Online appendix for the paper

Iterative Learning of Answer Set Programs from Context Dependent Examples

published in Theory and Practice of Logic Programming

Mark Law, Alessandra Russo∗ , Krysia Broda
Department of Computing, Imperial College London, SW7 2AZ
(*e-mail:* {mark.law09, a.russo, k.broda}@imperial.ac.uk)

submitted 22 April 2016; revised 8 July 2016; accepted 22 July 2016

Appendix A Proofs

In this section, we give the proofs of the theorems in the main paper. First, we prove the preliminary lemma (Lemma 1). Really, this is a corollary of the splitting set theorem (Lifschitz and Turner 1994). $e_U(P, X)$ is the partial evaluation of P with respect to X (over the atoms in U), which is described in (Lifschitz and Turner 1994).

Lemma 1
For any program P (consisting of normal rules, choice rules and constraints) and any set of pairs $S = \{\langle C_1, \mathtt{a_1}\rangle, \ldots, \langle C_n, \mathtt{a_n}\rangle\}$ such that none of the atoms $\mathtt{a_i}$ appear in P (or in any of the C's) and each $\mathtt{a_i}$ atom is unique: $AS(P \cup \{\mathtt{1\{a_1}, \ldots, \mathtt{a_n\}1.}\} \cup \{append(C_i, \mathtt{a_i})|\langle C_i, \mathtt{a_i}\rangle \in S\}) = \{A \cup \{\mathtt{a_i}\}|A \in AS(P \cup C_i), \langle C_i, \mathtt{a_i}\rangle \in S\}$

Proof
The answer sets of $\{\mathtt{1\{a_1}, \ldots, \mathtt{a_n\}1.}\}$ are $\{\mathtt{a_1}\}, \ldots, \{\mathtt{a_n}\}$, hence by the splitting set theorem (using $U = \{\mathtt{a_1}, \ldots, \mathtt{a_n}\}$ as a splitting set):

$$\begin{aligned} &AS(P \cup \{\mathtt{1\{a_1}, \ldots, \mathtt{a_n\}1.}\} \cup \{append(C_i, \mathtt{a_i})|\langle C_i, \mathtt{a_i}\rangle \in S\}) \\ &\quad = \left\{ A' \cup \{\mathtt{a_j}\} \;\middle|\; \begin{array}{c} \mathtt{a_j} \in \{\mathtt{a_1}, \ldots, \mathtt{a_n}\} \\ A' \in AS(e_U(P \cup \{append(C_i, \mathtt{a_i}) \mid \langle C_i, \mathtt{a_i}\rangle \in A\}, \{\mathtt{a_j}\})) \end{array} \right\} \\ &\quad = \{A \cup \{\mathtt{a_i}\}|A \in AS(P \cup C_i), \langle C_i, \mathtt{a_i}\rangle \in S\}. \quad \square \end{aligned}$$

Theorem 2
The complexity of deciding whether an $ILP^{context}_{LOAS}$ task is satisfiable is Σ^P_2-complete.

∗ This research is partially funded by the EPSRC project EP/K033522/1 "Privacy Dynamics".

Proof
Deciding satisfiability for ILP_{LOAS} is Σ_2^P-complete ((Law et al. 2015a)). It is therefore sufficient to show that there is a polynomial mapping from ILP_{LOAS} to $ILP^{context}_{LOAS}$ and a polynomial mapping from $ILP^{context}_{LOAS}$ to ILP_{LOAS}. The former is trivial (any ILP_{LOAS} task can be mapped to the same task in $ILP^{context}_{LOAS}$ with empty contexts). The latter follows from theorem 1. □

Theorem 3
ILASP2i terminates for any well defined $ILP^{context}_{LOAS}$ task.

Proof
Assume that the task $T = \langle B, S_M, E\rangle$ is well defined. This means that $T_1 = \mathcal{T}_{LOAS}(T)$ is a well defined ILP_{LOAS} task (every possible hypothesis has a finite grounding when combined with the background knowledge of T_1). Note that this also means that $T_2 = \mathcal{T}_{LOAS}(\langle B, S_M, \mathit{Relevant}\rangle)$ is well defined in each iteration as the size of the grounding of the background knowledge of T_2 combined with each hypothesis will be smaller than or equal to the size of the background in T_1 (the background knowledge of T_2 is almost a subset of the background in T_1, other than the extra choice rule, which is smaller).

