Scalable Non-observational Predicate Learning in ASP

Mark Law[1*], **Alessandra Russo**[1], **Krysia Broda**[1] and **Elisa Bertino**[2]
[1]Imperial College London, UK
[2]Purdue University, USA
{mark.law09, a.russo, k.broda}@imperial.ac.uk, bertino@purdue.edu

Abstract

Recently, novel ILP systems under the answer set semantics have been proposed, some of which are robust to noise and scalable over large hypothesis spaces. One such system is FastLAS, which is significantly faster than other state-of-the-art ASP-based ILP systems. FastLAS is, however, only capable of Observational Predicate Learning (OPL), where the learned hypothesis defines predicates that are directly observed in the examples. It cannot learn knowledge that is indirectly observable, such as learning causes of observed events. This class of problems, known as non-OPL, is known to be difficult to handle in the context of non-monotonic semantics. Solving non-OPL learning tasks whilst preserving scalability is a challenging open problem.

We address this problem with a new abductive method for translating examples of a non-OPL task to a set of examples, called *possibilities*, such that the original example is covered iff at least one of the possibilities is covered. This new method allows an ILP system capable of performing OPL tasks to be "upgraded" to solve non-OPL tasks. In particular, we present our new FastNonOPL system, which upgrades FastLAS with the new possibility generation. We compare it to other state-of-the-art ASP-based ILP systems capable of solving non-OPL tasks, showing that FastNonOPL is significantly faster, and in many cases more accurate, than these other systems.

1 Introduction

The goal of Inductive Logic Programming (ILP) [Muggleton, 1991] is to find a set of logical rules, called a hypothesis, that, together with some existing background knowledge, explains a set of examples. Many of the early ILP systems were tailored to solve *Observational Predicate Learning* (OPL) tasks, where the concept (or predicate) to be learned is directly observable from the examples. But in practice, learnable concepts often impact the observable world only indirectly, through some (already known) background knowledge. For instance, consider the following example policy learning problem. An organisation is migrating from one access control policy system to a new one based on security levels for files and clearance levels for people. Any person may access any file whose security level is lower than or equal to their clearance level. Rules defining the security levels of files and the clearance levels of people can be learned from logs of the previous policy that say which people should have access to which files. An instance of this problem is shown in Figure 1. Examples of `has_access` are given, and because the concepts of security level and clearance level impact `has_access` through the background knowledge, even though no explicit examples of these concepts are given, they are learnable. Learning concepts (or predicates) that are not directly observable is known as *non-Observational Predicate Learning* (non-OPL).

Non-OPL tasks are known to be difficult to handle, especially in the context of non-monotonic semantics. More modern non-monotonic ILP systems, such as [Ray, 2009; Corapi *et al.*, 2010; Corapi *et al.*, 2012; Law *et al.*, 2014], are able to solve non-OPL tasks but at the cost of scalability over large hypothesis spaces. On the other hand, FastLAS [Law *et al.*, 2020a], a novel non-monotonic ILP system under the answer set semantics [Gelfond and Lifschitz, 1988; Brewka *et al.*, 2011], has been proven to be significantly faster and more scalable to larger hypothesis spaces than other current state-of-the-art ASP-based ILP systems. But, FastLAS is only capable of solving OPL tasks and solving non-OPL learning tasks whilst preserving scalability still remains an open problem.

In this paper, we present a new system, called *FastNonOPL*, which maintains the scalability of FastLAS with respect to large hypothesis spaces, but expands the applicability to non-OPL tasks and programs with multiple answer sets. FastNonOPL relies upon a new approach called *possibility generation*, which translates each example of a given non-OPL task to a set of "observational" examples called *possibilities* such that the original example is covered if and only if at least one possibility is covered. In non-OPL tasks where each example leads to a single possibility, each of the original examples can be replaced with its unique possibility and FastLAS can be used to solve the translated task. We call such non-OPL tasks *non-branching*. In the case of *branching*

[*]Contact Author

File	Employment	Financial	Trade Secrets
f_1	No	Yes	No
f_2	Yes	Yes	No
f_3	No	No	Yes

Person	Role	Access
Alice	CEO	$\{f_1, f_2, f_3\}$
Bob	Manager	$\{f_1, f_2\}$
Charlie	Analyst	$\{f_1\}$
David	Cleaner	$\emptyset$

Background:

```
has_access(P,F) :- person(P), file(F), not denied(P,F).
denied(P,F) :- not cleared_to_see(P,L), sec_lv(F,L), person(P).
cleared_to_see(P,0..L) :- clr_lv(P,L).
```

Solution:

```
clr_lv(P,3) :- role(P,ceo).          sec_lv(F,3) :- trade_secrets(F).
clr_lv(P,2) :- role(P,manager).      sec_lv(F,2) :- employment(F).
clr_lv(P,1) :- role(P,analyst).      sec_lv(F,1) :- financial(F).
```

Figure 1: Policy Example. This representation allows files to have multiple security levels, the highest of which is effective.

non-OPL tasks, where examples lead to multiple possibilities, FastLAS cannot be applied directly. A modified version of FastLAS is used which tries to cover one possibility for each example, instead of attempting to cover all possibilities. The possibility generation approach is general, so in principle the modifications to FastLAS could be made to any "observational" ILP system to allow it to be used in a similar pipeline to solve non-OPL tasks.

We evaluate FastNonOPL on datasets which require systems to solve both branching and non-branching tasks. FastNonOPL maintains the scalability of FastLAS over the hypothesis space on non-branching tasks and can also be applied to branching tasks, outperforming the state-of-the-art ILASP [Law *et al.*, 2015a] ASP-based ILP system in terms of accuracy and running time.

