Inductive Learning using Constraint-driven Bias

Duangtida Athakravi[1], Dalal Alrajeh[1], Krysia Broda[1], Alessandra Russo[1⋆], and Ken Satoh[2]

[1] Imperial College London
{duangtida.athakravi07,dalal.alrajeh,k.broda,a.russo}@ic.ac.uk
[2] National Institute of Informatics
ksatoh@nii.ac.jp

Abstract. Heuristics such as the Occam Razor's principle have played a significant role in reducing the search for solutions of a learning task, by giving preference to most compressed hypotheses. For some application domains, however, these heuristics may become too weak and lead to solutions that are irrelevant or inapplicable. This is particularly the case when hypotheses ought to conform, within the scope of a given language bias, to precise domain-dependent structures. In this paper we introduce a notion of inductive learning through constraint-driven bias that addresses this problem. Specifically, we propose a notion of learning task in which the hypothesis space, induced by its mode declaration, is further constrained by domain-specific denials, and *acceptable hypotheses* are (brave inductive) solutions that conform with the given domain-specific constraints. We provide an implementation of this new learning task by extending the ASPAL learning approach and leveraging on its meta-level representation of hypothesis space to compute acceptable hypotheses. We demonstrate the usefulness of this new notion of learning by applying it to two class of problems - automated revision of software system goals models and learning of stratified normal programs.

Keywords: Inductive Logic Programming, Answer Set Programming, Meta-level Constraints

1 Introduction

Heuristics such as language bias and minimality have widely been used for controlling the search of hypotheses in a given inductive learning task. One of these is the Occam Razor's principle, for which preferred solutions among competing alternatives are generally those hypotheses with the fewest number of assumptions, also referred to as most compressed hypotheses. However, succinctness alone may not guarantee that the chosen hypothesis would be *acceptable* by the user. The user might have specific domain-dependent criteria for selecting or rejecting a hypothesis, in addition to the hypothesis having to be part of a language bias and covering given examples. Consider the following non-monotonic learning task:

⋆ This research is partially funded by the 7th Framework EU-FET project 600792 "ALLOW Ensembles", and the EPSRC project EP/K033522/1 "Privacy Dynamics".

$$B = \begin{cases} even(0) \\ num(0) \\ num(s(0)) \\ num(s(s(0))) \\ num(s(s(s(0)))) \\ num(s(s(s(s(0))))) \\ num(s(s(s(s(s(0)))))) \\ succ(X, s(X)) \leftarrow \\ \quad num(X), num(s(X)) \end{cases} \quad E = \begin{cases} even(s(s(0))) \\ odd(s(s(s(0)))) \\ odd(s(s(s(s(s(s(0))))))) \\ not\ even(s(0)) \\ not\ even(s(s(s(0)))) \\ not\ odd(0) \\ not\ odd(s(s(s(s(0))))) \end{cases} \quad M = \begin{cases} modeh(even(+num)) \\ modeh(odd(+num)) \\ modeb(even(+num)) \\ modeb(not\ even(+num)) \\ modeb(not\ odd(+num)) \\ modeb(succ(-num, +num)) \end{cases}$$

Under brave induction [24], a most compressed hypothesis could be the set of normal clauses

$$H_1 = \{even(X) \leftarrow not\ odd(X); odd(X) \leftarrow not\ even(X)\}$$

because together with B it has a stable model which includes the examples E. Another solution of this learning task, given for instance by the hypothesis

$$H_2 = \{even(X) \leftarrow succ(Y, X), not\ even(Y); odd(X) \leftarrow succ(Y, X), even(Y)\}$$

might be more reasonable in this application domain, but it would not be preferred in the search for most compressed hypothesis as it is bigger in size than H_1. In this paper we show how domain-specific constraints on the hypothesis space can provide additional information to the learner and guide the search towards more targeted solutions. We introduce a notion of inductive learning through *constraint-driven bias*. Specifically, we define a learning task in which domain-specific constraints on the structure of the hypothesis can be expressed by denials over the search space induced by the language bias, and solutions are *acceptable hypotheses* that conform to the given domain-specific constraints. For instance, in the above example, if a constraint were specified that requires hypotheses to contain the body literal *succ*, then H_1 would be considered to be a *non-acceptable* solution. We provide an implementation of this new learning task by extending the ASPAL learning approach [10]. We leverage on the fact that constraint satisfaction is already an integral part of the abductive search [18] used by ASPAL and show that the ASPAL meta-level representation of hypothesis space is well suited for the computation of acceptable hypotheses. The mechanism of our approach is an adaptation of abductive search with constraints [17] for the purpose of inductive learning using syntactic meta-level information. We express constraints in denial form only, unlike those in [7]. The form of abduction we use is more limited than [19] as we only assume grounded abducibles. We therefore do not require constructive negation [6] and dynamic constraints. We discuss soundness and completeness results of our learning task. We illustrate how this new notion of learning through constraint-driven bias can be used to learn stratified normal programs, and demonstrate its usefulness in addressing the practical open problem of goal model refinement in requirements engineering.

2 Background

We start by briefly describing the terminology used throughout this paper. A term is a constant, a variable, or an n-ary function defined over terms, and an atom is an n-ary predicate defined over terms. A *literal* is an atom l or its default negation *not* l. A *rule* is a normal clause of the form $h \leftarrow b_1, \ldots, b_n$, where atom h is its head and literals $b_1, \ldots, b_n$ are its bodies. A *fact* is a rule with no body.

A solution to a learning task is a set of rules called a *hypothesis*. Different learning task have been presented in the literature. In this paper we focus on the meta-level learning through abductive search approach proposed in [9], and its ASP implementation ASPAL [10]. ASPAL is an abductive learning framework for nonmonotonic ILP that translates an ILP task into an equivalent abductive logic programming task and computes inductive solutions as abductive assumptions over a declarative representation of the hypothesis space (i.e. top theory $\top$). In this learning framework, an ILP task $\langle B, M, E\rangle$ consists of a set of positive and negative examples E, expressed as ground literals, a normal logic program background theory B, consisting of a set of normal clauses, and mode declarations M. Mode declarations are of the form $modeh(s)$ or $modeb(s)$ corresponding respectively to head and body mode declarations where s is a schema for a predicate containing place-holders for its arguments ('+', '-' and '#' denoting respectively input variable, output variable and constant). A literal is compatible with a mode declaration if it is an instantiation of its schema with arguments instantiated according to the place-holders. A rule is compatible with M if (i) its head literal is compatible with (i.e. its predicate appear in) a head mode declaration from M; (ii) each body literal is compatible with a body mode declaration from M; and all of its variables are correctly linked as denoted by the input and output variable place-holders in the corresponding mode declarations. An hypothesis H is a solution to the ILP task if i) all rules in H are compatible with M; ii) $B \cup H$ is consistent; and iii) $B \cup H \models \bigwedge_{e \in E} e$[3]. ASPAL computes this learning task by an equivalent abductive task. An abductive task (ALP) is a tuple $\langle g, P, A, I\rangle$ with a set of ground facts g, called *observations*, a normal logic program P, a set of ground facts A, called *abducibles*, and a set of integrity constraints I. An abductive solution is a set $\Delta \subseteq A$ such that $P \cup \Delta$ is consistent, $P \cup \Delta \models g$ and $P \cup \Delta \models I$.

