Knowledge Representation in the Languages of Logic Programs under Answer Set Semantics.

Michael Gelfond

September, 2023

Introduction

My original interest in KR started with attempts to

- Achieve better understanding of the human mind and structure of the universe by discovery, refinement, and formalization of the basic categories of our language and thought.
- Develop a good methodology for the design and implementation of transparent and elaboration tolerant software systems.

In my view the most basic of these categories is that of

Rational Belief.

This view contributed substantially to the development of knowledge representation language Answer Set Prolog (ASP) and its extensions including

- ASP with sorts, sets and aggregates
- ASP capable of representing rare events
- ASP capable of representing random events and their probabilities
- Constraint ASP
- Causal ASP

A theory in an ASP based language defines possible collections of beliefs of the rational agent associated with this theory.

Such a collection, called an answer set, consists of literals satisfying

- the axioms of the theory and
- the Rationality Principle which prohibits the agent from believing things he is not forced to believe.

As an example, consider an ASP program consisting of a rule

$$\mathrm{p}(0) \leftarrow \mathrm{q}(0).$$

and two facts

$$\mathrm{q}(0). \qquad \neg \mathrm{p}(1).$$

The answer set of this program is

$$\{\mathrm{q}(0),\ \mathrm{p}(0),\ \neg \mathrm{p}(1)\}$$

The agent associated with the program believes $\mathsf{q}(0)$ and $\mathsf{p}(0)$ to be true and $\mathsf{p}(1)$ to be false.

He, however, has no beliefs concerning the truth values of $\mathsf{p}(2)$, $\mathsf{q}(1)$, etc.

Plan of the Talk

In what follows I use examples to discuss several ASP based languages and methodology of their use for solving a number of classical KR problems, such as representing

- defaults and their exceptions
- effects of actions
- random events and their probabilities
- causal relations

If time permits I also say a few words about industrial applications of these ideas.

Negations and Defaults

"$\neg p$" – "p is believed to be false".

"not p" – "it is not believed that p is true".

"not$\neg p$" – "it is not believed that p is false," i.e., "p may be true" or "p is possible".

Default "mothers normally love their children" is represented as

$$\begin{aligned} loves(X, Y) \quad \leftarrow \quad & mother(X, Y), \\ & \text{not } \neg loves(X, Y). \end{aligned}$$

If X is a mother of Y and she may love Y then she does.

The default, used together with

$$mother(mary, sam)$$

allows us to conclude

$$loves(mary, sam)$$

Exceptions to Defaults

There are three types:

Strong:	A new fact directly contradicts the default's conclusion.
Weak:	Stops application of the default without defeating its conclusion.
Indirect:	Exception discovered in the process of reasoning.

The first two are easily represented in ASP. The third is not.

Indirect Exceptions

Consider a program Π consisting of

- default "p is normally true",

$$p(X) \leftarrow \text{not } \neg p(X).$$

- and rule

$$q(X) \leftarrow p(X).$$

The program entails $q(0)$ but, after the addition of

$$\neg q(0)$$

it becomes inconsistent.

"0" is an indirect exception to the default. Intuitively we want Π to be consistent and entail $\neg q(0)$.

ASP with Rare Events (CR-Prolog)

To represent rare events we add to ASP a new language construct

$$\mathsf{l} \overset{+}{\leftarrow} \mathsf{body}$$

called consistency-restoring rule (cr-rule).

The rule says that if the reasoner associated with the program believes the body of the rule, then it "may possibly" believe its head.

However, this occasion is very rare and the rule may be used only if there is no way to obtain a consistent set of beliefs by using only regular rules of the program.