The rest of the paper is structured as follows. The next section recalls background material. The two subsequent sections present the FastNonOPL pipeline, and a method for possibility generation, together with proofs of correctness. These are followed by an evaluation of FastNonOPL. The paper concludes with discussions of related and future work.

2 Preliminaries

In this section we introduce basic notions and terminologies used throughout the paper. Given any atoms $\mathtt{h}$, $\mathtt{b_1}, \ldots, \mathtt{b_n}$, $\mathtt{c_1}, \ldots, \mathtt{c_m}$, a *normal rule* is of the form $\mathtt{h}$:- $\mathtt{b_1}, \ldots, \mathtt{b_n}$, not $\mathtt{c_1}, \ldots,$ not $\mathtt{c_m}$, where $\mathtt{h}$ is the *head* and $\mathtt{b_1}, \ldots, \mathtt{b_n}$, not $\mathtt{c_1}, \ldots,$ not $\mathtt{c_m}$ (collectively) is the *body* of the rule, and "not" represents negation as failure. A rule is *safe* if every variable in the rule occurs in at least one positive literal (i.e. the $\mathtt{b_i}$'s in the above rule) in the body of the rule. A program is a set of safe normal rules. The Herbrand Base of a program P, denoted HB_P, is the set of variable free (ground) atoms that can be formed from predicates and constants in P. The subsets of HB_P are called the (Herbrand) interpretations of P. Given a program P and an interpretation $I \subseteq HB_P$, the *reduct* P^I is constructed from the grounding of P in two steps: first, remove rules whose bodies contain the negation of an atom in I; then remove all negative literals from the remaining rules. Any $I \subseteq HB_P$ is an *answer set* of P if it is a minimal model of the reduct P^I. We denote the set of answer sets of a program P with $AS(P)$.

A *partial interpretation* e_{pi} is a pair of disjoint sets of atoms $\langle e_{pi}^{inc}, e_{pi}^{exc} \rangle$ called the *inclusions* and *exclusions* respectively. An interpretation I *extends* e_{pi} (written $I \triangleleft e_{pi}$) iff $e_{pi}^{inc} \subseteq I$ and $e_{pi}^{exc} \cap I = \emptyset$. A *weighted context-dependent partial interpretation* (WCDPI) is a tuple $e = \langle e_{id}, e_{pen}, e_{pi}, e_{ctx} \rangle$, where e_{id} is an identifier for e, e_{pen} is either a positive integer or ∞ called a penalty, e_{pi} is a partial interpretation and e_{ctx} is a program consisting of normal rules, called a *context*. A WCDPI e is *accepted* by a program P iff there is an answer set of $P \cup e_{ctx}$ that extends e_{pi}. Often, it is convenient to discuss WCDPI's without considering the penalty and identifier, as *CDPIs* of the form $\langle e_{pi}, e_{ctx} \rangle$.

Many ILP systems (e.g. [Muggleton, 1995; Ray, 2009; Srinivasan, 2001]) use mode declarations as a form of language bias to specify hypothesis spaces. A *mode bias* M is defined as a pair of sets of mode declarations $\langle M_h, M_b \rangle$, where M_h (resp. M_b) are called the *head* (resp. *body*) *mode declarations*. Each mode declaration is a literal whose abstracted arguments are either `var(t)` or `const(t)`, for some constant `t` (called a *type*). Informally, a literal is *compatible* with a mode declaration m if it can be constructed by replacing every instance of `var(t)` in m with a variable of type `t`, and every `const(t)` with a constant of type `t`.[1] For instance, `clr_lv(P,1)` is compatible with the mode declaration `clr_lv(var(person),const(lv))` (where `P` is a variable of type `person` and `1` is a constant of type `lv`).

Definition 1 *Given a mode bias $M = \langle M_h, M_b \rangle$, the* search space *$S_M$ is the set of normal rules R s.t. (i) the head of R is compatible with a mode declaration in M_h; (ii) each body literal of R is compatible with a mode declaration in M_b; and (iii) no variable occurs with two different types.*

We now recall the definition of a learning task, as used by the ILASP system [Law *et al.*, 2015a]. We consider a simplification which only allows positive examples.

Definition 2 *A* Positive Learning from Answer Sets *(ILP_{LAS}^{+}) task is a tuple $T = \langle B, M, E^+ \rangle$ where B is a program, M is a mode bias and E^+ is a finite set of WCDPIs. For any hypothesis $H \subseteq S_M$:*

- *For any $e \in E^+$, H* covers *e iff $B \cup H$ accepts e.*
- *$\mathcal{S}_{len}(H, T)$ is the number of literals in H plus the penalty of each example in T which is not covered by H.*

[1]The set of constants of each type is given with a task, together with the maximum number of variables in a rule, giving a set of variables $\mathtt{V_1}, \ldots, \mathtt{V_{max}}$ that can occur in each rule of a hypothesis. Whenever a variable `V` of type `t` occurs in a rule, the atom `t(V)` is added to the body of the rule to enforce the type.

- *H is an* optimal solution *of T iff $\mathcal{S}_{len}(H, T)$ is finite and there is no $H' \subseteq S_M$ s.t. $\mathcal{S}_{len}(H', T) < \mathcal{S}_{len}(H, T)$.*

The goal of any system for ILP^+_{LAS} is to find an optimal solution of an input ILP^+_{LAS} task.