The soundness of ILASP2 (Law et al. 2015a) can be used to show that H will always cover every example in *Relevant*; hence, at each step *re* must be an example which is in E but not in *Relevant*. As there are a finite number of examples in E, this means there can only be a finite number of iterations; hence, it remains to show that each iteration terminates. This is the case because, as $\mathcal{T}_{LOAS}(\langle B, S_M, \mathit{Relevant}\rangle)$ is well defined, the call to ILASP2 terminates ((Law et al. 2015a)) and *findRelevantExample* terminates (Appendix B). □

Theorem 4
ILASP2i is sound for any well defined $ILP^{context}_{LOAS}$ task, and returns an optimal solution if one exists.

Proof
If the ILASP2i algorithm returns a hypothesis then the while loop must terminate. For this to happen *findRelevantExample* must return `nil`. This means that H must cover every example in E. Hence ILASP2i is sound. As the algorithm terminates (see Theorem 3), the only way for a solution not to be returned is when $ILASP2$ returns `nil`. Since $ILASP2$ is complete (Law et al. 2015a), this is only possible when $\langle B, S_M, \mathit{Relevant}\rangle$ is unsatisfiable. But if $\langle B, S_M, \mathit{Relevant}\rangle$ is unsatisfiable then so is $\langle B, S_M, E\rangle$.

It remains to show that when a solution is returned, it is an optimal solution. Any solution H returned must be an optimal solution of $\langle B, S_M, \mathit{Relevant}\rangle$, (as ILASP2 returns an optimal solution). As it must also be a solution of $\langle B, S_M, E\rangle$, it must be an optimal solution (any shorter solution would be a solution of $\langle B, S_M, \mathit{Relevant}\rangle$, contradicting that H is an optimal solution for $\langle B, S_M, \mathit{Relevant}\rangle$). □

Appendix B *findRelevantExamples*

In this section, we describe (and prove the correctness of) the *findRelevantExamples* method which was omitted from the main paper. The method uses a meta encoding in ASP. Given a learning task and a hypothesis from the hypothesis space, this meta encoding is used to compute the set of examples that are covered and the set that are not covered. The meta encoding is formalised in definition 4, but we first introduce some notation in order to simplify the main definition. Some definitions are similar to those used in the ILASP2 meta representation (Law et al. 2015a).

Definition 1
For any ASP program P and predicate name `pred`, *reify*$(P, \texttt{pred})$ denotes the program constructed by replacing every atom $\texttt{a} \in P'$ (where P' is P with the weak constraints removed) by $\texttt{pred(a)}$. We use the same notation for sets of literals/partial interpretations, so for a set S: *reify*$(S, pred) = \{\texttt{pred(atom)} : \texttt{atom} \in S\}$.

Definition 2 formalises the way we represent weak constraints in our meta encoding. We use this representation to check whether ordering examples are covered. We use `as1` and `as2` to represent the atoms in two answer sets (`as1` and `as2` occur elsewhere in our encoding). The `w` atoms are then used to capture the penalties paid by each answer set at each level.

Definition 2
For any ASP program P, we write *weak*(P) to mean the program constructed from the weak constraints in P, translating each weak constraint $:\sim \texttt{b}_1, \ldots, \texttt{b}_\texttt{m}$, $\texttt{not b}_{\texttt{m}+1}, \ldots, \texttt{not b}_\texttt{n}.[\texttt{lev@wt}, \texttt{t}_1, \ldots, \texttt{t}_\texttt{k}]$ to the rules:

$$\left\{\begin{array}{l} \texttt{w(wt, lev, terms(t}_1, \ldots, \texttt{t}_\texttt{k}\texttt{), as1):- as1(b}_1\texttt{)}, \ldots, \texttt{as1(b}_\texttt{m}\texttt{)}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \texttt{not as1(b}_{\texttt{m}+1}\texttt{)}, \ldots, \texttt{not as1(b}_\texttt{n}\texttt{).} \\ \texttt{w(wt, lev, terms(t}_1, \ldots, \texttt{t}_\texttt{k}\texttt{), as2):- as2(b}_1\texttt{)}, \ldots, \texttt{as2(b}_\texttt{m}\texttt{)}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \texttt{not as2(b}_{\texttt{m}+1}\texttt{)}, \ldots, \texttt{not as2(b}_\texttt{n}\texttt{).} \end{array}\right\}$$

We now introduce a simplified version of the ASP program fragment which is used by ILASP2 to check whether one answer set dominates another. This is used in determining whether an ordering example is covered by a hypothesis. This makes use of the `w` atoms which are generated by the `w` rules in definition 2, and captures the definition of dominates given in Section 2.

