The DLVHEX System for Knowledge Representation: Recent Advances (System Description)

Christoph Redl

redl@kr.tuwien.ac.at

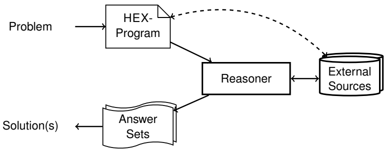
October 18, 2016

Outline

Motivation

HEX-Programs

- Extend ASP by external sources:



A HEX-program consists of rules of form

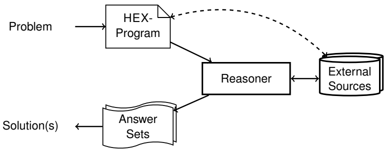
$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n,$$

with classical literals a_i, and classical literals or an external atoms b_j.

Motivation

HEX-Programs

- Extend ASP by external sources:



A HEX-program consists of rules of form

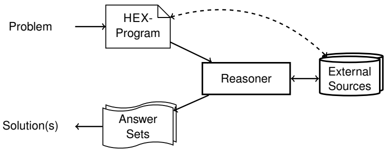
$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n,$$

with classical literals a_i, and classical literals or an external atoms b_j.
Example (external atom): $p(X, Y) \leftarrow url(U), \&rdf[U](X, Y, Z)$

Motivation

HEX-Programs

- Extend ASP by external sources:



A HEX-program consists of rules of form

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

with classical literals a_i, and classical literals or an external atoms b_j.
Example (external atom): $p(X,Y) \leftarrow url(U), \&rdf[U](X,Y,Z)$

Formally:
An external atom is of the form $\&p[q_1,\ldots,q_k](t_1,\ldots,t_l)$, where
p ...external predicate, q_i ...predicates or constants, t_j ...terms.

Semantics given by a $1+k+l$-ary Boolean oracle function $f_{\&p}$:
$I \models \&p[q_1,\ldots,q_k](t_1,\ldots,t_l)$ if $f_{\&p}(I,q_1,\ldots,q_k,t_1,\ldots,t_l) = \mathbf{T}$
(and $I \not\models \&p[q_1,\ldots,q_k](t_1,\ldots,t_l)$ otherwise).

Motivation

Implementation

DLVHEX

http://www.kr.tuwien.ac.at/research/systems/dlvhex

- Based on GRINGO and CLASP from the Potassco suite.
- Supported platforms: Linux-based, OS X, Windows.
- External sources are implemented as plugins using a plugin API (available for C++ or Python).

Motivation

Implementation

DLVHEX

http://www.kr.tuwien.ac.at/research/systems/dlvhex

- Based on GRINGO and CLASP from the Potassco suite.
- Supported platforms: Linux-based, OS X, Windows.
- External sources are implemented as plugins using a plugin API (available for C++ or Python).

This talk: presentation of

- **novelties done in the last three years** and
- **current state of the system.**

Outline

From Black-box to Grey-box

Previous Evaluation Bottleneck

- External sources were seen as black boxes.
- Behavior under an interpretation did not allow for drawing conclusions about other interpretations.
- Algorithms must be very general $\Rightarrow$ room for optimizations limited.

From Black-box to Grey-box

Previous Evaluation Bottleneck

- External sources were seen as black boxes.
- Behavior under an interpretation did not allow for drawing conclusions about other interpretations.
- Algorithms must be very general $\Rightarrow$ room for optimizations limited.

Idea

- Developers of external sources and/or implementer of HEX-program might have useful additional information.
- Provide a (predefined) list of possible properties of external sources.
- Let the developer and/or user specify which properties are satisfied.
- Algorithms exploit them for various purposes, most importantly:
  - efficiency improvements and
  - language flexibility (reducing syntactic restrictions).

From Black-box to Grey-box

Previous Evaluation Bottleneck

- External sources were seen as black boxes.
- Behavior under an interpretation did not allow for drawing conclusions about other interpretations.
- Algorithms must be very general $\Rightarrow$ room for optimizations limited.

Idea

- Developers of external sources and/or implementer of HEX-program might have useful additional information.
- Provide a (predefined) list of possible properties of external sources.
- Let the developer and/or user specify which properties are satisfied.
- Algorithms exploit them for various purposes, most importantly:
  - efficiency improvements and
  - language flexibility (reducing syntactic restrictions).

