Conflict-driven ASP Solving with External Source Access

Thomas Eiter, Michael Fink, Thomas Krennwallner, Christoph Redl

redl@kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

kbs Knowledge-Based Systems Group

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

Outline

HEX-Programs

HEX-programs extend ordinary ASP programs by external sources

Definition (HEX-programs)

A HEX-program consists of rules of form

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n,$$

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

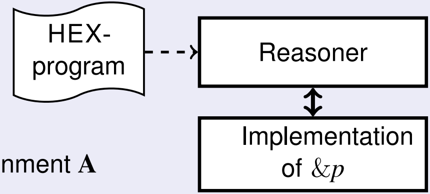
t_j ...terms

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 $\mathbf{A}$

iff $f_{\&p}(\mathbf{A}, q_1, \ldots, q_k, t_1, \ldots, t_l) = 1$.



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:

$$addr(\texttt{http://.../data1.rdf}).$$
$$addr(\texttt{http://.../data2.rdf}).$$
$$bel(X,Y) \leftarrow addr(U), \&rdf[U](X,Y,Z).$$

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:

$$addr(\texttt{http://.../data1.rdf}).$$
$$addr(\texttt{http://.../data2.rdf}).$$
$$bel(X,Y) \leftarrow addr(U), \&rdf[U](X,Y,Z).$$

$\&diff$

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

$$\begin{aligned} dom(X) &\leftarrow \#int(X). \\ nsel(X) &\leftarrow dom(X), \&diff[dom, sel](X). \\ sel(X) &\leftarrow dom(X), \&diff[dom, nsel](X). \\ &\leftarrow sel(X1), sel(X2), sel(X3), X1 \neq X2, X1 \neq X3, X2 \neq X3. \end{aligned}$$

Current Evaluation Method

Evaluating Program Π

1. Replace external atoms $\&g[\vec{p}](\vec{c})$ by ordinary ones $e_{\&g[\vec{p}]}(\vec{c})$ 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 $\mathbf{A}$ is subset-minimal under all compatible sets

Definition (Compatible Set)

A compatible set of a program Π is an assignment $\mathbf{A}$

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

(ii) $f_{\&g}(\mathbf{A}, \vec{p}, \vec{c}) = 1$ iff $\mathbf{T}e_{\&g[\vec{p}]}(\vec{c}) \in \mathbf{A}$ for all external atoms $\&g[\vec{p}](\vec{c})$ in Π

Definition (Answer Set)

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

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

(ii) $S \not\subset \{\mathbf{T}a \mid a \in A(\Pi)\} \cap \mathbf{A}$ for every compatible set $\mathbf{A}$ of Π.

Current Evaluation Method

Translation Approach

HEX-Program Π:

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

Guessing program $\hat{\Pi}$:

$$p(c_1).\ dom(c_1).\ dom(c_2).\ dom(c_3).$$
$$p(X) \leftarrow dom(X), e_{\&empty[p]}(X).$$
$$e_{\&empty[p]}(X) \vee \neg e_{\&empty[p]}(X) \leftarrow dom(X).$$

8 candidates, e.g.:
$\{\mathbf{T}p(c_1), \mathbf{T}p(c_2), \mathbf{T}dom(c_1), \mathbf{T}dom(c_2), \mathbf{T}dom(c_3),$
$\mathbf{F}e_{\&empty[p]}(c_1), \mathbf{T}e_{\&empty[p]}(c_2), \mathbf{F}e_{\&empty[p]}(c_3)\}$
Compatibility check: passed $\Rightarrow$ compatible set

Outline

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 $\hat{\Pi}$ 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* $\mathbf{T}a$ or $\mathbf{F}a$, where a is an atom.

Definition (Nogoods)

A nogood is a set of *signed literals*.

Definition (Solution to Nogoods)

An assignment $\mathbf{A}$ is a solution to a set of nogoods Δ iff $\delta \not\subseteq \mathbf{A}$ for all $\delta \in \Delta$.

Basic Definitions

Definition (Correct Nogoods)

A nogood δ is correct wrt. program Π, if all compatible sets of Π are solutions to δ.

