Exploiting Unfounded Sets for HEX-Program Evaluation

Thomas Eiter, Michael Fink, Thomas Krennwallner,
Christoph Redl, Peter Schüller

redl@kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

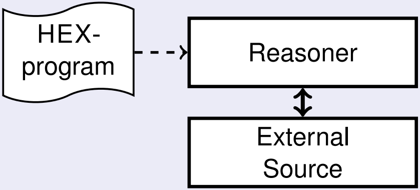
Knowledge-Based Systems Group

September 27, 2012

Motivation

HEX-Programs

- Extend ASP by external sources
- Scalability problems due to minimality checking



Contribution

- Exploit unfounded sets for minimality checking
- Search for unfounded sets encoded as separate search problem
- Much better scalability

Outline

1 Introduction

2 Answer Set Computation

3 Optimization and Learning

4 Implementation and Evaluation

5 Conclusion

Outline

1 Introduction

2 Answer Set Computation

3 Optimization and 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

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

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

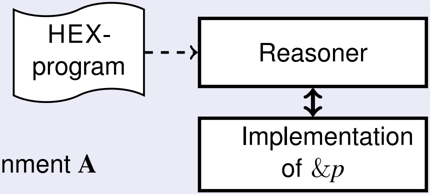
t_j ...terms

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:

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

Semantics of HEX-Programs

Definition (FLP-Reduct [Faber et al., 2004])

For an interpretation $\mathbf{A}$ over a program Π, the FLP-reduct $f\Pi^{\mathbf{A}}$ of Π wrt. $\mathbf{A}$ is the set $\{r \in \Pi \mid \mathbf{A} \models b, \text{ for all } b \in B(r)\}$ of all rules whose body is satisfied under $\mathbf{A}$.

Definition (Answer Set)

An interpretation $\mathbf{A}$ is an answer set of program Π iff it is a subset-minimal model of the FLP reduct $f\Pi^{\mathbf{A}}$.

Example

Program Π: $dom(a).dom(b).$

$$p(a) \leftarrow dom(a), \&g[p](a).$$
$$p(b) \leftarrow dom(b), \&g[p](b).$$

where $\&g$ implements the following mapping:

$$\emptyset \mapsto \{b\}; \{a\} \mapsto \{a\}; \{b\} \mapsto \emptyset; \{a,b\} \mapsto \{a,b\}$$

$\mathbf{A} = \{\mathbf{T}dom(a), \mathbf{T}dom(b), \mathbf{T}p(a), \mathbf{F}p(b)\}$ is a model but no subset-minimal model of

$$f\Pi^{\mathbf{A}} = \{dom(a); dom(b); p(a) \leftarrow dom(a), \&g[p](a)\}$$

Outline

1 Introduction

2 Answer Set Computation

3 Optimization and Learning

4 Implementation and Evaluation

5 Conclusion

Answer Set Computation

2-Step Algorithm

1. Compute a compatible set (=answer set candidate) [Eiter et al., 2012]
2. Check minimality

Answer Set Computation

2-Step Algorithm

1. Compute a compatible set (=answer set candidate) [Eiter et al., 2012]
2. Check minimality

The Naive Minimality Check

1. Let $\mathbf{A}$ be a compatible set
2. Compute $f\Pi^{\mathbf{A}}$
3. Check if there is a smaller model than $\mathbf{A}$

Problem: Reduct has usually many models
Note: In practice, smaller models are rarely found

Answer Set Computation

2-Step Algorithm

1. Compute a compatible set (=answer set candidate) [Eiter et al., 2012]
2. Check minimality

The Naive Minimality Check

1. Let $\mathbf{A}$ be a compatible set
2. Compute $f\Pi^{\mathbf{A}}$
3. Check if there is a smaller model than $\mathbf{A}$

Problem: Reduct has usually many models
Note: In practice, smaller models are rarely found

Complexity

Minimality check is Co-NP-complete, lifting the overall answer set existence problem to Π_2^P
(in presence of disjunctions and/or nonmonotonic external atoms)

Using Unfounded Sets [Faber, 2005]