To deal with inconsistency of Π we expand our representation of defaults. Now a default

$$p(X) \quad \leftarrow \quad not\, \neg p(X).$$

will come with Contingency Axiom:

$$\neg p(X) \overset{+}{\leftarrow}$$

The rest is unchanged.

$$q(X) \quad \leftarrow \quad p(X).$$

$$\neg q(0)$$

The answer set of the new program is

$$\{\neg q(0), \neg p(0),\ p(1),\ q(1) \dots\}$$

Effects of Actions

Direct effect of an action in "Causal ASP" is captured by causal law

$$\begin{array}{llll} m(f,a): & holds(f,I) & \leftarrow & occurs(a,I-1), \\ & & & body, \\ & & & \neg holds(f,I-1), \\ & & & \neg ab(m(f,a)) \end{array}$$

where $m(f,a)$ is the name of the law and time-steps occurring in the body are smaller then I.

The law says that, under normal circumstances, if "$body$" holds then the execution of "a" causes "f" to become true.

Normality of the law is established by default

$$\neg ab(M) \leftarrow \text{not } ab(M)$$

where M ranges over names of causal laws.

Direct and indirect exceptions to the default can be specified as usual.

The frame problem can be solved by the Inertia Axiom which says that things normally stay as they are. This is represented by default

$$\begin{array}{rcl} holds(F, I) & \leftarrow & holds(F, I-1), \\ & & \text{not } \neg holds(F, I) \end{array}$$

The problems of understanding and formalizing default reasoning and reasoning about actions and change remained unsolved for decades.

They are mostly solved now.

These solutions are used in solving multiple planning and diagnostic problems, problems in robotics, etc.

Randomness and Probability

So far we showed how such basic concepts as defaults and some causal relations can be expressed in terms of beliefs.

This is also true for randomness and probability.

This is not surprising since probability of an event can be viewed as the “degree of belief that it has taken place, or that it will take place” (G. Boole).

Reasoning about probability is simply reasoning about degrees of belief.

This can be done in P-log.

P-log is obtained from ASP by adding

- non-boolean functions and atoms of the form $f(\bar{t}) = y$
- a special sort of actions, called random experiments, described by statements of the form

  $$\text{random}(m, f(\bar{t}), p)$$

  where m is the name of an experiment randomly selecting the value of $f(\bar{t})$ from $\{Y : p(Y), Y \in range(f)\}$

- standard axioms

$$val(f(X), I) = y_0 \text{ or} \ldots \text{or } val(f(X), I) = y_n \leftarrow \begin{array}{l} random(m, f, p), \\ occurs(m, I - 1) \end{array}$$

  where $range(f) = \{y_0, \ldots, y_n\}$

- and

$$\leftarrow f(X) = Y, \neg p(Y)$$

- statements of the form

$$pr(m, f(\bar{t}) = y \mid_c body) = v$$

"Given $body$, the probability of $f(\bar{t})$ taking on the value y as the result of random experiment m is v".

Example: Monty Hall Problem

A player selects one of three closed doors, behind one of which there is a prize, put there by Monty.

After selection is made, Monty is obligated to open one of the remaining doors which does not contain the prize.

The player can switch his selection to the other unopened door, or stay with his original choice.

Does it matter if he switches? The answer is yes.

However, the lady who published the solution received thousands of letters from her readers—the vast majority of which, many with PhD in math, disagreed with her answer.

This phenomenon was clearly recognized by George Boole who wrote:

“I think it to be one of the peculiar difficulties of the theory of probabilities, that its difficulties sometimes are not seen. The solution of a problem may appear to be conducted according to the principles of the theory as usually stated; it may lead to a result susceptible of verification in particular instances; and yet it may be an erroneous solution.”

One possible explanation of this difficulty is the distance between the informal view of probability as “degree of belief” and its formal definition via “probability distribution” or other probabilistic models.

It is not always easy to build the probabilistic model well suited for a particular task. A wrong model can be the reason for an error.

One possible way to deal with the problem is to define probability with respect to explicitly stated knowledge base of the reasoner solving this task.

This is the approach of P-log.