The hypothesis space R_M of the ILP task, containing all the rules compatible with M, is encoded into a *top-theory* $\top$. For every rule $r = h \leftarrow \overline{b}$ in R_M, a corresponding declarative representation $h \leftarrow \overline{b}, rule(id(h \leftarrow \overline{b}), \overline{C})$, where $\overline{C}$ is the list of constant arguments in r, is included in $\top$. In this representation, the term $id(h \leftarrow \overline{b})$ is a *rule identifier* of the form $(m_h, m_1, l_{1,1}, \ldots, l_{1,p_1}, \ldots, m_n, l_{n,1}, \ldots, l_{n,p_n})$, where m_h is the label of the mode declaration for h, m_i is the label of the mode declaration for body literal b_i in $\overline{b}$, and for each m_j, the elements $l_{j,1}, \ldots, l_{j,p_j}$ are numbers denoting the variable arguments in r linked to b_i, counting from left to right. For instance, the rules in hypothesis H_2 are encoded as $rule((even, succ, 1, not\_even, 2), e)$ and $rule((odd, succ, 1, even, 2), e)$, where e represents the empty set of constant arguments and the mode declaration labels are *even*, *succ* and *not_even*. Given an ILP task $\langle B, M, E\rangle$, the equivalent abductive task generated by ASPAL is $\langle B', A^\top, I\rangle$ where $B' = B \cup \top \cup \{examples \leftarrow \bigwedge_{e \in E} e\}$, $A^\top$ is the set of abducible atoms $rule(id(h \leftarrow \overline{b}), \overline{C})$, I is the singleton set of integrity constraints $\{\bot \leftarrow not\ examples\}$. If $\Delta \subseteq A^\top$ is an abductive solution then the inverse translation of all abducibles in Δ, denoted $hyp(\Delta)$, is the corresponding inductive hypothesis H (see [8]).

[3] Throughout this paper we use $\models$ to refer to brave consequence.

In our constraint-driven learning approach we consider labels of literals compatible with mode declarations instead of rule identifiers, to allow us to express syntactic constraints directly of the literals that would form acceptable hypotheses. The label of a literal compatible with a mode declaration is composed of its predicate name and constant arguments. For a negated literal the label is prefixed with "*not_*". For instance, given a mode body declaration $modeb(not\ p(+int, \#int, -int))$ a compatible literal of the form $not\ p(X, 2, Z)$ will have the label $not\_p(2)$. In general, labels will be unground in the top theory (e.g. $not\_p(X)$), and their variables will be instantiated according to the compatible hypothesis that are abduced during the computation.

In what follows, for a given compatible rule r, we denote with $head(r)$ the label of its head literal and with $body(r)$ the list of labels of its body literals. In writing constraints over hypothesis structure we will often make use of variables of type labels and we will denote them with capital letters L_h and L_b. We will use the shorthand $L_h = head(r)$ and $L_b \in body(r)$, to denote that for a given compatible rule r there exists an instantiation Θ of the variables L_h and L_b such that $L_h\Theta = head(r)$ and $L_b\Theta =$ is the label of a body literal in r.

3 Learning using constraint-driven bias

Domain-dependent constraints are constraints on the hypothesis space that collectively characterise the set of *acceptable* hypotheses. We propose a set $\mathcal{L}_\mathcal{C}$ of eight primitives for expressing domain-specific constraints. The arguments of a primitive can be either variables or constants. Variables can be head label variables (L_h), which stand for labels of an head literal in a compatible rule, or body label variables (L_{b_i}) that stand for labels of a body literal in a compatible rule. Constants are either labels referring to a specific literal or integers. The intuitive meaning of these primitives is as follows: (i) $in\_some\_rule(L_h, L_b)$ – the hypothesis must include a rule with a combination of head and body literals labelled L_h and L_b; (ii) $in\_all\_rule(L_h, L_b)$ – all rules in the hypothesis with head literal labelled L_h must have a body literal labelled L_b; (iii) $in\_same\_rule(L_h, L_{b_1}, L_{b_2})$ – in the hypothesis, body literals labelled L_{b_1} and L_{b_2} must appear together in the body of a rule defining a head literal labelled L_h; (iv) $in\_diff\_rule(L_{h_1}, L_{b_1}, L_{h_2}, L_{b_2})$ – in the hypothesis, body literals labelled L_{b_1} and L_{b_2} must appear in two different rules, with heads labelled L_{h_1} and L_{h_2} respectively; (v) $max\_body(L_h, x)$ (resp. $min\_body(L_h, x)$) – in the hypothesis rules with head literal labelled L_h have at most (resp. at least) x body literals; (vii) $max\_head(L_h, x)$ (resp. $min\_head(L_h, x)$) – the hypothesis must include at most (resp. at least) x rules with head labelled L_h. Using the primitives in $\mathcal{L}_\mathcal{C}$ and given a background theory of an original inductive task, domain-dependent constraints are of the following form.

Definition 1. *Let B be a background knowledge expressed in a language $\mathcal{L}_\mathcal{B}$ and M a set of mode declarations with associated constraint primitives $\mathcal{L}_\mathcal{C}$. A domain-dependent constraint in $\mathcal{L}_\mathcal{C}$ is $\bot \leftarrow C, \overline{C_B}$ where C is a primitive of $\mathcal{L}_\mathcal{C}$ or its negated form and $\overline{C_B}$ is a conjunction of literals from the language $\mathcal{L}_\mathcal{B}$.*

The following definition states what we mean by an hypothesis (i.e. set of clauses), compatible with a given mode declaration M, to satisfy a domain-dependent constraint.