Definition 3 *A task $T = \langle B, M, E^+\rangle$ is* non-observational *if S_M contains a rule whose head uses a predicate which occurs in a rule body either in B or in a context in E^+.*

Example 1 *Reconsider the problem in Figure 1, which can be formalised as an ILP^+_{LAS} task. B is the background in the figure plus the facts defining the domain of each type (e.g. $\{\texttt{lv(1)}.\ \ldots\ \texttt{lv(3)}.\}$ to define the security/clearance levels). E^+ contains a set of examples, one for each person/file pair (there are 12 such examples). One such example e has the context $\{\texttt{role(alice, ceo)}.\ \texttt{financial(f1)}.\}$ and the partial interpretation $\langle\{\texttt{has\_access(alice, f1)}\}, \emptyset\rangle$ (for files that cannot be accessed, the $\texttt{has\_access}$ atom is in the exclusions). Finally, the mode declarations are as follows:*

$$M_h = \left\{ \begin{array}{l} \texttt{clr\_lv(var(person), const(lv))}, \\ \texttt{sec\_lv(var(file), const(lv))} \end{array} \right\}$$

$$M_b = \left\{ \begin{array}{l} \texttt{role(var(person), const(role))}, \\ \texttt{employment(var(file))}, \\ \texttt{trade\_secrets(var(file))}, \\ \texttt{financial(var(file))} \end{array} \right\}$$

This task is non-observational because there is a predicate in M_h which occurs in the body of a rule in B.

FastLAS [Law *et al.*, 2020a] is an algorithm for solving a subset of ILP^+_{LAS} tasks. At the core of the algorithm is a notion of an *OPT-sufficient* subset of the hypothesis space. A subset of the hypothesis space of a task is OPT-sufficient iff it either contains at least one optimal solution of the task or the original task is unsatisfiable. FastLAS works by first computing an OPT-sufficient subset of the hypothesis space, and then searching within it for an optimal solution. We omit the details of how FastLAS constructs this OPT-sufficient subset, as they are not necessary to understand the rest of the paper. FastLAS has a number of restrictions which limit its applicability; for instance, it is incapable of non-OPL or learning programs with multiple answer sets. Specifically, an ILP^+_{LAS} task $T = \langle B, M, E^+\rangle$ is a *FastLAS task* if for each $e \in E^+$, $|AS(B \cup e_{ctx})| = 1$ and no predicate in M_h occurs in M_b or in any rule body in $B \cup e_{ctx}$. The method presented in this paper lifts some of these restrictions to allow non-OPL and learning programs with multiple answer sets.

3 The FastNonOPL Pipeline

The subset of ILP^+_{LAS} tasks supported by FastNonOPL is formalised by the following definition. For any program P, *head_preds*(P) and *body_preds*(P) denote the predicates occurring in the heads and bodies (respectively) of the rules in P.

Definition 4 *An ILP^+_{LAS} task $T = \langle B, M, E^+\rangle$ is* non-recursive *if for each $e \in E^+$, $B \cup e_{ctx}$ can be partitioned into two programs bottom$(B \cup e_{ctx})$ and top$(B \cup e_{ctx})$ s.t. no predicate in M_h or the head of a rule in top$(B \cup e_{ctx})$ occurs in M_b or the body of a rule in bottom$(B \cup e_{ctx})$. In other words, for any $H \subseteq S_M$, head_preds(top$(B \cup e_{ctx}) \cup H$) and body_preds(bottom$(B \cup e_{ctx}) \cup H$) are disjoint.*

The intuition is that $B \cup e_{ctx} \cup H$ has three components: (1) a bottom program defining the predicates used in the bodies of the learned rules in H; (2) a middle program (H); and (3) a top program, which uses the predicates defined by H to define further predicates that may be used in the inclusions or exclusions of the example. Although this partitioning is not necessarily unique, we assume (w.l.o.g.) a fixed partitioning and refer to *the* bottom and top programs. The task is less restrictive than FastLAS's task, which would correspond, in these terms, to a non-recursive task with an empty top program and a bottom program with one answer set. For the rest of this paper all tasks are assumed to be non-recursive.

Example 2 *The task in Example 1 is a non-recursive task. For each of the examples $e \in E^+$, bottom$(B \cup e_{ctx})$ is the program $B_f \cup e_{ctx}$, where B_f is the set of facts defining the domains of types and top$(B \cup e_{ctx}) = B \backslash B_f$.*

FastNonOPL addresses the problem of non-OPL using a new approach for translating a "non-observational" example e into a set of "observational" examples. These new observational examples are the possible ways that e could be covered, given a background knowledge B, and so we refer to them as the *possibilities* of e. A possibility p is a CDPI $\langle p_{pi}, p_{as}\rangle$, with the rough intuition being that p_{as} is an answer set of the bottom program (*bottom*$(B \cup e_{ctx})$) and p_{pi} is a partial interpretation (over the ground instances of heads of rules in the hypothesis space), such that covering p_{pi} (i.e. proving the inclusions and none of the exclusions) is sufficient to cover the original partial interpretation e_{pi}. Examples may have a single possibility, many possibilities or even no possibilities. There are two ways that an example can have multiple possibilities: firstly, the bottom program could have multiple answer sets – in this case, any one of the answer sets could be used to cover the example; secondly, non-OPL can lead to multiple ways to prove the inclusions of a given example, leading to distinct partial interpretations p_{pi}. Note that in this definition (with a minor abuse of notation) p_{as} is treated as both an answer set and a set of facts.

