Solver Fast Prototyping for Reduct-based ELP Semantics

Stefania Costantini
Università de L'Aquila

Andrea Formisano
Università di Udine

39th Italian Conference on Computational Logic
Rome, June 26–28, 2024

Answer Set Programming

Syntax

An ASP program Π is a collection of propositional rules of the form

$r: \quad A_1 \vee \ldots \vee A_g \leftarrow L_1, \ldots, L_n$

- the A_is are atoms, the L_js are atoms or *naf*-literals
- the left-hand side and the right-hand side of r are called *head* and *body*, resp.
- a rule with empty body is a *fact*
- a rule with empty head is a *constraint*

Semantics

Semantics of Π is given in terms of *answer sets*

- a set M of atoms is an answer set for Π if it is a minimal model of the reduct Π^M
- Π^M obtained by deleting from Π the rules in which a body literal is false w.r.t. M

Epistemic Logic Programs

Extend ASP syntax:

- *objective* literals: those in "regular" ASP
- *subjective* literals of the form $\mathbf{K}\,L$, for L objective literal
- subjective literals can occur in bodies of rules

Example:

$$
\begin{aligned}
a &\leftarrow \; not\, b \\
b &\leftarrow \; not\, a \\
e &\leftarrow \; not\, \mathbf{K}\, f \\
f &\leftarrow \; not\, \mathbf{K}\, e
\end{aligned}
$$

Epistemic Logic Programs

Intuitive semantics:

- $\mathbf{K}\,L$ predicates on the truth of L in all answer sets of the program
- other operators: $\mathbf{M}L$ ($\equiv$ *not* $\mathbf{K}$ *not* L) ***not*** L ($\equiv$ *not* $\mathbf{K}$ L)
- notion of epistemic interpretation, a set of sets of atoms
- and of world view, a set of answer sets of a program
- a semantics $\mathcal{S}$ maps each program to a set of world views

The program

$$\begin{aligned} a &\leftarrow \; not\, b \\ b &\leftarrow \; not\, a \\ e &\leftarrow \; not\, \mathbf{K} f \\ f &\leftarrow \; not\, \mathbf{K}\, e \end{aligned}$$

has 2 world views made of 2 answer sets each:

$$[\{a, e\}, \{b, e\}] \qquad [\{a, f\}, \{b, f\}]$$

$\mathbf{K}\,e$ is true in the first world view, $\mathbf{K} f$ is true in the second one

Epistemic Logic Programs: which semantics?

Definition (Epistemic reduct [Gelfond 1991,1994])

Given a program Π and an epistemic interpretation $\mathcal{W}$, the reduct $\Pi^{\mathcal{W}}$ is obtained from Π by replacing each subjective literal $\mathbf{K}\,L$

- with $\top$ if $\mathcal{W} \models L$ (i.e., L is true in each set of $\mathcal{W}$)
- with $\bot$, otherwise

Then, $\mathcal{W}$ is a world view of Π if it is the set of answer sets of $\Pi^{\mathcal{W}}$

Example: the program

$$a \leftarrow \mathbf{K}\,a$$

has 2 world views: $[\{\}]$ and $[\{a\}]$

...the world view $[\{a\}]$ appears somehow counterintuitive...

Epistemic Logic Programs: which semantics?

Definition (Epistemic reduct [Gelfond 2011])

Given a program Π and an epistemic interpretation $\mathcal{W}$, the reduct $\Pi^{\mathcal{W}}$ is obtained from Π by

- replacing each subjective literal $\mathbf{K}\,L$ with $\bot$ if $\mathcal{W} \not\models L$
- remove all subjective literals of the form $not\,\mathbf{K}\,L$
- replacing each remaining subjective literal of the form $\mathbf{K}\,L$ with L

Then, $\mathcal{W}$ is a world view of Π if it is the set of answer sets of $\Pi^{\mathcal{W}}$

Example: Now, the program

$$a \leftarrow \mathbf{K}\,a$$

has 1 world view: $[\{\}]$

but the story is not over...

Epistemic Logic Programs: which semantics?

Many other proposals

- K15: Kahl, Watson, Balai, Gelfond, Zhang, 2015
- S16: Shen and Eiter, 2016
- F15: Fariñas del Cerro, Herzig, Su, 2015

- Su, 2019, 2021
- FAEEL: Cabalar, Fandinno, Fariñas del Cerro, 2020

These semantics differ and meet different desirable properties...

Epistemic Logic Programs: which semantics?