Problem Description

Player's knowledge in P-log:

$$\begin{aligned}\text{Declarations}: \quad & \mathsf{doors} = \{1,2,3\}\\ & \mathsf{selected}, \mathsf{prize}, \mathsf{open} : \mathsf{doors}\\ & \mathsf{canOpen} : \mathsf{doors} \rightarrow \mathsf{boolean}\end{aligned}$$

$$\begin{aligned}\text{Rules}: \quad \mathsf{canOpen}(D) &\leftarrow \ \mathsf{not}\ \neg \mathsf{canOpen}(D)\\ \neg \mathsf{canOpen}(D) &\leftarrow \ \mathsf{selected} = D\\ \neg \mathsf{canOpen}(D) &\leftarrow \ \mathsf{prize} = D\end{aligned}$$

The first rule is the default: "Normally, a door can be opened".

The next two rules are exceptions to this default.

For simplicity we assume that the player has already selected door one:

$$selected = 1$$

Values of variables $prize$ and $open$ are chosen by random experiments m_0 and m_1 performed by Monty.

Value of $prize$ is chosen from the set of all doors:

$$random(m_0, prize)$$

Value of $open$ is chosen from the set of doors which can be open according to the rules of the game:

$$random(m_1, open, canOpen)$$

According to the semantics of P-log:

$random(m_0, prize)$

can be replaced by rule

$$prize = 1 \text{ or } prize = 2 \text{ or } prize = 3$$

and

$random(m_1, open, canOpen)$ by rules

$$open = 1 \text{ or } open = 2 \text{ or } open = 3$$

$$\leftarrow \neg canOpen(D), open = D$$

Let us denote this program by Π.

Consider Π' obtained by adding to Π

- atoms
  $$\mathsf{obs}(\mathsf{open} = 2) \quad \mathsf{obs}(\mathsf{prize} \neq 2)$$
  where obs stands for the player's observation;
- To deal with observations we need general "Reality Check" axiom
  $$\leftarrow \mathsf{obs}(A), \neg A$$
  which guarantees that the program's conclusions do not contradict observations.

Π' defines possible worlds[1]:

$W_1 = \{prize = 1, canOpen(2), canOpen(3), open = 2\}$
$W_2 = \{prize = 3, canOpen(2), open = 2\}$

Probabilistic measures of W_1 and W_2 are

$$\mu(W_1) = 1/3 \times 1/2 = 1/6$$
$$\mu(W_2) = 1/3 \times 1 = 1/3$$

Changing the door doubles player's chances to win.

[1]Non-random $select = 1$ is not included.

Discussion

Probabilistic model of the domain consists of rules of the game and a particular scenario written in P-log.

It is very close to the English description of the story.

Formal model is the main part of problem solution. The rest is automatic.

Its worth noting that if rule

$$\neg canOpen(D) \leftarrow prize = D$$

were removed from the program then changing the door would not change the chance of winning.

Causality

Consider several stories in which an officer orders the group of guards to shoot a prisoner.

In the first story the guards follow the order.

In the second one guard refuses to do that.

In the third all of them disobey.

In the forth the officer orders the execution and later, unexpectedly, observes the prisoner to be alive.

In each story we will look for causal relations between actions of the officer and the guards and the final state of the prisoner.

Causal Theories

Reasoning about causal relations is done in a subclass of ASP programs called causal theories.

These limitations, together with the collection of general axioms included in every causal theory, allow causal interpretation of the $\leftarrow$ in the theory's laws.

Formalizing the stories

The signature of our theory T contains a group $g = \{1, 2\}$ of guards, actions $shoot(G)$ where $G \in g$ and $order(g, shoot)$ – "the officer orders the group to shoot", and inertial fluents $ordered(g, shoot)$ and $alive$.

$$\begin{aligned} m_0 :\ ordered(g, shoot, I) \quad \leftarrow \quad & occurs(order(g, shoot), I - 1)), \\ & \neg ordered(g, shoot, I - 1), \\ & \neg ab(m_0, I) \end{aligned}$$

$$\begin{aligned} m_1(G) :\ occurs(shoot(G), I) \quad \leftarrow \quad & ordered(g, shoot, I), \\ & in(G, g), \\ & \neg ab(m_1(G), I) \end{aligned}$$

$$\begin{aligned} m_2 : \neg alive(I) \quad \leftarrow \quad & card\{G : occurs(shoot(G), I-1)\} > 0, \\ & alive(I-1), \\ & \neg ab(m_2, I) \end{aligned}$$

where $card$ is the cardinality of the set.

