Online appendix for the paper

Learning Weak Constraints in Answer Set Programming

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 27 April 2015; revised 3 July 2015; accepted 21 July 2015

Appendix A The ILASP2 Meta Encoding

We present here the ILASP2 meta encoding which is omitted from the main paper. We first summarise some notation used in the encoding.

We will write $body^+(R)$ and $body^-(R)$ to refer to the positive and negative (respectively) literals in the body of a rule R. Given a program P, $weak(P)$ denotes the weak constraints in P and $non\_weak(P)$ denotes the set of rules in P which are not weak constraints.

Definition 1
For any ASP program P, predicate name *pred* and term `term` we will write $reify(P, pred, term)$ to mean the program constructed by replacing every atom $\texttt{a} \in P$ by $\texttt{pred}(\texttt{a}, \texttt{term})$. We will use the same notation for sets of literals/partial interpretations, so for a set S: $reify(S, pred, term) = \{\texttt{pred}(\texttt{atom}, \texttt{term}) : \texttt{atom} \in S\}$.

Definition 2
For any ASP program P and any atom `a`, $append(P, a)$ is the program constructed by appending `a` to every rule in P.

Definition 3
Given any term t and any positive example e, $cover(e, t)$ is the program:

$\texttt{cov(t):-in\_as}(\texttt{e}_1^{\texttt{inc}}, \texttt{t}), \ldots, \texttt{in\_as}(\texttt{e}_\texttt{n}^{\texttt{inc}}, \texttt{t}), \texttt{not in\_as}(\texttt{e}_1^{\texttt{exc}}, \texttt{t}), \ldots, \texttt{not in\_as}(\texttt{e}_\texttt{m}^{\texttt{exc}}, \texttt{t}).$
$\texttt{:-not cov(t).}$

The previous three definitions can be used in combination to test whether a program has an answer sets which extend given partial interpretations.

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

Example 1

Consider the program $P = \left\{ \begin{array}{l} \texttt{p:-not q.} \\ \texttt{q:-not p.} \end{array} \right\}$ and the partial interpretations $I_1 = \langle\{p\}, \emptyset\rangle$ and $I_2 = \langle\emptyset, \{p\}\rangle$.

The program $Q = append(reify(P, in\_as, X), as(X)) \cup \{\texttt{as(as1)}, \texttt{as(as2)}\} \cup cover(I_1, as1) \cup cover(I_2, as2)$ has the grounding:

```
in_as(p, as1) :- not in_as(q, as1), as(as1).
in_as(q, as1) :- not in_as(p, as1), as(as1).
in_as(p, as2) :- not in_as(q, as2), as(as2).
in_as(q, as2) :- not in_as(p, as2), as(as2).

as(as1).
as(as2).

cov(as1) :- in_as(p, as1).
:- not cov(as1).

cov(as2) :- not in_as(p, as2).
:- not cov(as2).
```

Without the two constraints, Q would have 4 answer sets (the combinations of `as1` and `as2` corresponding to the two answer sets of P). With the two constraints, the answer set represented by `as1` must extend I_1, and the answer set represented by `as2` must extend I_2. Hence, there is only one answer set of Q, {`as(as1)`, `as(as2)`, `in_as(p,as1)`, `in_as(q,as2)`, `cov(as1)`, `cov(as2)`}.

Note that the answer sets of P which extend I_1 and I_2 ({`p`} and {`q`}) can be extracted from the *in_as* atoms in this answer set of Q. If there were multiple answer sets of P extending one or more of the partial interpretations, then there would be multiple answer sets, representing all possible combinations such that the constraints are met.

Definition 4

Let p_1 and p_2 be distinct predicate names and t be a term. Given R, a weak constraint $:\sim \texttt{b}_1, \ldots, \texttt{b}_m$, not $\texttt{c}_1, \ldots,$ not $\texttt{c}_l$. $[\texttt{wt@lev}, \texttt{t}_1, \ldots, \texttt{t}_n]$, $meta_{weak}(R, p_1, p_2, t)$ is the rule:

$\texttt{w(wt,lev,args(t}_1,\ldots,\texttt{t}_n\texttt{),t):-p}_2\texttt{(X)},\texttt{p}_1\texttt{(b}_1\texttt{,t)},\ldots,\texttt{p}_1\texttt{(b}_m\texttt{,t)},\texttt{not p}_1\texttt{(c}_1\texttt{,t)},\ldots,\texttt{not p}_1\texttt{(c}_1\texttt{,t)}.$

For a set of weak constraints W, $meta_{weak}(W, p_1, p_2, t)$ is the set $\{meta_{weak}(R, p_1, p_2, t) \mid R \in W\}$.