Definition (Unfounded Set)

A set of atoms X is an unfounded set of Π wrt. (partial) assignment $\mathbf{A}$, iff for all $a \in X$ and all $r \in \Pi$ with $a \in H(r)$ at least one of the following holds:

1. $\mathbf{A} \not\models B(r)$
2. $\mathbf{A} \dot\cup \neg.X \not\models B(r)$
3. $\mathbf{A} \models h$ for some $h \in H(r) \setminus X$

(where $\mathbf{A} \dot\cup \neg.X = \{\mathbf{T}a \in \mathbf{A} \mid a \notin X\} \cup \{\mathbf{F}a \in \mathbf{A}\} \cup \{\mathbf{F}a \mid a \in X\}$)

Definition (Unfounded-free Assignments)

An assignment $\mathbf{A}$ is unfounded-free wrt. program Π, iff there is no unfounded set X of Π wrt. $\mathbf{A}$ such that $\mathbf{T}a \in \mathbf{A}$ for some $a \in X$.

Theorem

A model $\mathbf{A}$ of a program Π is is an answer set iff it is unfounded-free.

Using Unfounded Sets

Encode the search for unfounded sets as SAT instance

Unfounded Set Search Problem

Nogood Set $\Gamma_{\Pi}^{\mathbf{A}} = N_{\Pi}^{\mathbf{A}} \cup O_{\Pi}^{\mathbf{A}}$ over atoms $A(\hat{\Pi}) \cup \{h_r, l_r \mid r \in \Pi\}$ consisting of a necessary part $N_{\Pi}^{\mathbf{A}}$ and an optimization part $O_{\Pi}^{\mathbf{A}}$

- $N_{\Pi}^{\mathbf{A}} = \{\{\mathbf{F}a \mid \mathbf{T}a \in \mathbf{A}\}\} \cup \left(\bigcup_{r \in \Pi} R_r^{\mathbf{A}}\right)$
- $R_{r,\mathbf{A}} = H_{r,\mathbf{A}} \cup C_{r,\mathbf{A}}$, where
- $H_{r,\mathbf{A}} = \{\{\mathbf{T}h_r\} \cup \{\mathbf{F}h \mid h \in H(r)\}\} \cup \{\{\mathbf{F}h_r, \mathbf{T}h\} \mid h \in H(r)\}$
- $$C_{r,\mathbf{A}} = \begin{cases} \{\{\mathbf{T}h_r\} \cup \\ \quad \{\mathbf{F}a \mid a \in B_o^+(r), \mathbf{A} \models a\} \cup \{\mathbf{t}a \mid a \in B_e(\hat{r})\} \cup \\ \quad \{\mathbf{T}h \mid h \in H(r), \mathbf{A} \models h\}\} & \textit{if } \mathbf{A} \models B(r), \\ \{\} & \textit{otherwise} \end{cases}$$

Intuition: Solutions of $\Gamma_{\Pi}^{\mathbf{A}}$ correspond to potential unfounded sets of Π wrt. $\mathbf{A}$

Using Unfounded Sets

Each unfounded set corresponds to a solution of $\Gamma_{\Pi}^{\mathbf{A}}$

Definition (Induced Assignment of an Unfounded Set)

Let U be an unfounded set of a program Π wrt. assignment $\mathbf{A}$.
The assignment induced by U, denoted $I(U, \Gamma_{\Pi}^{\mathbf{A}})$, is

$$I(U, \Gamma_{\Pi}^{\mathbf{A}}) = I'(U, \Gamma_{\Pi}^{\mathbf{A}}) \cup \{\mathbf{F}a \mid a \in A(\Gamma_{\Pi}^{\mathbf{A}}), \mathbf{T}a \notin I'(U, \Gamma_{\Pi}^{\mathbf{A}})\}, \text{ where}$$

$$\begin{aligned} I'(U, \Gamma_{\Pi}^{\mathbf{A}}) = \quad & \{\mathbf{T}a \mid a \in U\} \cup \{\mathbf{T}h_r \mid r \in \Pi, H(r) \cap U \neq \emptyset\} \cup \\ & \{\mathbf{T}e_{\&g[\vec{p}]}(\vec{c}) \mid e_{\&g[\vec{p}]}(\vec{c}) \in A(\hat{\Pi}), \mathbf{A} \dot{\cup} \neg.U \models \&g[\vec{p}](\vec{c})\}. \end{aligned}$$