Definition (Learning Function)

A learning function for Π is a mapping $\Lambda : \mathcal{E} \times 2^{\mathcal{D}} \mapsto 2^{2^{\mathcal{D}}}$
$\mathcal{E}$ ... set of all external predicates with input list in Π
$\mathcal{D}$ ... set of all signed literals

Definition (Correct Learning Functions)

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

Algorithms

Algorithm: HEX-Eval

Input: A HEX-program Π
Output: All answer sets of Π

$\hat{\Pi} \leftarrow \Pi$ with all external atoms $\&[\vec{p}](\vec{c})$ replaced by $e_{\&[\vec{p}]}(\vec{c})$
Add guessing rules for all replacement atoms to $\hat{\Pi}$
$\nabla \leftarrow \emptyset$ /* set of dynamic nogoods */
$\Gamma \leftarrow \emptyset$ /* set of all compatible sets */
while $\mathbf{C} \neq \bot$ **do**
  $\mathbf{C} \leftarrow \bot$
  $inconsistent \leftarrow false$
  while $\mathbf{C} = \bot$ *and* $inconsistent = false$ **do**
    $\mathbf{A} \leftarrow$ HEX-CDNL$(\Pi, \hat{\Pi}, \nabla)$
    if $\mathbf{A} = \bot$ **then**
      $inconsistent \leftarrow true$
    else
      $compatible \leftarrow true$
      for *all external atoms* $\&[\vec{p}]$ *in* Π **do**
        Evaluate $\&[\vec{p}]$ under $\mathbf{A}$
        $\nabla \leftarrow \nabla \cup \Lambda(\&[\vec{p}], \mathbf{A})$
        Let $\mathbf{A}^{\&[\vec{p}](\vec{c})} = 1 \Leftrightarrow \mathbf{T}e_{\&[\vec{p}](\vec{c})} \in \mathbf{A}$
        if $\exists \vec{c}$: $f_{\&g}(\mathbf{A}, \vec{p}, \vec{c}) \neq \mathbf{A}^{\&[\vec{p}](\vec{c})}$ **then**
          Add $\mathbf{A}$ to ∇
          Set $compatible \leftarrow false$
      if $compatible$ **then** $\mathbf{C} \leftarrow \mathbf{A}$
  if $inconsistent = false$ **then**
    /* $\mathbf{C}$ is a compatible set of Π */
    $\nabla \leftarrow \nabla \cup \{\mathbf{C}\}$ and $\Gamma \leftarrow \Gamma \cup \{\mathbf{C}\}$

Output $\{\{\mathbf{T}a \in \mathbf{A} \mid a \in A(\Pi)\} \mid \mathbf{A} \in \Gamma\}$ which are subset-minimal

Algorithm: HEX-CDNL

Input: A program Π, its guessing program $\hat{\Pi}$, a set of correct nogoods ∇ of Π
Output: An answer set of $\hat{\Pi}$ (candidate for a compatible set of Π) which is a solution to all nogoods $d \in \nabla$, or $\bot$ if none exists

