Query Answering in (Resource-Based) answer set semantics

Stefania Costantini and Andrea Formisano

University of L'Aquila and University of Perugia, Italy

LRC'15, Corunna, Spain
Speaker: Stefania

Answer Set Semantics

The answer set semantics (AS) *extends* the well-founded semantics, which assigns to a logic program Π a unique, three-valued model $WFS(\Pi) = \langle W^+, W^- \rangle$.

In particular, the answer set semantics selects some of the (two-valued) classical models of given program Π so as for each atom A which is true w.r.t. an answer set M:

- A is supported in M by some rule of Π;
- consequently, the support of A does not depend (directly or indirectly) upon the negation of another true atom, including itself.

AS gave rise to ASP, very well-established logic programming paradigm.

Answer Set Semantics (cont'd)

- For *even cycles* such as $\{e \leftarrow not\ f.\ f \leftarrow not\ e.\}$, two answer sets can be found, in the example $\{e\}$ and $\{f\}$, respecting the above conditions.
- For odd cycles, such as unary odd cycles of the form $\{p \leftarrow not\ p.\}$ and ternary odd cycles of the form $\{a \leftarrow not\ b.\ b \leftarrow not\ c.\ c \leftarrow not\ a.\}$, it is not possible to fulfill the conditions in classical models. Thus, a program including such cycles is *inconsistent*, i.e., it has no answer sets, unless "handles" are provided from other parts of the program to make some atom of the cycle true/false.

In some sense, the answer set semantics is still three-valued: sometimes it is able to assign truth value to atoms, while sometimes (when the program is inconsistent) leaves them all undefined. Problem: lack of Relevance (Dix 1995)

Answer Set Semantics: Query-Answering

- Some attempts, also recent, preliminary program analisys and/or incremental answer set construction
  - Bonatti,P.A.,Pontelli,E.,Son,T.C.: Credulous resolution for answer set programming
    In Fox, D., Gomes, C.P., eds.: Proc. of the 23th AAAI Conference on Artificial Intelligence (2008)
  - Gebser, M., Gharib, M., Mercer, R.E., Schaub, T.: Monotonic answer set programming.
    J. Log. Comput.19(4) (2009) item Marple, K., Gupta, G.: Dynamic consistency checking in goal-directed answer set programming. TPLP14(4-5) (2014)

In a sense, the answer set semantics is still three-valued: sometimes it is able to assign truth value to atoms, while sometimes (when the program is inconsistent) leaves them all undefined.

Answer Set Semantics (AS): Autoepistemic Logic Characterization

Defined by Marek and Truszczyński (1997) for AS, drawing inspiration from Gelfond, 1997, i.e.:

> *not p* is not to be interpreted as $\neg p$, but instead as "I don' believe *p*", which is an assumption.

A rule

$$A \leftarrow A_1, \ldots, A_n, not\ B_1, \ldots, not\ B_m$$

in given program Π can be seen as standing for its "modal image"

$$L\,A_1 \wedge \ldots \wedge L\,A_n \wedge L \neg L\,B_1 \wedge \ldots \wedge L \neg L\,B_m \supset A$$

Extended Autoepistemic Logic Characterization (cont'd)

From modal images of single rules one can then get the modal image of the entire program.

Answer sets of Π coincide (after dropping modal atoms) with *reflexive autoepistemic expansions* of the modal image, where reflexive autoepistemic expansions are obtained as:

$$T = Cn(I \cup (\varphi \equiv L\varphi : \varphi \in T) \cup (\neg L\varphi : \varphi \notin T))$$

Extended Extended Autoepistemic Logic Characterization

Reflexive autoepistemic logic corresponds, according to Marek and Truszczyński, to the modal logic **SW5**. Specific axioms of **SW5**:

$$L\varphi \supset \varphi$$
$$L\varphi \supset L\,L\varphi$$
$$\neg\, L\neg\, L\varphi \supset (\varphi \supset L\varphi)$$

Modified modal image (Costantini and Formisano):

$$L\,A_1 \wedge \ldots \wedge L\,A_n \wedge L\neg\, L\,B_1 \wedge \ldots \wedge L\neg L\,B_m \supset L\,\dot{A}$$
$$L\,\dot{A} \wedge \neg\, L\neg\, L\,A \supset L\,A$$

Proposal: Resource-based Answer sets of Π, which coincide (after dropping modal atoms) with *reflexive autoepistemic expansions* of this modified modal image.