Proposition

Let U be an unfounded set of a program Π wrt. assignment $\mathbf{A}$ such that $\mathbf{A}^{\mathbf{T}} \cap U \neq \emptyset$. Then $I(U, \Gamma_{\Pi}^{\mathbf{A}})$ is a solution to $\Gamma_{\Pi}^{\mathbf{A}}$.

Using Unfounded Sets

Not each solution of $\Gamma_{\Pi}^{\mathbf{A}}$ corresponds to an unfounded set, but ...

Proposition

Let S be a solution to $\Gamma_{\Pi}^{\mathbf{A}}$ such that

(a) $\mathbf{T}e_{\&g[\vec{p}]}(\vec{c}) \in S$ and $\mathbf{A} \not\models \&g[\vec{p}](\vec{c})$ implies $\mathbf{A} \dot{\cup} \neg.U \models \&g[\vec{p}](\vec{c})$; and

(b) $\mathbf{F}e_{\&g[\vec{p}]}(\vec{c}) \in S$ and $\mathbf{A} \models \&g[\vec{p}](\vec{c})$ implies $\mathbf{A} \dot{\cup} \neg.U \not\models \&g[\vec{p}](\vec{c})$

where $U = \{a \mid a \in A(\Pi), \mathbf{T}a \in S\}$. Then U is an unfounded set of Π wrt. $\mathbf{A}$.

Our Approach

1. Compute a solution S of $\Gamma_{\Pi}^{\mathbf{A}}$
2. Check if truth value of external atom replacement $e_{\&g[\vec{p}]}(\vec{c})$ in S is equal to truth value of $\&g[\vec{p}](\vec{c})$ under $\mathbf{A} \dot{\cup} \neg.U$
3. If yes: S represents an unfounded set
4. If no: continue with next solution of $\Gamma_{\Pi}^{\mathbf{A}}$

Outline

1 Introduction

2 Answer Set Computation

3 Optimization and Learning

4 Implementation and Evaluation

5 Conclusion

Optimization and Learning

Optimization

Generate additional nogoods $O_{\Pi}^{\mathbf{A}}$ to prune search space

- Restrict search to atoms which are true in $\mathbf{A}$
- Try to avoid changes of truth values of external atoms

Learning

- Nogood exchange: Search for models $\leftrightarrow$ UFS search
- Learn nogoods from detected unfounded sets

Outline

1 Introduction

2 Answer Set Computation

3 Optimization and Learning

4 Implementation and Evaluation

5 Conclusion

Implementation

Implementation

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

Technology

- Basis: Gringo and CLASP
- CLASP serves also as SAT solver for UFS search
- Alternatively: self-made grounder and solver built from scatch

Benchmark Results

	n	5	6	7	8	9	10	11	12	13	...	20
all AS	explicit	10.9	94.3	—	—	—	—	—	—	—	—	—
	+EBL	4.3	34.8	266.1	—	—	—	—	—	—	—	—
	UFS	0.2	0.3	0.8	1.8	4.5	11.9	32.4	92.1	273.9	—	—
	+EBL	0.1	0.1	0.2	0.2	0.3	0.4	0.6	0.8	1.2	...	11.1
first AS	explicit	0.7	4.3	26.1	163.1	—	—	—	—	—	—	—
	+EBL	0.8	4.9	31.1	192.0	—	—	—	—	—	—	—
	UFS	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	...	0.5
	+EBL	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	...	0.3

Figure: Set Partitioning

#args	all answer sets		first answer set	
	Explicit	UFS	Explicit	UFS
5	1.47	1.13	0.70	0.62
6	4.57	2.90	1.52	1.27
7	19.99	10.50	3.64	2.77
8	80.63	39.01	9.46	6.94
9	142.95	80.66	30.12	20.97
10	240.46	122.81	107.14	63.50

