Conflict-driven ASP Solving with External Source Access

Thomas Eiter, Michael Fink, Thomas Krennwallner, Christoph Redl

redl@kr.tuwien.ac.at

September 05, 2012

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

September 05, 2012
Motivation

HEX-Programs

- Extend ASP by external sources
- Current algorithm based on a translation to ASP
- Scalability problems

Contribution

- New genuine algorithms
- Based on conflict-driven algorithms
- Much better scalability
Outline

1 Introduction

2 Algorithms with External Behavior Learning

3 Nogood Generation for External Behavior Learning

4 Implementation and Evaluation

5 Conclusion
Outline

1 Introduction

2 Algorithms with External Behavior Learning

3 Nogood Generation for External Behavior Learning

4 Implementation and Evaluation

5 Conclusion
HEX-Programs
HEX-programs extend ordinary ASP programs by external sources

Definition (HEX-programs)
A HEX-program consists of rules of form
\[ a_1 \lor \cdots \lor a_n \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n, \]
with classical literals \( a_i \), and classical literals or an external atoms \( b_j \).

Definition (External Atoms)
An external atom is of the form
\[ \& p[q_1, \ldots, q_k](t_1, \ldots, t_l), \]
\( p \) ... external predicate name
\( q_i \) ... predicate names or constants
\( t_j \) ... terms

Semantics:
1 + k + l-ary Boolean oracle function \( f_{\& p} \):
\[ \& p[q_1, \ldots, q_k](t_1, \ldots, t_l) \] is true under assignment \( A \)
iff \( f_{\& p}(A, q_1, \ldots, q_k, t_1, \ldots, t_l) = 1. \)
Examples

\texttt{&\textit{rdf}}

The \texttt{&\textit{rdf}} External Atom

- Input: URL
- Output: Set of triplets from RDF file

External knowledge base is a set of RDF files on the web:

\begin{align*}
\texttt{addr}(&\texttt{http://.../data1.rdf}). \\
\texttt{addr}(&\texttt{http://.../data2.rdf}). \\
\texttt{bel}(X, Y) &\leftarrow \texttt{addr}(U), \texttt{&\textit{rdf}[U]}(X, Y, Z).
\end{align*}
Examples

\&rdf

The \&rdf External Atom

- Input: URL
- Output: Set of triplets from RDF file

External knowledge base is a set of RDF files on the web:

\begin{align*}
addr(\text{http://.../data1.rdf}). \\
addr(\text{http://.../data2.rdf}). \\
bel(X, Y) \leftarrow addr(U), \&\text{rdf}[U](X, Y, Z).
\end{align*}

\&diff

\&diff[p, q](X): all elements X, which are in the extension of p but not of q:

\begin{align*}
dom(X) & \leftarrow \#\text{int}(X). \\
nsel(X) & \leftarrow dom(X), \&\text{diff}[dom, sel](X). \\
 sel(X) & \leftarrow dom(X), \&\text{diff}[dom, nsel](X). \\
 & \leftarrow sel(X1), sel(X2), sel(X3), X1 \neq X2, X1 \neq X3, X2 \neq X3.
\end{align*}
Current Evaluation Method

Evaluating Program $\Pi$

1. Replace external atoms $\&g[\textit{p}](\vec{c})$ by ordinary ones $e_{\&g}[\textit{p}](\vec{c})$ and guess their values $\rightarrow$ guessing program $\hat{\Pi}$
2. For each candidate, check if the truth values coincide with external sources
3. Check if $A$ is subset-minimal under all compatible sets

Definition (Compatible Set)

A compatible set of a program $\Pi$ is an assignment $A$

(i) which is an answer set [Gelfond and Lifschitz, 1991] of $\hat{\Pi}$, and

(ii) $f_{\&g}(A, \textit{p}, \vec{c}) = 1$ iff $Te_{\&g}[\textit{p}](\vec{c}) \in A$ for all external atoms $\&g[\textit{p}](\vec{c})$ in $\Pi$