Example 2

Consider the program P containing the two weak constraints:

```
:~ p(V).[1@2, V]
:~ q(V).[2@1, V]
```

$meta_{weak}(P, in\_as, as, X)$ is the program:

```
w(1, 2, args(V), X) :- as(X), in_as(p(V), X).
w(2, 1, args(V), X) :- as(X), in_as(q(V), X).
```

Note that for any program P, if we reify an interpretation $I = \{\mathtt{a_1}, \ldots, \mathtt{a_n}\}$ as $\{\mathtt{in\_as(a_1, id)}, \ldots, \mathtt{in\_as(a_n, id)}\}$ (the set $\mathit{reify}(I, \mathit{in\_as}, \mathit{id})$) then the atoms $\mathtt{w(wt, l, args(t_1, \ldots, t_m), id)}$ in the (unique) answer set of $\mathit{meta}_{\mathit{weak}}(\mathit{weak}(P), \mathit{in\_as}, \mathit{as}, X) \cup \mathit{reify}(I, \mathit{in\_as}, \mathit{id}) \cup \{\mathtt{as(id)}\}$ correspond exactly to the elements $(wt, l, t_1, \ldots t_m)$ of $\mathit{weak}(P, I)$.

For example, consider the interpretation $I = \{p(1), p(2), q(1)\}$. The unique answer set of $\mathit{meta}_{\mathit{weak}}(\mathit{weak}(P), \mathit{in\_as}, \mathit{as}, X) \cup \mathit{reify}(I, \mathit{in\_as}, \mathit{id}) \cup \{\mathtt{as(id)}\}$ is $\{\mathtt{as(id)}$, $\mathtt{in\_as(p(1), id)}$, $\mathtt{in\_as(p(2), id)}$, $\mathtt{in\_as(q(1), id)}$, $\mathtt{w(1, 2, args(1), id)}$, $\mathtt{w(1, 2, args(2), id)}$, $\mathtt{w(2, 1, args(1), id)}\}$. In this case, $\mathit{weak}(P, I) = \{(1, 2, 1), (1, 2, 2), (2, 1, 1)\}$.

Now that we have defined the predicate w to represent $\mathit{weak}(P, A)$ for each answer set A, we can use some additional rules to determine, given two interpretations, whether one dominates another.

Definition 5
Given any two terms $t1$ and $t2$, $\mathit{dominates}(t1, t2)$ is the program:

$$\left\{ \begin{array}{l} \mathtt{dom\_lv(t1, t2, L) :- lv(L), \#sum\{w(W, L, A, t1) = W, w(W, L, A, t2) = -W\} < 0.} \\ \mathtt{non\_dom\_lv(t1, t2, L) :- lv(L), \#sum\{w(W, L, A, t2) = W, w(W, L, A, t1) = -W\} < 0.} \\ \mathtt{non\_bef(t1, t2, L) :- lv(L), lv(L2), L < L2, non\_dom\_lv(t1, t2, L2).} \\ \mathtt{dom(t1, t2) :- dom\_lv(t1, t2, L), not\ non\_bef(t1, t2, L).} \end{array} \right\}$$

The intuition is that $\mathtt{dom(id1, id2)}$ (where $\mathtt{id1}$ and $\mathtt{id2}$ represent two answer sets A_1 and A_2) should be true if and only if A_1 dominates A_2. This is dependent on the atoms $\mathtt{dom\_lv(id1, id2, l)}$, which for each level l, should be true if and only if $P^l_{A_1} < P^l_{A_2}$; $\mathtt{non\_dom\_lv(id1, id2, l)}$, which for each level l, should be true if and only if $P^l_{A_2} < P^l_{A_1}$; and finally, $\mathtt{non\_bef(id1, id2, l)}$, which for each level l, should be true if and only if there is an $l_2 > l$ such that $P^{l_2}_{A_2} < P^{l_2}_{A_1}$.

Example 3
Consider again the program P and interpretation I from example 2. Consider also an additional interpretation $I' = \{p(1), p(2), p(3)\}$. $\mathit{weak}(P, I') = \{(1, 2, 1), (1, 2, 2), (1, 2, 3)\}$.

The unique answer set of $\mathit{meta}_{\mathit{weak}}(\mathit{weak}(P), \mathit{in\_as}, \mathit{as}, X) \cup \mathit{reify}(I, \mathit{in\_as}, \mathit{id}1) \cup \mathit{reify}(I', \mathit{in\_as}, \mathit{id}2) \cup \{\mathtt{as(id1)}, \mathtt{as(id2)}, \mathtt{lv(1)}, \mathtt{lv(2)}\} \cup \mathit{dominates}(\mathit{id}1, \mathit{id}2)$ contains $\mathtt{dom\_lv(id1, id2, 2)}$, because I dominates I' at level 2 (i.e. $P^2_I < P^2_{I'}$); contains $\mathtt{non\_dom\_lv(id1, id2, 1)}$, because I' dominates I at level 1; does not contain any *non_bef* atoms, because the only level at which I' dominates I is 1, which is not evaluated "before" any other level (it is the lowest level in the program); and finally, does contain $\mathtt{dom(id1, id2)}$ because I dominates I' at level 2 and there is no level "before" (higher than) level 2 at which I' dominates I. The presence of $\mathtt{dom(id1, id2)}$ in the answer set indicates that I dominates I'.