Definition 5 *Let T be a learning task, e be a WCDPI and Ab be the set of atoms which occur in the head of at least one rule in ground(S_M). A* possibility *p of e (w.r.t. T) is a CDPI $\langle p_{pi}, p_{as}\rangle$ s.t.:*

1. *$p_{as} \in AS($bottom$(B \cup e_{ctx}))$;*
2. *p^{inc}_{pi}, $p^{exc}_{pi} \subseteq Ab$;*
3. *$\forall \Delta \subseteq Ab$ s.t. $\Delta \lhd p_{pi}$, $\exists I \in AS(p_{as} \cup \Delta \cup$ top$(B \cup e_{ctx}))$ s.t. $I \lhd e_{pi}$.*

A possibility p is said to be *minimal* if there is no possibility p' of e s.t. $p' \neq p$, $p_{as} = p'_{as}$, $p'_{inc} \subseteq p_{inc}$ and $p'_{exc} \subseteq p_{exc}$. We write *poss*$(T, e)$ (resp. *poss*$^*(T, e)$) to denote the set of all possibilities (resp. minimal possibilities) of e.

While the task of generating possibilities shares some similarity with standard definitions of abduction – e.g. [Kakas *et al.*, 1992] – the definition of a possibility is considerably stronger than that of an abductive solution. Usually, abductive solutions are complete interpretations of the abducibles, meaning that if $\delta \not\in \Delta$, then δ is assumed to be false. Conversely, the first component of a possibility is a partial interpretation of the abducibles s.t. every extension Δ is similar to a traditional abductive solution. This stronger definition is required by FastNonOPL to guarantee the correctness when translating multiple examples.

Example 3 *Reconsider the task and example e in Example 1. There are four minimal possibilities. The context p_{as} of each is the answer set of bottom$(B \cup e_{ctx})$ ($\{\mathtt{role(alice, ceo).\ financial(f1).}\} \cup B_f$, where B_f is the set of facts defining the domains of types) and the four partial interpretations are:*

1. $p^1_{pi} = \langle\{\mathtt{clr\_lv(alice, 3)}\}, \emptyset\rangle$;
2. $p^2_{pi} = \langle\{\mathtt{clr\_lv(alice, 2)}\}, \{\mathtt{sec\_lv(f1, 3)}\}\rangle$;
3. $p^3_{pi} = \langle\{\mathtt{clr\_lv(alice, 1)}\}, \{\mathtt{sec\_lv(f1, 2)}, \mathtt{sec\_lv(f1, 3)}\}\rangle$; *and*
4. $p^4_{pi} = \langle\emptyset, \{\mathtt{sec\_lv(f1, 1)}, \mathtt{sec\_lv(f1, 2)}, \mathtt{sec\_lv(f1, 3)}\}\rangle$.

Each of the first three partial interpretations specifies that `alice` *must have a clearance level n such that* `f1` *does not have a security level higher than n (the maximum security level is 3). The final partial interpretation represents the case where* `f1` *has no security level. In this case, according to the background knowledge,* `f1` *can be accessed by Alice no matter what her clearance level is.*

The solution given in Figure 1 covers the possibility constructed using the first partial interpretation ($B \cup H \cup p_{as}$ has exactly one answer set and, as this answer set contains $\mathtt{clr\_lv(a, 3)}$, *it extends p^1_{pi}). Note that to actually learn the hypothesis in Figure 1, further examples would need to be given (including examples relating to managers and analysts).*

Theorem 1 shows that we can determine whether an example is accepted using its minimal possibilities. The proofs of all theorems are given in the online supplementary material document, available at https://github.com/spike-imperial/FastLAS/blob/master/fast_non_opl_proofs.pdf.

Theorem 1 *Let T be a learning task and e be a WCDPI. For any hypothesis $H \subseteq S_M$, $B \cup H$ accepts e iff there is at least one possibility $p \in poss^*(T, e)$ s.t. H accepts p.*

We call a learning task *non-branching* if each example has at most one possibility and *branching* if this is not the case. The significance of non-branching tasks is that if a general problem is guaranteed to always produce a non-branching learning task, then the single possibilities can be precomputed and solved using an "observational" ILP system such as FastLAS.[2] Theorem 2 demonstrates that even for branching tasks, most of the FastLAS algorithm can be used unchanged, computing an OPT-sufficient subset as before.

Theorem 2 *Let $T = \langle B, M, E^+\rangle$, and T_{poss} be the FastLAS task $\langle\emptyset, M, \{\langle p_{id}, 1, p_{pi}, p_{as}\rangle \mid e \in E^+, p \in poss^*(T, e)\}\rangle$. The subset of the hypothesis space constructed by FastLAS for T_{poss} is OPT-sufficient for T.*

After calculating the set of possibilities for each example, FastLAS can be used to generate an OPT-sufficient subset of the hypothesis space. Finding an optimal solution within this subset can then be done as in FastLAS's *solving* stage, but with the minor difference that (in the non-noisy case) FastNonOPL searches for a hypothesis that covers at least one possibility of each example (rather than covering each example). Just as in FastLAS, this final solving stage can be encoded in ASP[3] and solved using the ASP solver Clingo [Gebser *et al.*, 2016]. As this process will find an optimal solution within any OPT-sufficient subset, FastNonOPL is guaranteed to return an optimal solution of any non-recursive task.

4 Using Abduction to Generate Possibilities

This section describes the abductive method used to generate all possibilities for an example. The method works by iteratively extending a set of CDPIs. In each iteration it performs a kind of "anti-abduction" to search for *exceptions* to the CDPIs found in the previous iteration and performs "conventional" abduction[4] to find *fixes* to the exceptions. Example 4 shows a simple learning task to give the intuition.