Definition 2. *Let B be a background knowledge, H an hypothesis (set of normal clauses) and $\bot \leftarrow C, \overline{C_B}$ a domain-dependent constraint ic. The hypothesis H* satisfies *ic if and only if either $B \cup H \not\models \overline{C_B}$ or one of the following cases holds:*

- *$C = not\ in\_some\_rule(L_h, L_b)$ and there exists $r \in H$ s.t. $L_h = head(r)$ and $L_b \in body(r)$;*
- *$C = not\ in\_all\_rule(L_h, L_b)$ and either there is no rule $r \in H$ s.t. $L_h = head(r)$ or for all $r \in H$ s.t. $L_h = head(r)$ it is the case that $L_b \in body(r)$;*
- *$C = not\ in\_same\_rule(L_h, L_{b_1}, L_{b_2})$ and either there are no $r \in H$ with $L_h = head(r)$ with $L_{b_1} \in body(r)$ or $L_{b_2} \in body(r)$, or for all $r \in H$ s.t. $L_h = head(r)$ if $L_{b_1} \in body(r)$ then $L_{b_2} \in body(r)$, and vice versa;*
- *$C = not\ in\_diff\_rule(L_{h_1}, L_{b_1}, L_{h_2}, L_{b_2})$ and for some $r_1 \in H$ s.t. $L_{h_1} = head(r_1)$ and $L_{b_1} \in body(r_1)$ (resp. $L_{h_2} = head(r_2)$ and $L_{b_2} \in body(r_1)$) then it must also be the case that for some different rule $r_2 \in H$ with $L_{h_2} = head(r_2)$ (resp. $L_{h_1} = head(r_2)$) and $L_{b_2} \in body(r_2)$ (resp. $L_{b_1} \in body(r_2)$);*
- *$C = not\ max\_body(L_h, x)$ (resp. not $min\_body(L_h, x)$) and for all $r \in H$ if $L_h = head(r)$ then $|body(r)| \leq x$ (resp. $|body(r)| \geq x$);*
- *$C = not\ max\_head(L_h, x)$ (resp. not $min\_head(L_h, x)$) and for $R = \{r | r \in H$ and $L_h = head(r)\}$ then $|R| \leq x$ (resp. $|R| \geq x$).*

Primitives of $\mathcal{L}_\mathcal{C}$ can also occur positively in a domain-dependent constraint. If ic is $\bot \leftarrow C$, where C is one of the primitives, then a hypothesis H satisfies ic if H does not satisfy $\bot \leftarrow not\ C$. For instance, if C is the primitive $in\_some\_rule(L_h, L_b)$, then H satisfies $\bot \leftarrow in\_some\_rule(L_h, L_b)$ if H does not satisfy $\bot \leftarrow not\ in\_some\_rule(L_h, L_b)$. That is, there is not a rule $r \in H$ where $L_h = head(r)$ and $L_b \in body(r)$. We can now define the notion of *constraint biased search space*. Intuitively, this is the set of rules, compatible with a given mode declaration, that satisfies the given domain-dependent constraints.

Definition 3. *Let $\mathcal{P}(R_M)$ be the power set of rules compatible with M and ic be a domain-dependent constraint. The* constraint biased search space with respect to *ic, denoted $[\mathcal{P}(R_M)]_{ic}$, is the set $S \subseteq \mathcal{P}(R_M)$ where each H in S satisfies ic.*

This can be extended to the notion of constraint biased search space for a set of integrity constraints.

Definition 4. *The constraint biased search space of a set IC of domain-dependent constraints, denoted $[\mathcal{P}(R_M)]_{IC}$, is the intersection of the constraint biased search spaces with respect to each domain-dependent constraint in IC:*

$$[\mathcal{P}(R_M)]_{IC} = \bigcap_{ic \in IC} [\mathcal{P}(R_M)]_{ic}$$

We can now define our notion of learning task with constraint-driven bias and acceptable solutions.

Definition 5. *A learning task with constraint-driven bias is a tuple* $\langle B, M, E, IC\rangle$ *where* B *is background knowledge,* M *is a set of mode declarations,* E *is a set of examples and* IC *is a set of domain-dependent constraints.* H *is an* acceptable solution *if and only if (i)* $B \cup H \vDash \bigwedge_{e\in E} e$*; and (ii)* $H \in [\mathcal{P}(R_M)]_{IC}$.

Consider the example in Section 1. The constraint $\bot \leftarrow$ *not in_all_rule*$(L_h, succ)$ will rule out the hypothesis $H_1 = \{even(X) \leftarrow not\ odd(X); odd(X) \leftarrow not\ even(X)\}$.

4 Implementation

In this section we describe how the ASPAL system has been extended to compute acceptable solutions of a learning task with constraint-driven bias. In particular, we show how the primitives described in Section 3 can naturally be encoded as meta-constraints on the declarative representation of the hypothesis space used in ASPAL. Table 1 gives the declarative semantics, expressed in ASP, of each primitive in $\mathcal{L}_\mathcal{C}$. This declarative encoding makes use of the following meta-predicates: *is_rule*(R, L_h) represents a clause with identifier R and head literal labelled L_h; *in_rule*(R, L_h, L_b, I) represents a clause with identifier R, head literal labelled L_h, and body literal labelled L_b at position I. Note that the unique identifier R and the body literal position I are automatically generated by the learning system for each clause (and body) in the hypothesis space R_M. Given a rule $r = h \leftarrow b_1, \ldots, b_n$, with identifier R_{id}, of a learned hypothesis H, the associated meta-level information is the set of (ground) instances $\{$*in_rule*$(R_{id}, L_h, L_{b_1}, 1)$, $\ldots$, *in_rule*$(R_{id}, L_h, L_{b_n}, n)$, *is_rule*$(R_{id}, L_h)\}$. The meta-level information for a hypothesis H, denoted with $meta(H)$, is the union of the meta-level information of its rules.

The top theory $\top_M$ is given by $\top \cup \mathcal{M}$ where T is the ASPAL top-theory as described in Section 2 and $\mathcal{M}$ includes for each rule $h \leftarrow b_1, \ldots, b_n$, *rule*$(id(h \leftarrow b_1, \ldots, b_n), \bar{C})$ in $\top$, the following rule, which relates a rule in R_M with its meta-level information.

$$n+1\{is\_rule(i(\bar{C}), l_h), in\_rule(i(\bar{C}), l_h(\bar{C_h}), l_{b_1}(\bar{C_{b_1}}), 1), \ldots, in\_rule(i(\bar{C}), l_h(\bar{C_n}), l_{b_n}(\bar{C_{b_n}}), n)\}n+1 \leftarrow rule(id(h \leftarrow b_1, \ldots, b_n), \bar{C})$$

During the computation of a hypothesis, if the encoding *rule*$(id(h \leftarrow b_1, \ldots, b_n)$ of a rule r is abduced, then the above rule in $\mathcal{M}$ will enforce all the associated meta-level information of r to be also inferred if they satisfy the domain-specific constraints. That is, the hypothesis will satisfy the desired domain-dependent constraints. For example, the top theory for the example given in Section 1 will include the clauses:

$$even(X) \leftarrow succ(Y, X), not\ even(Y), rule((m1, m5, 1, m3, 2), c)$$
$$3\{is\_rule(1, even), in\_rule(1, even, succ, 1), in\_rule(1, even, not\_even, 2)\}3 \leftarrow rule((m1, m5, 1, m3, 2), c)$$

The set of abducibles in this case is $\{rule((m3, m5, 1, m6, 2), c)\}$ and the inferred meta-level information is the set $\{is\_rule(1, even), in\_rule(1, even, succ, 1), in\_rule(1, even, not\_even, 2)\}$.