Figure: Argumentation (plain)

Benchmark Results

#contexts	(no answer sets)				
	explicit check		UFS check		
	plain	+EBL	plain	+EBL	+UFL
3	8.61	4.68	7.31	2.44	0.50
4	86.55	48.53	80.31	25.98	1.89
5	188.05	142.61	188.10	94.45	4.62
6	209.34	155.81	207.14	152.32	14.39
7	263.98	227.99	264.00	218.94	49.42
8	293.64	209.41	286.38	189.86	124.23
9	—	281.98	—	260.01	190.56
10	—	274.76	—	247.67	219.83

Figure: Consistent MCSs

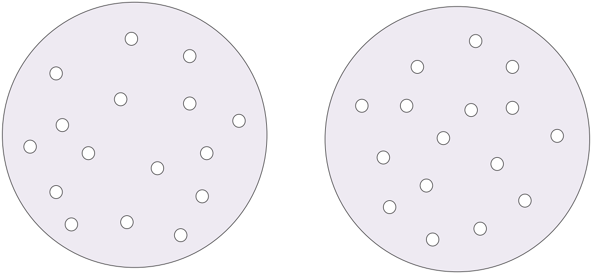
#contexts	enumerating all answer sets					finding first answer set				
	explicit check		UFS check			explicit check		UFS check		
	plain	+EBL	plain	+EBL	+UFL	plain	+EBL	plain	+EBL	+UFL
3	9.08	6.11	6.29	2.77	0.85	4.01	2.53	3.41	1.31	0.57
4	89.71	36.28	80.81	12.63	5.27	53.59	16.99	49.56	6.09	1.07
5	270.10	234.98	268.90	174.23	18.87	208.62	93.29	224.01	32.85	3.90
6	236.02	203.13	235.55	179.24	65.49	201.84	200.06	201.24	166.04	28.34
7	276.94	241.27	267.82	231.08	208.47	241.09	78.72	240.72	66.56	16.41
8	286.61	153.41	282.96	116.89	69.69	201.10	108.29	210.61	103.11	30.98
9	—	208.92	—	191.46	175.26	240.75	112.08	229.14	76.56	44.73
10	—	—	—	289.87	289.95	—	125.18	—	75.24	27.05

Figure: Inconsistent MCSs

Benchmark Results

Interesting Observations

- Search space for UFS check potentially smaller than for explicit check
- Even if they have the same size the UFS check is mostly faster:
  - Less overhead (SAT vs. ASP instance)
  - Easier for the solver to jump from one candidate to the next one



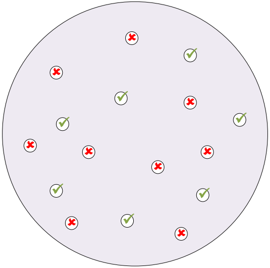
candidate smaller models of the reduct

candidate unfounded sets

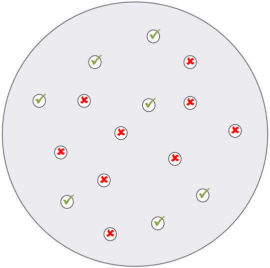
Benchmark Results

Interesting Observations

- Search space for UFS check potentially smaller than for explicit check
- Even if they have the same size the UFS check is mostly faster:
  - Less overhead (SAT vs. ASP instance)
  - Easier for the solver to jump from one candidate to the next one



candidate smaller models of the reduct

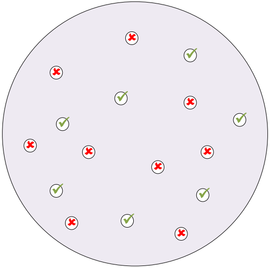


candidate unfounded sets

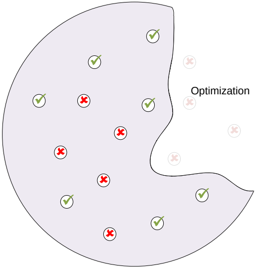
Benchmark Results

Interesting Observations

- Search space for UFS check potentially smaller than for explicit check
- Even if they have the same size the UFS check is mostly faster:
  - Less overhead (SAT vs. ASP instance)
  - Easier for the solver to jump from one candidate to the next one



