed [1, 68, 47]. Of these, [47] is the closest to our work: it presents a top-down learning algorithm that employs an abductive proof procedure for testing the coverage of examples. They consider a cost for each explanation by assigning a cost to every abducible literal. The minimum cost for explaining examples is then taken into account in a FOIL-like clause evaluation function. As ACL, the system can be applied to learn from incomplete background data.

Several other proposals for learning with incomplete information exist. In attributed-based or propositional learning one common way to handle incomplete information (i.e. missing attribute values) is to replace each example with a missing value with several examples, one for each of the possible values of the attribute, and to associate to each example a fractional weight, representing the conditional (with respect to the class of the example) probability of that particular value. The conditional probability of the different values is estimated with the relative frequency from the set of instances. This is the approach followed by ASSISTANT [8], CN2 [9] and C4.5 [66]. Various approaches to the handling of incomplete information are empirically compared in [65].

An early ILP system that is able to deal with missing information is that of LINUS [54]. This learns first order theories by first translating an ILP problem into an attribute-value representation and by then employing an attribute-value algorithm that handles incomplete information. In this way, it is able to deal both with missing arguments and missing facts in the background knowledge. The drawbacks of this approach are the large number of attributes that may be necessary and the restriction of the language of target programs to determinate Datalog clauses.

The systems FORCE2 [11], SKILit [39], CHILLIN [74] and FOIL-I [38] were designed to learn recursive predicates from incomplete information in the training set but not in the background knowledge. In [73] the authors propose several frameworks for learning from partial interpretations. A particular framework that can learn form incomplete information is that of learning from satisfiability [15]. This framework is more general than ACL as both the examples and the hypotheses can be clausal theories. On the other hand, theories learned by this framework correspond only to the integrity constraints part of an abductive theory with no (or a trivial default) rule part.

A problem that is related to learning from incomplete data is that of learning from noisy or in general imperfect data [24, 53]. This problem is handled by relaxing the requirements of consistency and completeness in the necessity and sufficiency stopping criteria and by adopting special heuristic functions for guiding the search. In general, these systems see incompleteness as a special case of noise and hence it may be that methods for handling noise are too coarse for incompleteness. Indeed, when we know in which predicates the incompleteness lies, then we would expect that we can use more specialised techniques, like the ACL framework, to get better results than the more general methods for noise. This is confirmed by some of the experiments presented in section 6 of this paper.

As we have seen, ACL can use integrity constraints as part of its background knowledge. Learning from integrity constraints was first examined in [13] and [12]. Recently, the system Progol [58] is able to learn from integrity constraints. However, in these cases, integrity constraints are used to impose conditions on the target predicates that need to be respected by the learned clauses. In ACL, instead, constraints impose conditions on background rather than target predicates and are used to restrict the assumptions of background facts rather than for specializing the clauses.

On the other hand, ACL also learns new integrity constraints as part of its final learned theory. Hence ACL involves a combination of explanatory (predictive) and confirmatory (descriptive) induction. Although several ILP systems (e.g. [58, 18, 14]) can produce theories that combine rules and integrity constraints, all of these use a single form of induction to generate both parts of the theory. In our work, abduction helps to integrate in a single framework the different ILP learning methods based on predictive and descriptive induction. From this point of view ACL falls within the framework of integration of these two ILP settings proposed in [20]. Finally, we point out that, as abductive theories are non-monotonic in nature, ACL can provide us with a method for learning non-monotonic theories. It can thus be used to address similar learning problems as those tackled in the work of [33, 5, 21].

## §8 Conclusions

We have studied the new learning framework of Abductive Concept Learning (ACL) setting up its theoretical foundations and developing a first system for it. This framework integrates abduction and induction extending the Inductive Logic Programming paradigm in order to learn abductive theories: both the background and target theories are abductive theories and deductive entailment as the coverage relation in ILP is replaced by an abductive entailment in the learning problem of ACL. The main application of ACL is learning from incomplete information.

The ACL problem can be decomposed into two subproblems, ACL1 and ACL2, the first consisting of learning the rule part of the abductive theory and the second consisting of learning the constraint part. ACL1 is an explanatory (predictive) learning problem, while ACL2 is a confirmatory (descriptive) learning problem. Based on this decomposition, a system for learning in this new framework has been developed that solves the ACL problem by first solving ACL1 and then ACL2. These separate problems are solved using and adapting algorithms and techniques from the existing ILP frameworks for explanatory concept learning from examples and descriptive learning from interpretations. In this way, ACL represents a non-trivial and useful integration of these two main ILP settings.

The ACL framework allows us also to tackle effectively the problem of multiple predicate learning, where each predicate is required to be learned from the incomplete data for the other predicates. By employing abduction we are able to link the learning of the different predicates and ensure the coherence among the definitions learned for them. A separated system for multiple predicate learning, called M-ACL, has been developed by suitably modifying the ACL system.

