Nested HEX-Programs

Thomas Eiter  Thomas Krennwallner  Christoph Redl

{eiter,tkren,redl}@kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

September 28, 2011
Motivation

2 Preliminaries: HEX-Programs

3 Nested HEX-Programs

4 Applications

5 Conclusion
Motivation

Answer-Set Programming and HEX

- Declarative programming formalism
- Shortcomings: No reasoning about subresults
  - Modular programming [JOTW09, EGV97]
  - Meta-reasoning over multiple answer sets
  - XASP [XBG07]: Call LPs under stable model semantics from XSB-Prolog
  - HEX-programs: Calls of procedural external sources
Motivation

Answer-Set Programming and HEX

- Declarative programming formalism
- Shortcomings: No reasoning about subresults
  - Modular programming [JOTW09, EGV97]
  - Meta-reasoning over multiple answer sets
  - XASP [XBG07]: Call LPs under stable model semantics from XSB-Prolog
  - HEX-programs: Calls of procedural external sources

Goal

- Fully declarative means for subprogram calls
- Relational input; Answer sets of subprograms = identifiable objects

Example

- Suppose $P$ computes shortest paths between two nodes in a graph
- Question: How to count the shortest paths (=answer sets of $P$)?
Outline

1. Motivation

2. Preliminaries: HEX-Programs

3. Nested HEX-Programs

4. Applications

5. Conclusion
Preliminaries: HEX-Programs

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
\[ \&p[q_1, \ldots, q_k](t_1, \ldots, t_l),\]
where
- \( p \) ... external predicate name or constants
- \( q_i \) ... predicate names
- \( t_j \) ... terms

Implementation: C++ function
Input: Extensions of \( q_i \)
Output: Set of \( l \)-Tuples

Reasoner
Implementation of \( \&p \)
Example

The \texttt{\&rdf} External Atom

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

Usage

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

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

1 Motivation

2 Preliminaries: HEX-Programs

3 Nested HEX-Programs

4 Applications

5 Conclusion
Architecture

Implementation

- On top of reasoner **DLVHEX**
- Main components:
  1. External Atoms
  2. Answer Set Cache and Handles
Subprogram Calls

Startup: How to call a subprogram?

Possible subprogram sources:
- separate file
- string constant

External Atoms:
\&\text{callhexfile}_n[f, p_1, \ldots, p_n](h) \quad \&\text{callhex}_n[c, p_1, \ldots, p_n](h)

Example: Separate file

\[
\begin{align*}
p_1(x, y) & \leftarrow p_2(a) \leftarrow p_2(b) \leftarrow \\
\text{handle}(H) & \leftarrow \&\text{callhexfile}_2[\text{sub.hex}, p_1, p_2](H)
\end{align*}
\]

Answer Set: \{\text{handle}(0)\}

Example: Embedded subprogram

\[
\begin{align*}
\text{handle}(H) & \leftarrow \&\text{callhex}_0[a \leftarrow b \leftarrow c \leftarrow a.](H)
\end{align*}
\]

Answer Set: \{\text{handle}(0)\}
Subprogram Calls

Caching Mechanism

- No unnecessary re-evaluation

Example

\[
\begin{align*}
h_1(H) & \leftarrow \text{&callhexfile}_0[\text{sub.hex}](H) \\
h_2(H) & \leftarrow \text{&callhexfile}_0[\text{sub.hex}](H) \\
h_3(H) & \leftarrow \text{&callhex}_0[a \leftarrow . \ b \leftarrow .](H)
\end{align*}
\]

Answer set: \{h_1(0), h_2(0), h_3(1)\} or \{h_1(1), h_2(1), h_3(0)\}.
Investigating Program Answers

What is “inside” a program’s answer?

Program answer is a set of answer sets

External Atom: $\text{answersets}[ph](ah)$

Example

$$ash(PH, AH) \leftarrow \text{callhex}_0[a \lor b \leftarrow .](PH), \text{answersets}[PH](AH)$$

Answer set: $\{ash(0, 0), ash(0, 1)\}$
Investigating Program Answers

What is “inside” a program’s answer?