Similarly, the unique answer set of $\mathit{meta}_{\mathit{weak}}(\mathit{weak}(P), \mathit{in\_as}, \mathit{as}, X) \cup \mathit{reify}(I, \mathit{in\_as}, \mathit{id}1) \cup \mathit{reify}(I', \mathit{in\_as}, \mathit{id}2) \cup \{\mathtt{as(id1)}, \mathtt{as(id2)}, \mathtt{lv(1)}, \mathtt{lv(2)}\} \cup \mathit{dominates}(\mathit{id}2, \mathit{id}1)$ contains $\mathtt{dom\_lv(id2, id1, 1)}$, because I' dominates I at level 1; contains $\mathtt{non\_dom\_lv(id2, id1, 2)}$, because I dominates I' at level 2; contains $\mathtt{non\_bef(id2, id1, 1)}$ as I dominates I' at level 2, which is evaluated "before" level 1;

and finally, does not contain $\texttt{dom(id2, id1)}$ because there is no level l in the program such that I' dominates I at l and I does not dominate I' at any higher level.

A.1 Encoding the search for positive hypotheses: T_{meta}

We now use the components described in the previous section to define a program T_{meta} whose answer sets correspond to the positive solutions of an ILP_{LOAS} task T.

Definition 6
Let T be the ILP_{LOAS} task $\langle B, S_M, E^+, E^-, O^b, O^c\rangle$. Then $T_{meta} = meta(B) \cup meta(S_M) \cup meta(E^+) \cup meta(E^-) \cup meta(O^b) \cup meta(O^c)$ where each meta component is as follows:

- $meta(B) = append(reify(non\_weak(B), in\_as, X), as(X)) \cup meta_{weak}(weak(B), in\_as, as, X).$
- $meta(S_M) =$
  $\{append(append(reify(R, in\_as, X), as(X)), in\_h(R_{id})) \mid R \in non\_weak(S_M)\}$
  $\cup \{append(W, in\_h(W_{id})) \mid W \in meta_{weak}(weak(S_M), in\_as, as, X)\}$
  $\cup \{\texttt{:\sim in\_h(R}_{\texttt{id}}\texttt{).[2} * |\texttt{R}|\texttt{@0, R}_{\texttt{id}}\texttt{]} \mid R \in S_M\}$
  $\cup \{\ \texttt{in\_h(R}_{\texttt{id}}\texttt{) : R} \in \texttt{S}_{\texttt{M}}\texttt{}.\ \}$
- $meta(E^+) = \left\{ \begin{array}{l} cover(e, e_{id}) \\ \texttt{as(e}_{\texttt{id}}\texttt{).} \end{array} \middle| \langle e^{inc}, e^{exc}\rangle \in E^+ \right\}$
- $meta(E^-) = \left\{ \begin{array}{l} \texttt{v\_i:-in\_as(e}_1^{\texttt{inc}}\texttt{, n}), \ldots, \texttt{in\_as(e}_{\texttt{n}}^{\texttt{inc}}\texttt{, n}), \\ \quad \texttt{not in\_as(e}_1^{\texttt{exc}}\texttt{, n}), \ldots, \\ \quad \texttt{not in\_as(e}_{\texttt{m}}^{\texttt{exc}}\texttt{, n}). \\ \texttt{as(n).} \end{array} \middle| \langle e^{inc}, e^{exc}\rangle \in E^- \right\}$
  $\cup \left\{ \begin{array}{l} \texttt{violating:-v\_i.} \\ \texttt{:\sim not violating.[1@0]} \end{array} \right\}$
- $meta(O^b) = \left\{ \begin{array}{l} \texttt{as(o}_{\texttt{id1}}\texttt{).} \quad \texttt{as(o}_{\texttt{id2}}\texttt{).} \\ cover(e^1, o_{id1}) \\ cover(e^2, o_{id2}) \\ dominates(o_{id1}, o_{id2}) \\ \texttt{:-not dom(o}_{\texttt{id1}}\texttt{, o}_{\texttt{id2}}\texttt{).} \end{array} \middle| \ o = \langle e_1, e_2\rangle \in O^b \right\} \cup \{\texttt{lv(l).} \mid l \in L\}$
- $meta(O^c) = \left\{ \begin{array}{l} dominates(e_1, e_2) \\ \texttt{v\_p(e}_{\texttt{id}}^1\texttt{, e}_{\texttt{id}}^2\texttt{):-} \\ \quad \texttt{not dom(e}_1\texttt{, e}_2\texttt{).} \end{array} \middle| \ \langle e_1, e_2\rangle \in O^c \right\} \cup \left\{ \begin{array}{l} \texttt{v\_p:-v\_p(T1, T2).} \\ \texttt{violating:-v\_p.} \end{array} \right\}$

The intuition is that the `in_h` atoms correspond to the rules in the hypothesis. Each rule $R \in S_M$ has a unique identifier R_{id} and if $\texttt{in\_h(R}_{\texttt{id}}\texttt{)}$ is true then R is considered to be part of the hypothesis H. These `in_h` atoms have been added to the bodies of the rules in the meta encoding so that a rule $R \in S_M$ only has an effect if it is part of H.