$\mathbf{A} \leftarrow \emptyset$ /* assignment over $A(\hat{\Pi}) \cup BA(\hat{\Pi}) \cup BA(sh(\hat{\Pi}))$ */
$dl \leftarrow 0$ /* decision level */
while *true* **do**
  $(\mathbf{A}, \nabla) \leftarrow$ Propagation$(\hat{\Pi}, \nabla, \mathbf{A})$
  if $\delta \subseteq \mathbf{A}$ *for some* $\delta \in \Delta_{\hat{\Pi}} \cup \Theta_{sh(\hat{\Pi})} \cup \nabla$ **then**
    if $dl = 0$ **then return** $\bot$
    $(\epsilon, k) \leftarrow$ Analysis$(\delta, \hat{\Pi}, \nabla, \mathbf{A})$
    $\nabla \leftarrow \nabla \cup \{\epsilon\}$
    $\mathbf{A} \leftarrow \mathbf{A} \setminus \{\sigma \in \mathbf{A} \mid k < dl(\sigma)\}$
    $dl \leftarrow k$
  else if $\mathbf{A}^{\mathbf{T}} \cup \mathbf{A}^{\mathbf{F}} = A(\hat{\Pi}) \cup BA(\hat{\Pi}) \cup BA(sh(\hat{\Pi}))$ **then**
    $U \leftarrow$ UnfoundedSet$(\hat{\Pi}, \mathbf{A})$
    if $U \neq \emptyset$ **then**
      let $\delta \in \lambda_{\hat{\Pi}}(U)$ such that $\delta \subseteq \mathbf{A}$
      if $\{\sigma \in \delta \mid 0 < dl(\sigma)\} = \emptyset$ **then return** $\bot$
      $(\epsilon, k) \leftarrow$ Analysis$(\delta, \hat{\Pi}, \nabla, \mathbf{A})$
      $\nabla \leftarrow \nabla \cup \{\epsilon\}$
      $\mathbf{A} \leftarrow \mathbf{A} \setminus \{\sigma \in \mathbf{A} \mid k < dl(\sigma)\}$
      $dl \leftarrow k$
    else
      return $\mathbf{A}^{\mathbf{T}} \cap A(\hat{\Pi})$
  else if *Heuristic decides to evaluate* $\&[\vec{p}]$ **then**
    Evaluate $\&[\vec{p}]$ under $\mathbf{A}$ and set $\nabla \leftarrow \nabla \cup \Lambda(\&[\vec{p}], \mathbf{A})$
  else
    $\sigma \leftarrow$ Select$(\hat{\Pi}, \nabla, \mathbf{A})$
    $dl \leftarrow dl + 1$
    $\mathbf{A} \leftarrow \mathbf{A} \circ (\sigma)$

Algorithms

Restricting to learning functions that are correct for Π, the following results hold.

Proposition

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

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

(ii) $\perp$*, then* Π *has no compatible set that is a solution to* ∇*.*

Proposition

HEX-*Eval computes all answer sets of* Π*.*

Outline

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 Π under assignment $\mathbf{A}$ is defined as

$$\Lambda_g(\&g[\vec{p}], \mathbf{A}) = \left\{ \mathbf{A}|_{\vec{p}} \cup \{\mathbf{F}e_{\&g[\vec{p}]}(\vec{c})\} \mid (\vec{c}) \in ext(\&g[\vec{p}], \mathbf{A}) \right\}$$

Example

$\&diff[p,q](X)$ with $ext(p,\mathbf{A}) = \{a,b\}$, $ext(q,\mathbf{A}) = \{a,c\}$
Learn: $\{\mathbf{T}p(a), \mathbf{T}p(b), \mathbf{F}p(c), \mathbf{T}q(a), \mathbf{F}q(b), \mathbf{T}q(c), \mathbf{F}e_{\&diff[p,q]}(b)\}$

Lemma

For all assignments $\mathbf{A}$, the nogoods $\Lambda_g(\&g[\vec{p}], \mathbf{A})$ are correct wrt. Π.

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 Π under assignment $\mathbf{A}$, such that $\&g$ is monotonic in $\vec{p_m} \subseteq \vec{p}$, is defined as

$$\Lambda_m(\&g[\vec{p}], \mathbf{A}) = \left\{ \{\mathbf{T}a \in \mathbf{A}|_{\vec{p_m}}\} \cup \mathbf{A}|_{\vec{p_n}} \cup \{\mathbf{F}e_{\&g[\vec{p}]}(\vec{c})\} \mid (\vec{c}) \in ext(\&g[\vec{p}], \mathbf{A}) \right\}$$

Example

$\&diff[p,q](X)$ with $ext(p, \mathbf{A}) = \{a, b\}$, $ext(q, \mathbf{A}) = \{a, c\}$, monotonic in p
Learn: $\{\mathbf{T}p(a), \mathbf{T}p(b), \mathbf{T}q(a), \mathbf{F}q(b), \mathbf{T}q(c), \mathbf{F}e_{\&diff[p,q]}(b)\}$

Lemma

For all assignments $\mathbf{A}$, the nogoods $\Lambda_m(\&g[\vec{p}], \mathbf{A})$ are correct wrt. Π.