Acceptable solutions of a learning task $\langle B, M, E, IC\rangle$, with constrain-driven bias IC, are computed using an *equivalent abductive task* $\langle \tilde{B}, A^\top, \tilde{I}\rangle$, where

Primitive predicate $\leftarrow P, \overline{C_B}$	Declarative semantics of $\leftarrow P, \overline{C_B}$
$\bot \leftarrow not\ in\_some\_rule(L_h, L_b), \overline{C_B}$	$ic\_cond \leftarrow in\_rule(Rid, L_h, L_b, I)$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$
$\leftarrow in\_some\_rule(L_h, L_b), \overline{C_B}$	$\leftarrow in\_rule(Rid, L_h, L_b, I), \overline{C_B}$
$\bot \leftarrow not\ in\_all\_rule(L_h, L_b), \overline{C_B}$	$\bot \leftarrow is\_rule(Rid, L_h),$ $not\ in\_rule(Rid, L_h, L_b, I), \overline{C_B}$
$\bot \leftarrow in\_all\_rule(L_h, L_b), \overline{C_B}$	$ic\_cond \leftarrow is\_rule(Rid, L_h)$ $ic\_cond \leftarrow is\_rule(Rid, L_h),$ $not\ in\_rule(Rid, L_h, L_b, I)$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$
$\bot \leftarrow not\ in\_same\_rule(L_h, L_{b_1}, L_{b_2}),$ $\overline{C_B}$	$\bot \leftarrow in\_rule(Rid, L_h, L_{b_1}, I_1),$ $not\ in\_rule(Rid, L_h, L_{b_2}, I_2), \overline{C_B}$ $\bot \leftarrow not\ in\_rule(Rid, L_h, L_{b_1}, I_1),$ $in\_rule(Rid, L_h, L_{b_2}, I_2), \overline{C_B}$
$\bot \leftarrow in\_same\_rule(L_h, L_{b_1}, L_{b_2}), \overline{C_B}$	$\bot \leftarrow in\_rule(Rid, L_h, L_{b_1}, I_1),$ $in\_rule(Rid, L_h, L_{b_2}, I_2), \overline{C_B}$
$\bot \leftarrow not\ in\_diff\_rule(L_{h_1}, L_{b_1}, L_{h_2}, L_{b_2}),$ $\overline{C_B}$	$ic\_cond \leftarrow in\_rule(Rid_1, L_{h_1}, L_{b_1}, I_1),$ $in\_rule(Rid_2, L_{h_2}, L_{b_2}, I_2),$ $Rid_1 \neq Rid_2$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$
$\bot \leftarrow in\_diff\_rule(L_{h_1}, L_{b_1}, L_{h_2}, L_{b_2}),$ $\overline{C_B}$	$\bot \leftarrow in\_rule(Rid_1, L_{h_1}, L_{b_1}, I_1),$ $in\_rule(Rid_2, L_{h_2}, L_{b_2}, I_2),$ $Rid_1 \neq Rid_2, \overline{C_B}$
$\bot \leftarrow not\ max\_body(L_h, x), \overline{C_B}$	$\bot \leftarrow is\_rule(Rid, L_h),$ $x + 1\{in\_rule(Rid, L_h, L_b, I)\}, \overline{C_B}$
$\bot \leftarrow max\_body(L_h, x), \overline{C_B}$	$ic\_cond \leftarrow is\_rule(Rid, L_h),$ $x + 1\{in\_rule(Rid, L_h, L_b, I)\}$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$
$\bot \leftarrow not\ min\_body(L_h, x), \overline{C_B}$	$\bot \leftarrow is\_rule(Rid, L_h),$ $0\{in\_rule(Rid, L_h, L_b, I)\}x - 1, \overline{C_B}$
$\bot \leftarrow min\_body(L_h, x), \overline{C_B}$	$ic\_cond \leftarrow is\_rule(Rid, L_h),$ $0\{in\_rule(Rid, L_h, L_b, I)\}x - 1$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$
$\bot \leftarrow not\ max\_head(L_h, x), \overline{C_B}$	$\bot \leftarrow x + 1\{is\_rule(Rid, L_h)\}, \overline{C_B}$
$\bot \leftarrow max\_head(L_h, x), \overline{C_B}$	$ic\_cond \leftarrow x + 1\{is\_rule(Rid, L_h)\}$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$
$\bot \leftarrow not\ min\_head(L_h, x), \overline{C_B}$	$\bot \leftarrow 0\{is\_rule(Rid, L_h)\}x - 1, \overline{C_B}$
$\bot \leftarrow min\_head(L_h, x), \overline{C_B}$	$ic\_cond \leftarrow 0\{is\_rule(Rid, L_h)\}x - 1$ $\bot \leftarrow not\ ic\_cond, \overline{C_B}$

Table 1. Declarative semantics of primitives of $\mathcal{L}_\mathcal{C}$

the background knowledge $\tilde{B} = B \cup \top_M \cup \{examples \leftarrow \bigwedge_{e \in E} e\}$, $A^\top$ is the same set of abducibles as in ASPAL, and $\tilde{I}$ is the set of integrity constraints $\{\bot \leftarrow not\ examples\} \cup IC^t$, where IC^t is the ASP representation of the domain-dependent constraints IC, based on the declarative semantics of the primitives given in Table 1. Acceptable hypotheses are then given by applying an inverse translation function on abductive solutions Δ of $\langle \tilde{B}, A^\top, \tilde{I} \rangle$.

In order to reason about the soundness and completeness of the implementation, we consider the constraints' ASP translation.

Proposition 1. *Let $\langle B, M, E, IC \rangle$ be a learning task with constraint-driven bias and let $\langle \tilde{B}, A^\top, \tilde{I} \rangle$ be an equivalent abductive task. For each $\Delta \subseteq A^\top$, $hyp(\Delta) \in [\mathcal{P}(R_M)]_{IC}$ if and only if $B \cup \top \cup \Delta \cup meta(hyp(\Delta)) \cup IC^t$ is satisfiable.*

Proof Outline: The proposition essentially shows that the ASP encoding of an hypothesis from a given constraint biased search space satisfies the ASP encoding of the given domain-dependent constraints. A proof outline is given which shows, by some cases, that this holds for single constraints ic with just one primitive.