Each of the terms t for which there is a fact $\texttt{as(t)}$ represents an answer set of $B \cup H$. As in the previous section the *cover* program $cover(I, t)$ is used to enforce that some of these answer sets extend particular partial interpretations.

There is one as(t) atom for each positive example e. The *cover* program is used to ensure that the corresponding answer set does extend e. There are two as(t) atoms for each brave ordering $\langle e_1, e_2 \rangle$. Two instances of the *cover* program are used to ensure that the first answer set extends e_1 and the second answer set extends e_2. We also use the *dominates* program from the previous section and a constraint to ensure that the first answer set dominates the second (hence the ordering is bravely respected).

For the negative examples and the cautious orderings, the aim is to generate violating in at least one answer set of the meta encoding corresponding to H, if H is indeed a violating solution (generating v_i if H does not cover a negative example and some instance of v_p if it does not respect a cautious ordering).

Firstly, for the negative examples, we have an extra fact as(n). As we have no constraints on the answer set of $B \cup H$ which this can correspond to, the intuition is that there is one answer set of the meta encoding for each answer set of $B \cup H$. For each negative example e^- there is a rule for v_i which will generate v_i if the answer set corresponding to as(n) extends e^-.

For the cautious orderings we use a similar approach. For any cautious ordering $\langle e_1, e_2 \rangle$, as e_1 and e_2 are positive examples, there are already two as(t) atoms which represent answer sets extending each of these interpretations; in fact, there will be one answer set of the meta encoding for each possible pair of answer sets of $B \cup H$ which extend these interpretations. Therefore, by using the *dominates* program, and generating a v_p atom if the answer set of $B \cup H$ corresponding to the first as(t) atom does not dominate the answer set of $B \cup H$ corresponding to the second as(t) atom in any answer set of the meta encoding, we ensure that violating will be true in at least one answer set of the meta encoding which corresponds to H.

Example 4

Consider the learning task:

$$B = \left\{ \begin{array}{l} \texttt{p(V):-r(V), not q(V).} \\ \texttt{q(V):-r(V), not p(V).} \\ \texttt{r(1).} \quad \texttt{r(2).} \\ \texttt{a:-not b.} \\ \texttt{b:-not a.} \end{array} \right\}$$

$$S_M = \left\{ \begin{array}{l} \texttt{q(1).} \\ \texttt{:\sim q(V).[1@1, V, r2]} \\ \texttt{:\sim b.[1@1, b, r3]} \end{array} \right\}$$

$$E^+ = \left\{ \begin{array}{l} \langle \{\texttt{p(2)}\}, \emptyset \rangle, \\ \langle \emptyset, \{\texttt{p(2)}\} \rangle, \\ \langle \{\texttt{a}\}, \{\texttt{b}\} \rangle, \\ \langle \emptyset, \{\texttt{a}\} \rangle \end{array} \right\}$$

$$E^- = \{ \ \langle \{\texttt{p(1)}\}, \emptyset \rangle \ \}$$

$$O^b = \{ \ \langle e_3^+, e_4^+ \rangle \ \}$$

$$O^c = \{ \ \langle e_1^+, e_2^+ \rangle \ \}$$

Figure A 1 shows T_{meta}. There are two optimal positive hypotheses (one containing each of the two weak constraints).

The positive hypothesis $:\sim \texttt{b} \cdot [\texttt{1@1}, \texttt{b}, \texttt{r3}]$ has a violating interpretation $\{\texttt{p(1)}, \texttt{q(2)}, \texttt{r(1)}, \texttt{r(2)}, \texttt{a}\}$. This corresponds to the following answer set of T_{meta}:

```
{ as(1), as(2), as(3), as(4), as(n), as(5), as(6), lv(1), in_as(r(1),1),
  in_as(r(1),2), in_as(r(1),3), in_as(r(1),4), in_as(r(1),n), in_as(r(1),5),
  in_as(r(1),6), in_as(r(2),1), in_as(r(2),2), in_as(r(2),3), in_as(r(2),4),
  in_as(r(2),n), in_as(r(2),5), in_as(r(2),6), in_as(q(1),3), in_as(q(1),4),
```

```
  in_as(q(1),6), in_as(p(1),1), in_as(p(1),2), in_as(p(1),n), in_as(p(1),5),
  in_as(p(2),1), in_as(q(2),2), in_as(q(2),3), in_as(q(2),4), in_as(q(2),n),
  in_as(q(2),5), in_as(q(2),6), in_as(a,1), in_as(a,2), in_as(a,3),
  in_as(b,4), in_as(a,n), in_as(a,5), in_as(b,6), in_h(r2),
  w(1,1,args(2,r2),2), w(1,1,args(1,r2),3), w(1,1,args(2,r2),3),
  w(1,1,args(1,r2),4), w(1,1,args(2,r2),4), w(1,1,args(2,r2),n),
  w(1,1,args(2,r2),5), w(1,1,args(1,r2),6), w(1,1,args(2,r2),6), cov(1),
  cov(2), cov(3), cov(4), v_i, violating, dom_lv(1,2,1), dom(1,2), cov(5),
  cov(6), dom_lv(5,6,1), dom(5,6) }
```