Example 4 *Consider a task with $B = \{$*`q:- not r, not s. r:-t, not u.`*$\}$, $S_M = \{$*`s. t. u.`*$\}$ and a set of examples E^+ including the CDPI $e = \langle\langle\{\mathtt{q}\}, \emptyset\rangle, \emptyset\rangle$. Our approach starts from a set of* partial possibilities *(of e) pp. Usually, this is a set of CDPIs of the form $\langle\langle\emptyset, \emptyset\rangle, A\rangle$, where A is an answer set of bottom$(B \cup e_{ctx})$, but in this case, bottom$(B \cup e_{ctx}) = \emptyset$, so there is only one CDPI p in pp (where $A = \emptyset$).*

Although in this case the empty hypothesis covers e, other examples in E^+ may require learning rules which cause e not to be covered, which is why the definition of possibilities in the previous section requires every extension of p to cover e. Note that p is not a (complete) possibility because there are some $\Delta \subseteq Ab$ s.t. $\Delta \lhd p_{pi}$ and $p_{as} \cup \Delta \cup top(B \cup e_{ctx})$ has no answer sets that extend e_{pi}. In this case, there are two minimal such Δ: $\{\mathtt{s}\}$ and $\{\mathtt{t}\}$. These are called exceptions *to p, and finding all such minimal exceptions is what we call anti-abduction. There are two ways of resolving exceptions. One way (called a* negative fix*) is to extend the exclusions of p so that none of the minimal exceptions can occur – in this instance, this yields the new possibility $\langle\langle\emptyset, \{\mathtt{s}, \mathtt{t}\}\rangle, \emptyset\rangle$. Extensions constructed in this way are guaranteed to be (complete) possibilities. The other way is to resolve an individual exception by abducing further inclusions. In this case, adding* `u` *to p^{inc}_{pi} resolves the first exception – we call $\{\mathtt{u}\}$ a* positive fix *of the exception. Extensions constructed in this way may still*

[2] To use the possibility generation method described in this paper with another observational ILP system it must (a) support CDPIs and (b) either support, or be modified to support groups of examples such that at least one should be covered. Note that as ILP^+_{LAS} tasks can be translated to brave induction tasks (used by many other ASP-based ILP systems), many ASP-based ILP systems for brave induction can be used, provided they satisfy (b). We modified FastLAS to support (b).

[3]The encoding in the online supplementary material document.

[4]Both "anti-abduction" and "conventional" abduction are achieved with ASP encodings, which are given in the supplementary material document.

Algorithm 1 $abduce\_possibilities(T, e)$

```
$pp = \{\langle\langle\emptyset,\emptyset\rangle, \{\mathtt{a.} \mid \mathtt{a} \in A\}\rangle | A \in AS(bottom(B \cup e_{ctx}))\}$;
$poss = \emptyset$;
while $pp \neq \emptyset$ do
    $pp' = \emptyset$;
    for $p \in pp$ do
        $p\_exceptions = m\_except(T, e, p)$;
        for $\Delta \in p\_exceptions$ do
            for $\Delta^{+}_{fix} \in m\_fix^{+}(T, e, p, \Delta)$ do
                $pp' = pp' \cup \{\langle\langle p^{inc}_{pi} \cup \Delta^{+}_{fix}, \emptyset\rangle, p_{as}\rangle\}$;
        for $\Delta^{-}_{fix} \in m\_fix^{-}(T, p\_exceptions)$ do
            $poss = poss \cup \{\langle\langle p^{inc}_{pi}, \Delta^{-}_{fix}\rangle, p_{as}\rangle\}$;
    $pp = pp'$;
return $poss$;
```

have exceptions, and thus require further iterations. These concepts are formalised by the next definition.

Definition 6 *Let T be a learning task and e be a WCDPI, $p = \langle p_{pi}, p_{as}\rangle$ be a CDPI and Ab be the set of atoms which occur in the head of at least one rule in ground(S_M).*

- *Any $\Delta \subseteq Ab$ is an* exception *to p if $\Delta \lhd p_{pi}$ and $\nexists A \in AS(p_{as} \cup \Delta \cup top(B \cup e_{ctx}))$ s.t. $A \lhd e_{pi}$.*
- *Any $\Delta^{+}_{fix} \subseteq Ab$ is a* positive fix *of an exception Δ to p if $\exists A \in AS(p_{as} \cup \Delta \cup \Delta^{+}_{fix} \cup top(B \cup e_{ctx}))$ s.t. $A \lhd e_{pi}$.*
- *Any $\Delta^{-}_{fix} \subseteq Ab \backslash p^{inc}_{pi}$ is a* negative fix *of a set of exceptions exs to p if $\forall \Delta \in exs$, $\Delta^{-}_{fix} \cap \Delta \neq \emptyset$.*

The sets of all minimal exceptions of p, minimal positive fixes of an exception Δ and minimal negative fixes of a set of exceptions exs are denoted by $m\_except(T, e, p)$, $m\_fix^{+}(T, e, p, \Delta)$ and $m\_fix^{-}(T, exs)$, respectively.