- ic is $\bot \leftarrow not\ in\_some\_rule(L_h, L_b)$: $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff, by Definitions 2-3, there exists $r \in hyp(\Delta)$ s.t. $L_h = head(r)$ and $L_b \in body(r)$. Therefore $in\_rule(Rid, L_h, L_b, I) \in meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$, for some value of I and Rid the rule identifier of r. Hence $\delta \cup meta(hyp(\delta)) \cup ic^t$ is satisfiable.
- ic is $\bot \leftarrow not\ in\_all\_rule(L_h, L_b)$: $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff, by Definitions 2-3, either there is no rule $r \in hyp(\Delta)$ s.t. $L_h = head(r)$ or for all $r \in hyp(\Delta)$ s.t. $L_h = head(r)$ it is the case that $L_b \in body(r)$. In the first case, $is\_rule(Rid, L_h) \notin meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$, for any rule identifier Rid making $\delta \cup meta(hyp(\delta)) \cup ic$ satisfiable. In the second case, $in\_rule(Rid, L_h, L_b, I) \in meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$, for some value of I and Rid so making $\delta \cup meta(hyp(\delta)) \cup ic^t$ satisfiable.
- ic is $\bot \leftarrow not\ in\_same\_rule(L_h, L_{b_1}, L_{b_2})$: $hyp(\Delta) \subseteq [\mathcal{P}(R_M)]_{ic}$ iff, by Definitions 2-3, either there are no $r \in hyp(\Delta)$ with $L_h = head(r)$ with $L_{b_1} \in body(r)$ or $L_{b_2} \in body(r)$, or for all $r \in hyp(\Delta)$ s.t. $L_h = head(r)$ if $L_{b_1} \in body(r)$ then $L_{b_2} \in body(r)$, and vice versa. In the first case either $in\_rule(Rid, L_h, L_{b_1}, I_1)) \notin meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$, for any rule identifier Rid and $in\_rule(Rid, L_h, L_{b_2}, I_2)) \notin meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$, so making $\delta \cup meta(hyp(\delta)) \cup ic^t$ is satisfiable. In the second case, both atoms $in\_rule(Rid, L_h, L_{b_1}, I_1))$ and $in\_rule(Rid, L_h, L_{b_2}, I_2))$ are in $meta(hyp(\delta))$ so making $\delta \cup meta(hyp(\delta)) \cup ic^t$ is satisfiable. The two ASP denials that form ic^t work together to ensure that both literals must be in the same rule if either one of them is.
- ic is $\bot \leftarrow not\ in\_diff\_rule(L_{h_1}, L_{b_1}, L_{h_2}, L_{b_2})$: $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff, by Definitions 2-3, for some $r_1 \in hyp(\Delta)$ s.t. $L_{h_1} = head(r_1)$ and $L_{b_1} \in body(r_1)$ (resp. $L_{h_2} = head(r_2)$ and $L_{b_2} \in body(r_1)$) there must also exist some different rule $r_2 \in hyp(\Delta)$ with $L_{h_2} = head(r_2)$ (resp. $L_{h_1} = head(r_2)$) and $L_{b_2} \in body(r_2)$ (resp. $L_{b_1} \in body(r_2)$). Consider some $r_1 \in hyp(\Delta)$ s.t.

$L_{h_1} = head(r_1)$ and $L_{b_1} \in body(r_1)$ (resp. $L_{h_2} = head(r_2)$). In this case, both $in\_rule(Rid, L_{h_1}, L_{b_1}, I_1)$ and $in\_rule(Rid, L_{h_1}, L_{b_2}, I_2)$ are in $meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$ for some values of Rid_1, Rid_2, I_1 and I_2, so making $\delta \cup meta(hyp(\delta)) \cup ic^t$ is satisfiable.

- ic is $\bot \leftarrow not\ max\_body(L_h, x)$ ($not\ min\_body(L_h, x)$): $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff, by Definitions 2-3, for all $r \in hyp(\Delta)$ if $L_h = head(r)$ then $|body(r)| \leq x$ (resp. $|body(r)| \geq x$). Let $r \in hyp(\Delta)$ such that $L_h = head(r)$. So $is\_rule(Rid, L_h) \in meta(hyp(\delta))$, where $\delta \in A^\top$ and $hyp(\delta) = r$ and up to x $in\_rule(Rid, L_h, L_b, I) \in meta(hyp(\delta))$. Hence $\delta \cup meta(hyp(\delta)) \cup ic^t$ is satisfiable.
- ic is $\bot \leftarrow not\ max\_head(L_h, x)$ ($not\ min\_head(L_h, x)$): $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff, by Definitions 2-3, $|\{r | r \in hyp(\Delta) \text{ and } L_h = head(r)\}| \leq x$ (resp. $\geq x$). So, $|\{isrule(R_{id}, L_h) : isrule(R_{id}, L_h) \in meta(hyp(\delta))\}| \leq x$, where $\delta \in A^\top$ and $hyp(\delta) = r$. Hence $\delta \cup meta(hyp(\delta)) \cup ic^t$ is satisfiable.

It is easy to show that applying the above cases to each ic appearing in a given set of domain-dependent constraints, the proposition holds for the set $hyp(\Delta)$. Moreover, an integrity constraint ic^t that, in addition to a single primitive, includes background literals $\overline{C_B}$ will be satisfied either if $\overline{C_B}$ is not entailed by $B \cup hyp(\Delta)$, or if $\overline{C_B}$ is entailed by $B \cup hyp(\Delta)$ and the single primitive is satisfied. Thus $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff $B \cup \top \cup \Delta \cup meta(hyp(\Delta)) \cup ic^t$ is satisfiable, and consequently $hyp(\Delta) \in [\mathcal{P}(R_M)]_{ic}$ iff $B \cup \top \cup \Delta \cup meta(hyp(\Delta)) \cup ic^t$ is satisfiable for all $ic \in IC$.

The next theorem shows that the ASP implementation is sound and complete with respect to our notion of acceptable solution of a learning task with constrain-driven bias.

Theorem 1. *Let $\langle B, M, E, IC \rangle$ be a learning task with constraint-driven bias. The set of rule $hyp(\Delta)$ is an acceptable solution of this task if and only if Δ is an abductive solution of the abductive task $\langle \tilde{B}, A^\top, \tilde{I} \rangle$.*

Proof Outline: We consider first the completeness case. For $hyp(\Delta)$ to be an acceptable solution, $hyp(\Delta) \in [\mathcal{P}(R_M)]_{IC}$ and $B \cup hyp(\Delta) \models \bigwedge_{e \in E} e$ bravely. The latter means that there must exist an answer set AS of $B \cup hyp(\Delta)$ that extends $\bigwedge_{e \in E} e$. By Proposition 1 $B \cup \top \cup \Delta \cup meta(hyp(\Delta)) \cup IC^t$ is satisfiable. We also know that $B \cup \top \cup \Delta$ has an answer set that proves the examples because unfolding (answer set-preserving transformation) $\top$ with respect to Δ yields $B \cup hyp(\Delta)$. So adding $\{examples \leftarrow \bigwedge_{e \in E} e\} \cup \{\bot \leftarrow not\ examples\}$ will make $\tilde{B} \cup \Delta \cup \tilde{I}$ satisfiable, which implies that Δ is an abductive solution of $\langle \tilde{B}, A^\top, \tilde{I} \rangle$.