Definition 3
dominates is the program:

$$\left\{\begin{array}{l} \texttt{dom\_lv(L):- lv(L), \#sum\{w(W, L, A, as1) = W, w(W, L, A, as2) = -W\} < 0.} \\ \texttt{non\_dom\_lv(L):- lv(L), \#sum\{w(W, L, A, as2) = W, w(W, L, A, as1) = -W\} < 0.} \\ \texttt{non\_bef(L):- lv(L), lv(L2), L < L2, non\_dom\_lv(L2).} \\ \texttt{dominated:- dom\_lv(L), not non\_bef(L).} \end{array}\right\}$$

In (Law et al. 2015a), multiple instances of *dominates* were included in the same meta encoding, and hence the program was slightly more complicated in order to track the different instances. The main structure of the program is the same however, and hence the same results apply. The result we need for this paper is proven (for the more general program) in (Law et al. 2015b) and is given by Lemma 2.

Lemma 2
Let I_1 and I_2 be interpretations, P be an ASP program and L be the set of levels used in the weak constraints in P. The unique answer set of $\mathit{dominates} \cup \{\texttt{lv(l).} \mid l \in L\} \cup \mathit{weak}(P) \cup \mathit{reify}(I_1, \texttt{as1}) \cup \mathit{reify}(I_2, \texttt{as2})$ contains the atom `dominated` if and only if I_1 dominates I_2 wrt the weak constraints in P.

Definition 4 captures the meta encoding we use in *findRelevantExamples*. This encoding is made of 6 components. $\mathcal{R}_1$ captures the background knowledge and hypothesis – by reifying $B \cup H$, the `as1` and `as2` atoms represent two answer sets A_1 and A_2, and the *dominates* program (together with $\mathit{weak}(B \cup H)$ and the priority levels) checks whether A_1 dominates A_2. The programs $\mathcal{R}_2$ to $\mathcal{R}_5$ are used to check whether each type of example is covered. These programs make use of the predicate `test_on` of arity 2 and the `test` predicate of arity 1. The meaning of $\texttt{test}(\texttt{ex}_\texttt{id})$ is that the example ex should be tested. There is a choice rule in $\mathcal{R}_6$ to say that each example should be tested. For the positive and negative examples, this means that they should be tested on `as1` (meaning to check whether it is possible that an answer set of $B \cup H$ extends this example). For an ordering example $\langle ex_1, ex_2 \rangle$ it is slightly more involved: ex_1 should be tested on `as1` and ex_2 should be tested on `as2` (and the ordering should be checked).

Definition 4
Let T be the $ILP^{context}_{LOAS}$ task $\langle B, S_M, \langle E^+, E^-, O^b, O^c \rangle\rangle$ and H be a hypothesis such that $H \subseteq S_M$. Let L be the set of all priority levels in $B \cup H$ $\mathcal{R}(T, H)$ is the ASP program $\mathcal{R}_1(B \cup H) \cup \mathcal{R}_2(E^+) \cup \mathcal{R}_3(E^-) \cup \mathcal{R}_4(O^b) \cup \mathcal{R}_5(O^c) \cup \mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$, where the individual components are as follows:

- $\mathcal{R}_1(B \cup H) = \mathit{reify}(B \cup H, \texttt{as1}) \cup \mathit{reify}(B \cup H, \texttt{as2}) \cup \mathit{weak}(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup \mathit{dominates}$
- $\mathcal{R}_2(E^+) = \left\{ \begin{array}{l} \texttt{cov(as1):- test\_on(ex}_\texttt{id}\texttt{, as1),} \\ \quad \texttt{as1(e}_1^{\texttt{inc}}\texttt{)}, \ldots, \texttt{as1(e}_\texttt{m}^{\texttt{inc}}\texttt{),} \\ \quad \texttt{not as1(e}_1^{\texttt{exc}}\texttt{)}, \ldots, \texttt{not as1(e}_\texttt{n}^{\texttt{exc}}\texttt{)} \\ \texttt{cov(as2):- test\_on(ex}_\texttt{id}\texttt{, as2),} \\ \quad \texttt{as2(e}_1^{\texttt{inc}}\texttt{)}, \ldots, \texttt{as2(e}_\texttt{m}^{\texttt{inc}}\texttt{),} \\ \quad \texttt{not as2(e}_1^{\texttt{exc}}\texttt{)}, \ldots, \texttt{not as2(e}_\texttt{n}^{\texttt{exc}}\texttt{)} \\ \texttt{:- not cov(as1), test\_on(ex}_\texttt{id}\texttt{, as1).} \\ \texttt{:- not cov(as2), test\_on(ex}_\texttt{id}\texttt{, as2).} \\ \mathit{append}(\mathit{reify}(C, \texttt{as1}), \texttt{test\_on(ex}_\texttt{id}\texttt{, as1})) \\ \mathit{append}(\mathit{reify}(C, \texttt{as2}), \texttt{test\_on(ex}_\texttt{id}\texttt{, as2})) \end{array} \middle| \begin{array}{c} ex \in E^+, \\ ex = \langle e, C \rangle, \\ e = \langle \{\texttt{e}_1^\texttt{i}, \ldots, \texttt{e}_\texttt{m}^\texttt{i}\}, \{\texttt{e}_1^\texttt{e}, \ldots, \texttt{e}_\texttt{n}^\texttt{e}\} \rangle \end{array} \right\}$
- $\mathcal{R}_3(E^-) = \left\{ \begin{array}{l} \texttt{violated:- test\_on(ex}_\texttt{id}\texttt{, as1),} \\ \quad \texttt{as1(e}_1^{\texttt{inc}}\texttt{)}, \ldots, \texttt{as1(e}_\texttt{m}^{\texttt{inc}}\texttt{),} \\ \quad \texttt{not as1(e}_1^{\texttt{exc}}\texttt{)}, \ldots, \texttt{not as1(e}_\texttt{n}^{\texttt{exc}}\texttt{).} \\ \mathit{append}(\mathit{reify}(C, \texttt{as1}), \texttt{test\_on(ex}_\texttt{id}\texttt{, as1})) \\ \texttt{:- not violated, test\_on(ex}_\texttt{id}\texttt{, as1).} \end{array} \middle| \begin{array}{c} ex \in E^-, \\ ex = \langle e, C \rangle, \\ e = \langle \{\texttt{e}_1^\texttt{i}, \ldots, \texttt{e}_\texttt{m}^\texttt{i}\}, \{\texttt{e}_1^\texttt{e}, \ldots, \texttt{e}_\texttt{n}^\texttt{e}\} \rangle \end{array} \right\}$
- $\mathcal{R}_4(O^b) = \{ \texttt{:- test(o}_\texttt{id}\texttt{), not dominated.} \mid o \in O^b \}$
- $\mathcal{R}_5(O^c) = \{ \texttt{:- test(o}_\texttt{id}\texttt{), dominated.} \mid o \in O^c \}$
- $\mathcal{R}_6(\{ex_1, \ldots ex_m\}, \{o_1, \ldots o_n\}) = \{ \texttt{1\{test(ex}_1\texttt{)}, \ldots, \texttt{test(ex}_\texttt{m}\texttt{), test(o}_1\texttt{)}, \ldots, \texttt{test(o}_\texttt{n}\texttt{)\}1.} \}$

$$\cup\{\ \texttt{test\_on(ex}_\texttt{i}\texttt{,as1):- test(ex}_\texttt{i}\texttt{).}\ \mid\ ex_i \in \{ex_1, \ldots, ex_m\}\ \}$$
$$\cup\left\{\begin{array}{l}\texttt{test\_on(ex}_1\texttt{,as1):- test(o}_\texttt{i}\texttt{).}\\ \texttt{test\_on(ex}_2\texttt{,as2):- test(o}_\texttt{i}\texttt{).}\end{array}\ \middle|\ \begin{array}{c}o_i \in \{o_1, \ldots, o_n\}\\ o_i = \langle ex_1, ex_2\rangle\end{array}\right\}$$

Theorem 5
Let T be any $ILP^{context}_{LOAS}$ task and H be any subset of the hypothesis space.

1. $\forall ex \in E^+$, $\exists A \in AS(\mathcal{R}(T,H))$ st $\texttt{test(ex}_\texttt{id}\texttt{)} \in A$ iff H covers ex.
2. $\forall ex \in E^-$, $\exists A \in AS(\mathcal{R}(T,H))$ st $\texttt{test(ex}_\texttt{id}\texttt{)} \in A$ iff H does not cover ex.
3. $\forall o \in O^b$, $\exists A \in AS(\mathcal{R}(T,H))$ st $\texttt{test(o}_\texttt{id}\texttt{)} \in A$ iff H bravely respects o.
4. $\forall o \in O^c$, $\exists A \in AS(\mathcal{R}(T,H))$ st $\texttt{test(o}_\texttt{id}\texttt{)} \in A$ iff H does not cautiously respect o.

