FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Knowledge-Based Systems Group

31st AAAI Conference on Artificial Intelligence (AAAI-17), San Francisco, California, USA, February 4–9, 2017

Efficient Evaluation of Answer Set Programs with External Sources Based on External Source Inlining

Christoph Redl
redl@tuwien.ac.at

1. Motivation

HEX-programs extend ASP by **external sources**:

- Rule bodies may contain external atoms of the form
$$\&p[q_1, \dots, q_k](t_1, \dots, t_l),$$
where
p ... external predicate name,
q_i ... predicate names or constants: $\tau(\&p, i) \in \{\mathbf{pred}, \mathbf{const}\}$,
t_j ... terms.

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

Example: Set Partitioning

$$P = \left\{ \begin{array}{l} d(a_1). \dots . d(a_n). \\ r_1 : p(X) \leftarrow d(X), \&diff[d, q](X). \\ r_2 : q(X) \leftarrow d(X), \&diff[d, p](X). \end{array} \right\}$$

Problem:

- Calling external sources during solving is expensive.
- This is in particular the case for cyclic external sources.
- Reasons include both algorithmic and technical overhead (e.g. caching effects).

Our solution:

- Compile the HEX-program to an ordinary ASP-program by inlining external sources.
- To this end, we employ support sets (i.e., sets of input atoms which make the external atom true).
- Although inlining leads to an exponential blowup in the worst case, it is known that for certain types of external sources this is not the case!

2. Support Sets

- Let $e = \&g[\vec{y}](\vec{x})$ be an external atom in a program P.
- **Intuition:**
A positive (resp. negative) support set is a set of positive or negated input atoms of e, whose satisfaction implies that e is true (resp. false).
- **Formally:**
A support set for e is a consistent set $S_\sigma = S_\sigma^+ \cup S_\sigma^-$ with $\sigma \in \{\mathbf{T}, \mathbf{F}\}$, $S_\sigma^+ \subseteq HB_{\mathcal{C}}(\mathcal{P})$, and $S_\sigma^- \subseteq \neg HB_{\mathcal{C}}(\mathcal{P})$ s.t. $A \supseteq S_\sigma^+$ and $A \cap \neg S_\sigma^- = \emptyset$ implies $A \models e$ if $\sigma = \mathbf{T}$ and $A \not\models e$ if $\sigma = \mathbf{F}$ for all assignments A.

Example: Set Partitioning (cont'd)

A positive support set of $\&diff[d, q](b)$ in P is $S_\mathbf{T} = \{\mathbf{T}d(b), \mathbf{F}q(b)\}$ since for all A: $A \models d(b)$ and $A \not\models q(b)$ implies $A \models \&diff[d, q](b)$.

- **Important concept: complete families of support sets:**
A family (=set) of support sets $\mathcal{S}_\sigma$ for external atom e is complete, if it contains all possibilities how to satisfy resp. falsify the external atom.
Formally:
A positive resp. negative family of support sets $\mathcal{S}_\sigma$ with $\sigma \in \{\mathbf{T}, \mathbf{F}\}$ for external atom e is a set of positive resp. negative support sets of e; $\mathcal{S}_\sigma$ is complete if for each assignment A with $A \models e$ resp. $A \not\models e$ there is an $S_\sigma \in \mathcal{S}_\sigma$ s.t. $A \supseteq S_\sigma^+$ and $A \cap \neg S_\sigma^- = \emptyset$.

3. Inlining of External Atoms – Our Encoding

- Due to cyclic and nonmonotonic external atoms, inlining is not trivial (the formalism is on the second level of the polynomial hierarchy).
- Our encoding is based on the saturation technique.