Consider now the soundness case. Let AS be an answer set of $\tilde{B} \cup \Delta \cup \tilde{I}$. As $\tilde{I} = \{\bot \leftarrow examples\} \cup IC^t$, and $\{examples \leftarrow \bigwedge_{e \in E} e\}$ is in $\tilde{B}$, then AS satisfies the examples. The examples are only satisfiable by $B \cup \top \cup \Delta$, and unfolding $\top$ with respect to Δ yields $B \cup hyp(\Delta)$, thus $B \cup hyp(\Delta)$ satisfies the examples. Since $\tilde{B} \cup \Delta \cup \tilde{I}$ is satisfiable then $B \cup \top \cup \Delta \cup meta(hyp(\Delta)) \cup IC^t$ is satisfiable. Hence it follows by Proposition 1 that $hyp(\Delta) \in [\mathcal{P}(R_M)]_{IC}$, and therefore according to Definition 5 $hyp(\Delta)$ is acceptable.

5 Case Studies

5.1 Requirements Engineering

In the area of requirements engineering, goal models are directed acyclic graphs for representing links between objectives the software system is expected to meet. In these models high-level goals are linked to their sub-goals through refinement links such that satisfying the conjunction of the sub-goals is a sufficient condition for satisfying its parent. A library of goal patterns (expressed in propositional temporal logic [23]) that guarantees these refinement semantics is given in [11]. An example is given in Figure 1. Figure 1.b shows a simplified goal model for a Flight Control System (FCS) from [2] that was obtained using the refinement pattern in Figure 1.a. The symbol $\Box$ means always, $\bigcirc$ means at the next time point, $\rightarrow$ is for logical implication, and *MovingOnRunway*, *ThrustEnabled* and *WheelsTurning* are propositional atoms The top-level goal, labelled g, states that whenever the plane is moving on the runway, the reverse thrust shall be enabled at the next time point. The goal g has two children labelled $c1$ and $c2$ respectively. The left child $c1$ says that whenever the plane is moving on the runway, the wheels are turning, whilst the right child $c2$ says that whenever the wheels are turning, the reverse thrust is enabled next.

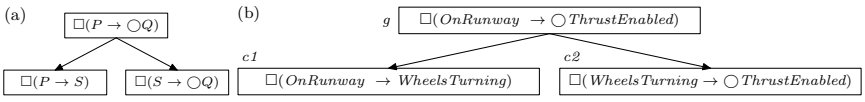

Fig. 1. (a) A goal pattern from [11]; (b) FCS goal refinement model with goals labelled g, $c1$ and $c2$.

One problem with early goal-driven models (such as the one in Figure 1.b) is that they may be incorrect or too weak and hence need to be continually revised until a complete and correct specification is reached. Consider a scenario in which the runway surface is wet, the plane is moving on the runway, the wheel sensors are flagged and the wheels are not turning, hence violating the goal $c1$. In such circumstances, the idealistic or conflicting goals need to be revised to avoid any potential system failures. Not only so, but all goals dependent on the problematic goal (e.g., children, other sub-goals and parent goals) may also need to be modified to ensure that the correctness of the goal model is maintained. Very limited formal support is available for this crucial task [21]. The purpose of this case study is to demonstrate how theory revision with domain-dependent constraints on mode declarations provides formal, automated support to sound revisions of goal models, something not achievable by current non-monotonic ILP systems like ASPAL.

To revise the model, each assertion in Figure 1.b is mapped into a set of rules, using an Event Calculus formalism [20] similar to the one presented in [1], that captures the syntactic formulation of the goals[4]. The predicates *a_holds*/3

[4] We do not present here details of the mapping from temporal logic to logic programs as this is outside the scope of this paper.

(resp. *c_holds*/3) represent the notion that the antecedent (resp. consequent) of the goal holds at a time point in a scenario.

$$R = \left\{ \begin{array}{l} a\_holds(c1, T, S) \leftarrow holds\_at(onRunway, T, S) \\ c\_holds(c1, T, S) \leftarrow holds\_at(wheelsTurning, T, S) \\ a\_holds(c2, T, S) \leftarrow holds\_at(wheelsTurning, T, S) \\ c\_holds(c2, T, S) \leftarrow holds\_at(thrustEnabled, T, S) \end{array} \right\}$$

Violating scenarios are included in the background theory as a series of *happens* facts, e.g.,[5]

$$B = \left\{ \begin{array}{ll} happens(rain, 0, s1) & happens(landPlane, 1, s1) \\ happens(switchPulseOn, 1, s1) & happens(enableReverseThrust, 2, s1) \\ ... & \end{array} \right\}$$

The aim of the learning task is to revise the goals appearing in Figure 1.b, by replacing the condition *WheelsTurning* in the sub-goals $c1$ and $c2$ which is not true in the violating scenario with another that does. We use our new notion of learning to compute acceptable revisions of the goal models, i.e., revisions that preserve the semantics of the goal model. An example constraint is that literals appearing in the body of a rule with head label *a_holds* must also appear in the body of a rule with head label *c_holds*. This is given below.

$$IC = \left\{ \begin{array}{l} \bot \leftarrow in\_diff\_rule(a\_holds(C1), BL1, c\_holds(C2), BL2), \\ \quad right\_child\_of(G, C1), left\_child\_of(G, C2), BL1 \neq BL2, C1 \neq C2 \\ \bot \leftarrow in\_diff\_rule(c\_holds(C1), BL1, a\_holds(C2), BL2), \\ \quad left\_child\_of(G, C1), right\_child\_of(G, C2), BL1 \neq BL2, C1 \neq C2 \\ \bot \leftarrow not\ min\_body(a\_holds(C), 1) \\ \bot \leftarrow not\ min\_body(c\_holds(C), 1) \end{array} \right\}$$

Using ASPAL without domain-specific constraints to compute revisions would generate the revised theory R'_1, given below, as most compressed hypothesis, which does not preserve the refinement semantics of the goal model (where the same assertion S must appear in both $c1$ and $c2$ according to the pattern in Figure 1.a), nor the correctness of goal models (requiring the parent goal to hold the conjunction of its children holds).

$$R'_1 = \left\{ \begin{array}{l} a\_holds(c1, T, S) \leftarrow holds\_at(onRunway, T, S) \\ c\_holds(c1, T, S) \\ a\_holds(c2, T, S) \leftarrow holds\_at(wheelsPulseOn, T, S) \\ c\_holds(c2, T, S) \leftarrow holds\_at(thrustEnabled, T, S) \end{array} \right\}$$

On the other hand, using the constraint-driven bias extension of ASPAL with domain-specific constraints that capture the goal-model refinement semantics we are able to compute the acceptable revised theory R'_2, given below, which although not minimal, satisfies the refinement semantics.

$$R'_2 = \left\{ \begin{array}{l} a\_holds(c1, T, S) \leftarrow holds\_at(onRunway, T, S) \\ c\_holds(c1, T, S) \leftarrow holds\_at(wheelsPulseOn, T, S) \\ a\_holds(c2, T, S) \leftarrow holds\_at(wheelsPulseOn, T, S) \\ c\_holds(c2, T, S) \leftarrow holds\_at(thrustEnabled, T, S) \end{array} \right\}$$

[5] The complete program is available from www.doc.ic.ac.uk/∼da04/ilp14/goalmodel.lp.

