

Inductive Learning using Constraint-driven Bias

Duangtida Athakravi, Dalal Alrajeh, Krysia Broda, and Alessandra Russo

Imperial College London

{duangtida.athakravi07,dalal.alrajeh,k.broda,a.russo}@ic.ac.uk

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 restrictive and lead to solutions that are irrelevant or inapplicable. This is particularly the case when hypothesis solutions 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 language for expressing domain-specific constraints over a given hypothesis space, and define a notion of *acceptable hypothesis* as solutions of an ILP task that conform to given domain-specific constraints over a language bias. We show how the ASPAL learning approach can be extended to compute acceptable hypothesis, by leveraging on its meta-level representation of hypothesis space and apply our new notion of learning to support revision of software system goals models.

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 Occam Razor’s principle, for which preferred solutions among competing alternative 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:

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

Under brave induction [10], a most compressed hypothesis would be the set of normal clauses $H_1 = \{even(X) \leftarrow not\ odd(X); odd(X) \leftarrow not\ even(X)\}$. The hypothesis $H_2 = \{even(X) \leftarrow succ(Y, X), not\ odd(Y); odd(X) \leftarrow succ(Y, X), not$

$even(Y)\}$, another solution of the learning task, would be more reasonable in the given 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*. We propose a language for expressing domain-specific constraints over a given hypothesis space and define a notion of *acceptable hypothesis* as solutions of an ILP task that conform to given domain-specific constraints over the language bias. For instance, in the above example, if a constraint were specified that requires hypothesis clauses to contain the body literal $succ/2$, then H_1 would be considered to be a non-*acceptable* solution. We show how the ASPAL learning approach [3] is well suited to compute acceptable hypotheses, by leveraging on its meta-level representation of hypothesis space and the fact that constraint satisfaction is already an integral part of the abductive inference framework [7] used by ASPAL. We then demonstrate the usefulness of this new notion of learning through constraint-driven bias in addressing the practical open problem of goal model refinement in requirements engineering.

2 Background

We briefly summarise the meta-level learning through abductive search (TAL), proposed in [2], and its ASP implementation (ASPAL) [3]. The TAL approach is a top-directed abductive learning framework for nonmonotonic ILP that translates an ILP task into an equivalent abductive task and computes inductive solutions as abductive assumptions over a declarative representation of the hypothesis space (i.e. top theory $\top$).

In the learning framework, a given ILP task with a set of ground examples E , background theory B and an hypothesis space R_M , is transformed into an equivalent ALP task with background knowledge $B \cup \top \cup \{examples \leftarrow \bigwedge_{e \in E} e\}$, set of abducibles $A^\top$, and integrity constraint $\{\perp \leftarrow not\ examples\}$. Clauses in $\top$ correspond to clauses in R_M . If $r = h \leftarrow \bar{b}$ is a clause in R_M , its corresponding declarative representation in $\top$ is $h \leftarrow \bar{b}, rule(id(h \leftarrow \bar{b}), \bar{C})$, where $\bar{C}$ is the list of constant arguments in r , $id(h \leftarrow \bar{b})$ is its encoding as a tuple $(m_h, m_1, l_{1,1}, \dots, l_{1,p_1}, \dots, m_n, l_{n,1}, \dots, l_{n,p_n})$, where m_h is the identifier of the mode declaration for h , m_i is the identifier of the mode declaration for body literal b_i in $\bar{b}$, and for each m_i , the elements $l_{j,1}, \dots, l_{j,p_j}$ are numbers denoting the variable arguments in r linked to b_i , counting from left to right. The condition $rule(id(h \leftarrow \bar{b}), \bar{C})$ is an abducible in $A^\top$. Suppose $\Delta \subseteq A^\top$ is the abductive answer to the generated ALP task, the inverse translation of each abducible in Δ is a clause in the corresponding inductive hypothesis H .