A positive external atom e in a program P with a complete family of positive support sets $\mathcal{S}_\mathbf{T}$ is inlined as follows (negative ones are handled similarly):

$$P_{[e]} = \{x_e \leftarrow S_\mathbf{T}^+ \cup \{\bar{a} \mid \neg a \in S_\mathbf{T}^-\} \mid S_\mathbf{T} \in \mathcal{S}_\mathbf{T}\} \tag{1}$$
$$\cup \left\{ \begin{array}{l} \bar{a} \leftarrow \mathbf{not}\, a; \bar{a} \leftarrow x_e \\ a \vee \bar{a} \leftarrow \mathbf{not}\, \bar{x}_e \end{array} \;\middle|\; a \in I(e, P) \right\} \tag{2}$$
$$\cup \{\bar{x}_e \leftarrow \mathbf{not}\, x_e\} \tag{3}$$
$$\cup P|_{e \rightarrow x_e} \tag{4}$$

where $\bar{a}$ is a new atom for each a, x_e and $\bar{x}_e$ are new atoms for e, and $P|_{e \rightarrow x_e} = \bigcup_{r \in P} r|_{e \rightarrow x_e}$ where $r|_{e \rightarrow x_e}$ denotes r with e replaced by x_e.

5. Implementation and Experiments

We implemented our novel inlining approach in the **DLVHEX** solver and compared it to two previous evluation approaches for HEX-programs:

- traditional: Respect external atoms in the core algorithms.
- sup.sets: Use support sets only for external atom verification.

We considered several benchmark problems, including:

1. House problem (abstraction of configuration problems):

n	all answer sets			first answer set		
	traditional	sup.sets	**inlining**	traditional	sup.sets	**inlining**
7	251.68 (81)	83.24 (3)	**22.21 (2)**	22.25 (2)	3.19 (0)	**1.53 (0)**
8	266.22 (85)	183.48 (43)	**59.54 (11)**	61.33 (10)	22.42 (1)	**3.10 (0)**
9	272.70 (85)	263.01 (85)	**86.07 (13)**	76.74 (12)	56.57 (12)	**6.18 (0)**
10	278.26 (83)	275.47 (83)	**121.39 (16)**	102.86 (12)	98.96 (12)	**11.97 (0)**
11	292.05 (85)	300.00 (100)	**167.00 (45)**	158.73 (41)	176.44 (49)	**22.52 (0)**
12	300.00 (100)	300.00 (100)	**180.43 (41)**	159.64 (47)	210.52 (51)	**40.43 (0)**

2. DL-programs (integration of ASP with description logics):

n	all answer sets			first answer set		
	traditional	sup.sets	**inlining**	traditional	sup.sets	**inlining**
20	1.08 (0)	0.34 (0)	**0.31 (0)**	0.34 (0)	0.34 (0)	**0.31 (0)**
30	27.73 (3)	0.98 (0)	**0.34 (0)**	5.66 (0)	0.98 (0)	**0.34 (0)**
40	145.06 (35)	16.68 (2)	**0.40 (0)**	84.73 (14)	16.74 (2)	**0.40 (0)**
50	249.78 (76)	80.69 (15)	**0.48 (0)**	213.45 (60)	80.61 (15)	**0.47 (0)**
60	285.70 (90)	184.25 (47)	**0.57 (0)**	265.61 (85)	184.23 (47)	**0.57 (0)**
70	298.13 (99)	254.00 (74)	**0.72 (0)**	297.17 (99)	254.06 (73)	**0.72 (0)**

6. Conclusion and Outlook

Main results:

- Novel evaluation algorithm for HEX-programs and an implementation.
- Experiments show a significant (up to exponential) speedup.

Future work:

- Refinements and optimizations of the rewriting.
- Heuristics for deciding when to rewrite.

8. References

- Alviano, M., Faber, W., and Gebser, M. (2015).
Rewriting recursive aggregates in answer set programming: back to monotonicity.
CoRR, abs/1507.03923.
- Darwiche, A. and Marquis, P. (2011).
A knowledge compilation map.
CoRR, abs/1106.1819.
- Eiter, T., Fink, M., Krennwallner, T., Redl, C., and Schüller, P. (2014).
Efficient HEX-program evaluation based on unfounded sets.
Journal of Artificial Intelligence Research, 49:269–321.

FWF
Der Wissenschaftsfonds.