<table>
<tr><th rowspan="2">Program</th><th colspan="4">World views</th></tr>
<tr><th>G94</th><th>G11/FAEEL</th><th>K15</th><th>F15/S16</th></tr>
<tr><td>$a \leftarrow not\, \mathbf{K}\, not\, a$</td><td colspan="2">$[\emptyset],\ [\{a\}]$</td><td colspan="2">$[\{a\}]$</td></tr>
<tr><td>$a \vee b$
$a \leftarrow not\, \mathbf{K}\, not\, b$</td><td colspan="2">none</td><td colspan="2">$[\{a\}]$</td></tr>
<tr><td>$a \vee b$
$a \leftarrow \mathbf{K}\, not\, b$</td><td colspan="2">$[\{a\}],\ [\{a\},\{b\}]$</td><td colspan="2">$[\{a\},\{b\}]$</td></tr>
<tr><td>$a \leftarrow b$
$b \leftarrow not\, \mathbf{K}\, not\, a$</td><td colspan="2">$[\emptyset],\ [\{a,b\}]$</td><td colspan="2">$[\{a,b\}]$</td></tr>
<tr><td>$a \leftarrow not\, \mathbf{K}\, not\, b$
$b \leftarrow not\, \mathbf{K}\, not\, a$</td><td colspan="2">$[\emptyset],\ [\{a,b\}]$</td><td colspan="2">$[\{a\},\{b\}]$</td></tr>
<tr><td>$a \leftarrow not\, \mathbf{K}\, not\, b \wedge not\, b$
$b \leftarrow not\, \mathbf{K}\, not\, a \wedge not\, a$</td><td colspan="3">$[\emptyset], [\{a\},\{b\}]$</td><td>$[\{a\},\{b\}]$</td></tr>
<tr><td>$a \leftarrow \mathbf{K}\, a$</td><td>$[\emptyset], [\{a\}]$</td><td colspan="3">$[\emptyset]$</td></tr>
<tr><td>$a \leftarrow \mathbf{K}\, a$
$a \leftarrow not\, \mathbf{K}\, a$</td><td>$[\{a\}]$</td><td colspan="3">none</td></tr>
</table>

Epistemic Logic Programs: properties

Some “desirable” properties of ELP semantics

- Epistemic splitting property:
  Reminiscent of splitting in ASP: an ELP can be split in *top/bottom* parts (w.r.t. epistemic operators) and world views can be computed incrementally
- Subjective constraint monotonicity:
  given Π program and c subjective constraint, it holds that
  “*$\mathcal{W}$ is a world view of $\Pi \cup \{c\}$ iff $\mathcal{W}$ is a world view of Π and $\mathcal{W}$ satisfies c*”.
- Foundedness:
  Intuition: Atoms occurring in sets within a world view cannot have been derived via positive cyclic dependencies
  For example semantics G94 does not meet foundedness: it admits $[\{a\}]$ as world view of the program $a \leftarrow \mathbf{K}\, a$
  In such world view a is derived from the fact that $\mathbf{K}\, a$ holds in the unique answer set $\{a\}$

Epistemic Logic Programs: another semantics!

A new reduct-based semantics

- consider subjective literals $\mathbf{K}\,A$ and $\mathbf{K}\,not\,A$ as knowledge atoms
- an ELP Π is seen as an ASP program involving knowledge atoms
- CF24-adaptation of Π w.r.t. $\mathcal{W}$ is obtained by
  - whenever $\mathcal{W} \models G$, in all non-unit rules with head G substitute head G with $\mathbf{K}\,G$, and add new rule $G \leftarrow \mathbf{K}\,G$
  - whenever $\mathcal{W} \models not\,G$, add new rule $\mathbf{K}\,not\,G \leftarrow not\,G$
- $\mathcal{W}$ is a CF24-world view of Π if is the set of answer sets of the CF24-adaptation of Π w.r.t. $\mathcal{W}$ (ignoring knowledge atoms)

WORK IN PROGRESS:
assessing the properties of CF24, such as foundedness

ELP-solver fast prototyping

When looking for the “right” semantics, while designing/developing a new one, or just to compare different semantics

- it might be useful to put candidate semantics to trial on simple examples
- Do not want to implement a full-blown solver
- All reduct-based semantics for ELP share a “common characteristics”

Goal: design a generic solver pipeline that can be easily instantiated to compute different reduct-based semantics

The ELP-solver pipeline

Let be given: a semantics $\mathcal{S}$ based on a notion $\mathcal{R}$ of reduct and (possibly) requiring a post-processing $\mathcal{P}$ to filter the candidate world views (some semantics impose additional minimality criterion)