Several experiments were performed with data from various sources to test ACL1 (and ACL) on problems of learning from incomplete information in comparison with other systems such as FOIL, mFOIL and, when appropriate, with c4.5. The performance of ACL1 in terms of accuracy and compactness of the learned theory was superior to FOIL, which does not have any special facility for missing information, and comparable (in many cases marginally better) to those of mFOIL and c4.5. ACL1 also compared favourable with the ICL-Sat system adapted from ICL for learning with partial interpretations. Other experiments were also done that confirmed the ability of M-ACL to learn multiple predicates. The computational efficiency of ACL1 was inferior to that of c4.5 and FOIL. This is due partly to the initial non-optimized implementation of ACL1 but mainly to the fact that ACL1 computes the additional output of supporting assumptions for the learned rules that are useful for the further learning of constraints in the second phase of ACL. These "proof of the principle" experiments demonstrate the ability of ACL to learn with incomplete information and its appropriate use for multiple predicate learning.

There are several directions for further work. One such direction is the more efficient development of the ACL system where the checking of integrity constraints is controlled. Another possibility is the study of predicate invention within the ACL framework through the natural use of new (unknown) abducible predicates. At a deeper level another important direction of future work concerns the separation of the full ACL problem to its two subproblems. The development of the ACL algorithm and system in this paper was heavily based on the separation of the full ACL problem into the ACL1 and ACL2 subproblems, adapting traditional ILP techniques to solve these. Further work is needed to examine other ways of synthesizing these subproblems and more importantly to develop algorithms that would search directly the full space of abductive theories. This involves the definition of generality orderings for this space and the development of suitable refinement operators that would allow the simultaneous learning of both parts (rules and constraints) of an abductive theory.

***Acknowledgment*** We would like to thank the members of the $ILP^2$ project and the participants of the ECAI'96 and IJCAI'97 workshops on Abduction and Induction for many useful discussions. We also thank Evelina Lamma, Paola Mello and Luis Moniz Pereira for their comments on a preliminary version of this paper. This work has been partially supported by the European Union, ESPRIT project $ILP^2$ and the Department of Planning of the Government of Cyprus.

## *References*

1) A. Abe. The relation between abductive hypotheses and inductive hypotheses. In Flach and Kakas [30].
2) H. Adé and M. Denecker. RUTH: An ILP theory revision system. In *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, 1994.
3) H. Adé and M. Denecker. AILP: Abductive inductive logic programming. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.
4) K.R. Apt and M. Bezem. Acyclic programs. In *Proceedings of the 7th Interna-*

*tional Conference on Logic Programming*, pages 579–597, Jerusalem, 1990.

5) M. Bain and S. Muggleton. Non-monotonic learning. In J.E. Hayes-Michie and E. Tyugu, editors, *Machine Intelligence*, volume 12. Oxford University Press, 1991.

6) H. Blockeel and L. De Raedt. Inductive database design. In *Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems*, volume 1079 of *Lecture Notes in Artificial Intelligence*, pages 376–385. Springer-Verlag, 1996.

7) E. Keogh C. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

8) B. Cestnik, I. Knononenko, and I. Bratko. ASSISTANT 86: A knowledge elicitation tool for sophisticated users. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning*, pages 31–45. Sigma Press, Wilmslow, UK, 1987.

9) P. Clark and R. Boswell. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

10) W. W. Cohen. Abductive explanation-based learning: A solution to the multiple inconsistent explanation problem. *Machine Learning*, 8:167–219, 1992.

11) William W. Cohen. Pac-learning a restricted class of recursive logic programs. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 86–92, Washington, D.C., 1993.

12) L. De Raedt. *Interactive Theory Revision: An Inductive Logic Programming Approach*. Academic Press, 1992.

13) L. De Raedt and M. Bruynooghe. Belief updating from integrity constraints and queries. *Artificial Intelligence*, 53:291–307, 1992.

14) L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

15) L. De Raedt and L. Dehaspe. Learning from satisfiability. Technical report, Katholieke Universiteit Leuven, 1996.

16) L. De Raedt and S. Džeroski. First order $jk$-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.

17) L. De Raedt, N. Lavrač, and S. Džeroski. Multiple predicate learning. In S. Muggleton, editor, *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, pages 221–240. J. Stefan Institute, 1993.

18) L. De Raedt and W. Van Lear. Inductive constraint logic. In *Proceedings of the 5th International Workshop on Algorithmic Learning Theory*, 1995.

19) M. Denecker and D. De Schreye. SLDNFA: an abductive procedure for normal abductive programs. In K. R. Apt, editor, *Proceedings of the International Joint Conference and Symposium on Logic Programming*, pages 686–700, 1992.

20) Y. Dimopoulos, S. Džeroski, and A. C. Kakas. Integrating explanatory and descriptive learning in ILP. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

21) Y. Dimopoulos and A. C. Kakas. Learning non-monotonic logic programs: Learning exceptions. In *Proceedings of the 8th European Conference on Machine Learning*, 1995.

22) Y. Dimopoulos and A. C. Kakas. Abduction and inductive learning. In *Advances in Inductive Logic Programming.* IOS Press, 1996.