Definition (Answer Set)

An (DLVHEX) answer set of $\Pi$ is any set $S \subseteq \{Ta \mid a \in A(\Pi)\}$ such that

(i) $S = \{Ta \mid a \in A(\Pi)\} \cap A$ for some compatible set $A$ of $\Pi$ and

(ii) $S \nsubseteq \{Ta \mid a \in A(\Pi)\} \cap A$ for every compatible set $A$ of $\Pi$. 
Current Evaluation Method

Translation Approach

HEX-Program \( \Pi \):

\[
p(c_1). \, dom(c_1). \, dom(c_2). \, dom(c_3).
p(X) \leftarrow dom(X), \& \text{empty}[p](X).
\]

Guessing program \( \hat{\Pi} \):

\[
p(c_1). \, dom(c_1). \, dom(c_2). \, dom(c_3).
p(X) \leftarrow dom(X), e \& \text{empty}[p](X).
\]

\[
ee \& \text{empty}[p](X) \lor \neg e \& \text{empty}[p](X) \leftarrow \text{dom}(X).
\]

8 candidates, e.g.:

\[
\{Tp(c_1), Tp(c_2), T\text{dom}(c_1), T\text{dom}(c_2), T\text{dom}(c_3),
Fe \& \text{empty}[p](c_1), Te \& \text{empty}[p](c_2), Fe \& \text{empty}[p](c_3)\}
\]

Compatibility check: passed \( \Rightarrow \) compatible set
Outline

1. Introduction

2. Algorithms with External Behavior Learning

3. Nogood Generation for External Behavior Learning

4. Implementation and Evaluation

5. Conclusion
Novel Algorithm

Idea

- Use a conflict-driven (disjunctive) ASP solver [Drescher et al., 2008]
- Program $\hat{\Pi}$ is represented as a set of nogoods
Novel Algorithm

Idea

- Use a conflict-driven (disjunctive) ASP solver [Drescher et al., 2008]
- Program $\Pi$ is represented as a set of nogoods
- Introduce additional nogoods to describe external sources

Definition (Assignment)

An assignment is a consistent set of signed literals $T_a$ or $F_a$, where $a$ is an atom.

Definition (Nogoods)

A nogood is a set of signed literals.

Definition (Solution to Nogoods)

An assignment $A$ is a solution to a set of nogoods $\Delta$ iff $\delta \not\subseteq A$ for all $\delta \in \Delta$. 
Basic Definitions

**Definition (Correct Nogoods)**

A nogood $\delta$ is **correct wrt. program** $\Pi$, if all compatible sets of $\Pi$ are solutions to $\delta$.

**Definition (Learning Function)**

A **learning function** for $\Pi$ is a mapping $\Lambda : \mathcal{E} \times 2^D \mapsto 2^2^D$

*\mathcal{E} \ldots set of all external predicates with input list in $\Pi$

*D \ldots set of all signed literals

**Definition (Correct Learning Functions)**

A learning function $\Lambda$ is **correct** for a program $\Pi$, if and only if all $d \in \Lambda(\&g[\vec{p}], \Lambda)$ are correct for $\Pi$, for all $\&g[\vec{p}]$ in $\mathcal{E}$ and $\Lambda \in 2^D$. 
Algorithms

**Algorithm: HEX-Eval**

**Input:** A HEX-program \( \Pi \)

**Output:** All answer sets of \( \Pi \)

\[
\tilde{\Pi} \leftarrow \Pi \text{ with all external atoms } &\tilde{\Pi}(\tilde{e}) \text{ replaced by } e &\tilde{\Pi}(\tilde{e})
\]