Extended Autoepistemic Logic Characterization: Example

Unary odd cycle

$$p \leftarrow not\, p$$

Modified modal image:

$$L \neg L\, p \supset L\, \dot{p}$$
$$L\, \dot{p} \wedge \neg L \neg L\, p \supset L\, p$$

Unique reflexive autoepistemic expansion $\emptyset$

Extended Autoepistemic Logic Characterization: Example

Unary odd cycle with positive dependencies

$$p \leftarrow a$$
$$a \leftarrow not\ p$$

Modified modal image:

$$L \neg L\, p \supset L\, \dot{a}$$
$$L\, \dot{a} \wedge \neg L \neg L\, a \supset L\, a L\, a \supset L\, \dot{p}$$
$$L\, \dot{p} \wedge \neg L \neg L\, p \supset L\, p$$

Unique reflexive autoepistemic expansion $\{a\}$, while unique classical model $\{p\}$ not supported.

$\{a\}$ is not a classical model, but it is a supported set of atoms (w.r.t. given program) and in this sense it is also maximal.

Extended Autoepistemic Logic Characterization: Example

Ternary odd cycle with positive dependencies

$$a \leftarrow not\ b$$
$$b \leftarrow not\ c$$
$$c \leftarrow not\ a$$

Modified modal image:

$$L \neg L\, b \supset L\, \dot{a}$$
$$L\, \dot{a} \wedge \neg L \neg L\, a \supset L\, a$$
$$L \neg L\, c \supset L\, \dot{c}$$
$$L\, \dot{c} \wedge \neg L \neg L\, c \supset L\, c$$
$$L \neg L\, a \supset L\, \dot{c}$$
$$L\, \dot{c} \wedge \neg L \neg L\, c \supset L\, c$$

Three reflexive autoepistemic expansion, namely $\{a\}$, $\{b\}$,$\{c\}$, depending upon which negative assumption you choose to make, e.g., $\{a\}$ from $L \neg L\, b$.

Resource-Based answer set semantics (RAS)

- Every program is consistent.
- Consequence: constraints have to be defined in a separate 'layer'.
- Regains important properties of non-monotonic formalisms (Dix 1995), namely Relevance and Modularity.
- Allows for prolog-style query-answering.
- Same complexity as AS.

RAS: Linear Logic Characterization

Stems from the linear logic formulation of ASP that we proposed in the past (in honor of David Pearce), where answer sets as maximal tensor conjunctions provable from linear logic theory corresponding to given ASP program.

Negation as a resource (whence the name RAS): negation *not A* of atom *A* as a resource that is unlimitedly available unless *A* is proved. Therefore:

1. *not A* becomes unavailable if *A* is proved;
2. whenever *not A* has been used, *A* can no longer be proved.

Program $p \leftarrow \mathit{not}\ p$, empty answer set.

Program $a \leftarrow \mathit{not}\ b.\ b \leftarrow \mathit{not}\ c.\ c \leftarrow \mathit{not}\ a.$, three resource-based answer sets, $\{a\}$, $\{b\}$ and $\{c\}$.

Transposition into ASP and Complexity

In ASP: facts remains the same, each modal rule transposed as follows, where A' stands for $\dot{A}$, and A'' stands for $\neg L \neg L A$. For simplicity, A and $L A$ as well as $\not\!\!A$ and $L \neg L A$ are assumed to coincide.

$$A' \leftarrow A_1, \ldots, A_n, \mathit{not}\ B_1, \ldots, \mathit{not}\ B_m.$$
$$A \leftarrow A', A''.$$
$$\leftarrow A'', \mathit{not}\ A.$$
$$A'' \leftarrow \mathit{not}\ \mathit{no}A''.$$
$$\mathit{no}A'' \leftarrow \mathit{not}\ nA''.$$

The answer sets of the resulting program that maximize the assumptions A'' coincide, after removing the fresh atoms, with the resource-based answer sets of Π. Thus, they can be computed by using an answer set solver.

Implication: *RAS has the same complexity as AS*.

RAS: AS-like Characterization

Considers program Π as divided into layers according to Lifschitz & Turner Splitting Theorem.

- Applies modified Gelfond & Lifschitz Γ operator incrementally over layers, based upon modified immediate consequence operator.
- Resource-based answer sets are maximally supported sets of atoms (MCSs) w.r.t. given program Π.
- Answer sets (if any exists) are among the resource-based answer sets.

Query-answering under RAS

RAS enjoys Relevance and Modularity, i.e., every conclusion A can be derived from rules A depends upon, and subprograms can be to some extent semantically independent.

- Relevance allows for top-down prolog-like query answering.
- Relevance and modularity allow for contextual query-answering and optimized constraint checking.

A query-answering procedure for logic programming under RAS can be obtained, e.g., by suitably modifying and extending XSB-Resolution.

XSB-Resolution

Correct and complete query-answering procedure for datalog with negation under the well-founded semantics.