Important:
User specifies them but does **not** need to know how they are exploited!

Specifying Properties

How to specify them?

- During development of external source using the plugin API.

Specifying Properties

How to specify them?

- During development of external source using the plugin API.
- As part of the HEX-program using property tags $\langle \cdots \rangle$.

Specifying Properties

How to specify them?

- During development of external source using the plugin API.
- As part of the HEX-program using property tags $\langle \cdots \rangle$.
  Example:
  $\&greaterThan[p, 10]()$ is true if $\sum_{p(c) \in I} c > 10$.
  It is monotonic for positive integers.

Specifying Properties

How to specify them?

- During development of external source using the plugin API.
- As part of the HEX-program using property tags $\langle \cdots \rangle$.
  Example:
  $\&greaterThan[p, 10]()$ is true if $\sum_{p(c) \in I} c > 10$.
  It is monotonic for positive integers.

Available properties (examples)

- **Functionality**: $\&add[X, Y](Z)\langle functional \rangle$
  Adds integers X and Y and is true for their sum Z.
  It provides exactly one output for a given input.

Specifying Properties

How to specify them?

- During development of external source using the plugin API.
- As part of the HEX-program using property tags $\langle \cdots \rangle$.
  Example:
  $\&greaterThan[p, 10]()$ is true if $\sum_{p(c) \in I} c > 10$.
  It is monotonic for positive integers.

Available properties (examples)

- **Functionality**: $\&add[X, Y](Z)\langle functional \rangle$
  Adds integers X and Y and is true for their sum Z.
  It provides exactly one output for a given input.
- **Well-ordering**: $\&decrement[X](Z)\langle wellordering\ 0\ 0 \rangle$
  Decrements a given integer.
  The 0-th output is no greater than the 0-th input (wrt. some ordering).

Specifying Properties

How to specify them?

- During development of external source using the plugin API.
- As part of the HEX-program using property tags $\langle \cdots \rangle$.
  Example:
  $\&greaterThan[p, 10]()$ is true if $\sum_{p(c) \in I} c > 10$.
  It is monotonic for positive integers.

Available properties (examples)

- **Functionality**: $\&add[X, Y](Z)\langle functional\rangle$
  Adds integers X and Y and is true for their sum Z.
  It provides exactly one output for a given input.
- **Well-ordering**: $\&decrement[X](Z)\langle wellordering\ 0\ 0\rangle$
  Decrements a given integer.
  The 0-th output is no greater than the 0-th input (wrt. some ordering).
- **Three-valued semantics**:
  The external source can be evaluated under partial interpretations.

Specifying Properties

How to specify them?

- During development of external source using the plugin API.
- As part of the HEX-program using property tags $\langle \cdots \rangle$.
  Example:
  $\&greaterThan[p, 10]()$ is true if $\sum_{p(c) \in I} c > 10$.
  It is monotonic for positive integers.

Available properties (examples)

- **Functionality**: $\&add[X, Y](Z)\langle functional\rangle$
  Adds integers X and Y and is true for their sum Z.
  It provides exactly one output for a given input.
- **Well-ordering**: $\&decrement[X](Z)\langle wellordering\ 0\ 0\rangle$
  Decrements a given integer.
  The 0-th output is no greater than the 0-th input (wrt. some ordering).
- **Three-valued semantics**:
  The external source can be evaluated under partial interpretations.
- …

Exploiting Properties for Efficiency Improvement

Conflict-driven Solving

- ASP program is internally represented by nogoods (sets of literals which cannot be simultaneously true).
- Additional nogoods learned from conflicting interpretations.
- HEX-solver further learns nogoods from external sources which describe parts of their behavior to avoid future wrong guesses.

Exploiting Properties for Efficiency Improvement

Conflict-driven Solving

- ASP program is internally represented by nogoods (sets of literals which cannot be simultaneously true).
- Additional nogoods learned from conflicting interpretations.
- HEX-solver further learns nogoods from external sources which describe parts of their behavior to avoid future wrong guesses.

Example

- We evaluate $\&diff[p,q](X)$ under $I = \{p(a), q(b)\}$.
- It is true for $X = a$ (and false otherwise), i.e., $I \models \&diff[p,q](a)$.
- $\Rightarrow$ Learn nogood $N = \{p(a), \neg q(a), \neg p(b), q(b), \neg \&diff[p,q](a)\}$.