23) B. Duval. Abduction for Explanation Based Learning. In *Proceedings of the European Working Session on Learning*, number 482 in LNCS, pages 348–360, 1991.

24) S. Džeroski and I. Bratko. Handling noise in inductive logic programming. In S. Muggleton, editor, *Proceedings of the 2nd International Workshop on Inductive Logic Programming*, Report ICOT TM-1182, 1992.

25) K. Eshghi and R. A. Kowalski. Abduction compared with Negation by Failure. In *Proceedings of the 6th International Conference on Logic Programming*, 1989.

26) F. Esposito, E. Lamma, D. Malerba, P. Mello, M. Milano, F. Riguzzi, and G. Semeraro. Learning abductive logic programs. In Flach and Kakas [28]. Available on-line at `http://www.cs.bris.ac.uk/~flach/ECAI96/`.

27) P. A. Flach. *Conjectures: An Inquiry Concerning the Logic of Induction.* PhD thesis, Katholieke Universiteit Brabant, 1995.

28) P. A. Flach and A. C. Kakas, editors. *Proceedings of the ECAI'96 Workshop on Abductive and Inductive Reasoning*, 1996. Available on-line at `http://www.cs.bris.ac.uk/~flach/ECAI96/`.

29) P. A. Flach and A. C. Kakas, editors. *Proceedings of the IJCAI'97 Workshop on Abductive and Inductive Reasoning*, 1997. Available on-line at `http://www.cs.bris.ac.uk/~flach/IJCAI97/`.

30) P. A. Flach and A. C. Kakas, editors. *Abductive and Inductive Reasoning.* Pure and Applied Logic. Kluwer, 1998.

31) P. Flener. Inductive logic program synthesis with Dialogs. In S. Muggleton, editor, *Inductive Logic Programming: Selected Papers from the 6th International Workshop*, pages 175–198. Springer-Verlag, 1997.

32) L. Giordano and A. Martelli. Generalized stable model semantics, truth maintenance and conflict resolution. In *Proceedings of the 7th International Conference on Logic Programming*, pages 427–411, Jerusalem, 1990. MIT Press.

33) N. Helft. Induction as nonmonotonic inference. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 149–156. Morgan Kaufmann, 1989.

34) K. Inoue. Hypothetical reasoning in logic programs. *Journal of Logic Programming*, 18:191–227, 1994.

35) K. Inoue and H. Haneda. Learning abductive and nonmonotonic logic programsg. In P. A. Flach and A. C. Kakas, editors, *Abductive and Inductive Reasoning*, Pure and Applied Logic. Kluwer, 1999.

36) K. Inoue and C. Sakama. On the equivalence between disjunctive and abductive logic programs. In *In proceedings of ICLP94*, pages 489–503, 1994.

37) K. Inoue and C. Sakama. Abductive framework for nonmonotonic theory change. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 204–210, 1995.

38) N. Inuzuka, M. Kamo, N. Ishii, H. Seki, and H. Itoh. Top-down induction of logic programs from incomplete samples. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 119–136. Stockholm University, Royal Institute of Technology, 1996.

39) A. Jorge and P. Brazdil. Architecture for iterative learning of recursive definitions. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 206–218. IOS Press, 1996.

40) A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2:719–770, 1993.

41) A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in AI and Logic Programming*, volume 5, pages 233–306. Oxford University Press, 1997.

42) A. C. Kakas and P. Mancarella. Database updates through abduction. In R. Sacks-Davis D. McLeod and H. Schek, editors, *Proceedings of the 16th International Conference on Very Large Databases, VLDB-90*, pages 650–661. Morgan Kaufmann, 1990.

43) A. C. Kakas and P. Mancarella. Generalized stable models: a semantics for abduction. In *Proceedings of the 9th European Conference on Artificial Intelligence*, 1990.

44) A. C. Kakas and P. Mancarella. On the relation between truth maintenance and abduction. In *Proceedings of the 2nd Pacific Rim International Conference on Artificial Intelligence*, 1990.

45) A. C. Kakas and F. Riguzzi. Learning with abduction. In *Proceedings of the 7th International Workshop on ILP*, 1997.

46) J. Kalbfleish. *Probability and Statistical Inference*, volume II. Springer, New York, 1979.

47) T. Kanai and S. Kunifuji. Extending inductive generalization with abduction. In Flach and Kakas [30].

48) K. Khan, S. Muggleton, and R. Parson. Repeat learning using predicate invention. In *Proceedings of the 8th International Workshop on Inductive Logic Programming*, 1998.

49) B. Kijsirikul, M. Numao, and M. Shimura. Discrimination-based constructive induction of logic programs. In *Proceedings of the 10th National Conference on Artificial Intelligence.* Morgan Kaufmann, 1992.

50) E. Lamma, P. Mello, M. Milano, and F. Riguzzi. Integrating induction and abduction in logic programming. In P. P. Wang, editor, *Proceedings of the Third Joint Conference on Information Sciences*, volume 2, pages 203–206, 1997.