- Definite success and failure for atoms true or false w.r.t. the well-founded model.
- Efficient tabling mechanism: a *table* is associated to given program, recording atoms which succeed or fail, but also information about whether one subgoal depends on another, and whether the dependency is through negation (in order to detect undefined literals).
- prolog-like backtracking, previous table state restored upon backtracking.

RAS-XSB-Resolution

- Table $\mathcal{T}able(\Pi)$ associated to given program Π
- Definite success and failure and basic tabling "borrowed" from XSB-resolution. Both A and *not* A are recorded in $\mathcal{T}able(\Pi)$ upon success.
- Extended tabling:
  - $\mathcal{T}able(\Pi)$ is initialized by inserting, for each atom A occurring as the conclusion of some rule in Π, a fact *yesA* (fresh atom), meaning that A has still to be evaluated.
  - Insertion of either A or *not* A into the table "absorbs" *yesA* and prevents further evaluation attempts.

RAS-XSB-Resolution: specific features

Managing unary negative odd cycles (possibly with intermediate positive dependencies)

- Atom *A* is *forced to failure* if any possible derivation incurs into *not A* directly, i.e., not through layers of negation.
- In consequence of failure of *A*, fact *yesA* is removed from $\mathcal{T}able(\Pi)$ (if present).

RAS-XSB-Resolution: specific features

Managing non-unary negative cycles (possibly with intermediate positive dependencies)

- Literal *not A* is *allowed to succeed* if *A* does not fail, rather any derivation of *not A* incurs through layers of negation again into *not A* (undefined under XSB-Resolution);
- In consequence of success of *not A*, fact *yesA* is removed from $\mathcal{T}able(\Pi)$ (if present), and fact *not A* is added to $\mathcal{T}able(\Pi)$.
- In case however *not A* is *allowed to succeed*, if the parent subgoal fails then *yesA* is restored and *not A* is removed.

Properties of RAS-XSB-Resolution: Basic

Thanks to Relevance of RAS we obtain soundness and correctness.

Theorem

RAS-XSB-resolution is correct and complete w.r.t. resource Answer Set semantics, in the sense that, given program Π*, query* $?- A$ *succeeds under RAS-XSB-resolution with an initialized* $\mathcal{T}able(\Pi)$ *iff there exists resource-based answer set* M *for* Π *where* $A \in M$*.*

Properties of RAS-XSB-Resolution: Basic

Thanks to Modularity of RAS we get contextual query-answering.

Theorem

RAS-XSB-resolution is contextually correct and complete w.r.t. resource Answer Set semantics, in the sense that, given program Π *and query sequence* $?- A_1, \ldots, ?- A_k$, $k > 1$, *we have that, for* $\{B_1, \ldots, B_r\} \subseteq \{A_1, \ldots, A_k\}$ *and* $\{D_1, \ldots, D_s\} \subseteq \{A_1, \ldots, A_k\}$, *the queries* $?- B_1, \ldots, ?- B_r$ *succeed while* $?- D_1, \ldots, ?- D_s$ *fail under RAS-XSB-resolution, iff there exists resource-based answer set* M *for* Π *where* $\{B_1, \ldots, B_r\} \subseteq M$ *and* $\{D_1, \ldots, D_s\} \cap M = \emptyset$.

In practice, the table is not reset.

Properties of RAS-XSB-Resolution: Constraints

Constraints of the form $\leftarrow C$, C atom. M is *admissible* w.r.t. such a constraint if $C \notin M$. Thanks to Modularity and Relevance we get locality in constraint-checking.

Theorem

Let Π *be an admissible program w.r.t. the constraints* $\leftarrow H_1, \ldots, \leftarrow H_h$ *(it has admissible answer sets). Let* $\leftarrow H_1, \ldots, \leftarrow H_k$, $k \leq h$ *be the relevant constraints for a query* $?\text{–}A$. *For each of the* H_i*s, let* $H_i' = not\ H_i$. *If* $?\text{–}A$ *succeeds and subsequent queries* $?\text{–}\ H_i'$, $i \leq k$, *contextually succeeds as well, then there exists some admissible resource-based answer set* M *for* Π *with* $A \in M$.

Concluding Remarks and Future Directions

- Checking constraints on the state of $\mathcal{T}able(\Pi)$ left by a query may (together with 'smart' heuristics) alleviate the efficiency problem of constraint-checking.
- RAS-XSB-resolution is applicable to non-ground queries on ground programs. Extension: non-ground programs (principles and techniques proposed by Bonatti, Pontelli & Son).
- Under way: full detailed definition obtained by suitably extending XSB definitions.
- Future work: full efficient implementation.