Exploiting Properties for Efficiency Improvement

Conflict-driven Solving

- ASP program is internally represented by nogoods (sets of literals which cannot be simultaneously true).
- Additional nogoods learned from conflicting interpretations.
- HEX-solver further learns nogoods from external sources which describe parts of their behavior to avoid future wrong guesses.

Example

- We evaluate $\&diff[p,q](X)$ under $I = \{p(a), q(b)\}$.
- It is true for $X = a$ (and false otherwise), i.e., $I \models \&diff[p,q](a)$.
- $\Rightarrow$ Learn nogood $N = \{p(a), \neg q(a), \neg p(b), q(b), \neg\&diff[p,q](a)\}$.

Exploiting Properties

- Known properties used to shrink nogoods to their essential part.
- Example: $\&diff[p,q](X)$ is monotonic in p:
  Shrink above nogood N to $N' = \{p(a), \neg q(a), q(b), \neg\&diff[p,q](a)\}$.
  (If $p(b)$ turns to true, $\&diff[p,q](a)$ is still true $\Rightarrow \neg p(b)$ not needed.)

Exploiting Properties for Language Flexibility

Grounding and Safety

- External atoms may introduce new constants: value invention.
- $\Rightarrow$ Can lead to programs which cannot be finitely grounded.

Exploiting Properties for Language Flexibility

Grounding and Safety

- External atoms may introduce new constants: value invention.
- $\Rightarrow$ Can lead to programs which cannot be finitely grounded.

Example

$$\Pi = \left\{ \begin{array}{l} r_1: start(s). \\ r_2: scc(X) \leftarrow start(X). \quad r_3: scc(Y) \leftarrow scc(X), \&edge[X](Y). \end{array} \right\}$$

Exploiting Properties for Language Flexibility

Grounding and Safety

- External atoms may introduce new constants: value invention.
- $\Rightarrow$ Can lead to programs which cannot be finitely grounded.

Example

$$\Pi = \left\{ \begin{array}{l} r_1 : start(s). \\ r_2 : scc(X) \leftarrow start(X). \quad r_3 : scc(Y) \leftarrow scc(X), \&edge[X](Y). \end{array} \right\}$$

Solution: Syntactic Restrictions (Safety)

- Traditionally: strong safety; essentially no recursive value invention!

Exploiting Properties for Language Flexibility

Grounding and Safety

- External atoms may introduce new constants: value invention.
- $\Rightarrow$ Can lead to programs which cannot be finitely grounded.

Example

$$\Pi = \left\{ \begin{array}{l} r_1: start(s). \\ r_2: scc(X) \leftarrow start(X). \quad r_3: scc(Y) \leftarrow scc(X), \&edge[X](Y). \end{array} \right\}$$

Solution: Syntactic Restrictions (Safety)

- Traditionally: strong safety; essentially no recursive value invention!
- But: overly restrictive.

Exploiting Properties for Language Flexibility

Grounding and Safety

- External atoms may introduce new constants: value invention.
- ⇒ Can lead to programs which cannot be finitely grounded.

Example

$$\Pi = \left\{ \begin{array}{l} r_1 : start(s). \\ r_2 : scc(X) \leftarrow start(X). \quad r_3 : scc(Y) \leftarrow scc(X), \&edge[X](Y). \end{array} \right\}$$

Solution: Syntactic Restrictions (Safety)

- Traditionally: strong safety; essentially no recursive value invention!
- But: overly restrictive.

Exploiting Properties

- Properties may allow for identifying finite groundability even in presence of recursive value invention (in some cases).
- Example:
  Known finiteness of the graph above allows for establishing safety.

Outline

Python Programming Interface

More convenient interface

Previously only C++ support, but Python preferred by many developers:

- No overhead due to makefiles, compilation, linking, etc.
- High-level features.
- Negligible overhead compared to plugins implemented in C++.

Python Programming Interface

More convenient interface

Previously only C++ support, but Python preferred by many developers:

- No overhead due to makefiles, compilation, linking, etc.
- High-level features.
- Negligible overhead compared to plugins implemented in C++.