51) E. Lamma, P. Mello, M. Milano, and F. Riguzzi. Integrating induction and abduction in logic programming. To appear on Information Sciences, 1998.

52) N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.

53) N. Lavrač, S. Džeroski, and I. Bratko. Handling imperfect data in inductive logic programming. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 48–64. IOS Press, 1996.

54) N. Lavrač, S. Džeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with LINUS. In Y. Kodratoff, editor, *Proceedings of the 5th European Working Session on Learning*, volume 482 of *Lecture Notes in Artificial Intelligence*, pages 265–281. Springer-Verlag, 1991.

55) J. Lloyd. *Foundations of Logic Programming.* Springer Verlag, Berlin, second edition, 1987.

56) R. Mooney. Integrating abduction and induction in machine learning. In Flach and Kakas [30].

57) S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–317, 1991.

58) S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

59) S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

60) P. O'Rourke. Abduction and explanation-based learning: Case studies in diverse domains. *Computational Intelligence*, 10:295–330, 1994.

61) G.D. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

62) G.D. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6, pages 101–124. Edinburgh University Press, 1971.

63) D. Poole, R. G. Goebel, and Aleliunas. Theorist: a logical reasoning system for default and diagnosis. In Cercone and McCalla, editors, *The Knowledge Fronteer: Essays in the Representation of Knowledge*, Lecture Notes in Computer Science, pages 331–352. Springer-Verlag, 1987.

64) J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

65) J. R. Quinlan. Unknown attribute values in induction. In *Proceedings of the Sixth International Machine Learning Workshop*, pages 164–168, San Mateo, CA, 1991. Morgan Kaufmann.

66) J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

67) B.L. Richards and R.J. Mooney. Refinement of first-order Horn-clause domain theories. *Machine Learning*, 19(2):95–131, 1995.

68) C. Sakama. Computing induction through abduction. In Flach and Kakas [30].

69) K. Satoh and N. Iwayama. A query evaluation method for abductive logic programming. In *In proceedings of the 1992 Joint International Conference and Symposium on Logic Programming*, pages 671–685, 1992.

70) I. Stahl. Predicate invention in inductive logic programming. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 34–47. IOS Press, 1996.

71) C. Thompson and R. Mooney. Inductive learning for abductive diagnosis. In *Proceedings of the 12th National Conference on Artificial Intelligence*, 1994.

72) R. Wirth and P. O'Rorke. Constraints on predicate invention. In L. Birnbaum and G. Collins, editors, *Proceedings of the 8th International Workshop on Machine Learning*, pages 457–461. Morgan Kaufmann, 1991.

73) S. Wrobel and S. Džeroski. The ILP description learning problem: Towards a genearl model-leve definition of data mining in ILP. In *Proceedings of the Fachgruppentreffen Maschinelles Lernen*, 1995.

74) J.M. Zelle, R.J. Mooney, and J.B. Konvisser. Combining top-down and bottom-up techniques in inductive logic programming. In W.W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning*, pages 343–351. Morgan Kaufmann, 1994.

# §1 Proof of Theorem 3.1 on Equivalence of ACL with ACL1 and ACL2

**Theorem 1.18**
Let $T_{ACL1} = \langle P \cup P', A, I\rangle$, $\Delta^+$ and $\Delta^-$ be the solution of ACL1 given training sets $E^+$ and $E^-$, background theory $T = \langle P, A, I\rangle$ and space of possible programs $\mathcal{P}$. Moreover, let $T' = \langle P \cup P', A, I \cup I'\rangle$ be the solution to ACL2 given the previous solution of ACL1 and hypothesis space $\mathcal{I}$. Then $T'$ is a solution to the ACL problem that has $E^+$ and $E^-$ as training sets, $T$ as background theory and $\mathcal{P}$ and $\mathcal{I}$ as spaces of possible programs and constraints.

**Proof** We first prove that $T' \models_A E^+$ and then that $\forall e^- \in E^-, T' \not\models_A e^-$.
*Proof of* $T' \models_A E^+$*:* from ACL1 we have that $M_{P\cup P'}(\Delta^+) \models E^+$. From ACL1 and ACL2 we have, respectively, that $M_{P\cup P'}(\Delta^+) \models I$ and $M_{P\cup P'}(\Delta^+) \models I'$, therefore $M_{P\cup P'}(\Delta^+) \models I \cup I'$. This, together with $M_{P\cup P'}(\Delta^+) \models E^+$, proves that $\Delta^+$ is an abductive explanation for $E^+$ in $T'$.
*Proof of* $\forall e^- \in E^-, T' \not\models_A e^-$*:* from ACL1 we have that $T_{ACL1} \models_A not\_E^-$ with $\Delta^-$. From the definition of strong abductive explanation of a conjunction of goals (definition 2.8) $\Delta^-$ is also a strong abductive explanation for $not\_e^-$ for every $e^- \in E^-$. Therefore, from property 2.1 in section 2 we have

$$\forall \Delta_{e^-} : T_{ACL1} \models_A e^- \text{ with } \Delta_{e^-}, \quad \exists \bar{l} \in \Delta_{e^-} : l \in \Delta^-$$

Since the integrity constraints in $T'$ are a superset of those in $T_{ACL1}$ and the rule part is the same, the set of explanations for $e^-$ in $T'$ is a subset of those for $e^-$ in $T_{ACL1}$.

