Explaining Inconsistency in Answer Set Programs and Extensions

Christoph Redl

redl@kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Knowledge-Based Systems Group

July 6, 2017

Outline

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.

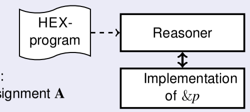
Definition (External Atoms)

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 $f_{\&p}(\mathbf{A}, q_1, \ldots, q_k, t_1, \ldots, t_l) = 1$.



Motivation

Main idea

- Inconsistent programs are programs without answer sets.
- Question: Why it such a program inconsistent?

Motivation

Main idea

- Inconsistent programs are programs without answer sets.
- Question: Why it such a program inconsistent?
- Previous work (e.g. on debugging):
  Focus on explaining why a particular interpretation is not an answer set.
  Focus on human-readable explanations (unsupported atoms, odd loops, etc).

Motivation

Main idea

- Inconsistent programs are programs without answer sets.
- Question: Why it such a program inconsistent?
- Previous work (e.g. on debugging):
  Focus on explaining why a particular interpretation is not an answer set.
  Focus on human-readable explanations (unsupported atoms, odd loops, etc).
- Here: Focus on explanations which be easily expressed as constraints.

Motivation

Main idea

- Inconsistent programs are programs without answer sets.
- Question: Why it such a program inconsistent?
- Previous work (e.g. on debugging):
  Focus on explaining why a particular interpretation is not an answer set.
  Focus on human-readable explanations (unsupported atoms, odd loops, etc).
- Here: Focus on explanations which be easily expressed as constraints.
- Envisaged application: improvements for programs with multiple components.

Motivation

Motivation

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)

Motivation

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.

Motivation

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.

Motivation

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.
- A meta-programming technique for computing IRs for normal programs.

Motivation

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.
- A meta-programming technique for computing IRs for normal programs.
- A procedural algorithm for computing IRs for general programs.

Outline

Formalizing Inconsistency Reasons

Basic idea

- Consider programs P extended by atoms $I \subseteq \mathcal{D}$ from a given domain $\mathcal{D}$.

Formalizing Inconsistency Reasons

Basic idea

- Consider programs P extended by atoms $I \subseteq \mathcal{D}$ from a given domain $\mathcal{D}$.
- Goal:
  Express reasons for inconsistency of $P \cup facts(I)$ in terms of I
  (where $facts(I) = \{a \leftarrow \mid a \in I\}$),
  i.e., a criterion wrt. I which guarantees that $P \cup facts(I)$ is inconsistent.

Formalizing Inconsistency Reasons

Basic idea

- Consider programs P extended by atoms $I \subseteq \mathcal{D}$ from a given domain $\mathcal{D}$.
- Goal:
  Express reasons for inconsistency of $P \cup facts(I)$ in terms of I
  (where $facts(I) = \{a \leftarrow \mid a \in I\}$),
  i.e., a criterion wrt. I which guarantees that $P \cup facts(I)$ is inconsistent.
- Proposed notion:
  Use pairs of sets of atoms from $\mathcal{D}$ which must occur resp. must not occur in I such that $P \cup facts(I)$ is inconsistent.

Formalizing Inconsistency Reasons

Formally:

Definition (Inconsistency Reason (IR))

Let P be a HEX-program and $\mathcal{D}$ be a domain of atoms.

An inconsistency reason (IR) of P wrt. $\mathcal{D}$ is a pair $R = (R^+, R^-)$ of sets of atoms $R^+ \subseteq \mathcal{D}$ and $R^- \subseteq \mathcal{D}$ with $R^+ \cap R^- = \emptyset$ s.t. $P \cup facts(I)$ is inconsistent for all $I \subseteq \mathcal{D}$ with $R^+ \subseteq I$ and $R^- \cap I = \emptyset$.

Formalizing Inconsistency Reasons

Formally:

Definition (Inconsistency Reason (IR))

Let P be a HEX-program and $\mathcal{D}$ be a domain of atoms.

An inconsistency reason (IR) of P wrt. $\mathcal{D}$ is a pair $R = (R^+, R^-)$ of sets of atoms $R^+ \subseteq \mathcal{D}$ and $R^- \subseteq \mathcal{D}$ with $R^+ \cap R^- = \emptyset$ s.t. $P \cup facts(I)$ is inconsistent for all $I \subseteq \mathcal{D}$ with $R^+ \subseteq I$ and $R^- \cap I = \emptyset$.

Example

Consider $P = \{\leftarrow a, \mathrm{not}\, c;\ d \leftarrow b.\}$ and $\mathcal{D} = \{a, b, c\}$.

An IR is $R = (\{a\}, \{c\})$ because $P \cup facts(I)$ is inconsistent for all $I \subseteq \mathcal{D}$ whenever $a \in I$ and $c \notin I$.

Formal Results

Proposition

For all HEX-programs P and domains $\mathcal{D}$ such that $P \cup facts(I)$ is inconsistent for some set $I \subseteq \mathcal{D}$ of atoms, then there is an IR of P wrt. $\mathcal{D}$.