candidate smaller models of the reduct

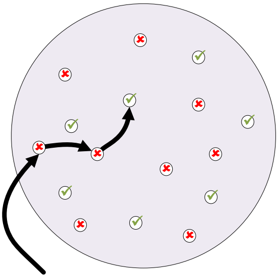


candidate unfounded sets

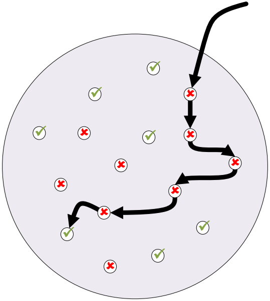
Benchmark Results

Interesting Observations

- Search space for UFS check potentially smaller than for explicit check
- Even if they have the same size the UFS check is mostly faster:
  - Less overhead (SAT vs. ASP instance)
  - Easier for the solver to jump from one candidate to the next one



candidate smaller models of the reduct

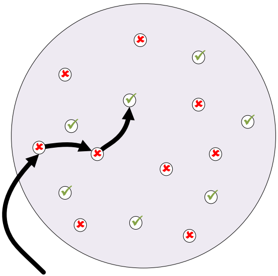


candidate unfounded sets

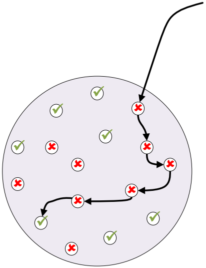
Benchmark Results

Interesting Observations

- Search space for UFS check potentially smaller than for explicit check
- Even if they have the same size the UFS check is mostly faster:
  - Less overhead (SAT vs. ASP instance)
  - Easier for the solver to jump from one candidate to the next one



candidate smaller models of the reduct

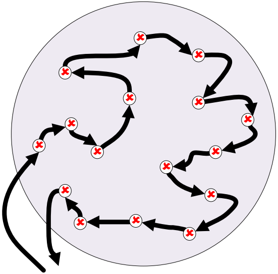


candidate unfounded sets

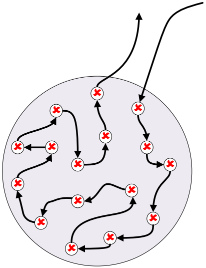
Benchmark Results

Interesting Observations

- Search space for UFS check potentially smaller than for explicit check
- Even if they have the same size the UFS check is mostly faster:
  - Less overhead (SAT vs. ASP instance)
  - Easier for the solver to jump from one candidate to the next one



candidate smaller models of the reduct



candidate unfounded sets

Outline

1 Introduction

2 Answer Set Computation

3 Optimization and Learning

4 Implementation and Evaluation

5 Conclusion

Conclusion

Evaluating HEX-Programs

- Compute a compatible set, then check if it is unfounded-free
- Encoded as nogood set consisting of a necessary and optimization part
- Unfounded sets allow for learning nogoods

Implementation and Evaluation

- Prototype implementation based on Gringo and CLASP
- Experiments show significant improvements by UFS-based minimality check
- Further speedup by optimization part and learning

Future Work

- Unfounded set check over partial interpretations
- Decision criterion for necessity of UFS-check
- Further restriction of search space to the relevant part

References

Eiter, T., Fink, M., Krennwallner, T., and Redl, C. (2012).
Conflict-driven ASP solving with external sources.
Theory and Practice of Logic Programming: Special Issue ICLP.
To appear.

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.

Faber, W. (2005).
Unfounded sets for disjunctive logic programs with arbitrary aggregates.
In *In Logic Programming and Nonmonotonic Reasoning, 8th International Conference (LPNMR'05), 2005*, pages 40–52. Springer Verlag.

Faber, W., Leone, N., and Pfeifer, G. (2004).
Recursive aggregates in disjunctive logic programs: Semantics and complexity.
In *In Proceedings of European Conference on Logics in Artificial Intelligence (JELIA*, pages 200–212. Springer.