The constraints $I'$ generated by ACL2 make inconsistent each of the complements in $\Delta^-$ and hence, for every such $\Delta_{e^-}$, there exists an $\bar{l} \in \Delta_{e^-}$ such that $\{\bar{l}\}$ is inconsistent with $I'$. From the restricted form of the integrity constraints in $I'$, any superset of $\{\bar{l}\}$, in particular $\Delta_{e^-}$, cannot satisfy the integrity constraints. Therefore, any $\Delta_{e^-}$ is not a consistent extension of $T'$ and hence $T' \not\models_A e^-$ as required. ■

# §2 Proof of Theorem 4.1 on Soundness of ACL

Let us first give the proof of proposition 2.1 that will be needed for proving theorem 4.1.

**Proposition 2.1**
Let $T = \langle P, A, I\rangle$ be an abductive theory in its three-valued version and let $\Delta_1$ and $\Delta_2$ be two strong abductive explanations of, respectively, $G_1$ and $G_2$, where $G_1$ and $G_2$ can be either positive or negative goals. If $\Delta_1 \cup \Delta_2$ is self-consistent, then $\Delta_1 \cup \Delta_2$ is a strong abductive explanation for $G_1 \wedge G_2$.

**Proof** We first consider the case where $G_1$ and $G_2$ are two positive goals. We need to verify the two conditions of definition 2.7.

Let us first prove that $M(\Delta_1 \cup \Delta_2)$ is a generalized model. Consider $\Delta_2$ as a self-consistent extension of $\Delta_1$. Since $\Delta_1$ is a strong abductive explanation, any self-consistent extension $\Delta'$ of $\Delta_1$ for which $M(\Delta') \models I$, is such that $M(\Delta_1 \cup \Delta') \models I$. Taking $\Delta' = \Delta_2$, since $\Delta_2$ is an abductive explanation, $M(\Delta_2) \models I$ holds and so $M(\Delta_1 \cup \Delta_2) \models I$. Therefore $M(\Delta_1 \cup \Delta_2)$ is a generalized model. Since $P \cup \Delta_1 \cup \Delta_2$ is a definite logic program, $M(\Delta_1 \cup \Delta_2) \supseteq M(\Delta_1)$ and $M(\Delta_1 \cup \Delta_2) \supseteq M(\Delta_2)$ therefore $M(\Delta_1 \cup \Delta_2) \models G_1$ and $M(\Delta_1 \cup \Delta_2) \models G_2$, so $\Delta_1 \cup \Delta_2$ is an abductive explanation for both $G_1$ and $G_2$.

To show that $\Delta_1 \cup \Delta_2$ satisfies the second condition of definition 2.7 consider a set $\Delta'$ such that $\Delta' \cup \Delta_1 \cup \Delta_2$ is self-consistent and $M(\Delta') \models I$. We need to prove that $M(\Delta' \cup \Delta_1 \cup \Delta_2) \models I$. Consider the set $\Delta'' = \Delta' \cup \Delta_2$. Since $\Delta_1$ is strong and $\Delta'' \cup \Delta_1$ is self-consistent, if $M(\Delta'') \models I$ then $M(\Delta_1 \cup \Delta'') \models I$ would follow. But $M(\Delta'') \models I$ is true since $\Delta_2$ is strong, $\Delta' \cup \Delta_2$ is self-consistent and $M(\Delta') \models I$. Therefore the second condition of definition 2.7is proved.

Consider now the case where we have two negative goals $G_1 = not\_O_1$ and $G_2 = not\_O_2$. In order for $\Delta_1 \cup \Delta_2$ to be a strong abductive explanation for $G_1$ and $G_2$, we need to show that the conditions of definition 2.8 are satisfied. The fact that $M(\Delta_1 \cup \Delta_2)$ is a strong generalized model can be proved in the same way as for positive goals. To show that $M(\Delta_1 \cup \Delta_2) \not\models O_1$ (and similarly $M(\Delta_1 \cup \Delta_2) \not\models O_2$) we note that $\Delta_1$ is a strong abductive explanation for $not\_O_1$ and hence if $\Delta_2$ were an abductive explanation for $O_1$, then $\Delta_1 \cup \Delta_2$ would not be self-consistent which contradicts the hypothesis of the statement. Next we show the second condition of definition 2.8, i.e., that for every $\Delta'$ that is an abductive explanation for $O_1$ (or $O_2$), then $(\Delta_1 \cup \Delta_2) \cup \Delta'$ is not self-consistent. This follows directly from the fact that if $\Delta'$ is an explanation for $O_1$ ($O_2$), since $\Delta_1$ ($\Delta_2$) is strong, then $\Delta' \cup \Delta_1$ ($\Delta' \cup \Delta_2$) is not self-consistent and hence $\Delta_1 \cup \Delta_2 \cup \Delta'$ is not self-consistent. The other case where one of the goals is positive and the other is negative can be shown similarly.