The idea of Algorithm 1 is to iteratively interleave the search for minimal exceptions and fixes to a set of CDPIs. In an arbitrary iteration, each CDPI p in pp is taken in turn. The first step is to compute the set of minimal exceptions to p. Note that if there are no minimal exceptions, p is a possibility and is directly added to the set *poss* (because $m\_fix^{-}(T, \emptyset) = \{\emptyset\}$). If there are minimal exceptions, they are resolved using the two methods described in Example 4: the two inner loops search for positive and negative fixes to the exceptions. The following theorem proves that Algorithm 1 is guaranteed to terminate and return a set of possibilities that includes the set of minimal possibilities for any WCDPI. This can then be filtered to remove non-minimal possibilities (in practice, filtering takes place during the execution of the algorithm to reduce unnecessary computation).

Theorem 3 *For any learning task T and any WCDPI e, $abduce\_possibilities(T, e)$ is guaranteed to terminate, and returns a set S such that $poss^{*}(T, e) \subseteq S \subseteq poss(T, e)$.*

5 Evaluation

This section presents an evaluation of FastNonOPL[5] on non-OPL tasks, demonstrating that it outperforms other ASP-based ILP systems. In particular, it is significantly faster than ILASP [Law *et al.*, 2015a]. The ILASP system consists of a collection of algorithms, each with various parameters. For both scenarios in this evaluation, we performed initial experiments to determine the best performing version of ILASP for the scenario and report results only for this version.[6] Specifically, for the agent scenario we used ILASP2i [Law *et al.*, 2016], and for the CAVIAR scenario we used ILASP4 [Law, 2020]. Full details of the flags used can be found in the supplementary material document.[7]

Agent Experiments

We investigate the problem of an agent learning how to navigate a grid (inspired by [Law *et al.*, 2014]). The agent starts with complete knowledge of the map, which is a `10x10` grid containing walls, locked cells (which can be unlocked by visiting the corresponding key) and link cells (which allow the agent to go directly to another cell) but no knowledge of the meaning of the various components of the grid, nor how they impact which actions the agent can take in future time points. At each time point, the agent is informed of which actions it can take by an oracle. The agent must learn a definition of `valid_action`, defining the actions that are valid at each time point, from examples of previous *episodes* involving the agent, which are labelled as either `valid` or `invalid`. An episode is labelled as `valid` iff every action executed by the agent in that episode is a `valid_action`. This can be captured by the following (background knowledge) rules.

```
valid :- not invalid.

invalid :- time(T), execute(A, T),
           not valid_action(A, T).
```

This task is branching because `invalid` episodes only imply that at least one action in the episode must not have been a `valid_action`. The correct definition of `valid_action` is:

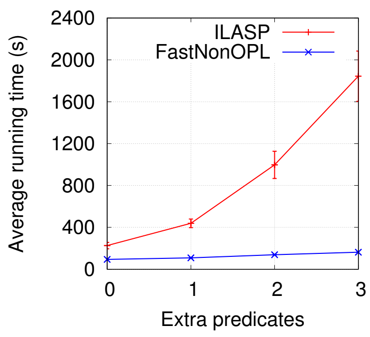
```
valid_action(C1, T) :- agent_at(C2, T),
  not wall(C1, C2), adjacent(C1, C2),
  unlocked(C1, T).

valid_action(C1, T) :- agent_at(C2, T),
  link(C2, C1), unlocked(C1, T).
```

ILASP and FastNonOPL were run on a set of 10 learning tasks, each consisting of 50 valid episodes and 50 invalid episodes (all learning tasks are available at https://github.com/spike-imperial/FastLAS/blob/master/FastLAS2/data).

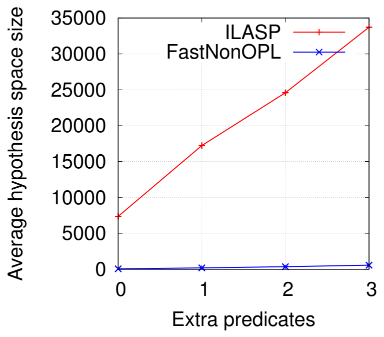
[5]FastNonOPL can be run by downloading the latest version of FastLAS (currently 2.0.0) from https://spike-imperial.github.io/FastLAS/ and running FastLAS with the `--nopl` flag.

[6]For details on the differences between the various ILASP systems, see [Law *et al.*, 2020b].

[7]All experiments for FastNonOPL, FastLAS and ILASP were run on an Ubuntu 18.04 desktop machine with a 3.4 GHz Intel® Core™ i7-6700 processor and with 16GB RAM. The results for OLED are quoted from [Katzouris *et al.*, 2016].



(a)



(b)

System	Average F_1	Average Time
ILASP	0.846	488.7s (8.3s)
OLED	0.792	107s
FastNonOPL	0.917	297.6s (4.3s)

(c)

System	Average F_1	Average Time
ILASP	0.827	749.0s (12.9s)
FastNonOPL	0.883	502.9s (12.2s)

(d)

Figure 2: (a) and (b) show the average running time and hypothesis space size for ILASP and FastNonOPL on the agent problem with varying numbers of extra predicates in the bias; (c) and (d) show the F_1 scores and running times for ILASP, OLED and FastNonOPL on the non-branching CAVIAR scenario and for ILASP and FastNonOPL on the branching scenario. OLED's time is quoted from [Katzouris *et al.*, 2016] and is not directly comparable with the other times.