Note that the violating interpretation can be extracted from the $\texttt{in\_as}(\_,\texttt{n})$ atoms and the hypothesis can be extracted from the $\texttt{in\_h}$ atoms.

Similarly, the violating pair $\langle\{\texttt{p}(2),\texttt{q}(1),\texttt{r}(1),\texttt{r}(2),\texttt{a}\},\{\texttt{p}(1),\texttt{q}(2),\texttt{r}(1),\texttt{r}(2),\texttt{a}\}\rangle$ can be extracted from the answer set:

```
{ as(1), as(2), as(3), as(4), as(n), as(5), as(6), lv(1), in_as(r(1),1),
  in_as(r(1),2), in_as(r(1),3), in_as(r(1),4), in_as(r(1),n), in_as(r(1),5),
  in_as(r(1),6), in_as(r(2),1), in_as(r(2),2), in_as(r(2),3), in_as(r(2),4),
  in_as(r(2),n), in_as(r(2),5), in_as(r(2),6), in_as(q(1),1), in_as(q(1),4),
  in_as(q(1),6), in_as(p(1),2), in_as(p(1),3), in_as(p(1),n), in_as(p(1),5),
  in_as(p(2),1), in_as(q(2),2), in_as(q(2),3), in_as(q(2),4), in_as(q(2),n),
  in_as(q(2),5), in_as(q(2),6), in_as(a,1), in_as(a,2), in_as(a,3),
  in_as(b,4), in_as(a,n), in_as(a,5), in_as(b,6), w(1,1,args(1,r2),1),
  in_h(r2), w(1,1,args(2,r2),2), w(1,1,args(2,r2),3), w(1,1,args(1,r2),4),
  w(1,1,args(2,r2),4), w(1,1,args(2,r2),n), w(1,1,args(2,r2),5),
  w(1,1,args(1,r2),6), w(1,1,args(2,r2),6), cov(1), cov(2), cov(3), cov(4),
  v_i, violating, v_p(1,2), v_p, cov(5), cov(6), dom_lv(5,6,1), dom(5,6) }
```

```
% meta(B)
in_as(p(V),X) :- in_as(r(V),X),
  not in_as(q(V),X), as(X).
in_as(q(V),X) :- in_as(r(V),X),
  not in_as(p(V),X), as(X).
in_as(r(1),X) :- as(X).
in_as(r(2),X) :- as(X).
in_as(a,X) :- not in_as(b,X), as(X).
in_as(b,X) :- not in_as(a,X), as(X).

% meta(S_M)
in_as(q(1),X) :- as(X), in_h(r1).
w(1,1,args(V,r2),X) :- in_as(q(V),X),
  as(X), in_h(r2).
w(1,1,args(b,r3),X) :- in_as(b,X),
  as(X), in_h(r3).
0 {in_h(r1), in_h(r2), in_h(r3)} 2.
:~ in_h(r1).[2@0,r1]
:~ in_h(r2).[2@0,r2]
:~ in_h(r3).[2@0,r3]

% meta(E^+)
as(1).
as(2).
as(3).
as(4).
cov(1) :- in_as(p(2),1).
cov(2) :- not in_as(p(2),2).
cov(3) :- in_as(a,3), not in_as(b,3).
cov(4) :- not in_as(a,4).
:- not cov(1).
:- not cov(2).
:- not cov(3).
:- not cov(4).
```

```
% meta(E^-)
v_i :- in_as(p(1),n).
as(n).
violating :- v_i.
:~ not violating.[1@0, violating]

% meta(O^b)
as(5).
as(6).
cov(5) :- in_as(a,5), not in_as(b,5).
cov(6) :- not in_as(a,6).
:- not cov(5).
:- not cov(6).
dom_lv(5,6,L) :- lv(L),
  #sum{w(W,L,A,5)=W, w(W,L,A,6)=-W} < 0.
wrong_dom_lv(5,6,L) :- lv(L),
  #sum{w(W,L,A,6)=W, w(W,L,A,5)=-W} < 0.
wrong_bef(5,6,L) :- lv(L), L < L2,
  wrong_dom_lv(5,6,L2).
dom(5,6) :- dom_lv(5,6,L),
  not wrong_bef(5,6,L).
:- not dom(5,6).
lv(1).

% meta(O^c)
dom_lv(1,2,L) :- lv(L),
  #sum{w(W,L,A,1)=W, w(W,L,A,2)=-W} < 0.
wrong_dom_lv(1,2,L) :- lv(L),
  #sum{w(W,L,A,2)=W, w(W,L,A,1)=-W} < 0.
wrong_bef(1,2,L) :- lv(L), L < L2,
  wrong_dom_lv(1,2,L2).
dom(1,2) :- dom_lv(1,2,L),
  not wrong_bef(1,2,L).
v_p(1,2) :- not dom(1,2).
v_p :- v_p(X,Y).
violating :- v_p.
```