Add guessing rules for all replacement atoms to \( \tilde{\Pi} \)

\[
\nabla \leftarrow \emptyset \text{ /* set of dynamic nogoods */}
\]

\[
\Gamma \leftarrow \emptyset \text{ /* set of all compatible sets */}
\]

\[
\hat{\Pi} \text{ : HEX-program }
\]

\[
\Pi \neq \emptyset \text{ : All answer sets of } \Pi \text{ with all external atoms }
\]

if inconsistent

\[
\text{C} \leftarrow \emptyset \text{ /* assignment over } A(\tilde{\Pi}) \cup BA(\tilde{\Pi}) \cup BA(sh(\tilde{\Pi})) * /
\]

\[
dl \leftarrow 0 \text{ /* decision level */}
\]

while true do

\[
(A, \nabla) \leftarrow \text{Propagation}(\tilde{\Pi}, \nabla, A)
\]

\[
\text{if } \delta \subseteq A \text{ for some } \delta \in \Delta_{\tilde{\Pi}} \cup \Theta_{sh(\tilde{\Pi})} \cup \nabla \text{ then }
\]

\[
\text{if } dl = 0 \text{ then return } \perp
\]

\[
(\epsilon, k) \leftarrow \text{Analysis}(\delta, \tilde{\Pi}, \nabla, A)
\]

\[
\nabla \leftarrow \nabla \cup \{\epsilon\}
\]

\[
A \leftarrow A \setminus \{\sigma \in A \mid k < dl(\sigma)\}
\]

\[
dl \leftarrow k
\]

else if \( A^T \cup A^F = A(\tilde{\Pi}) \cup BA(\tilde{\Pi}) \cup BA(sh(\tilde{\Pi})) \) then

\[
U \leftarrow \text{UnfoundedSet}(\tilde{\Pi}, A)
\]

\[
\text{if } U \neq \emptyset \text{ then }
\]

\[
\text{let } \delta \in \Lambda_{\tilde{\Pi}}(U) \text{ such that } \delta \subseteq A
\]

\[
\text{if } \{\sigma \in \delta \mid 0 < dl(\sigma)\} = \emptyset \text{ then return } \perp
\]

\[
(\epsilon, k) \leftarrow \text{Analysis}(\delta, \tilde{\Pi}, \nabla, A)
\]

\[
\nabla \leftarrow \nabla \cup \{\epsilon\}
\]

\[
A \leftarrow A \setminus \{\sigma \in A \mid k < dl(\sigma)\}
\]

\[
dl \leftarrow k
\]

else

\[
\text{return } A^T \cap A(\tilde{\Pi})
\]

else if Heuristic decides to evaluate \( &\tilde{\Pi}(\tilde{e}) \) then

\[
\text{Evaluate } &\tilde{\Pi}(\tilde{e}) \text{ under } A \text{ and set } \nabla \leftarrow \nabla \cup \Lambda( &\tilde{\Pi}(\tilde{e}), A)
\]

else

\[
\sigma \leftarrow \text{Select}(\tilde{\Pi}, \nabla, A)
\]

\[
dl \leftarrow dl + 1
\]

\[
A \leftarrow A \cup (\sigma)
\]

Output \( \{Ta \in A \mid a \in A(\Pi)\} \) which are subset-minimal

**Algorithm: HEX-CDNL**

**Input:** A program \( \Pi \), its guessing program \( \hat{\Pi} \), a set of correct nogoods \( \nabla \) of \( \Pi \)

**Output:** An answer set of \( \hat{\Pi} \) (candidate for a compatible set of \( \Pi \)) which is a solution to all nogoods \( d \in \nabla \), or \( \perp \) if none exists

\[
A \leftarrow \emptyset \text{ /* assignment over } A(\hat{\Pi}) \cup BA(\hat{\Pi}) \cup BA(sh(\hat{\Pi})) * /
\]

\[
dl \leftarrow 0 \text{ /* decision level */}
\]

while true do

\[
(A, \nabla) \leftarrow \text{Propagation}(\hat{\Pi}, \nabla, A)
\]

\[
\text{if } \delta \subseteq A \text{ for some } \delta \in \Delta_{\hat{\Pi}} \cup \Theta_{sh(\hat{\Pi})} \cup \nabla \text{ then }
\]