Proof

1. Let $ex = \langle e, C\rangle$ be a CDPI in E^+ st $e = \langle\{\texttt{e}^\texttt{i}_1, \ldots \texttt{e}^\texttt{i}_\texttt{m}\}, \{\texttt{e}^\texttt{e}_1, \ldots, \texttt{e}^\texttt{e}_\texttt{n}\}\rangle$.
   H covers $ex \Leftrightarrow \exists A \in AS(B \cup H \cup C)$ st A extends e
   $\Leftrightarrow \exists A \in AS(\mathit{reify}(B \cup H \cup C, \texttt{as1}))$ st A extends $\mathit{reify}(e, \texttt{as1})$
   $\Leftrightarrow \mathit{reify}(B \cup H \cup C, \texttt{as1}) \cup \left\{\begin{array}{l}\texttt{cov(as1):- as1(e}_1\texttt{),}\ldots\texttt{,as1(e}_\texttt{m}\texttt{),}\\ \qquad\qquad\texttt{not as1(e}_1\texttt{),}\ldots\texttt{,not as1(e}_\texttt{n}\texttt{).}\\ \texttt{:- not cov(as1).}\end{array}\right\}$
   is satisfiable (we refer to this program as P_1 later in the proof).
   $\Leftrightarrow \mathit{reify}(B \cup H, \texttt{as1}) \cup \mathit{append}(\mathit{reify}(C, \texttt{as1}), \texttt{test\_on(ex}_\texttt{id}\texttt{,as1)})$
   $$\cup\left\{\begin{array}{l}\texttt{cov(as1):- test\_on(ex}_\texttt{id}\texttt{,as1),}\\ \quad\texttt{as1(e}_1\texttt{),}\ldots\texttt{,as1(e}_\texttt{m}\texttt{),}\\ \quad\texttt{not as1(e}_1\texttt{),}\ldots\texttt{,not as1(e}_\texttt{n}\texttt{).}\\ \texttt{:- not cov(as1),test\_on(ex}_\texttt{id}\texttt{,as1).}\end{array}\right\}\cup\mathcal{R}_6(E^+\cup E^-, O^b\cup O^c)$$
   has an answer set which contains $\texttt{test(ex}_\texttt{id}\texttt{)}$ (we refer to this program as P_2). This follows from the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set – $\{\texttt{test(ex}_\texttt{id}\texttt{),test\_on(ex}_\texttt{id}\texttt{,as1)}\}$ is an answer set of the bottom program, leading to P_1 as the partially evaluated top program
   $\Leftrightarrow \mathcal{R}(T,H)$ has an answer set which contains $\texttt{test(ex}_\texttt{id}\texttt{)}$. Again, this is by the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, as $P_2 \subseteq \mathcal{R}(T,H)$ and each of the extra rules in $\mathcal{R}(T,H)$ which are not in P_2 contain a `test_on` or `test` atom in the body that is not in the answer set $\{\texttt{test(ex}_\texttt{id}\texttt{),test\_on(ex}_\texttt{id}\texttt{,as1)}\}$ and hence they are removed from the partially evaluated top program.
2. Let $ex = \langle e, C\rangle$ be a CDPI in E^- st $e = \langle\{\texttt{e}^\texttt{i}_1, \ldots \texttt{e}^\texttt{i}_\texttt{m}\}, \{\texttt{e}^\texttt{e}_1, \ldots, \texttt{e}^\texttt{e}_\texttt{n}\}\rangle$.
   H does not cover $ex \Leftrightarrow \exists A \in AS(B \cup H \cup C)$ st A extends e
   $\Leftrightarrow \exists A \in AS(\mathit{reify}(B \cup H \cup C, \texttt{as1}))$ st A extends $\mathit{reify}(e, \texttt{as1})$
   $\Leftrightarrow \mathit{reify}(B \cup H \cup C, \texttt{as1}) \cup \left\{\begin{array}{l}\texttt{violated:- as1(e}_1\texttt{),}\ldots\texttt{,as1(e}_\texttt{m}\texttt{),}\\ \quad\texttt{not as1(e}_1\texttt{),}\ldots\texttt{,not as1(e}_\texttt{n}\texttt{).}\\ \texttt{:- not violated.}\end{array}\right\}$ is satisfiable (we refer to this program as P_3 later in the proof)

$\Leftrightarrow$ *reify*$(B \cup H, \texttt{as1}) \cup$ *append*(*reify*$(C, \texttt{as1}), \texttt{test\_on}(\texttt{ex}_{\texttt{id}}, \texttt{as1}))$

$$\cup \left\{ \begin{array}{l} \texttt{violated:- test\_on}(\texttt{ex}_{\texttt{id}}, \texttt{as1}), \\ \quad \texttt{as1}(\texttt{e}_1), \ldots, \texttt{as1}(\texttt{e}_{\texttt{m}}), \\ \quad \texttt{not as1}(\texttt{e}_1), \ldots, \texttt{not as1}(\texttt{e}_{\texttt{n}}). \\ \texttt{:- not violated}, \texttt{test\_on}(\texttt{ex}_{\texttt{id}}, \texttt{as1}). \end{array} \right\} \cup \mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$$

has an answer set which contains $\texttt{test}(\texttt{ex}_{\texttt{id}})$ (we refer to this program as P_4). This follows by the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, $\{\texttt{test}(\texttt{ex}_{\texttt{id}}), \texttt{test\_on}(\texttt{ex}_{\texttt{id}}, \texttt{as1})\}$ is an answer set of the bottom program, leading to P_3 as the partially evaluated top program.