The accuracy of each hypothesis was evaluated on a set of 1000 episodes.[8] Both ILASP2i and FastNonOPL are guaranteed to return an optimal solution of any learning task and therefore achieve the same average accuracy of 99.5%. To evaluate the relative scalability of the two systems we ran each learning task with different language biases. All biases contained the predicates which occur in the target solution and between 0 and 3 extra predicates which occur in the background knowledge but not in the target solution. Figure 2 shows the average hypothesis space size and running time for both systems. As ILASP enumerates the hypothesis space in full, the average hypothesis space size for ILASP is the full set of rules that ILASP considers (as ILASP bounds the number of literals in a rule, this is still smaller than the full hypothesis space). On the other hand FastNonOPL uses FastLAS to construct a smaller OPT-sufficient hypothesis space, which is well over two orders of magnitude smaller than the space constructed by ILASP. This is reflected by the shorter running times for FastNonOPL.

CAVIAR

We compared FastNonOPL to OLED [Katzouris *et al.*, 2016], FastLAS and ILASP on the CAVIAR dataset, which consists of data gathered from a video stream [Fisher *et al.*, 2004]. The dataset has been manually annotated, adding information such as the positions of people and when two people are interacting. We consider a task from [Katzouris *et al.*, 2016], in which the aim is to learn initiating and terminating conditions for two people meeting.

ILASP struggles with large hypothesis spaces [Law, 2018], because it enumerates the hypothesis space in full. A small subset of the hypothesis space (from [Law *et al.*, 2018b]) used by OLED was used in the ILASP experiments, restricting the number of literals in the body, employing several "common sense" constraints, such as a person cannot be walking and running at the same time. This restricted hypothesis space contains 3370 rules.

On the other hand, FastNonOPL does not need to generate the hypothesis space in full, and can therefore use a much larger hypothesis space. On real data, where the best hypothesis is not likely to be known beforehand, this has the advantage that larger hypothesis spaces are likely to contain better solutions than handcrafted smaller hypothesis spaces. In the FastNonOPL experiments, we used the bias from [Law *et al.*, 2020a], which represents a hypothesis space containing over 2^{42} non-isomorphic rules. The results[9] of performing 10-fold cross validation are shown in Figure 2. FastNonOPL is significantly faster than ILASP, and as FastNonOPL is able to use a larger hypothesis space, it returns better quality solutions with a higher average F_1 score. OLED is faster than both ILASP and FastNonOPL on this dataset; however, as it does not guarantee optimality, this is expected. OLED's average F_1 score is significantly lower than the other systems.

As this task is non-branching, it can be preprocessed and run in FastLAS (similarly to the FastNonOPL approach, but with domain-specific preprocessing, rather than FastNonOPL's general approach). This experiment was performed in [Law *et al.*, 2020a], and we repeated it for a comparison with FastNonOPL. FastLAS achieved a similar F_1 score of 0.920 (the small difference is caused by the two systems finding different optimal solutions in some folds) with a lower running time of 75.9s. The lower running time is because FastLAS does not need to run the possibility generation phase of FastNonOPL, because the tasks have already been preprocessed to remove the need for any non-OPL.

Branching Experiment

We performed a second experiment, in which the learner was given less information in each example. The examples were now labelled as either having no meetings or at least one meeting, rather than explicitly labelling each meeting that was occuring. This is a branching task, because in each example where a meeting is occuring, the learner can "choose" from all pairs of people in the scene. As this is a branching

[8]The accuracy is $\frac{tp+tn}{tp+tn+fp+fn}$, where *tp*, *tn*, *fp*, *fn* are the numbers of true positives, true negatives, false positives and false negatives.

[9]The F_1 score is the harmonic mean of the precision $(tp/(tp+fp))$ and the recall $(tp/(tp+fn))$. We use F_1 scores, micro averaged across the folds, for comparison with the OLED result.

task, only ILASP and FastNonOPL are capable of solving it.[10] FastNonOPL again outperforms ILASP, both in terms of the quality of the learned solution and the running time, showing that even for branching tasks, FastNonOPL maintains FastLAS's increased scalability (w.r.t. the size of the hypothesis space) over ILASP. The F_1 scores of both systems are slightly lower than in the simpler non-branching experiment, which is unsurprising because the learner is given less information in each example. However, just as before, FastNonOPL's larger hypothesis space compared to ILASP means that FastNonOPL is able to find a better quality solution than ILASP.

6 Related Work

FastNonOPL uses the FastLAS [Law *et al.*, 2020a] system at its core. FastNonOPL is much more general than FastLAS, as it supports non-observational predicate learning and programs with multiple answer sets, meaning that although FastNonOPL can solve any task that FastLAS can solve, the converse does not hold. When FastNonOPL is executed on a task that FastLAS can solve, all examples are guaranteed to have exactly one possibility.

The generation of an OPT-sufficient subset of the hypothesis space in FastLAS (and therefore FastNonOPL), is related to early *bottom clause* ILP approaches used by Progol [Muggleton, 1995], Aleph [Srinivasan, 2001] and later generalised by HAIL [Ray *et al.*, 2003]. A key difference is that the early systems used iterative approaches to construct a hypothesis. A single positive example (corresponding to a single inclusion of an example in this paper) is considered in each iteration. The systems compute the best rule (or rules in the case of HAIL) that covers the example, and add it to the hypothesis. This means that none of these systems guarantee finding an optimal solution, as although each iteration might find an optimal rule to add to the hypothesis, the final hypothesis may still be sub-optimal. Theorem 2 shows that FastNonOPL, like FastLAS, searches an OPT-sufficient subset of the hypothesis space, and is therefore guaranteed to return an optimal solution.