3 Learning using Meta-level Integrity Constraints

The class of domain-dependent constraints proposed in this paper are constraints on mode declarations that collectively characterise the class of acceptable hypotheses. We assume all mode declarations have unique labels: the label of a literal, compatible with a mode head or body, is given by its predi-

cate name and its constant as arguments. A negated body literal has the prefix “not_” added to its label. For instance, given a mode body declaration $modeb(not\ p(+int, \#int, -int))$, an associated label would be $not_p(2)$, where 2 is a constant of the given constant type. Notice that this gives literals different associated labels for different constants used if their mode declarations have constant arguments. We define an abstract language $\mathcal{L}_C$ to express our domain-specific constraints. The language is composed of six primitives whose arguments are either variables or constants. Variables can be mode head variables, which stand for any mode head declaration, or mode body variables that stand for any mode body declaration. Constants are labels referring to specific mode declarations. The primitives for expressing constraints on acceptable solutions are as follows, where the arguments can be either variables or constants: i) $in_some_rule(L_h, L_b)$, which means that there must exist a clause with a head literal L_h and body literal L_b ; ii) $in_all_rule(L_h, L_b)$, for all clauses with head literal L_h must have body literal L_b ; iii) $in_same_rule(L_h, L_{b_1}, L_{b_2})$, meaning that all clauses with the head literal L_h must have both body literals L_{b_1} and L_{b_2} , or neither of them; iv) $in_diff_rule(L_{b_1}, L_{b_2})$, which states that body literals L_{b_1} and L_{b_2} must appear in different clauses; v) $max_body(L_h, X)$, specifying that clauses with head literal L_h must have at most X number of body literals; and finally vi) $min_body(L_h, X)$ which says that clauses with head literal L_h must have at least X number of body literals. Not all of these primitives are orthogonal to each other: $in_same_rule(L_h, L_{b_1}, L_{b_2})$ can, for instance, be expressed using $in_some_rule(L_h, L_b)$. For convenience we have separated them into different predicates. From these primitives, we can now define the domain-dependent constraints. Using $\mathcal{L}_C$ and the background theory of the original inductive task defined using language $\mathcal{L}_B$, the user can specify integrity constraints as follows.

Definition 1. Let B be a background knowledge expressed in a language $\mathcal{L}_B$ and M a set of mode declarations with associated constraint language $\mathcal{L}_C$. A domain-dependent constraint in $\mathcal{L}_C$ is $\perp \leftarrow C_1, \dots, C_k, not\ C_{k+1}, \dots, not\ C_n, \overline{C_B}$ where C_i , for $1 \leq i \leq n$, is a primitive of $\mathcal{L}_C$ and $\overline{C_B}$ is a conjunction of literals from the language $\mathcal{L}_B$ only.

Consider the example in Section 1. The constraint $\perp \leftarrow not\ in_all_rule(L_h, succ)$ makes hypothesis $\{even(X) \leftarrow not\ odd(X); odd(X) \leftarrow not\ even(X)\}$ not acceptable.

Table 1 gives the declarative semantics of each primitive in $\mathcal{L}_C$. The meta-predicate $is_rule(R, L_h)$ represents a clause with identifier R and head literal labelled L_h , the meta-predicate $in_rule(R, L_h, L_b)$ indicates a clause with identifier R , head literal labelled L_h and body literal labelled L_b . Note that the unique identifier R is assumed to be automatically generated by the learning system for each clause in the hypothesis space, differently from the head and body literal labels which are specified by the user as part of the learning task definition.

If $h \leftarrow b_1, \dots, b_n$ is a learned clause H with identifier R_{id} , its associated set of meta predicates, denoted with $meta(H)$, includes the instances $is_rule(R_{id}, L_h)$, $in_rule(R_{id}, L_h, L_{b_1})$, $\dots$, $in_rule(R_{id}, L_h, L_{b_n})$.

Definition 2. Let $\langle B, M, E, IC \rangle$ be an inductive learning task with background knowledge B , mode declarations M , examples E and domain-specific constraints