$\Leftrightarrow \mathcal{R}(T, H)$ has an answer set which contains $\texttt{test}(\texttt{ex}_{\texttt{id}})$. Again, this is by the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, as $P_4 \subseteq \mathcal{R}(T, H)$ and each of the extra rules in $\mathcal{R}(T, H)$ which are not in P_4 contain a `test_on` or `test` atom in the body that is not in the answer set $\{\texttt{test}(\texttt{ex}_{\texttt{id}}), \texttt{test\_on}(\texttt{ex}_{\texttt{id}}, \texttt{as1})\}$ and hence they are removed from the partially evaluated top program.

3. Let $o = \langle ex1, ex2 \rangle$ be a CDOE in O^b st $ex1 = \langle e1, C_1 \rangle$, $ex2 = \langle e2, C_2 \rangle$, $e_1 = \langle \{\texttt{e1}^{\texttt{i}}_1, \ldots, \texttt{e1}^{\texttt{i}}_{\texttt{m}}\}, \{\texttt{e1}^{\texttt{e}}_1, \ldots, \texttt{e1}^{\texttt{e}}_{\texttt{n}}\} \rangle$ and $e_2 = \langle \{\texttt{e2}^{\texttt{i}}_1, \ldots, \texttt{e2}^{\texttt{i}}_{\texttt{j}}\}, \{\texttt{e2}^{\texttt{e}}_1, \ldots, \texttt{e2}^{\texttt{e}}_{\texttt{k}}\} \rangle$.
H bravely respects $o \Leftrightarrow \exists A_1 \in AS(B \cup H \cup C_1), \exists A_2 \in AS(B \cup H \cup C_2)$ st A_1 extends e_1, A_2 extends e_2 and $A_1 \prec_{B \cup H} A_2$
$\Leftrightarrow \exists A_1 \in AS($*reify*$(B \cup H \cup C_1, \texttt{as1})), \exists A_2 \in AS($*reify*$(B \cup H \cup C_2, \texttt{as2}))$ st A_1 extends *reify*$(e_1, \texttt{as1})$, A_2 extends *reify*$(e_2, \texttt{as2})$ and `dominated` is in the unique answer set of $A_1 \cup A_2 \cup$ *weak*$(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup$ *dominates* (by Lemma 2)
$\Leftrightarrow$ *reify*$(B \cup H \cup C_1, \texttt{as1}) \cup$ *reify*$(B \cup H \cup C_2, \texttt{as2}) \cup$ *weak*$(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup$ *dominates*)

$$\cup \left\{ \begin{array}{l} \texttt{cov(as1):- as1}(\texttt{e1}^{\texttt{i}}_1), \ldots, \texttt{as1}(\texttt{e1}^{\texttt{i}}_{\texttt{m}}), \\ \quad \texttt{not as1}(\texttt{e1}^{\texttt{e}}_1), \ldots, \texttt{not as1}(\texttt{e1}^{\texttt{e}}_{\texttt{n}}). \\ \texttt{:- not cov(as1).} \\ \texttt{cov(as2):- as2}(\texttt{e2}^{\texttt{i}}_1), \ldots, \texttt{as2}(\texttt{e2}^{\texttt{i}}_{\texttt{j}}), \\ \quad \texttt{not as2}(\texttt{e2}^{\texttt{e}}_1), \ldots, \texttt{not as2}(\texttt{e2}^{\texttt{e}}_{\texttt{k}}). \\ \texttt{:- not cov(as2).} \\ \texttt{:- not dominated.} \end{array} \right\} \text{ is satisfiable (we}$$

refer to this program as P_5 later in the proof)
$\Leftrightarrow$ *reify*$(B \cup H \cup C_1, \texttt{as1}) \cup$ *reify*$(B \cup H \cup C_2, \texttt{as2}) \cup$ *weak*$(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup$ *dominates*)