Formal Results

Proposition

For all HEX-programs P and domains $\mathcal{D}$ such that $P \cup facts(I)$ is inconsistent for some set $I \subseteq \mathcal{D}$ of atoms, then there is an IR of P wrt. $\mathcal{D}$.

Alternative characterization based on unfounded sets:

Definition (Unfounded Set [Faber, 2005])

Given a HEX-program P and an assignment A, let U be any set of atoms appearing in P. Then, U is an *unfounded set for P wrt. A* if, for each rule $r \in P$ with $H(r) \cap U \neq \emptyset$, at least one of the following conditions holds:

(i) some literal of $B(r)$ is false wrt. A; or

(ii) some literal of $B(r)$ is false wrt. $A \setminus U$; or

(iii) some atom of $H(r) \setminus U$ is true wrt. A.

Formal Results

Proposition

Let P be a ground HEX-program and $\mathcal{D}$ be a domain. Then a pair of sets of atoms (R^+, R^-) with $R^+ \subseteq \mathcal{D}$, $R^- \subseteq \mathcal{D}$ and $R^+ \cap R^- = \emptyset$ is an IR of P iff for all classical models M of P either (i) $R^+ \not\subseteq M$ or (ii) there is a nonempty unfounded set U of P wrt. M such that $U \cap M \neq \emptyset$ and $U \cap (\mathcal{D} \setminus R^-) = \emptyset$.

Relaxed proposition (sound but not complete for determining IRs):

Proposition

Let P be a ground HEX-program and $\mathcal{D}$ be a domain such that $H(P) \cap \mathcal{D} = \emptyset$. For a pair of sets of atoms (R^+, R^-) with $R^+ \subseteq \mathcal{D}$ and $R^- \subseteq \mathcal{D}$, if for all classical models M of P we either have (i) $R^+ \not\subseteq M$ or (ii) $R^- \cap M \neq \emptyset$ then (R^+, R^-) is an IR of P.

Outline

Complexity Results

Reasoning problem	Program class: Normal ASP	Program class: General HEX
Checking an IR candidate	$coNP$-c	Π_2^P-c
Checking existence of an IR	Π_2^P-c	Π_3^P-c
Checking a subset-minimal IR candidate	D_1^P-c	D_2^P-c
Checking existence of a minimal IR	Π_2^P-c	Π_3^P-c

Table: Summary of Complexity Results

Outline

Inconsistency Reasons for Normal ASP-Programs

A Meta-Program for Inconsistency Detection

For a normal ASP program P one can construct a positive disjunctive meta-program M^P with the following properties (e.g. [Eiter and Polleres, 2006]):

- M^P is always consistent;
- if P is inconsistent, then M^P has a single answer set $A_{sat} = A(M^P)$ containing all atoms in M^P including a dedicated atom $incons$ which does not appear in P; and
- if P is consistent, then the answer sets of M^P correspond one-to-one to those of P and none of them contains $incons$.

Computing Inconsistency Reasons

Inconsistency Reasons for Normal ASP-Programs

A Meta-Program for Inconsistency Detection

For a normal ASP program P one can construct a positive disjunctive meta-program M^P with the following properties (e.g. [Eiter and Polleres, 2006]):

- M^P is always consistent;
- if P is inconsistent, then M^P has a single answer set $A_{sat} = A(M^P)$ containing all atoms in M^P including a dedicated atom $incons$ which does not appear in P; and
- if P is consistent, then the answer sets of M^P correspond one-to-one to those of P and none of them contains $incons$.

Then, the atom $incons$ in the answer set(s) of M^P represents inconsistency of the original program P.

Inconsistency Reasons for Normal ASP-Programs

A Meta-Program for IR Computation

IRs of P can then be computed using the meta-program:

$$
\begin{aligned}
\tau(\mathcal{D}, P) = {} & M^P && (1)\\
& \cup \{a^+ \vee a^- \vee a^x \mid a \in \mathcal{D}\} \ \cup && (2)\\
& \cup \{a \leftarrow a^+;\ \leftarrow a, a^-;\ a \vee \bar{a} \leftarrow a^x;\ \mid a \in \mathcal{D}\} && (3)\\
& \cup \{\leftarrow \mathrm{not}\, incons\}, && (4)
\end{aligned}
$$

where a^+, a^-, a^x and $\bar{a}$ are new atoms for all atoms $a \in \mathcal{D}$.

Proposition

Let P be an ordinary normal program and $\mathcal{D}$ be a domain. Then (R^+, R^-) is an IR of P wrt. $\mathcal{D}$ iff $\tau(\mathcal{D}, P)$ has an answer set $A \supseteq \{a^\sigma \mid \sigma \in \{+,-\}, a \in R^\sigma\}$.

Inconsistency Reasons for General HEX-Programs

Challenges

- Deciding if a general HEX-program has an IR is Σ_3^P-complete but reasoning tasks over a general HEX-program are only on the second level of the polynomial hierarchy.

Computing Inconsistency Reasons

Inconsistency Reasons for General HEX-Programs

Challenges

- Deciding if a general HEX-program has an IR is Σ_3^P-complete but reasoning tasks over a general HEX-program are only on the second level of the polynomial hierarchy.