5.2 Hypothesis Stratification

In this second case study, we show more generally how constraints on mode declarations could be used to compute stratified programs [3] as acceptable solutions of a learning task. In our non-monotonic setting this is useful to guarantee the uniqueness of a hypothesis' interpretation. We consider two instances of this learning task. In the first case the given background knowledge is empty and we want to compute a set of stratified programs that proves the examples. In the second case the learning task has a given (stratified) background knowledge and we want to compute a hypothesis such that the union of the hypothesis and given background knowledge is stratified and proves the examples.

Let's consider the first case. Example of domain-dependent constraints, for guiding the search to stratified programs, are as follows:

$$\bot \leftarrow in\_some\_rule(H,B), is\_positive(B), stratum(H,L), not\ defined(B,L)$$
$$\bot \leftarrow in\_some\_rule(H,B), is\_negative(B), stratum(H,L), not\ predefined(B,L)$$
$$\bot \leftarrow not\ min\_head(H,1), stratum(H,L), L \neq 1$$

The first constraint checks that positively occurring body literals are defined within the same or lower stratum than the head stratum. The second constraint checks that negatively occurring body literals are defined in a lower stratum than the head stratum, and the third constraint ensures that literals exclusively used in the body (i.e. are not used in any head) are in the lowest stratum. Auxiliary predicates can be defined to help reasoning about basic concepts related to stratification. These include facts about labels of positive literals (using a predicate *is_positive*), labels of negative literals (using a predicate *is_negative*), and facts about labels of corresponding negative and positive pairs of literals (using the predicate *negative_positive_pair*/2). A notion of strata can be expressed using the ASP choice rule to ensure that each literal in the hypothesis is assigned a unique stratum; the predicate *level* asserts an integer limit on the number of strata allowed in a hypothesis, and the number of levels needed is at most equal to the number of predicates that can appear in the hypothesis:

$$1\{stratum(H,L) : level(L)\}1 \leftarrow is\_positive(H)$$
$$\bot \leftarrow stratum(H,L1), stratum(H,L2), L1 \neq L2$$

Other key auxiliary predicates are defined below. Predicates $defined$ and $predefined$ are used to check that positive and negative body literals are appropriately defined in a lower or equal stratum to where they are used:

$$defined(B,L1) \leftarrow level(L1), stratum(B,L2), L2 \leq L1$$
$$predefined(NegB,L1) \leftarrow level(L1), negative\_positive\_pair(NegB,B),$$
$$stratum(B,L2), L2 < L1$$

We have considered two learning tasks[6] of this kind, where no examples was given, the background included only the definition of the auxiliary predicates, described above, and the mode declaration were defined as shown in Table 2.

[6] Clingo's option `--project` was used when running the ASP programs for solving the examples using the constraints to ensure that each hypothesis is output only once regardless of the number of ways it could be stratified.

Task	Head declaration	Body declaration
1	modeh(p(+arg,+arg))	modeb(q(+arg,+arg))
	modeh(q(+arg,+arg))	modeb(not q(+arg,+arg))
		modeb(p(+arg,+arg))
		modeb(not p(+arg,+arg))
2	modeh(p(+arg,+arg))	modeb(q(+arg,+arg))
	modeh(q(+arg,+arg))	modeb(not q(+arg,+arg))
	modeh(a(+arg,+arg))	modeb(p(+arg,+arg))
		modeb(not p(+arg,+arg))
		modeb(a(+arg,+arg))
		modeb(not a(+arg,+arg))

Table 2. Mode declarations used in the two learning tasks for testing the stratification constraints. All tasks were run to find hypotheses with at most two clauses and maximum two body literals per clause.

Each learning task was solved for a different range of allowable values for variables of type *int*, from 1 to 5, 1 to 10, and 1 to 20.
Table 3 shows the number of unique hypotheses computed in each instance, with and without the constraints. In Table 3 are the recorded results of the case study[7]. For both tasks the time taken to find all hypotheses using the constraints is smaller than the learning task without constraints. We can conclude that for these experiments that domain-specific constraints help improving the efficiency of the task of learning stratified programs. More importantly, for the non-constrained tasks, the hypotheses produced are not all stratified, thus further parsing of the results is required to find only the stratified hypotheses.

Task	Range for `arg`	Without constraints		With constraints	
		No. of hypotheses	Time (s)	No. of hypotheses	Time (s)
1	1-5	25126	3.588	8614	1.716
	1-10		9.282		4.820
	1-15		34.788		14.960
2	1-5	302860	219.306	158482	198.433
	1-10		1494.942		1082.694
	1-15		4640.889		2346.723

Table 3. Number of hypotheses learnt with and without using the constraints, and the time taken to learn all hypotheses for when the argument of the rules can be integer within the ranges 1 to 5, 1 to 10 and 1 to 15.

The second type of problem we have considered is when we want to learn programs that together with an existing background knowledge are stratified.

[7] All tasks were run using the ASP solver Clingo 3 [13] on a 2.13GHz laptop computer with 4GB memory

The stratified constraint has now to apply to the union of the given background and a computed hypothesis. To achieve this a meta-level representation of the existing background knowledge is constructed using is_rule and in_rule, adding instances of the predicates $is\_positive$, $is\_negative$ and $negative\_positive\_pair$ as necessary. This case study only deals with ungrounded rules, but the constraint is still applicable to rules with constants. Consider the even and odd example in Section 1. The predicates $is\_positive$ and $is\_negative$ can be extended as follows:

$$\begin{array}{ll} is\_positive(succ, succ) & is\_negative(not\_even, not\_even) \\ is\_positive(even, even) & is\_negative(not\_odd, not\_odd) \\ is\_positive(even, even(X)) \leftarrow num(X) & is\_positive(odd, odd) \end{array}$$

The original domain-dependent constraints are in this case:

$$IC = \left\{ \begin{array}{l} \leftarrow in\_some\_rule(H, B\_label), is\_positive(B\_name, B\_label), \\ \quad stratum(H, L), not\ defined(B\_name, L) \\ \leftarrow in\_some\_rule(H, B\_label), is\_negative(B\_name, B\_label), \\ \quad stratum(H, L), not\ predefined(B\_name, L) \end{array} \right\}$$

Solving for hypotheses containing at most two rules and at most three body literals, a total of ten hypotheses can be found including the following:

$$\left\{ \begin{array}{l} odd(A) \leftarrow succ(B, A), succ(C, B), not\ even(C) \\ even(A) \leftarrow succ(B, A), succ(C, B), even(C) \end{array} \right\}$$

$$\left\{ \begin{array}{l} even(A) \leftarrow succ(B, A), succ(C, B), even(C) \\ odd(A) \leftarrow succ(B, A), even(B) \end{array} \right\}$$

$$\left\{ \begin{array}{l} even(A) \leftarrow succ(B, A), succ(C, B), even(C) \\ odd(A) \leftarrow not\ even(A) \end{array} \right\}$$