Fig. A 1: An example of T_{meta}.

A.2 Encoding classes of violating hypotheses: VR_{meta}

The previous section set out how to construct a meta level ASP program T_{meta} whose answer sets correspond to the positive hypotheses. It is also able to identify violating hypotheses, but has no way of eliminating them (as only a subset of the meta level answer sets corresponding to a violating hypothesis will identify it as violating, so eliminating these answer sets does not necessarily eliminate the hypothesis).

We therefore present a new meta level program which, combined with T_{meta}, eliminates any hypothesis which is violating for a given set of violating reasons. As

violating reasons are full answer sets (or pair of full answer sets), we do not have to generate answer sets. It is enough to check whether the answer sets we have as part of the violating reasons are still answer sets of $B \cup H$. The standard way to do this is check whether these answer sets are the minimal model of the reduct of $B \cup H$ with respect to this answer set. The next two definitions define a meta level program which, given an interpretation, computes the minimal model of the reduct of a program with respect to that interpretation. This reduct construction is close to the simplified reduct for choice rules from (Law et al. 2015).

Definition 7

Given any choice rule $R = \texttt{l\{h}_1, \ldots, \texttt{h}_n\texttt{\}u:-body}$, *reductify*$(R)$ is the program:

$$\left\{ \begin{array}{l} \texttt{mmr(h}_1\texttt{,X):-reify(body}^+\texttt{,mmr,X),reify(body}^-\texttt{,not in\_vs,X),} \\ \qquad \texttt{l\{in\_vs(h}_1\texttt{,X)}, \ldots, \texttt{in\_vs(h}_n\texttt{,X)\}u,in\_vs(h}_1\texttt{,X)}\cdot \\ \qquad\qquad \ldots \\ \texttt{mmr(h}_n\texttt{,X):-reify(body}^+\texttt{,mmr,X),reify(body}^-\texttt{,not in\_vs,X),} \\ \qquad \texttt{l\{in\_vs(h}_1\texttt{,X)}, \ldots, \texttt{in\_vs(h}_n\texttt{,X)\}u,in\_vs(h}_n\texttt{,X)}\cdot \\ \texttt{mmr}(\bot\texttt{,X):-reify(body}^+\texttt{,mmr,X),reify(body}^-\texttt{,not in\_vs,X),} \\ \qquad \texttt{u} + \texttt{1\{in\_vs(h}_1\texttt{,X)}, \ldots, \texttt{in\_vs(h}_n\texttt{,X)\}}\cdot \\ \texttt{mmr}(\bot\texttt{,X):-reify(body}^+\texttt{,mmr,X),reify(body}^-\texttt{,not in\_vs,X),} \\ \qquad \texttt{\{in\_vs(h}_1\texttt{,X)}, \ldots, \texttt{in\_vs(h}_n\texttt{,X)\}l} - \texttt{1}\cdot \end{array} \right\}$$

Definition 8

Let P be an ASP program such that P_1 is the set of normal rules in P, P_2 is the set of constraints in P and P_3 is the set of choice rules in P.

$$\begin{aligned} \textit{reductify}(P) = & \left\{ \begin{array}{l} \texttt{mmr(head(R),X):-reify(body}^+\texttt{(R),mmr,X),} \\ \quad \texttt{reify(body}^-\texttt{(R),not in\_vs,X),vs(X).} \end{array} \middle| R \in P_1 \right\} \\ & \cup \left\{ \begin{array}{l} \texttt{mmr}(\bot\texttt{,X):-reify(body}^+\texttt{(R),mmr,X),} \\ \quad \texttt{reify(body}^-\texttt{(R),not in\_vs,X),vs(X).} \end{array} \middle| R \in P_2 \right\} \\ & \cup \{\textit{reductify}(R) \mid R \in P_3\}. \end{aligned}$$