The ELP-solver pipeline

Let be given: a semantics $\mathcal{S}$ based on a notion $\mathcal{R}$ of reduct and (possibly) requiring a post-processing $\mathcal{P}$ to filter the candidate world views (some semantics impose additional minimality criterion)

The procedure is composed of a sequence of modules

1. A module $M_{\mathcal{W}}$ that computes all epistemic interpretations $\mathcal{W}_1, \ldots, \mathcal{W}_k$ for Π (i.e., all sets of subsets of At_{Π})
2. A module M_{red} that applies, for each $\mathcal{W}_i$, the reduct $\mathcal{R}$ to Π, and generates the reduct program Π_i
3. A module M_{ASP} that computes the set SMs_i of answer sets Π_i
4. In case a post-processing $\mathcal{P}$ is required, a module $M_{\mathcal{P}}$ applying $\mathcal{P}$ to select the desired candidate SMs_i's;
5. A module M_{chk} that checks each SMs_i and selects those which are world views w.r.t. $\mathcal{S}$ (i.e., they coincide with the corresponding $\mathcal{W}_i$)

The pipeline in ASP Chef

How to quickly implement the pipeline?

The pipeline in ASP Chef

How to quickly implement the pipeline?

using ASP Chef

The ASP Chef system

- Notion of recipe, a sequence of ingredients
- Ingredients are instances of basic operations processing/producing a sequence of sets of atoms
- ... etc ... the details in the tutorial by Mario Alviano

The notion of ASP Chef recipe matches the idea of ELP-solver pipeline

The recipe — input ELP encoding

- The input ELP program have to be encoded as a set of facts
- specific terms to represent **K** and *naf*-literals
  - *not* L is represented by `neg(`L`)`
  - **K** L is represented by `k(`L`)`
- each rule

$$A_1 \vee \ldots \vee A_g \leftarrow L_1, \ldots, L_n$$

is encoded as the fact

rule(head(A_1,...,A_g), body(L_1,...,L_n)).

Example:

```
rule(head(a), body(neg(b))).
rule(head(b), body(neg(a))).
rule(head(e), body(neg(k(f)))).
rule(head(f), body(neg(k(e)))).
```

The recipe — main steps

The main steps/ingredients:

- extract, rule heads, rule bodies, atoms, objective and subjective literals from input ELP encoding (`hlit/1, blit/1, rule_body/2, rule_head/2,...`)
- guess epistemic interpretations
- detect epistemic consequences of each guessed epistemic interpretation (`modeledByW/1`)
- compute the reduct (w.r.t. an epistemic interpretation)
- compute answer sets of the reduct
- check if the epistemic interpretation is a world view

G94 Semantics

A SEARCH MODELS ingredient evaluating the following ASP program computes the G94-reduct by determining a rewriting of input ELP lits/rules

```
% detect ``substitutes'' for rule literals:
red_blit(k(L),true) :- blit(k(L)), modeledByW(L).
red_blit(k(L),false) :- blit(k(L)), not modeledByW(L).
red_blit(neg(L),neg(L)) :- blit(neg(L)), @functor(L)!="k".
red_blit(L,L) :- blit(L), @functor(L)!="neg",
    @functor(L)!="k".
red_blit(neg(k(L)),false) :- blit(neg(k(L))), modeledByW(L).
red_blit(neg(k(L)),true) :- blit(neg(k(L))), not modeledByW(L).

% reduced rules head and body literals :
red_rule_head(rule(H,B), @argument(H,I)) :- rule(H,B), I =
    1..@arity(H).
red_rule_body(rule(H,B), R) :- rule_body(rule(H,B), L),
    red_blit(L,R).
```

Computing answer sets of the reduct

A SEARCH MODELS ingredient evaluating the following ASP computes the answer sets of a reduct program

```
% detect falsified reduced rules bodies:
red_body_false(R) :- red_rule_body(R,false).

% infer true literals w.r.t. reduced rules
true(L):red_rule_head(rule(H,B),L) :-
    rule(H,B),~not red_body_false(rule(H,B));
  true(N):red_rule_body(rule(H,B),N), @functor(N)!="neg",
      @functor(N)!="true";
  not true(M):red_rule_body(rule(H,B),neg(M)),
      @functor(M)!="neg", @functor(M)!="false";
  not not true(M):red_rule_body(rule(H,B),neg(neg(M))),
      @functor(M)!="false".
```