In order to find only the more succinct hypotheses, additional constraints could be used. For instance, the constraint $\bot \leftarrow not\ max\_body(odd, 2)$, which states that rules defining the concept of *odd* should have a maximum of two body literals, would rule out the hypothesis

$$\left\{ \begin{array}{l} odd(A) \leftarrow succ(B, A), succ(C, B), not\ even(C) \\ even(A) \leftarrow succ(B, A),\ succ(C, B), even(C) \end{array} \right\}$$

6 Conclusion and Related Work

In this paper we have proposed a new notion of learning, where domain-dependent constraints can be specified as part of the learning task and implemented in terms of integrity constraints on the syntactic structure of compatible with the given mode declaration. *Acceptable* hypotheses are set of clauses that are compatible with the mode declaration, satisfy the domain-dependent constraints and (bravely) cover all the examples. We have shown how the meta-level representation and abductive search for inductive solutions is particularly suited to handle this type of learning tasks. This is demonstrated by considering the ASPAL learning system and extending it to support learning using constraint-driven bias. The new extended implementation has also been proved to be sound and complete with respect to our proposed new notion of learning. Finally, we have illustrated its applicability to two type of problems – goal model refinements in

requirements engineering and computation of stratified programs from a given search space. Preliminary experimental results show that the computational time is not affected by the extra constraint satisfaction problem, but in contrast it helps in reducing the magnitude of the number of computed hypotheses.

The notion of meta-level top theory for reasoning about hypotheses and for expressing constraints on answer sets is related to the approach in [12] where meta-level information is used to express constraints and preferences. Somewhat related to our approach is the work in [15] where knowledge is represented as causal graphs and constraints are added by specifying impossible connections between nodes. Similarly to the work in [15], Metagol$_D$ [22] handles the problem of learning through a meta-interpreter top theories by lifting the entire learning task into a second-order representation and making use of a meta-interpreter – theory of allowable rule formats – for restricting the hypothesis space. Our work differs from these two approaches in that we generally use meta-level representation only to represent the hypothesis space and constraints, rather than the entire background knowledge and examples. This is obviously with the exception of the stratification problem presented, where the global property (.e. stratification) that the constraints try to maintain in the search for hypotheses by definition applies not only to the learned hypothesis but also to an existing background knowledge.

More related to our work is the notion of production fields for enforcing constraints on the hypothesis [14]: a tuple $\langle L, C \rangle$ could be expressed where L is a literal and C is the condition to be satisfied for L to be included in the hypothesis. As far as we are aware, in past works, the condition in a production field has mainly been used to specify conditions like length of the clause or depth of a term. Our approach is able to enforce similar constraints for body declarations using the primitives $max\_body/2$ and $min\_body/2$. Furthermore, as shown in our case studies more complex constraints across different clauses can also be imposed. Other more loosely related approaches that have made use of enhanced language bias to apply additional constraints to the hypothesis space include [4] and [5]. They both add a cardinality constraint to each mode declaration to specify the minimal or maximal number of times the mode declaration can be used in the hypothesis. Past work that use integrity constraints in ILP includes [16], which focuses on how to check for constraint satisfiability, and with respect to meta-level information only uses arguments' types. This differs from our work which directly reasons on the structure of the hypothesis, and allows for the constraints to be defined over relationships between different rules.

In this paper we have only considered using one primitive of $\mathcal{L}_\mathcal{C}$ in each integrity constraint. For future work we intend to extend it to multiple primitives in the same constraint. This requires a more refined encoding in ASP in order to consider the primitives and their negated forms can be safely represented.

References

1. Alrajeh, D., Kramer, J., van Lamsweerde, A., Russo, A., Uchitel, S.: Generating obstacle conditions for requirements completeness. In: ICSE. pp. 705–715 (2012)

2. Alrajeh, D., Kramer, J., Russo, A., Uchitel, S.: Elaborating requirements using model checking and inductive learning. IEEE Trans. on SE 39(3), 361–383 (2013)
3. Apt, K.R., Blair, H.A., Walker, A.: Foundations of deductive databases and logic programming. chap. Towards a Theory of Declarative Knowledge, pp. 89–148 (1988)
4. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artif. Intell. 101(1-2), 285–297 (1998)
5. Bragaglia, S., Ray, O.: Nonmonotonic learning in large biological networks. In: 24th Int. Conference on Inductive Logic Programming (2014)
6. Chan, D.: Constructive negation based on the completed database. In: Logic Programming, Proceedings of the Fifth International Conference and Symposium, 1988 (2 Volumes). pp. 111–125 (1988)
7. Christiansen, H.: Executable specifications for hypothesis-based reasoning with prolog and constraint handling rules. J. Applied Logic 7(3), 341–362 (2009)
8. Corapi, D.: Nonmonotonic Inductive Logic Programming as Abductive Search. Ph.D. thesis, Imperial College London (2011)
9. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming as abductive search. In: Hermenegildo, M.V., Schaub, T. (eds.) ICLP (Technical Communications). LIPIcs, vol. 7, pp. 54–63 (2010)
10. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming in answer set programming. In: ILP. vol. 7207, pp. 91–97. Springer (2011)
11. Darimont, R., van Lamsweerde, A.: Formal refinement patterns for goal-driven requirements elaboration. In: FSE. pp. 179–190. ACM (1996)
12. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: Computing preferred answer sets by meta-interpretation in answer set programming. TPLP. 3(4), 463–498 (Jul 2003)
13. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Clingo* = ASP + control: Preliminary report. In: ICLP'14. vol. 14(4-5) (2014)
14. Inoue, K.: Induction as consequence finding. ML 55(2), 109–135 (May 2004)
15. Inoue, K., Doncescu, A., Nabeshima, H.: Completing causal networks by meta-level abduction. Machine Learning 91(2), 239–277 (2013)
16. Jorge, A., Brazdil, P.: Integrity constraints in ilp using a monte carlo approach. In: In Proc. 6th International Workshop on Inductive Logic Programming. pp. 137–151. Springer-Verlag (1996)
17. Kakas, A., Michael, A., Mourlas, C.: Aclp: Abductive constraint logic programming (2000)
18. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive logic programming. J. Log. Comput. 2(6), 719–770 (1992)
19. Kakas, A.C., Nuffelen, B.V., Denecker, M.: A-system: Problem solving through abduction. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, 2001. pp. 591–596. Morgan Kaufmann (2001)
20. Kowalski, R.A., Sergot, M.: A logic-based calculus of events. New generation computing 4(1), 67–95 (1986)
21. van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE trans. on SE 26(10), 978–1005 (2000)
22. Lin, D., Dechter, E., Ellis, K., Tenenbaum, J.B., Muggleton, S.: Bias reformulation for one-shot function induction. In: ECAI 2014. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 525–530 (2014)
23. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Springer (1992)
24. Sakama, C., Inoue, K.: Brave induction: a logical framework for learning from incomplete information. Machine Learning 76(1), 3–35 (2009)