■

**Theorem 2.22 (Soundness)**
The ACL algorithm is sound.

**Proof** ACL finds a solution $T'$ of ACL by solving the ACL1 and ACL2 subproblems in sequence. Theorem 3.1 states that the combination of the solutions of ACL1 and ACL2 gives a solution for ACL. Therefore, to prove the soundness of ACL, it is sufficient to prove that the solutions found by the algorithms for ACL1 and ACL2 satisfy their respective subproblem definitions.

For the second phase of ACL2, this is guaranteed by the correctness of the ICL [18] algorithm or of any other sound method used for discriminating between positive and negative interpretations. It remains therefore to prove that the procedure ACL1 is sound with respect to the ACL1 definition, i.e. that, given the background theory $T = \langle P, A, I\rangle$ and training sets $E^+$ and $E^-$, the program $T_{ACL1} = \langle P \cup P', A, I\rangle$ and the sets $\Delta^+$ and $\Delta^-$ that are generated by the algorithm are such that

$$T_{ACL1} \models_A E^+ \text{ with } \Delta^+ \tag{1}$$

$$T_{ACL1} \models_A \mathit{not\_E}^- \text{ with } \Delta^- \tag{2}$$

$$\Delta^+ \cup \Delta^- \text{ is self-consistent} \tag{3}$$

ACL1 learns the program $T_{ACL1}$ by iteratively adding a new clause to the current hypothesis, initially empty. Each clause is tested by trying an abductive derivation for each positive and for each (negated) negative example.

Suppose that clauses are learned in the following order: $c_1, \ldots, c_l$. Let $H_1, \ldots, H_l$ be the successive partial hypotheses, with $H_0 = \emptyset$ and $H_k = H_{k-1} \cup \{c_k\}$, and let $T_k = \langle P \cup H_k, A, I\rangle$. Let also $E_k^+ = \{e_{k,1}^+, \ldots, e_{k,n_k}^+\}$ be the set of positive examples whose conjunction is covered by clause $c_k$ and let $E^+ = \{e_1^+, \ldots, e_n^+\}$, $E^- = \{e_1^-, \ldots, e_m^-\}$ be the complete sets of positive and negative examples.

For each clause $c_k$, we define two sets of abductive assumptions $\Delta_k^{in}$ and $\Delta_k^{out}$. $\Delta_k^{in}$ is the initial set of assumptions under which the testing of examples with this clause starts. $\Delta_k^{out}$ is the final set of assumptions produced in the derivations of all the examples in $E_k^+$ and in $E^-$. The input sets $\Delta_k^{in}$ are defined recursively via $\Delta_k^{in} = \Delta_{k-1}^{in} \cup \Delta_{k-1}^{out}$ for $k = 2, \ldots, l$, with $\Delta_1^{in} = \emptyset$. The output sets $\Delta_k^{out}$ are given by $\Delta_k^{out} = \Delta_k^+ \cup \Delta_k^-$ with

$$\Delta_k^+ = \bigcup_{i=1,\ldots,n_k} \Delta_{e_{k,i}^+}$$

$$\Delta_k^- = \bigcup_{j=1,\ldots,m} \Delta_{k,not\_e_j^-}$$

where $\Delta_{e_{k,i}^+}$ is the explanation for example $e_{k,i}^+$ and $\Delta_{k,not\_e_J^-}$ is the explanation for $not\_e_j^-$ in the theory $T_k = \langle P \cup H_k, A, I\rangle$.

We will show that each abductive explanation $\Delta_{e_{k,i}^+}$ and $\Delta_{k,not\_e_J^-}$ is a strong abductive explanation in the theory $T_k$. These explanations are constructed successively with the explanation for each example forming part of the input for the next example. Therefore, if the input sets $\Delta_k^{in}$ are strong, then also the individual explanations are strong, by the correctness (with respect to definition 2.9) of the abductive derivation used by the algorithm and the property of proposition 2.1 that the union of strong explanations is strong. Note also that the successive test of the examples by the abductive derivation in the algorithm ensures that these individual explanations are self-consistent with each other required for the application of proposition 2.1.

Hence we need to show that $\Delta_k^{in}$ are strong abductive extensions in $T_k$, for $k = 1, \ldots, l$. We do this by induction on $k$. For $k = 1$, $\Delta_1^{in} = \emptyset$ which is a strong abductive extension because, by the assumptions on the hypothesis spaces of the integrity constraints and programs, it always satisfies any set of constraints and it trivially satisfies the strong property in definition 2.7. Suppose that $\Delta_k^{in}$ is strong in $T_k$, we have to prove that $\Delta_{k+1}^{in}$ is strong in $T_{k+1}$. We first prove that $\Delta_{k+1}^{in}$ is strong in $T_k$. $\Delta_{k+1}^{in} = \Delta_k^{in} \cup \Delta_k^{out}$ is the union of strong abductive extensions of $T_k$: $\Delta_k^{in}$ is strong by the inductive hypothesis and $\Delta_k^{out}$ is strong because is the union of strong explanations computed successively by the correct abductive derivations of the algorithm starting from the strong extension $\Delta_k^{in}$. Also the derivations ensures that all these explanations are self-consistent with each other. Therefore, by proposition 2.1, $\Delta_{k+1}^{in}$ is a strong abductive extension of $T_k$.