$$\cup \left\{ \begin{array}{l} \texttt{cov(as1):- test\_on}(\texttt{ex1}_{\texttt{id}}, \texttt{as2}), \texttt{as1}(\texttt{e1}^{\texttt{i}}_1), \ldots, \texttt{as1}(\texttt{e1}^{\texttt{i}}_{\texttt{m}}), \\ \quad \texttt{not as1}(\texttt{e1}^{\texttt{e}}_1), \ldots, \texttt{not as1}(\texttt{e1}^{\texttt{e}}_{\texttt{n}}). \\ \texttt{:- not test\_on}(\texttt{ex1}_{\texttt{id}}, \texttt{as1}), \texttt{cov(as1).} \\ \texttt{cov(as2):- test\_on}(\texttt{ex2}_{\texttt{id}}, \texttt{as2}), \texttt{as2}(\texttt{e2}^{\texttt{i}}_1), \ldots, \texttt{as2}(\texttt{e2}^{\texttt{i}}_{\texttt{j}}), \\ \quad \texttt{not as2}(\texttt{e2}^{\texttt{e}}_1), \ldots, \texttt{not as2}(\texttt{e2}^{\texttt{e}}_{\texttt{k}}). \\ \texttt{:- test\_on}(\texttt{ex2}_{\texttt{id}}, \texttt{as2}), \texttt{not cov(as2).} \\ \texttt{:- test}(\texttt{o}_{\texttt{id}}), \texttt{not dominated.} \end{array} \right\}$$

$$\cup \mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$$

has an answer set which contains $\texttt{test}(\texttt{o}_{\texttt{id}})$ (we refer to this program as P_6). This follows by the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, $\{\texttt{test}(\texttt{o}_{\texttt{id}}), \texttt{test\_on}(\texttt{ex1}_{\texttt{id}}, \texttt{as1}), \texttt{test\_on}(\texttt{ex2}_{\texttt{id}}, \texttt{as2})\}$ is an answer set of the bottom program, leading to P_5 as the partially evaluated top program
$\Leftrightarrow \mathcal{R}(T, H)$ has an answer set which contains $\texttt{test}(\texttt{o}_{\texttt{id}})$. Again, this is by the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, as $P_6 \subseteq \mathcal{R}_6(T, H)$ and each of the extra rules which are in $\mathcal{R}_6(T, H)$ but not in P_6 contain a $\texttt{test\_on}$ or $\texttt{test}$ atom which is not in the answer set $\{\texttt{test}(\texttt{o}_{\texttt{id}}), \texttt{test\_on}(\texttt{ex1}_{\texttt{id}}, \texttt{as1}), \texttt{test\_on}(\texttt{ex2}_{\texttt{id}}, \texttt{as2})\}$ and hence they are removed from the partially evaluated top program

4. Let $o = \langle ex1, ex2 \rangle$ be a CDOE in O^c st $ex1 = \langle e1, C_1 \rangle$, $ex2 = \langle e2, C_2 \rangle$, $e_1 = \langle \{\texttt{e1}^{\texttt{i}}_1, \ldots, \texttt{e1}^{\texttt{i}}_{\texttt{m}}\}, \{\texttt{e1}^{\texttt{e}}_1, \ldots, \texttt{e1}^{\texttt{e}}_{\texttt{n}}\} \rangle$ and $e_2 = \langle \{\texttt{e2}^{\texttt{i}}_1, \ldots, \texttt{e2}^{\texttt{i}}_{\texttt{j}}\}, \{\texttt{e2}^{\texttt{e}}_1, \ldots, \texttt{e2}^{\texttt{e}}_{\texttt{k}}\} \rangle$
H does not cautiously respect $o \Leftrightarrow \exists A_1 \in AS(B \cup H \cup C_1), \exists A_2 \in AS(B \cup H \cup C_2)$ st A_1 extends e_1, A_2 extends e_2 and $A_1 \not\preceq_{B \cup H} A_2$
$\Leftrightarrow \exists A_1 \in AS(\mathit{reify}(B \cup H \cup C_1, \texttt{as1})), \exists A_2 \in AS(\mathit{reify}(B \cup H \cup C_2, \texttt{as2}))$ st A_1 extends $\mathit{reify}(e_1, \texttt{as1})$, A_2 extends $\mathit{reify}(e_2, \texttt{as2})$ and $\texttt{dominated}$ is not in the unique answer set of $A_1 \cup A_2 \cup \mathit{weak}(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup \mathit{dominates}$ (by Lemma 2)
$\Leftrightarrow \mathit{reify}(B \cup H \cup C_1, \texttt{as1}) \cup \mathit{reify}(B \cup H \cup C_2, \texttt{as2}) \cup \mathit{weak}(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup \mathit{dominates}$)

$$\cup \left\{ \begin{array}{l} \texttt{cov(as1):- as1(e1}^{\texttt{i}}_1\texttt{)}, \ldots, \texttt{as1(e1}^{\texttt{i}}_{\texttt{m}}\texttt{)}, \\ \quad \texttt{not as1(e1}^{\texttt{e}}_1\texttt{)}, \ldots, \texttt{not as1(e1}^{\texttt{e}}_{\texttt{n}}\texttt{).} \\ \texttt{:- not cov(as1).} \\ \texttt{cov(as2):- as2(e2}^{\texttt{i}}_1\texttt{)}, \ldots, \texttt{as2(e2}^{\texttt{i}}_{\texttt{j}}\texttt{)}, \\ \quad \texttt{not as2(e2}^{\texttt{e}}_1\texttt{)}, \ldots, \texttt{not as2(e2}^{\texttt{e}}_{\texttt{k}}\texttt{).} \\ \texttt{:- not cov(as2).} \\ \texttt{:- dominated.} \end{array} \right\} \text{ is satisfiable (we}$$