Primitive predicate P	Declarative semantics of $\leftarrow P$
$not\ in_some_rule(L_h, L_b)$	$ic_cond \leftarrow in_rule(Rid, L_h, L_b)$ $\perp \leftarrow not\ ic_cond$
$in_some_rule(L_h, L_b)$	$\leftarrow in_rule(Rid, L_h, L_b)$
$not\ in_all_rule(L_h, L_b)$	$\perp \leftarrow is_rule(Rid, L_h), not\ in_rule(Rid, L_h, L_b)$
$in_all_rule(L_h, L_b)$	$ic_cond \leftarrow is_rule(Rid, L_h)$ $ic_cond \leftarrow is_rule(Rid, L_h), not\ in_rule(Rid, L_h, L_b)$ $\perp \leftarrow not\ ic_cond$
$not\ in_same_rule(L_h, L_{b_1}, L_{b_2})$	$\perp \leftarrow in_rule(Rid, L_h, L_{b_1}), not\ in_rule(Rid, L_h, L_{b_2})$ $\perp \leftarrow not\ in_rule(Rid, L_h, L_{b_1}), in_rule(Rid, L_h, L_{b_2})$
$in_same_rule(L_h, L_{b_1}, L_{b_2})$	$\perp \leftarrow in_rule(Rid, L_h, L_{b_1}), in_rule(Rid, L_h, L_{b_2})$
$not\ in_diff_rule(L_{b_1}, L_{b_2})$	$ic_cond \leftarrow in_rule(Rid_1, L_{h_1}, L_{b_1}), in_rule(Rid_2, L_{h_2}, L_{b_2}),$ $Rid_1 \neq Rid_2$ $\perp \leftarrow not\ ic_cond$
$in_diff_rule(L_{b_1}, L_{b_2})$	$\perp \leftarrow in_rule(Rid_1, L_{h_1}, L_{b_1}), in_rule(Rid_2, L_{h_2}, L_{b_2}),$ $Rid_1 \neq Rid_2$
$not\ max_body(L_h, X)$	$\perp \leftarrow is_rule(Rid, L_h), X + 1\{in_rule(Rid, L_h, L_b)\}$
$max_body(L_h, X)$	$ic_cond \leftarrow is_rule(Rid, L_h), X + 1\{in_rule(Rid, L_h, L_b)\}$ $\perp \leftarrow not\ ic_cond$
$not\ min_body(L_h, X)$	$\perp \leftarrow is_rule(Rid, L_h), 0\{in_rule(Rid, L_h, L_b)\}X - 1$
$min_body(L_h, X)$	$ic_cond \leftarrow is_rule(Rid, L_h), 0\{in_rule(Rid, L_h, L_b)\}X - 1$ $\perp \leftarrow not\ ic_cond$

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

IC. The set of clauses H is an acceptable solution if and only if (i) $H \subseteq R_M$, where R_M is the hypothesis space induced by M ; ii) $B \cup H \models \bigwedge_{e \in E} e$; and iii) $B \cup H \cup meta(H) \models \bigwedge_{ic \in IC} ic$, where $meta(H)$ is the set of meta-predicate instances associated with H .

Leveraging on the meta-level abductive search approach of ASPAL, we have been able to extend the implementation to compute acceptable solutions of a learning task with domain-specific constraints. The ASPAL top theory $\top$ has been extended to include as abducibles not only the ASPAL rule encoding instances [3] but also meta-predicate instances. Each clause $h \leftarrow b_1, \dots, b_n$, $rule(id(h \leftarrow b_1, \dots, b_n), \bar{C})$ in the top theory is replaced with the clauses:

$$\begin{aligned}
& h \leftarrow b_1, \dots, b_n, rule(id(h \leftarrow b_1, \dots, b_n), \bar{C}), \\
& \quad is_rule(i(\bar{C}), l_h), in_rule(i(\bar{C}), l_h, l_{b_1}), \dots, in_rule(i(\bar{C}), l_h, l_{b_n})). \\
& \leftarrow 1\{rule(id(h \leftarrow b_1, \dots, b_n), \bar{C}), \\
& \quad is_rule(i(\bar{C}), l_h), in_rule(i(\bar{C}), l_h, l_{b_1}), \dots, in_rule(i(\bar{C}), l_h, l_{b_n}))\}n + 1.
\end{aligned}$$

where $i(\bar{C})$ is a unique identifier for the clause, l_h is the label of head predicate, and l_{b_j} is the label of predicate b_j , for $1 \leq j \leq n$. During the computation, the abductive assumption of a rule encoding $rule(id(h \leftarrow b_1, \dots, b_n))$ of a possible hypothesis is controlled by the constraint which imposes all associated meta-predicates to be abduced provided that they satisfy the domain-specific constraints. The top theory for the example given in Section 1 includes the clauses:

$$\begin{aligned}
& even(X) \leftarrow succ(Y, X), not\ even(Y), rule((m1, m5, 1, m3, 2), c), \\
& \quad is_rule(1, even), in_rule(1, even, succ), in_rule(1, even, not_even).
\end{aligned}$$

$$\perp \leftarrow 1\{rule((m3, m5, 1, m6, 2), c), is_rule(1, even), in_rule(1, even, succ), in_rule(1, even, not_even)\}3.$$

The set of abducibles is in this case $\{rule((m3, m5, 1, m6, 2), c), is_rule(1, even), in_rule(1, even, succ), in_rule(1, even, not_even)\}$.