We still need to show that $\Delta^{in}_{k+1}$ is a strong extension of $T_{k+1}$. This can be done by directly verifying the conditions in the definition 2.7 of strong abductive extension. Since the integrity constraints $I$ and the background program $P$ do not contain any target predicate, their satisfaction is independent from the addition of any clause for the target predicates. Therefore, as $\Delta^{in}_{k+1}$ satisfies $I$ in $T_k$, it does so in $T_{k+1}$ as well. We also need to show that, for any set $\Delta'$ such that $\Delta^{in}_{k+1} \cup \Delta'$ is self-consistent and $\Delta'$ satisfies $I$ in $T_{k+1}$, $\Delta^{in}_{k+1} \cup \Delta'$ must also do so in $T_{k+1}$. From the independence of $I$ and $P$ from the target predicates, $\Delta'$ satisfies $I$ in $T_{k+1}$ implies that $\Delta'$ satisfies $I$ in $T_k$. Since $\Delta^{in}_{k+1}$ is strong in $T_k$, then $\Delta^{in}_{k+1} \cup \Delta'$ satisfies $I$ in $T_k$. Again, the independence of $I$ and $P$ from the target predicates gives that $\Delta^{in}_{k+1} \cup \Delta'$ satisfies $I$ in $T_{k+1}$.

We can now show the ACL1 conditions with

$$\Delta^+ = \bigcup_{k=1,\ldots,l} \bigcup_{i=1,\ldots,n_k} \Delta_{e^+_{k,i}}$$

$$\Delta^- = \bigcup_{k=1,\ldots,l\ j=1,\ldots,m} \Delta_{k,not\ e^-_j}$$

which, by construction, are the final sets returned by the ACL1 algorithm. We first show that all the explanations for the individual examples are strong abductive explanations in the final theory $T_l = T_{ACL1}$ from the fact that they are strong in their respective theories $T_k$. This follows in the same way as we have shown above that $\Delta^{in}_{k+1}$ is strong in $T_{k+1}$ from the fact that it is strong in $T_k$.

We also know that all these individual explanations are self-consistent with each other. This follows directly from their successive construction in the algorithm satisfying the abductive derivability of definition 2.9. Hence $\Delta^+ \cup \Delta^-$ is self-consistent and the third condition (3) of ACL1 is satisfied. Moreover, by proposition 2.1, the union $\Delta^+$ is then also a strong abductive explanation of $E^+_1, \ldots, E^+_l$ in $T_{ACL1}$. From the sufficiency stopping criterion we have that $E^+_1 \cup \ldots \cup E^+_l = E^+$, therefore $\Delta^+$ is a strong abductive explanation of $E^+$ in $T_{ACL1}$ and condition 1 is satisfied. Similarly, by proposition 2.1, the union $\Delta^-$ is a strong abductive explanation of $E^-$ in $T_{ACL1}$ and condition 2 is satisfied.

■

## §3 Abductive Proof Procedure

In the following we recall the abductive proof procedure for ALP, taken from [44)], used as a basis for the abductive coverage procedure in the ACL1 algorithm.

This ALP procedure is applied to abductive theories $T = \langle P, A, I\rangle$ in their positive form. Thus the abducibles $A$ contain predicates ($a \in A$) for positive assumptions and predicates ($not\_a \in A$) for negative assumptions. The integrity constraints in $I$ are restricted to have a denial form, $\neg(B_1 \wedge \ldots \wedge B_m \wedge \neg A_1 \wedge \ldots \wedge \neg A_k)$ (written here in logic programming style as goals $\leftarrow (B_1, \ldots, B_m, \neg A_1, \ldots, \neg A_k)$, with at least one abducible with no definition in $P$ appearing in $B_1, \ldots, B_m$. Integrity constraints in the range-restricted clausal form, $A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m$, are first transformed into the equivalent denial above before they are used by the abductive procedure.

This procedure also assumes that the program $P$ of $T$ contains no definitions for the abducible predicates ie. no rule (or fact) in $P$ has in its head an abducible predicate. When the program contains such definitions the abductive theory $T = \langle P, A, I\rangle$ can be first transformed so that no such definitions exist. For each abducible predicate $p$ that contains a partial definition in $P$ we add a new abducible $\delta_p$ to the set of abducibles $A$, we remove $p$ from $A$ and we add the rule $p(\vec{X}) \leftarrow \delta_p(\vec{X})$ to the program $P$. In this way, if $p(\vec{c})$ can not be derived using the partial definition for $p$, it can be derived by abducing $\delta_p(\vec{c})$ thus effectively abducing $p$.