refer to this program as P_7 later in the proof)
$\Leftrightarrow \mathit{reify}(B \cup H \cup C_1, \texttt{as1}) \cup \mathit{reify}(B \cup H \cup C_2, \texttt{as2}) \cup \mathit{weak}(B \cup H) \cup \{\texttt{lv(l).} \mid \texttt{l} \in L\} \cup \mathit{dominates}$)

$$\cup \left\{ \begin{array}{l} \texttt{cov(as1):- test\_on(ex1}_{\texttt{id}}\texttt{, as2), as1(e1}^{\texttt{i}}_1\texttt{)}, \ldots, \texttt{as1(e1}^{\texttt{i}}_{\texttt{m}}\texttt{)}, \\ \quad \texttt{not as1(e1}^{\texttt{e}}_1\texttt{)}, \ldots, \texttt{not as1(e1}^{\texttt{e}}_{\texttt{n}}\texttt{).} \\ \texttt{:- not test\_on(ex1}_{\texttt{id}}\texttt{, as1), cov(as1).} \\ \texttt{cov(as2):- test\_on(ex2}_{\texttt{id}}\texttt{, as2), as2(e2}^{\texttt{i}}_1\texttt{)}, \ldots, \texttt{as2(e2}^{\texttt{i}}_{\texttt{j}}\texttt{)}, \\ \quad \texttt{not as2(e2}^{\texttt{e}}_1\texttt{)}, \ldots, \texttt{not as2(e2}^{\texttt{e}}_{\texttt{k}}\texttt{).} \\ \texttt{:- test\_on(ex2}_{\texttt{id}}\texttt{, as2), not cov(as2).} \\ \texttt{:- test(o}_{\texttt{id}}\texttt{), dominated.} \end{array} \right\}$$
$$\cup \mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$$

has an answer set which contains the atom $\texttt{test}(\texttt{o}_{\texttt{id}})$ (we refer to this program as P_8). This follows from the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, $\{\texttt{test}(\texttt{o}_{\texttt{id}}), \texttt{test\_on}(\texttt{ex1}_{\texttt{id}}, \texttt{as1}), \texttt{test\_on}(\texttt{ex2}_{\texttt{id}}, \texttt{as2})\}$ is an answer set of the bottom program, leading to P_7 as the partially evaluated top program
$\Leftrightarrow \mathcal{R}(T, H)$ has an answer set which contains $\texttt{test}(\texttt{o}_{\texttt{id}})$. Again, this is by

the splitting set theorem, using the atoms in $\mathcal{R}_6(E^+ \cup E^-, O^b \cup O^c)$ as a splitting set, as $P_8 \subseteq \mathcal{R}_6(T, H)$ and each of the extra rules which are in $\mathcal{R}_6(T, H)$ but not in P_8 contain a `test_on` or `test` atom which is not in the answer set $\{\texttt{test}(\texttt{o}_{\texttt{id}}), \texttt{test\_on}(\texttt{ex1}_{\texttt{id}}, \texttt{as1}), \texttt{test\_on}(\texttt{ex2}_{\texttt{id}}, \texttt{as2})\}$ and hence they are removed from the partially evaluated top program.

□

findRelevantExamples(T, H) works by constructing $\mathcal{R}(T, H)$ and computing its answer sets. For each example ex, whether of not ex is covered by T can be computed from the answer sets, using the results in Theorem 5. The first example which is not covered is returned. If no such example is found, `nil` is returned. The correctness of *findRelevantExamples* follows directly from Theorem 5. If the task T is well defined then $\mathcal{R}(T, H)$ will ground finitely (and have a finite number of answer sets), and therefore solving $\mathcal{R}(T, H)$ for answer sets will terminate in a finite time; hence as there are a finite number of examples, *findRelevantExamples* will terminate in a finite time.

References

Law, M., Russo, A., and Broda, K. 2015a. Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming 15,* 4-5, 511–525.

Law, M., Russo, A., and Broda, K. 2015b. Proof of the soundness and completeness of ILASP2. `https://www.doc.ic.ac.uk/~ml1909/Proofs_for_ILASP2.pdf`.

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. In *ICLP*. Vol. 94. 23–37.