Example 5

Consider the program $P = \left\{ \begin{array}{l} \texttt{p:-not q.} \\ \texttt{q:-not p.} \end{array} \right\}$

$$\textit{reductify}(P) = \left\{ \begin{array}{l} \texttt{mmr(p,X):-not in\_vs(q,X),vs(X).} \\ \texttt{mmr(q,X):-not in\_vs(p,X),vs(X).} \end{array} \right\}$$

We can check whether $\{\texttt{p}\}$ is an answer set by combining *reductify*(P) with $\{\texttt{vs(vs1)}, \texttt{in\_vs(p,vs1)}\}$. The answer set of this program is $\{\texttt{vs(vs1)}, \texttt{in\_vs(p,vs1)}, \texttt{mmr(p,vs1)}\}$, from which the minimal model, $\{\texttt{p}\}$, can be extracted. This shows that $\{\texttt{p}\}$ is indeed an answer set of P.

Definition 9

Let T be the ILP_{LOAS} task $\langle B, S_M, E^+, E^-, O^b, O^c \rangle$ and VR be the set of violating reasons $VI \cup VP$, where VI are violating interpretations and VP are violating pairs.

$VR_{meta}(T)$ is the program $meta(VI) \cup meta(VP) \cup meta(Aux)$ where the meta components are defined as follows:

- $meta(VI) = \left\{ \begin{array}{l} reify(I, in\_vs, I_{id}) \\ \texttt{:-not nas(I}_{\texttt{id}}\texttt{).} \\ \texttt{vs(I}_{\texttt{id}}\texttt{).} \end{array} \middle| I \in VI \right\}$
- $meta(VP) = \left\{ \begin{array}{l} dominates(vp_{id}1, vp_{id}2) \\ reify(I_1, in\_vs, vp_{id}1) \\ reify(I_2, in\_vs, vp_{id}2) \\ \texttt{vs(vp}_{\texttt{id1}}\texttt{).} \\ \texttt{vs(vp}_{\texttt{id2}}\texttt{).} \\ \texttt{:-not nas(vp}_{\texttt{id1}}\texttt{),not nas(vp}_{\texttt{id2}}\texttt{),} \\ \quad \texttt{not dom(vp}_{\texttt{id1}}\texttt{,vp}_{\texttt{id2}}\texttt{).} \end{array} \middle| vp = \langle I_1, I_2 \rangle \in VP \right\}$
- $meta(Aux) =$

$$\left\{ \begin{array}{l} reductify(B) \\ \quad \cup \left\{ \begin{array}{l} \texttt{nas(X):-in\_vs(ATOM,X),not mmr(ATOM,X).} \\ \texttt{nas(X):-not in\_vs(ATOM,X),mmr(ATOM,X).} \end{array} \right\} \\ \quad \cup \{append(reductify(R), in\_hyp(R_{id})) \mid R \in non\_weak(S_M)\} \\ \quad \cup \{append(meta_{weak}(W, in\_vs, vs, X), in\_hyp(W_{id})) \mid W \in weak(S_M)\} \\ \quad \cup \{meta_{weak}(W, in\_vs, vs, X) \mid W \in weak(B)\} \\ \quad \cup \{\texttt{lv(l).} \mid l \in L\} \end{array} \right\}$$

This meta encoding uses the reductify program to check whether the various interpretations in each violating reason is still an answer set of $B \cup H$. There is a constraint for each of the violating interpretation, ensuring that it is no longer an answer set of $B \cup H$. Similarly, there is a constraint for each violating pair that says, if both interpretations are still answer sets of $B \cup H$, then the first must dominate the second. This is checked by using the *dominates* program as before (the weak constraints are also translated as before).

Example 6
Recall B, S_M, E^+, E^-, O^b and O^c from example 4 and let VI be the set containing the violating interpretation $\{\texttt{p(1)}, \texttt{p(2)}, \texttt{r(1)}, \texttt{r(2)}, \texttt{a}\}$, VP the set containing the violating pair $\langle\{\texttt{p(2)}, \texttt{q(1)}, \texttt{r(1)}, \texttt{r(2)}, \texttt{a}\}, \{\texttt{q(1)}, \texttt{q(2)}, \texttt{r(1)}, \texttt{r(2)}, \texttt{a}\}\rangle$ and let VR be the set of violating reasons $VI \cup VP$. Then figure A 2 shows $VR_{meta}(T)$.