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 Π under assignment $\mathbf{A}$ is defined as

$$\Lambda_f(\&g[\vec{p}], \mathbf{A}) = \left\{ \{\mathbf{T}e_{\&g[\vec{p}]}(\vec{c}), \mathbf{T}e_{\&g[\vec{p}]}(\vec{c'})\} \mid \vec{c} \neq \vec{c'} \right\}$$

Example

$\&concat[ab, c](X)$
Learn: $\{\mathbf{T}e_{\&concat[ab,c]}(abc), \mathbf{T}e_{\&concat[ab,c]}(ab)\}$

Lemma

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

Outline

Implementation

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

#	all models			first model		
	DLV	CLASP w/o EBL	CLASP w EBL	DLV	CLASP w/o EBL	CLASP w EBL
1	0.07	0.08	0.07	0.08	0.07	0.07
5	0.20	0.16	0.10	0.08	0.08	0.07
10	12.98	9.56	0.17	0.56	0.28	0.07
11	38.51	21.73	0.19	0.93	0.63	0.08
12	89.46	49.51	0.19	1.69	1.13	0.08
13	218.49	111.37	0.20	3.53	2.31	0.10
14	—	262.67	0.28	8.76	3.69	0.10
⋮	—	—	⋮	⋮	⋮	⋮
18	—	—	0.45	128.79	62.58	0.12
19	—	—	0.42	—	95.39	0.10
20	—	—	0.54	—	91.16	0.11

Bird-Penguin

#	DLV	CLASP w/o EBL	CLASP w EBL
1	0.50	0.15	0.14
5	1.90	1.98	0.59
6	4.02	4.28	0.25
7	8.32	7.95	0.60
8	16.11	16.39	0.29
9	33.29	34.35	0.35
10	83.75	94.62	0.42
11	229.20	230.75	4.45
12	—	—	1.10
⋮	—	—	⋮
20	—	—	2.70

HEX − Program :

$$
\begin{aligned}
birds(X) &\leftarrow DL[Bird](X).\\
flies(X) &\leftarrow birds(X), \text{not } neg\_flies(X).\\
neg\_flies(X) &\leftarrow birds(X), DL[Flier \uplus flies; \neg Flier](X).
\end{aligned}
$$

Ontology :

$$
\begin{aligned}
Flier &\sqsubseteq \neg NonFlier\\
Penguin &\sqsubseteq Bird\\
Penguin &\sqsubseteq NonFlier
\end{aligned}
$$

Benchmark Results

Wine Classification

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

Inst.	concept completion		speedup	
	w/o EBL	w EBL	max	avg
wine_0	25	31	33.02	6.93
wine_1	16	25	16.05	5.78
wine_2	14	22	11.82	4.27
wine_3	4	17	10.09	4.02
wine_4	4	17	6.83	2.87
wine_5	4	16	5.22	2.34
wine_6	4	13	2.83	1.52
wine_7	4	12	1.81	1.14
wine_8	4	4	1.88	1.08

Explaining Inconsistency in Multi-context Systems

#contexts	DLV	CLASP w/o EBL	CLASP w EBL
3	0.07	0.05	0.04
4	1.04	0.68	0.14
5	0.23	0.15	0.05
6	2.63	1.44	0.12
7	8.71	4.39	0.17

Outline

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

Drescher, C., Gebser, M., Grote, T., Kaufmann, B., König, A., Ostrowski, M., and Schaub, T. (2008).
Conflict-driven disjunctive answer set solving.
In *KR'08*, pages 422–432. AAAI Press.

Eiter, T., Ianni, G., Schindlauer, R., and Tompits, H. (2005).
A uniform integration of higher-order reasoning and external evaluations in answer-set programming.
In *In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05*, pages 90–96. Professional Book.
URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.8944.

Gebser, M., Ostrowski, M., and Schaub, T. (2009).
Constraint answer set solving.
In *ICLP*, pages 235–249.

Gelfond, M. and Lifschitz, V. (1991).
Classical negation in logic programs and disjunctive databases.
New Generation Computing, 9:365–385.
URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.7150.