The procedure is composed of two phases: abductive derivation and consistency derivation.

**Abductive derivation**

An abductive derivation from $(G_1\ \Delta_1)$ to $(G_n\ \Delta_n)$ in $\langle P, A, I\rangle$ via a safe selection rule $R$, of a literal from a goal, is a sequence

$$(G_1\ \Delta_1), (G_2\ \Delta_2), \ldots, (G_n\ \Delta_n)$$

such that each $G_i$ has the form $\leftarrow L_1, \ldots, L_k$, $R(G_i) = L_j$ and $(G_{i+1}\ \Delta_{i+1})$ is obtained according to one of the following rules:

(1) If $L_j$ is not abducible, then $G_{i+1} = C$ and $\Delta_{i+1} = \Delta_i$ where $C$ is the resolvent of some clause in $P$ with $G_i$ on the selected literal $L_j$;

(2) If $L_j$ is abducible and $L_j \in \Delta_i$, then $G_{i+1} = \ \leftarrow L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_k$ and $\Delta_{i+1} = \Delta_i$;

(3) If $L_j$ is a ground abducible, $L_j \notin \Delta_i$ and $\overline{L_j} \notin \Delta_i$ and there exists a *consistency derivation* from $(\{L_j\}\ \Delta_i \cup \{L_j\})$ to $(\{\}\ \Delta')$ then $G_{i+1} = \ \leftarrow L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_k$ and $\Delta_{i+1} = \Delta'$.

Steps (1) and (2) are SLD-resolution steps with the rules of $P$ and abductive assumptions already computed, respectively. In step (3) a new abductive assumption is required and it is added to the current set of assumptions provided it is consistent.

**Consistency derivation**

A consistency derivation for an abducible $\alpha$ from $(\alpha,\ \Delta_1)$ to $(F_n\ \Delta_n)$ in $\langle P, A, I\rangle$ is a sequence

$$(\alpha\ \Delta_1), (F_1\ \Delta_1), (F_2\ \Delta_2), \ldots, (F_n\ \Delta_n)$$

where :

(i) $F_1$ is the union of all goals of the form $\leftarrow L_1, \ldots, L_n$ obtained by resolving the abducible $\alpha$ with the denials in $I$ with no such goal been empty;

(ii) for each $i > 1$, let $F_i$ have the form $\{\leftarrow L_1, \ldots, L_k\} \cup F_i'$, then for some $j = 1, \ldots, k$ $L_j$ is selected and $(F_{i+1}\ \Delta_{i+1})$ is obtained according to one of the following rules:

(C1) If $L_j$ is not abducible, then $F_{i+1} = C' \cup F_i'$ where $C'$ is the set of all resolvents of clauses in $P$ with $\leftarrow L_1, \ldots, L_k$ on the literal $L_j$ and the empty goal $[] \notin C'$, and $\Delta_{i+1} = \Delta_i$;

(C2) If $L_j$ is abducible, $L_j \in \Delta_i$ and $k > 1$, then
$F_{i+1} = \{\leftarrow L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_k\} \cup F_i'$
and $\Delta_{i+1} = \Delta_i$;

(C3) If $L_j$ is abducible, $\overline{L_j} \in \Delta_i$ then $F_{i+1} = F_i'$ and $\Delta_{i+1} = \Delta_i$;

(C4) If $L_j$ is a ground abducible, $L_j \notin \Delta_i$ and $\overline{L_j} \notin \Delta_i$, and there exists an *abductive derivation* from $(\leftarrow \overline{L_j}\ \Delta_i)$ to $([]\ \Delta')$ then $F_{i+1} = F_i'$ and $\Delta_{i+1} = \Delta'$;

(C5) If $L_j$ is equal to $\neg A$ with $A$ a ground atom and there exists an *abductive derivation* from $(\leftarrow A\ \Delta_i)$ to $([]\ \Delta')$ then $F_{i+1} = F_i'$ and $\Delta_{i+1} = \Delta'$.

In case (C1) the current branch splits into as many branches as the number of resolvents of $\leftarrow L_1, \ldots, L_k$ with the clauses in $P$ on $L_j$. If the empty clause is one of such resolvents the whole consistency check fails. In case (C2) the goal under consideration is made simpler if literal $L_j$ belongs to the current set of assumptions $\Delta_i$. If the goal contains only one literal then the derivation fails. In case (C3) the current branch is already consistent under the assumptions in $\Delta_i$, and this branch is dropped from the consistency checking. In case (C4) the current branch of the consistency search space can be dropped provided $\leftarrow \overline{L_j}$ is abductively provable. In case (C5), like (C4) the current branch fails and can be dropped provided that we can show that the atom $A$ holds.

Given an initial goal (query) $G$ and initial set of assumptions $\Delta_{in}$, possibly empty, the procedure succeeds, and returns the set of abducibles $\Delta_{out}$ iff there exists an abductive derivation from $(G\ \Delta_{in})$ to $([]\ \Delta_{out})$. In this case, we also say that the abductive derivation succeeds.