\[
\text{if } dl = 0 \text{ then return } \perp
\]

\[
(\epsilon, k) \leftarrow \text{Analysis}(\delta, \hat{\Pi}, \nabla, A)
\]

\[
\nabla \leftarrow \nabla \cup \{\epsilon\}
\]

\[
A \leftarrow A \setminus \{\sigma \in A \mid k < dl(\sigma)\}
\]

\[
dl \leftarrow k
\]

else if \( A^T \cup A^F = A(\hat{\Pi}) \cup BA(\hat{\Pi}) \cup BA(sh(\hat{\Pi})) \) then

\[
U \leftarrow \text{UnfoundedSet}(\hat{\Pi}, A)
\]

\[
\text{if } U \neq \emptyset \text{ then }
\]

\[
\text{let } \delta \in \Lambda_{\hat{\Pi}}(U) \text{ such that } \delta \subseteq A
\]

\[
\text{if } \{\sigma \in \delta \mid 0 < dl(\sigma)\} = \emptyset \text{ then return } \perp
\]

\[
(\epsilon, k) \leftarrow \text{Analysis}(\delta, \hat{\Pi}, \nabla, A)
\]

\[
\nabla \leftarrow \nabla \cup \{\epsilon\}
\]

\[
A \leftarrow A \setminus \{\sigma \in A \mid k < dl(\sigma)\}
\]

\[
dl \leftarrow k
\]

else

\[
\text{return } A^T \cap A(\hat{\Pi})
\]

else if Heuristic decides to evaluate \( &\hat{\Pi}(\hat{e}) \) then

\[
\text{Evaluate } &\hat{\Pi}(\hat{e}) \text{ under } A \text{ and set } \nabla \leftarrow \nabla \cup \Lambda( &\hat{\Pi}(\hat{e}), A)
\]

else

\[
\sigma \leftarrow \text{Select}(\hat{\Pi}, \nabla, A)
\]

\[
dl \leftarrow dl + 1
\]

\[
A \leftarrow A \cup (\sigma)
\]
Restricting to learning functions that are correct for $\Pi$, the following results hold.

**Proposition**

*If for input $\Pi$, $\hat{\Pi}$ and $\nabla$, HEX-CDNL returns*

(i) *an interpretation $\mathbf{A}$, then $\mathbf{A}$ is an answer set of $\hat{\Pi}$ and a solution to $\nabla$;*

(ii) $\bot$, *then $\Pi$ has no compatible set that is a solution to $\nabla$.*

**Proposition**

*HEX-Eval computes all answer sets of $\Pi$.***
Outline

1. Introduction
2. Algorithms with External Behavior Learning
3. Nogood Generation for External Behavior Learning
4. Implementation and Evaluation
5. Conclusion
Concrete Learning Functions

**Idea:** learn that input implies output

**Definition**

The learning function for a general external predicate with input list $&g[\vec{p}]$ in program $\Pi$ under assignment $A$ is defined as

$$\Lambda_g(\&g[\vec{p}], A) = \{A|\vec{p} \cup \{Fe_{\&g[\vec{p}]}(\vec{c})\} \mid (\vec{c}) \in ext(\&g[\vec{p}], A)\}$$

**Example**

$\&diff[p, q](X)$ with $ext(p, A) = \{a, b\}$, $ext(q, A) = \{a, c\}$

Learn: $\{Tp(a), Tp(b), Fp(c), Tq(a), Fq(b), Tq(c), Fe_{\&diff[p,q]}(b)\}$

**Lemma**

*For all assignments $A$, the nogoods $\Lambda_g(\&g[\vec{p}], A)$ are correct wrt. $\Pi$.***
Concrete Learning Functions