4 Case Study

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 [9]) that guarantees these refinement semantics is given in [4]. An example of a goal model for a Flight Control System described in [1] is given in Figure 1 in which the top-level goal states that whenever the plane is moving on the runway, the reverse thrust shall be enabled at the next time point.

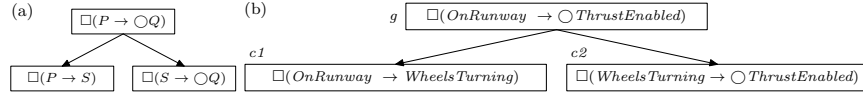


Fig. 1. (a) A goal pattern from [4] where $\Box$ means always, $\bigcirc$ means at the next time point, $\rightarrow$ for logical implication, (b) FCS goal model with goals labelled g , $c1$ and $c2$.

One problem with early driven goal models 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 wheels sensors are flagged and the wheels are not turning, hence violating the goal $c1$. To revise the model, each assertion in Figure 1.b is mapped into a set of rules, using an Event Calculus formalism [8] similar to the one presented in [1], that captures the syntactic formulation of the goals¹. The predicates $a_holds/3$ (resp. $c_holds/3$) represent the notion that the antecedent (resp. consequent) of the goal holds at time point in the scenario.

$$R = \left\{ \begin{array}{l} a_holds(c1, T, S) \leftarrow holds_at(onRunway, T, S). \\ c_holds(c1, T, S) \leftarrow holds_at(wheelsTurning, T, S). \\ a_holds(c2, T, S) \leftarrow holds_at(wheelsTurning, T, S). \\ c_holds(c2, T, S) \leftarrow holds_at(thrustEnabled, T, S). \end{array} \right\}$$

We use our new notion of learning to compute acceptable revisions of the goals 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 a_holds label as head, must also appear in the body of a rule with c_holds label as head.

$$IC = \left\{ \begin{array}{l} \perp \leftarrow in_some_rule(a_holds(C1), BL), right_child_of(G, C1), \\ \quad left_child_of(G, C2), C1 \neq C2, not\ in_some_rule(c_holds(C2), BL). \\ \perp \leftarrow in_some_rule(c_holds(C1), BL), left_child_of(G, C1), \\ \quad right_child_of(G, C2), C1 \neq C2, not\ in_some_rule(a_holds(C2), BL). \end{array} \right\}$$

¹ We do not present here details of the mapping from temporal logic to logic programs as this is outside the scope of this paper.

Using ASPAL without domain-specific constraints to compute revisions would generate the revised theory R'_1 as most compressed hypothesis, which does not preserve the refinement semantics of the goal model. On the other hand, using ASPAL with domain-specific constraints that capture the goal-model refinement semantics gives us the acceptable revised theory R'_2 , which although not minimal, satisfies the refinement semantics.

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

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

5 Conclusion and Related Work

In this paper we have proposed the idea of domain-specific constraints as a way for defining and computing *acceptable* hypotheses and briefly shown its use for goal model refinements in requirements engineering. The use of meta-predicates in the top theory of the ASPAL implementation, for expressing constraints on answer sets, is related to the approach in [5] where meta-level information is used to express constraints and preferences. Somewhat related to our approach is the work in [6] where knowledge is represented as causal graphs and constraints are added by specifying impossible connections between nodes. Our approach differs in that meta-level representation is only used for representing hypothesis space and constraints, and not the background knowledge or examples.

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. 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)
3. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming in answer set programming. In: ILP. vol. 7207, pp. 91–97. Springer (2011)
4. Darimont, R., van Lamsweerde, A.: Formal refinement patterns for goal-driven requirements elaboration. In: FSE. pp. 179–190. ACM (1996)
5. 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)
6. Inoue, K., Doncescu, A., Nabeshima, H.: Completing causal networks by meta-level abduction. Machine Learning 91(2), 239–277 (2013)
7. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive logic programming. J. Log. Comput. 2(6), 719–770 (1992)
8. Kowalski, R.A., Sergot, M.: A logic-based calculus of events. New generation computing 4(1), 67–95 (1986)
9. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Springer (1992)
10. Sakama, C., Inoue, K.: Brave induction: a logical framework for learning from incomplete information. Machine Learning 76(1), 3–35 (2009)