Answer Set Programming in Intuitionistic Logic

Aleksy Schubert and Paweł Urzyczyn
Uniwersytet Warszawski

TYPES 2016, Novi Sad

Logic Programming

This program:

p :− .
q :− p, r.
s :− p.

has the unique model:

$\{p, s\}$

(or implicitly $\{p, s, \neg q, \neg r\}$)

What is Answer Set Programming?

It is a PR-oriented renaming of

stable model semantics,

an approach to deal with negation in logic programs.

Like this one:

$s :- \neg p, q.$
$r :- \neg s, q.$
$p :- r.$
$q :- \neg s.$

What is a model of this program?

$s \mathrel{:-} \neg p, q.$
$r \mathrel{:-} \neg s, q.$
$p \mathrel{:-} r.$
$q \mathrel{:-} \neg s.$

First try: $\{p, r\}$ (i.e., implicitly $\{p, r, \neg q, \neg s\}$).

$s \mathrel{:-} \neg p, q.$
$r \mathrel{:-} \neg s, q.$
$p \mathrel{:-} r.$
$q \mathrel{:-} \neg s.$

This will not work. It proves q, and it should not!
This model is *unstable*, because it is *unsound*.

What is a model of this program?

s :– $\neg p, q$.
r :– $\neg s, q$.
p :– r.
q :– $\neg s$.

Second try: $\{p, s\}$ (i.e., implicitly $\{p, s, \neg q, \neg r\}$).

s :– $\neg p, q$.
r :– $\neg s, q$.
p :– r.
q :– $\neg s$.

It proves nothing, and we want to derive p and s.
This model is unstable, because it is *insufficient*.

What is a model of this program?

s :– $\neg p, q$.
r :– $\neg s, q$.
p :– r.
q :– $\neg s$.

That will work: $\{p, q, r\}$ (i.e., implicitly $\{p, q, r, \neg s\}$).

s :– $\neg p, q$.
r :– $\neg s, q$.
p :– r.
q :– $\neg s$.

This is a *stable model* or *answer set* for the program.
It proves exactly p, q, and r.

Things are not so easy in general

Some programs have no stable model at all,
for example this one:

$p :- \neg p.$

Some programs have more than one stable model,
for example this one:

$p :- \neg q.$
$q :- \neg p.$

has two stable models, namely $\{p, \neg q\}$ and $\{q, \neg p\}$.

The existence of a stable model is an NP-complete problem.

Existence and Entailment

Write $P \models_{SMS} X$, when every stable model of P satisfies X.

A program P has no stable model
if and only if $P \models_{SMS} X$,
for some X that does not occur in P.

The stable entailment is therefore co-NP-complete.

∘

Interpretation in Intuitionistic Propositional Calculus (IPC)

Given a program P and an atom X
we define a formula φ so that:

$P \models_{SMS} X$ if and only if $\vdash_{\mathbf{int}} \varphi$.

∘

What does $P \models_{SMS} X$ mean?

It means that, for every stable model M of P,

(1) Either X holds in M, or

(2) The model M is unstable, because:

(2a) It is unsound (some $Y \notin M$ has a proof), or

(2b) It is insufficient (some $Y \in M$ has no proof).

∘

Our formula

The formula φ is of shape

$\psi_1 \to \cdots \to \psi_d \to 0$

so proving it amounts to proving the judgment

$\psi_1, \ldots, \psi_d \ \vdash \ 0$.

We select the assumptions $\psi_1, \ldots, \psi_d$ so that any proof of 0 must force either (1) or (2a) or (2b) in every model.

Taking every model into account

The initial proof goal is 0. Let $X_1, \ldots, X_n$ be all propositional atoms in P, including X. The first n assumptions are:

$$\psi_1 = (X_1 \to 1) \to (\overline{X}_1 \to 1) \to 0,$$
$$\ldots$$
$$\psi_n = (X_n \to n) \to (\overline{X}_n \to n) \to n - 1.$$

In order to prove 0 we must prove the goal n under all possible choices of X_i vs $\overline{X}_i$.

Every such choice represents a different model.

If X simply holds in a model...

...then we use an assumption formula $X \to n$
and the proof is completed.

Otherwise the model is unstable. . .

. . . and we must *prove it.*

We include all clauses of P as assumptions,
but we rename all $\neg X_j$ as $\overline{X}_j$, and all X_j as X_j!

$\circ$

The easy case. . .

. . . occurs when the model is unsound.
It proves some X_i, but we have chosen $\overline{X}_i$.

This case is handled by assumption formulas of the form

$$\overline{X}_i \to X_i! \to n$$

so that the goal n can be proved if we have some X_i
and we can derive $X_i!$ from the (renamed) program P.

The other case occurs. . .

. . . when the model is unstable because it is insufficient.
It cannot prove some X_i which occurs in the present choice.
For this we have assumption formulas of the form

$$X_i \to X_i? \to n$$

Here, $X_i?$ means „X_i is not derivable in the model".

∘

An insufficient example[1]

The model $\{p, q, \neg r, \neg s\}$ is insufficient for the program:

$$p \text{ :- } \neg r, q. \qquad q \text{ :- } \neg s, p. \qquad p \text{ :- } \neg r, s.$$

We will prove $p?$ using these assumptions:

$$(p? \to K_1) \to (p? \to K_3) \to p?$$

$$(q? \to K_2) \to q? \qquad r? \qquad s?$$

$$q? \to K_1, \qquad p? \to K_2, \qquad r? \to K_3,$$

$$r \to K_1, \qquad s \to K_2, \qquad s? \to K_3$$

To prove $p?$ one must derive both K_1 and K_3 from the additional assumption $p?$. The latter cannot be easier.

To derive K_1 we may try proving $q?$
That is, proving K_2 with the added assumption $q?$
We obtain K_2 from $p?$ This represents a loop in a proof.

[1] :)

Summing up. . .

We know how to represent ASP in IPC.

We are sure that the target fragment of IPC is co-NP-complete.

So: we can program ASP in IPC.