Idea: learn that parts of the input imply output

Definition

The learning function for an external predicate \&g with input list \( \vec{p} \) in program \( \Pi \) under assignment \( A \), such that \&g is monotonic in \( p_m \subseteq \vec{p} \), is defined as

\[
\Lambda_m(\&g[\vec{p}], A) = \left\{ \{ Ta \in A |_{p_m} \} \cup A |_{p_n} \cup \{ Fe_{\&g[\vec{p}]}(\vec{c}) \} \mid (\vec{c}) \in ext(\&g[\vec{p}], A) \right\}
\]

Example

\&diff\[p, q\](X) with \( ext(p, A) = \{a, b\} \), \( ext(q, A) = \{a, c\} \), monotonic in \( p \)

Learn: \{Tp(a), Tp(b), Tq(a), Fq(b), Tq(c), Fe_{\&diff\[p, q\]}(b)\}

Lemma

For all assignments \( A \), the nogoods \( \Lambda_m(\&g[\vec{p}], A) \) are correct wrt. \( \Pi \).
Concrete Learning Functions

**Idea:** multiple output tuples exclude each other

**Definition**

The learning function for a functional external predicate \( &g \) with input list \( \vec{p} \) in program \( \Pi \) under assignment \( A \) is defined as

\[
\Lambda_f( &g[\vec{p}], A ) = \{ \{ Te_{&g[\vec{p}]}(\vec{c}), Te_{&g[\vec{p}]}(\vec{c}') \} \mid \vec{c} \neq \vec{c}' \}
\]

**Example**

\( &\text{concat}[ab, c](X) \)
Learn: \( \{ Te_{&\text{concat}[ab, c]}(abc), Te_{&\text{concat}[ab, c]}(ab) \} \)

**Lemma**

*For all assignments \( A \), if \( &g \) is functional, the nogoods \( \Lambda_f( &g[\vec{p}], A ) \) are correct wrt. \( \Pi \).*
Outline

1 Introduction

2 Algorithms with External Behavior Learning

3 Nogood Generation for External Behavior Learning

4 Implementation and Evaluation

5 Conclusion
Implementation

Prototype implementation: DLVHEX
Written in C++
External sources loaded via plugin interface

Technology

Basis: Gringo and CLASP
External Behavior Learning exploits CLASP’s SMT interface
Alternatively: self-made grounder and solver built from scratch
Benchmark Results

Set Partitioning

<table>
<thead>
<tr>
<th>#</th>
<th>all models</th>
<th>first model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DLV w/o EBL</td>
<td>CLASP w/o EBL</td>
</tr>
<tr>
<td>1</td>
<td>0.07</td>
<td>0.08</td>
</tr>
<tr>
<td>5</td>
<td>0.20</td>
<td>0.16</td>
</tr>
<tr>
<td>10</td>
<td>12.98</td>
<td>9.56</td>
</tr>
<tr>
<td>11</td>
<td>38.51</td>
<td>21.73</td>
</tr>
<tr>
<td>12</td>
<td>89.46</td>
<td>49.51</td>
</tr>
<tr>
<td>13</td>
<td>218.49</td>
<td>111.37</td>
</tr>
<tr>
<td>14</td>
<td>—</td>
<td>262.67</td>
</tr>
<tr>
<td>18</td>
<td>—</td>
<td>0.45</td>
</tr>
<tr>
<td>19</td>
<td>—</td>
<td>0.42</td>
</tr>
<tr>
<td>20</td>
<td>—</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Bird-Penguin