XHAIL [Ray, 2009] also uses abduction to find the heads of rules in the hypothesis. However, the abductive task only requires finding a set of atoms Δ s.t. $B \cup \Delta$ entails every example. If this set Δ does not lead to an inductive solution, it finds another Δ using an iterative deepening strategy on the size of Δ. This method will return an inductive solution if one exists; however, it may be sub-optimal, especially on learning tasks with noisy examples, which can adversely affect the performance of the learned program on unseen data. Computing all abductive solutions would address this problem, but is computationally infeasible. By computing partial interpretations that capture classes of abductive solutions, the set of minimal possibilities captures the full set of abductive solutions without needing to compute them all.

[10]OLED must be able to count the number of true positives, false positives and false negatives (over the training data) caused by a rule. In branching tasks a rule does not have a fixed count; for example, terminating two people meeting might clearly cause a false negative in the original CAVIAR task, but in the branching task, false negatives can only occur if there are no people meeting.

The OLED (Online Learning of Event Definitions) [Katzouris *et al.*, 2016] system is an ILP system that is specifically targeted at learning Event Calculus axioms from large amounts of sequential data. Similarly to XHAIL, OLED may return sub-optimal inductive solutions. Although OLED is capable of some non-OPL, it is not able to solve branching tasks. Our evaluation showed that on non-branching tasks, FastNonOPL outperformed OLED in terms of average F_1 score, but OLED was faster.

The ILASP [Law *et al.*, 2015a] systems also learn ASP programs. Our evaluation shows that FastNonOPL is significantly faster than ILASP when applied on the same learning tasks. This increase in speed is because ILASP starts by constructing the hypothesis space in full, whereas FastNonOPL constructs a small OPT-sufficient subset of the hypothesis space. On the other hand, ILASP is much more general as it can (resources permitting) learn any ASP program [Law *et al.*, 2018a], including programs with choice rules and weak constraints [Law *et al.*, 2015b], and supports recursion and predicate invention [Law, 2018].

7 Conclusion and Future Work

Non-OPL is an important feature of modern ILP systems, allowing them to be applied to many more datasets. Although the FastLAS [Law *et al.*, 2020a] system is highly scalable, its inability to perform non-OPL is therefore a severe limitation. FastNonOPL's new *possibility generation* enables non-OPL, meaning that FastNonOPL increases on the applicability of FastLAS, while still maintaining its scalability w.r.t. the hypothesis space (as shown in the evaluation). The next step will be to extend FastNonOPL even further, to support learning recursive rules and predicate invention.

Acknowledgements

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Corapi *et al.*, 2010] Domenico Corapi, Alessandra Russo, and Emil Lupu. Inductive logic programming as abductive search. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

[Corapi *et al.*, 2012] Domenico Corapi, Alessandra Russo, and Emil Lupu. Inductive logic programming in answer set programming. In *Inductive Logic Programming*, pages 91–97. Springer, 2012.

[Fisher *et al.*, 2004] Robert Fisher, Jose Santos-Victor, and James Crowley. CAVIAR: Context aware vision using image-based active recognition. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/, 2004. Accessed: 2019-08-28.

[Gebser *et al.*, 2016] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with clingo 5. In *Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

[Kakas *et al.*, 1992] Antonis C Kakas, Robert A. Kowalski, and Francesca Toni. Abductive logic programming. *Journal of logic and computation*, 2(6):719–770, 1992.

[Katzouris *et al.*, 2016] Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Online learning of event definitions. *Theory and Practice of Logic Programming*, 16(5-6):817–833, 2016.

[Law *et al.*, 2014] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs. In Eduardo Fermé and João Leite, editors, *Proceedings of the Fourteenth European Conference on Logics in Artificial Intelligence, 2014, Funchal, Madeira, Portugal, September 24-26, 2014.*, volume 8761 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2014.

[Law *et al.*, 2015a] Mark Law, Alessandra Russo, and Krysia Broda. The ILASP system for learning answer set programs. www.ilasp.com, 2015.

[Law *et al.*, 2015b] Mark Law, Alessandra Russo, and Krysia Broda. Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming*, 15(4-5):511–525, 2015.

[Law *et al.*, 2016] Mark Law, Alessandra Russo, and Krysia Broda. Iterative learning of answer set programs from context dependent examples. *Theory and Practice of Logic Programming*, 16(5-6):834–848, 2016.

[Law *et al.*, 2018a] Mark Law, Alessandra Russo, and Krysia Broda. The complexity and generality of learning answer set programs. *Artificial Intelligence*, 259:110–146, 2018.

[Law *et al.*, 2018b] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs from noisy examples. *Advances in Cognitive Systems*, 2018.

[Law *et al.*, 2020a] Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. FastLAS: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *AAAI*. Association for the Advancement of Artificial Intelligence, 2020.

[Law *et al.*, 2020b] Mark Law, Alessandra Russo, and Krysia Broda. The ILASP system for inductive learning of answer set programs. *The Association for Logic Programming Newsletter*, 2020.

[Law, 2018] Mark Law. *Inductive Learning of Answer Set Programs*. PhD thesis, Imperial College London, 2018.

[Law, 2020] Mark Law. Conflict-driven inductive logic programming. *arXiv preprint arXiv:2101.00058*, 2020.

[Muggleton, 1991] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Muggleton, 1995] Stephen Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3-4):245–286, 1995.

[Ray *et al.*, 2003] Oliver Ray, Krysia Broda, and Alessandra Russo. Hybrid abductive inductive learning: A generalisation of progol. In *Inductive Logic Programming*, pages 311–328. Springer, 2003.

[Ray, 2009] Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3):329–340, 2009.

[Srinivasan, 2001] Ashwin Srinivasan. The aleph manual. *Machine Learning at the Computing Laboratory, Oxford University*, 2001.