G11 Semantics

A SEARCH MODELS ingredient evaluating the following ASP program computes the G11-reduct

```
% detect ``substitutes'' for rule literals:
red_blit(k(L),false) :- blit(k(L)), not modeledByW(L).
red_blit(neg(k(L)),false) :- blit(neg(k(L))), modeledByW(L).
red_blit(neg(k(L)),true) :- blit(neg(k(L))),
    not modeledByW(@argument(@argument(L,1),1)).
red_blit(k(L),L) :- blit(k(L)), modeledByW(L).
red_blit(neg(L),neg(L)) :- blit(neg(L)), @functor(L)!="k".
red_blit(L,L) :- blit(L), @functor(L)!="neg",
    @functor(L)!="k".

% reduced rules head and body literals (same as G94):
red_rule_head(rule(H,B), @argument(H,I)) :- rule(H,B), I =
    1..@arity(H).
red_rule_body(rule(H,B), R) :- rule_body(rule(H,B), L),
    red_blit(L,R).
```

K15 Semantics

A SEARCH MODELS ingredient evaluating the following ASP program computes the K15-reduct

```
% detect ``substitutes'' for rule literals:
red_blit(k(L),false) :- blit(k(L)), not modeledByW(L).
red_blit(neg(k(L)),true) :- blit(neg(k(L))), not modeledByW(L).
red_blit(neg(k(neg(L))),neg(neg(L))) :- blit(neg(k(neg(L)))),
    modeledByW(neg(L)).
red_blit(k(L),L) :- blit(k(L)), modeledByW(L).
red_blit(neg(k(L)),neg(L)) :- blit(neg(k(L))),
    @functor(L)!="neg", modeledByW(L).
red_blit(neg(L),neg(L)) :- blit(neg(L)), @functor(L)!="k".
red_blit(L,L) :- blit(L), @functor(L)!="neg",
    @functor(L)!="k".

% reduced rules head and body literals (same as G94):
red_rule_head(rule(H,B), @argument(H,I)) :- rule(H,B), I =
    1..@arity(H).
red_rule_body(rule(H,B), R) :- rule_body(rule(H,B), L),
    red_blit(L,R).
```

CF24 Semantics

The following ASP program computes the CF24-reduct

```
red_blit(k(neg(L)),knowN(L)) :- blit(k(neg(L))).
red_blit(k(L),knowP(L)) :- blit(k(L)), @functor(L)!="neg".
red_blit(neg(L),neg(L)) :- blit(neg(L)), @functor(L)!="k".
red_blit(L,L) :- blit(L), @functor(L)!="neg", @functor(L)!="k".
red_blit(neg(k(L)),neg(knowP(L))):-blit(neg(k(L))),red_blit(k(L),knowP(L)).
red_blit(neg(k(L)),neg(knowN(L))):-blit(neg(k(L))),red_blit(k(L),knowN(L)).
red_hlit(A,knowP(A)) :- hlit(A), modeledByW(A).
red_hlit(A,A) :- hlit(A), not modeledByW(A).
nonunit(R) :- rule_body(R, L), L!=true.
changehead(rule(H,B)) :- nonunit(rule(H,B)), I=1..@arity(H),
     L=@argument(H,I), @functor(L)!="neg",modeledByW(L).

red_rule_head(rule(H,B), knowP(L)) :- changehead(rule(H,B)),
     I=1..@arity(H), L=@argument(H,I), @functor(L)!="neg",modeledByW(L).
red_rule_head(rule(H,B), @argument(H,I)) :- rule(H,B), I=1..@arity(H),
     not changehead(rule(H,B)).
red_rule_body(rule(H,B), R) :- rule_body(rule(H,B), L), red_blit(L,R).
rule(head(L),body(knowP(L))) :- modeledByW(L), @functor(L)!="neg".
red_rule_head(rule(head(L),body(knowP(L))),L) :- modeledByW(L),
     @functor(L)!="neg".
red_rule_body(rule(head(L),body(knowP(L))),knowP(L)) :- modeledByW(L),
     @functor(L)!="neg".
rule(head(knowN(L)),body(neg(L))) :- modeledByW(neg(L)).
red_rule_head(rule(head(knowN(L)),body(neg(L))),knowN(L)):-modeledByW(neg(L)).
red_rule_body(rule(head(knowN(L)),body(neg(L))),neg(L)):-modeledByW(neg(L)).
```

Conclusion

- Many semantics exists for ELP, no consensus
- some meet desirable properties (many do not)
- many are defined by introducing a notion of (epistemic) reduct
- our own proposal for a (founded) reduct-based semantics
- fast ELP-solver pipeline
- prototype implementation of the pipeline in ASP Chef

Next/current work?

- improve the ASP Chef implementation
- improve efficiency (use of epistemic guesses, i.e., epistemic literals holding in a world view)
- other semantics not (directly) based on a notion of reduct?
- complete the study of CF24, its relation with other semantics, properties it satisfies, ...
- ...