<table>
<thead>
<tr>
<th>#</th>
<th>DLV w/o EBL</th>
<th>CLASP w/o EBL</th>
<th>CLASP w EBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.50</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td>5</td>
<td>1.90</td>
<td>1.98</td>
<td>0.59</td>
</tr>
<tr>
<td>6</td>
<td>4.02</td>
<td>4.28</td>
<td>0.25</td>
</tr>
<tr>
<td>7</td>
<td>8.32</td>
<td>7.95</td>
<td>0.60</td>
</tr>
<tr>
<td>8</td>
<td>16.11</td>
<td>16.39</td>
<td>0.29</td>
</tr>
<tr>
<td>9</td>
<td>33.29</td>
<td>34.35</td>
<td>0.35</td>
</tr>
<tr>
<td>10</td>
<td>83.75</td>
<td>94.62</td>
<td>0.42</td>
</tr>
<tr>
<td>11</td>
<td>229.20</td>
<td>230.75</td>
<td>4.45</td>
</tr>
<tr>
<td>12</td>
<td>—</td>
<td>—</td>
<td>1.10</td>
</tr>
<tr>
<td>13</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>14</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>18</td>
<td>—</td>
<td>—</td>
<td>2.70</td>
</tr>
</tbody>
</table>

HEX — Program:

\[\begin{align*}
birds(X) & \leftarrow \text{DL}[\text{Bird}](X) . \\
flies(X) & \leftarrow \text{birds}(X), \text{not neg \_flies}(X) . \\
\text{neg \_flies}(X) & \leftarrow \text{birds}(X), \text{DL}[	ext{Flier} \oplus \text{flies}; \neg \text{Flier}](X) .
\end{align*}\]

Ontology:

\[\begin{align*}
\text{Flier} & \sqsubseteq \neg \text{NonFlier} \\
\text{Penguin} & \sqsubseteq \text{Bird} \\
\text{Penguin} & \sqsubseteq \text{NonFlier}
\end{align*}\]
Benchmark Results

Wine Classification

“A wine is white by default, unless it is derivable that it is red”

<table>
<thead>
<tr>
<th>Inst.</th>
<th>concept completion speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o EBL</td>
</tr>
<tr>
<td>wine.0</td>
<td>25</td>
</tr>
<tr>
<td>wine.1</td>
<td>16</td>
</tr>
<tr>
<td>wine.2</td>
<td>14</td>
</tr>
<tr>
<td>wine.3</td>
<td>4</td>
</tr>
<tr>
<td>wine.4</td>
<td>4</td>
</tr>
<tr>
<td>wine.5</td>
<td>4</td>
</tr>
<tr>
<td>wine.6</td>
<td>4</td>
</tr>
<tr>
<td>wine.7</td>
<td>4</td>
</tr>
<tr>
<td>wine.8</td>
<td>4</td>
</tr>
</tbody>
</table>

Explaining Inconsistency in Multi-context Systems

<table>
<thead>
<tr>
<th>#contexts</th>
<th>DLV w/o EBL</th>
<th>CLASP w/o EBL</th>
<th>CLASP w EBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.07</td>
<td>0.05</td>
<td>0.04</td>
</tr>
<tr>
<td>4</td>
<td>1.04</td>
<td>0.68</td>
<td>0.14</td>
</tr>
<tr>
<td>5</td>
<td>0.23</td>
<td>0.15</td>
<td>0.05</td>
</tr>
<tr>
<td>6</td>
<td>2.63</td>
<td>1.44</td>
<td>0.12</td>
</tr>
<tr>
<td>7</td>
<td>8.71</td>
<td>4.39</td>
<td>0.17</td>
</tr>
</tbody>
</table>
Outline

1 Introduction

2 Algorithms with External Behavior Learning

3 Nogood Generation for External Behavior Learning

4 Implementation and Evaluation

5 Conclusion
Conclusion

External Behavior Learning (EBL)

- Provide novel genuine algorithms for HEX-program evaluation
- Use customizable learning functions
- Learn additional nogoods from external source evaluations
- Uninformed vs. informed learning

Implementation and Evaluation

- Prototype implementation based on Gringo and CLASP
- Experiments show significant improvements by EBL

Future Work

- Identify further properties for informed learning
- Language for writing user-defined learning functions
References

Conflict-driven disjunctive answer set solving.

A uniform integration of higher-order reasoning and external evaluations in answer-set programming.
URL:

Constraint answer set solving.
In *ICLP*, pages 235–249.

Classical negation in logic programs and disjunctive databases.
URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.7150.