Now that we have ruled out any hypothesis with these violating reasons, one optimal answer set of this program is:

```
{ as(1), as(2), as(3), as(4), as(n), as(5), as(6), lv(1), in_vs(p(1),v1),
  in_vs(p(2),v1), in_vs(r(1),v1), in_vs(r(2),v1), in_vs(a,v1), vs(v1),
  in_vs(p(2),v2), in_vs(q(1),v2), in_vs(r(1),v2), in_vs(r(2),v2), in_vs(a,v2),
  vs(v2), in_vs(q(1),v3), in_vs(q(2),v3), in_vs(r(1),v3), in_vs(r(2),v3),
  in_vs(a,v3), vs(v3), in_as(r(1),1), in_as(r(1),2), in_as(r(1),3),
  in_as(r(1),4), in_as(r(1),n), in_as(r(1),5), in_as(r(1),6), in_as(r(2),1),
  in_as(r(2),2), in_as(r(2),3), in_as(r(2),4), in_as(r(2),n), in_as(r(2),5),
  in_as(r(2),6), in_as(q(1),1), in_h(r1), in_as(q(1),2), in_as(q(1),3),
  in_as(q(1),4), in_as(q(1),n), in_as(q(1),5), in_as(q(1),6), in_as(p(2),1),
  in_as(q(2),2), in_as(p(2),3), in_as(q(2),4), in_as(p(2),n), in_as(p(2),5),
  in_as(q(2),6), in_as(a,1), in_as(a,2), in_as(a,3), in_as(b,4), in_as(a,n),
  in_as(a,5) in_as(b,6), w(1,1,args(1,r2),1), in_h(r2), w(1,1,args(1,r2),2),
  w(1,1,args(2,r2),2), w(1,1,args(1,r2),3), w(1,1,args(1,r2),4),
  w(1,1,args(2,r2),4), w(1,1,args(1,r2),n), w(1,1,args(1,r2),5),
```

```
  w(1,1,args(1,r2),6), w(1,1,args(2,r2),6), cov(1), cov(2), cov(3), cov(4),
  w(1,1,ts(1),v2), w(1,1,ts(1),v3), w(1,1,ts(2),v3), dom_lv(1,2,1), dom(1,2),
  cov(5), cov(6), dom_lv(5,6,1), dom(5,6), mmr(r(1),v1), mmr(r(1),v2),
  mmr(r(1),v3), mmr(r(2),v1), mmr(r(2),v2), mmr(r(2),v3), mmr(p(1),v1),
  mmr(p(2),v1), mmr(p(2),v2), mmr(q(1),v2), mmr(q(1),v3), mmr(q(2),v3),
  mmr(a,v1), mmr(a,v2), mmr(a,v3), mmr(q(1),v1), nas(v1), dom_lv(v2,v3,1),
  dom(v2,v3) }
```

This answer set corresponds to the hypothesis:

```
q(1).
:~ q(V).[1@1,V,r2]
```

As the optimality of this answer set is 5, we know that this hypothesis cannot have any violating reasons, as otherwise, there would be an answer set with optimality 4 corresponding to the hypothesis (which would have been returned as optimal). Hence, the hypothesis must be an optimal inductive solution of the task.

```
% meta(VI)
in_vs(p(1),v1).
in_vs(p(2),v1).
in_vs(r(1),v1).
in_vs(r(2),v1).
in_vs(a,v1).
vs(v1).
:- not nas(v1).

% meta(VP)
in_vs(p(2),v2).
in_vs(q(1),v2).
in_vs(r(1),v2).
in_vs(r(2),v2).
in_vs(a,v2).
vs(v2).

in_vs(q(1),v3).
in_vs(q(2),v3).
in_vs(r(1),v3).
in_vs(r(2),v3).
in_vs(a,v3).
vs(v3).

dom_lv(v2,v3,L) :- lv(L),
   #sum{w(W,L,A,v2)=W,
        w(W,L,A,v3)=-W} < 0.
wrong_dom_lv(v2,v3,L) :- lv(L),
   #sum{w(W,L,A,v3)=W,
        w(W,L,A,v2)=-W} < 0.
wrong_bef(v2,v3,L) :- lv(L), L < L2,
   wrong_dom_lv(1,2,L2).
dom(v2,v3) :- dom_lv(v2,v3,L),
   not wrong_bef(v2,v3,L).

:- not nas(v2), not nas(v3),
   not dom(v2,v3).

% meta(Aux)
mmr(p(V),X) :- mmr(r(V),X),
  not in_vs(q(V),X), vs(X).
mmr(q(V),X) :- mmr(r(V),X),
  not in_vs(p(V),X), vs(X).
mmr(r(1),X) :- vs(X).
mmr(r(2),X) :- vs(X).
mmr(a,X) :- not in_vs(b,X), vs(X).
mmr(b,X) :- not in_vs(a,X), vs(X).

mmr(q(1),X) :- vs(X), in_h(r1).
w(1,1,ts(V),X) :- vs(X),
  in_vs(q(V),X), in_h(r2).
w(1,1,args(b,r3),X) :- vs(X),
  in_vs(b,X), in_h(r3).

nas(X) :- in_vs(A,X), not mmr(A,X).
nas(X) :- not in_vs(A,X), mmr(A,X).
```

Fig. A 2: An example of $VR_{meta}(T)$.

References

Law, M., Russo, A., and Broda, K. 2015. Simplified reduct for choice rules in ASP. Tech. Rep. DTR2015-2, Imperial College of Science, Technology and Medicine, Department of Computing.