# Nested HEX-Programs

Thomas Eiter Thomas Krennwallner Christoph Redl

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

September 28, 2011

# Outline

# 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 = identifible 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

# Preliminaries

## 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 $b_j$.

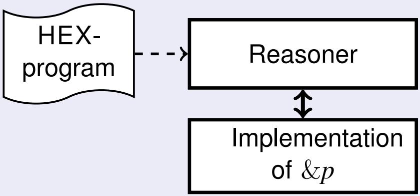
## Definition (External Atoms)

An external atom is of the form

$$\&p[q_1, \ldots, q_k](t_1, \ldots, t_l),$$

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

# Example

## The $\&rdf$ External Atom

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

## Usage

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

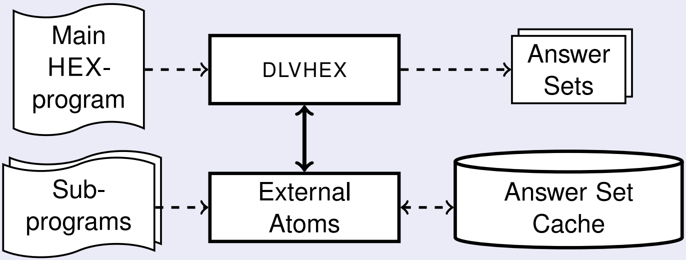
$$addr(\texttt{http://.../data1.rdf}).$$
$$addr(\texttt{http://.../data2.rdf}).$$
$$bel(X, Y) \leftarrow addr(U), \&rdf[U](X, Y, Z).$$

# Outline

# 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:

$\&callhexfile_n[f, p_1, \ldots, p_n](h)$ $\&callhex_n[c, p_1, \ldots, p_n](h)$

## Example: Separate file

$$\begin{aligned} p_1(x,y) &\leftarrow \quad p_2(a) \leftarrow \quad p_2(b) \leftarrow \\ handle(H) &\leftarrow \&callhexfile_2[\texttt{sub.hex}, p_1, p_2](H) \end{aligned}$$

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

## Example: Embedded subprogram

$$handle(H) \leftarrow \&callhex_0[\texttt{a} \leftarrow .\ \texttt{b} \leftarrow .\ \texttt{c} \leftarrow \texttt{a}.](H)$$

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

# Subprogram Calls

## Caching Mechanism

- No unnecessary re-evaluation

## Example

$$\begin{array}{rcl} h_1(H) & \leftarrow & \&callhexfile_0[\texttt{sub.hex}](H) \\ h_2(H) & \leftarrow & \&callhexfile_0[\texttt{sub.hex}](H) \\ h_3(H) & \leftarrow & \&callhex_0[\texttt{a} \leftarrow .\ \texttt{b} \leftarrow .](H) \end{array}$$

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: $\&answersets[ph](ah)$

## Example

$$ash(PH, AH) \quad \leftarrow \quad \&callhex_0[\mathtt{a} \vee \mathtt{b} \leftarrow .](PH), \&answersets[PH](AH)$$

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: $\&answersets[ph](ah)$

## Example

$$ash(PH, AH) \quad \leftarrow \quad \&callhex_0[\mathtt{a} \vee \mathtt{b} \leftarrow .](PH), \&answersets[PH](AH)$$

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

## Motivating Example: Counting paths

Program `paths.hex` specifies a graph and computes all shortest paths between $S$ and $D$ (defined by $s(S)$ and $d(D)$).

$$\begin{aligned} P_{cnt} = \{s(node_1). & \quad d(node_7). \\ as(AH) & \leftarrow \&callhexfile_2[\mathtt{paths.hex}, s, d](PH), \&answersets[PH](AH) \\ number(D) & \leftarrow as(C), D = C + 1, \text{not}\, as(D) \\ exists\_path & \leftarrow number(D) \\ number(0) & \leftarrow \text{not}\, exists\_path\} \end{aligned}$$

# 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

$$\begin{aligned} preds(P, A) \quad \leftarrow \quad & \&callhex_0[\mathtt{node(a).\ node(b).\ edge(a,b).}](PH), \\ & \&answersets[PH](AH), \&predicates[PH, AH](P, A) \end{aligned}$$

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

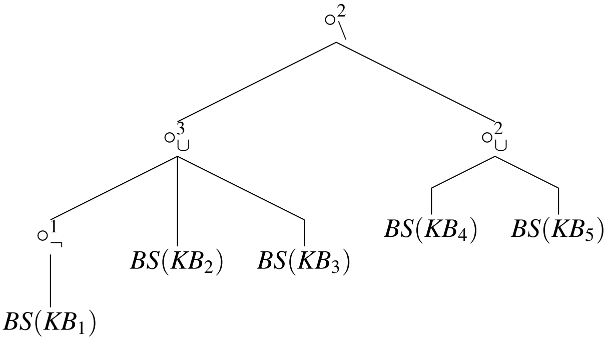
$$
\begin{aligned}
h(PH, AH) &\leftarrow \&callhex_0[\texttt{node(a). node(b). node(c). edge(a,b).} \\
&\qquad\qquad \texttt{edge(c,a).}](PH), \\
&\quad\ \&answersets[PH](AH) \\
edge(W, V) &\leftarrow h(PH, AH), \&arguments[PH, AH, \texttt{edge}](I, 0, V), \\
&\quad\ \&arguments[PH, AH, \texttt{edge}](I, 1, W) \\
node(V) &\leftarrow h(PH, AH), \&arguments[PH, AH, \texttt{node}](I, 0, V)
\end{aligned}
$$

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

# Outline

# 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

# 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)
  `http://www.ub.tuwien.ac.at/dipl/2010/AC07808210.pdf`

- 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