Program answer is a set of answer sets

External Atom: \&answersets[\textit{ph}](\textit{ah})

Example

\[
\text{\textit{ash}(PH, AH) \leftarrow \&\text{callhex}_{0}[a \lor b \leftarrow .](PH), \&\text{answersets}[PH](AH)}
\]

Answer set: \{\textit{ash}(0, 0), \textit{ash}(0, 1)\}

Motivating Example: Counting paths

Program \texttt{paths.hex} specifies a graph and computes all shortest paths between \textit{S} and \textit{D} (defined by \textit{s}(\textit{S}) and \textit{d}(\textit{D})).

\[
P_{\text{cnt}} = \{s(\text{node}_1). \quad d(\text{node}_7).
\]

\[
\text{as(AH) \leftarrow \&\text{callhexfile}_{2}[\texttt{paths.hex}, s, d](PH), \&\text{answersets}[PH](AH)}
\]

\[
\text{number(D) \leftarrow \text{as}(C), D = C + 1, \text{not as}(D)}
\]

\[
\text{exists\_path \leftarrow number(D)}
\]

\[
\text{number}(0) \leftarrow \text{not exists\_path}
\]
Internals of Answer Sets

What is the content of an answer set?
A set of literals over predicate symbols with certain arities

External Atom: $\&predicates[ph, ah](pred, arity)$

Example

\[
preds(P, A) \leftarrow \&callhex_0[node(a). node(b). edge(a, b).](PH), \\
\&answersets[PH](AH), \&predicates[PH, AH](P, A)
\]

Answer Set: $\{preds(node, 1), preds(edge, 2)\}$
Extracting Literals

Which literals are in an answer sets?

Literals are of form \( L_i = p(c_1, \ldots, c_k) \)

Describe set of literals \( \{L_{i_1}, \ldots, L_{i_n}\} \) as set of triples \((i, a, c_{A+1})\)

- Unique literal index \(i\)
- Argument index \(0 \leq a \leq (k - 1)\) and \(s\) (sign)
- Argument values \(c_{a+1}\)

External Atom: \&arguments[ph, ah, pred](i, a, c_{a+1})

Example: Reverse a directed graph

\[
\begin{align*}
    h(\text{PH}, \text{AH}) & \leftarrow \ & \text{&callhex}_0[\text{node}(a). \text{node}(b). \text{node}(c). \text{edge}(a, b). \text{edge}(c, a)].(\text{PH}), \\
    & \quad \text{&answersets}[\text{PH}](\text{AH}) \\
    \text{edge}(W, V) & \leftarrow \ & h(\text{PH}, \text{AH}), \ & \text{&arguments}[\text{PH}, \text{AH}, \text{edge}](I, 0, V), \\
    & \quad \text{&arguments}[\text{PH}, \text{AH}, \text{edge}](I, 1, W) \\
    \text{node}(V) & \leftarrow \ & h(\text{PH}, \text{AH}), \ & \text{&arguments}[\text{PH}, \text{AH}, \text{node}](I, 0, V)
\end{align*}
\]

Answer Set: \{h(0, 0), \text{node}(a), \text{node}(b), \text{node}(c), \text{edge}(b, a), \text{edge}(a, c)\}
Outline

1 Motivation

2 Preliminaries: HEX-Programs

3 Nested HEX-Programs

4 Applications

5 Conclusion
Applications

- **MELD** belief set merging system

- **Aggregate Functions**
  e.g. answer set counting

- **Generalized Quantifiers**
  e.g. cautious vs. brave reasoning; user-defined entailment relation

- **Preferences over Answer Sets**
  e.g. optimization tasks
Outline

1 Motivation

2 Preliminaries: HEX-Programs

3 Nested HEX-Programs

4 Applications

5 Conclusion
Conclusion

Summary

- Reasoning about sets of answer sets of (an)other program(s)
- Modularity by subprogram calls
- http://www.kr.tuwien.ac.at/research/dlvhex/meld.html

Nested HEX-Programs

- Set of External Atoms
- Answer Set Cache and Handles

Applications

- Belief Set Merging
- Generalization of aggregates
- Preferences (optimization tasks, etc)
References

Janhunen T., Oikarinen E., Tompits H., Woltran S
Modularity Aspects of Disjunctive Stable Models
In: Journal of Artificial Intelligence Research 35 (2009) 813–857

Eiter, T., Gottlob, G., Veith, H.:
Modular Logic Programming and Generalized Quantifiers
In: Proceedings of the 4th International Conference on Logic Programming and
Nonmonotonic Reasoning (1997) 290–309

Redl, C.:
Development of a belief merging framework for dlvhex.
Master’s thesis, Vienna University of Technology (June 2010)

Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.:
dlvhex: A system for integrating multiple semantics in an ASP framework.
In: WLP’06. (2006) 206–210

Redl, C., Eiter, T., Krennwallner, T.:
Declarative Belief Set Merging using Merging Plans.
13th International Symposium on Practical Aspects of Declarative Languages (2011)
99–114