- $\Rightarrow$ Computing the IRs of a general HEX-program cannot be polynomially reduced to a meta-HEX-program (unless $\Sigma_3^P = \Sigma_2^P$).

Inconsistency Reasons for General HEX-Programs

Challenges

- Deciding if a general HEX-program has an IR is Σ_3^P-complete but reasoning tasks over a general HEX-program are only on the second level of the polynomial hierarchy.
- $\Rightarrow$ Computing the IRs of a general HEX-program cannot be polynomially reduced to a meta-HEX-program (unless $\Sigma_3^P = \Sigma_2^P$).
- Remedies:
  - Giving up completeness
    For HEX-programs P without disjunctions we can specify an encoding for computing its IRs, which is sound but not complete.

Inconsistency Reasons for General HEX-Programs

Challenges

- Deciding if a general HEX-program has an IR is Σ_3^P-complete but reasoning tasks over a general HEX-program are only on the second level of the polynomial hierarchy.
- $\Rightarrow$ Computing the IRs of a general HEX-program cannot be polynomially reduced to a meta-HEX-program (unless $\Sigma_3^P = \Sigma_2^P$).
- Remedies:
  - Giving up completeness
    For HEX-programs P without disjunctions we can specify an encoding for computing its IRs, which is sound but not complete.
  - Using a procedural algorithm

Outline

Learning Over Multiple Program Components

HEX-program evaluation is based on program splitting.

Learning Over Multiple Program Components

HEX-program evaluation is based on program splitting.

Example

- We want to form a committee of employees.
- Some pairs of persons have conflicts of interests.
- The competences of the committee depend on the persons involved.
- Competences can depend nonmonotonically on the members.
- Constraints define the competences the committee should have.

Learning Over Multiple Program Components

HEX-program evaluation is based on program splitting.

Example

- We want to form a committee of employees.
- Some pairs of persons have conflicts of interests.
- The competences of the committee depend on the persons involved.
- Competences can depend nonmonotonically on the members.
- Constraints define the competences the committee should have.

$$
\begin{aligned}
P = \{ & r_1 \colon in(X) \vee out(X) \leftarrow person(X). \\
& r_2 \colon \leftarrow in(X), in(Y), conflict(X, Y). \\
& r_3 \colon comp(X) \leftarrow \&competences[in](X). \\
& r_4 \colon \leftarrow \mathrm{not}\ comp(technical), \mathrm{not}\ comp(financial). \}
\end{aligned}
$$

Learning Over Multiple Program Components

HEX-program evaluation is based on program splitting.

Example

- We want to form a committee of employees.
- Some pairs of persons have conflicts of interests.
- The competences of the committee depend on the persons involved.
- Competences can depend nonmonotonically on the members.
- Constraints define the competences the committee should have.

$$
\begin{aligned}
P = \{ & r_1: in(X) \vee out(X) \leftarrow person(X). \\
& r_2: \leftarrow in(X), in(Y), conflict(X,Y). \\
& r_3: comp(X) \leftarrow \&competences[in](X). \\
& r_4: \leftarrow \text{not } comp(technical), \text{not } comp(financial). \}
\end{aligned}
$$

$$\boxed{P_1 = \{r_1, r_2\}} \longleftarrow \boxed{P_2 = \{r_3, r_4\}}$$

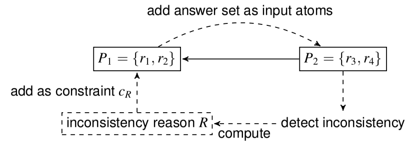
Figure: Evaluation graph of the program from the previous example

Learning Over Multiple Program Components

Main Idea:
Associate an IR R of a later program component with a constraint c_R which we propagate to predecessors.

Learning Over Multiple Program Components

Main Idea:
Associate an IR R of a later program component with a constraint c_R which we propagate to predecessors.



Outline

Conclusion

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)

Conclusion

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.

Conclusion

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.

Conclusion

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.
- A meta-programming technique for computing IRs for normal programs.

Conclusion

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.
- A meta-programming technique for computing IRs for normal programs.
- A procedural algorithm for computing IRs for general programs.

Conclusion

Contribution

- Characterizing of inconsistency in terms of atoms in the facts (EDB).
  ⇒ Notion of inconsistency reasons (IRs)
  That is, we identify classes of instances which are inconsistent.
- Complexity results regarding the computation of IRs.
- A meta-programming technique for computing IRs for normal programs.
- A procedural algorithm for computing IRs for general programs.

Ongoing Work

- Application of the results for algorithmic improvements.

References I

Eiter, T. and Polleres, A. (2006).
Towards automated integration of guess and check programs in answer set programming: a meta-interpreter and applications.
TPLP, 6(1-2):23–60.

Faber, W. (2005).
Unfounded sets for disjunctive logic programs with arbitrary aggregates.
In *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005), Diamante, Italy, September 5-8, 2005*, volume 3662, pages 40–52. Springer.