Example

Program

$$\Pi = \left\{ \begin{array}{l} r_1 : start(s). \\ r_2 : scc(X) \leftarrow start(X). \quad r_3 : scc(Y) \leftarrow scc(X), \&edge[X](Y). \end{array} \right\}$$

compute the strongly connected component of a node s in a graph.

Implementation of $\&edge[X](Y)$:

```
def edge(x):
  graph=((1,2),(1,3),(2,3))           # simplified implementation
  for edge in graph:                  # search for out-edges of node x
    if edge[0]==x.intValue():
      dlvhex.output((edge[1],))       # output edge target

def register():
  prop = dlvhex.ExtSourceProperties() # inform dlvhex about
  prop.addFiniteOutputDomain(0)       # finiteness of the graph
  dlvhex.addAtom("edge", (dlvhex.CONSTANT, ), 1, prop)
```

Further Improvements

Availability

- Pre-compiled binaries for major platforms available (previously distributed only as sourcecode).
- Online demo: `http://www.kr.tuwien.ac.at/research/systems/dlvhex/demo.php`.

Interoperability

- Support for all features of the ASP-Core-2 standard.
- Support for input/ouput in CSV format (interoperability with tools and spreadsheet programs).

Outline

Applications of HEX-Programs

- Multi-context Systems (interconnection of knowledge-bases)
- DL-programs (integration of ASP with ontologies)
- Constraint ASP (programs with constraint atoms)
- Physics simulation (e.g. AngryBirds agent)
- Route planning (possibly semantically enriched)
- Robotics applications (planning)
- ACTHEX (programs with action atoms)
- ...

Outline

Conclusion

DLVHEX

Two main categories of improvements:

Exploiting external source properties

- Plugin developer or HEX-programmer tags guaranteed properties.
- Algorithms exploit these properties where applicable.
- User does not need to know how they are exploited to benefit.
- Used for efficiency improvements and language flexibility.

Usability and System Improvements

- New programming interface (API) for Python-based plugins.
- Binaries for Linux, OS X and Windows available.
- Online demo allows for testing in the browser.
- Support for ASP-Core-2 standard and for input/output in CSV format.

http://www.kr.tuwien.ac.at/research/systems/dlvhex

References I

Bögl, M., Eiter, T., Fink, M., and Schüller, P. (2010).
The MCS-IE system for explaining inconsistency in multi-context systems.
In *In Proceedings of the Twelfth European Conference on Logics in Artificial Intelligence (JELIA 2010)*, pages 356–359.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Roland Kaminski, T. K., Leone, N., Ricca, F., and Schaub, T. (2013).
ASP-Core-2 Input Language Format.

Eiter, T., Fink, M., Krennwallner, T., and Redl, C. (2016a).
Domain expansion for asp-programs with external sources.
Artif. Intell., 233:84–121.

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.

References II

Eiter, T., Kaminski, T., Redl, C., and Weinzierl, A. (2016b).
Exploiting partial assignments for efficient evaluation of answer set programs with external source access.
In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016), July 9–15, 2016, New York City, New York, USA*.

Eiter, T., Krennwallner, T., Prandtstetter, M., Rudloff, C., Schneider, P., and Straub, M. (2016c).
Semantically enriched multi-modal routing.
Int. J. Intelligent Transportation Systems Research, 14(1):20–35.

Erdem, E., Patoglu, V., and Schüller, P. (2016).
A Systematic Analysis of Levels of Integration between High-Level Task Planning and Low-Level Feasibility Checks.
AI Communications, IOS Press.

References III

Ianni, G., Calimeri, F., Germano, S., Humenberger, A., Redl, C., Stepanova, D., Tucci, A., and Wimmer, A. (2016).
Angry-HEX: an artificial player for angry birds based on declarative knowledge bases.
IEEE Transactions on Computational Intelligence and AI in Games.

Zirtiloglu, H. and Yolum, P. (2008).
Ranking semantic information for e-government: complaints management.
In Duke, A., Hepp, M., Bontcheva, K., and Vilain, M. B., editors, *Proceedings of the First International Workshop on Ontology-supported Business Intelligence, OBI 2008, Karlsruhe, Germany, October 27, 2008*, volume 308 of *ACM International Conference Proceeding Series*, page 5. ACM.