The rule says that shooting by at least one guard will cause the prisoner's death.

The first story is formalized by causal theory $\mathsf{T}(S_1) = \mathsf{T} \cup S_1$ where

$$S_1 = \langle \mathrm{init}(\mathit{alive}), \mathrm{do}(\mathsf{order}(g, \mathit{shoot}), 0) \rangle$$

$\mathsf{do}(A, \mathrm{I})$ stands for deliberate execution of action A at step I.

$\mathrm{do}(\mathsf{order}(g, \mathit{shoot}), 0)$ initiates a chain of events leading to the prisoner's death.

It is called the death's "deliberate cause".

Shootings by the particular guards are "factual" or "physical" causes of this events.

In the second story T is used together with scenario

$$S_2 = \langle \text{init}(\mathit{alive}), \text{do}(\mathsf{order}(g, \mathit{shoot}), 0), \mathsf{do}(\neg \mathit{shoot}(1), 1) \rangle$$

where $\mathsf{do}(\neg \mathit{shoot}(1), 1)$ indicates that the first guard deliberately refused to follow the order.

Since the second shot anyway do$(\mathsf{order}(g, \mathit{shoot}), 0)$ is still the deliberate cause of death but now there is only one factual cause.

In the third story T is used together with

$$S_3 = S_1 \cup \{\mathsf{do}(\neg shoot(1), 1), \mathsf{do}(\neg shoot(2), 1)\}$$

The orders were deliberately disobeyed by all guards and the prisoner remained alive. This event has neither deliberate nor factual cause – it is simply a consequence of inertia.

One can, however, say that guards deliberate refusal to shoot,

$$\mathsf{do}(\neg shoot(1), 1), \mathsf{do}(\neg shoot(2), 1)$$

"prevented the death of the prisoner expected in S_1".

Finally, the forth story is represented by

$$S_4 = \langle \mathrm{init}(alive), \mathrm{do}(order(g, shoot), 0), obs(alive, 3) \rangle$$

The unexpected observation has a “causal explanation”

$$\{do(\neg shoot(1), 1), do(\neg shoot(2), 1)\}$$

The notion of causal theory together with the collection of definitions of various causal relations allowed us to obtain reasonable representations and reasonable answers to causal queries for all the examples we were able to investigate so far.

Discussion

ASP based languages were useful in understanding basic categories of language and thought and the development of KR methodology.

Some of the languages have efficient reasoning systems and non-trivial mathematical theory.

Others are still under development.

It is important to complete this work.

There are also more basic categories to look at.

Another big task is the axiomatization of large parts of commonsense knowledge.

An Industrial Application

A story by Gerhard Friedrich – former head of the Department for Configurations and Diagnosis at Siemens, Germany.

Worked on configuration problem for more than 20 years. Used different methods, including procedural programming, constraint programming, etc.

The problem is difficult. E. g. in telecommunication domain, just focusing on hardware:

- Up to 30.000 modules of 200 types
- Up to 1.000 frames of 50 types
- Up to 200 racks of 20 types
- Up to 10.000 cables of 50 types

Use of ASP allowed

- Reduction of initial development cost by 66%
- Reduction of yearly maintenance cost by 80%
- Productivity increase by 300% (no additional staff)
- Enhanced user interaction: explanations, incremental configuration, repair ...

In this case ASP delivered on its promise.

Gerhard gave an example of ASP program consisting of 12 rules.

Quote: " By this simple ASP program, we solved real world instances which could NOT be solved by the in-house tool"

In his estimate "ASP is a historical leap for AI".

We still need to

(a) improve efficiency of algorithms and implementations for ASP and its extensions;

(b) improve the methodology of refining initial knowledge bases to make them executable;

(c) teach knowledge representation and declarative programming.

References

Chitta Baral, Knowledge representation, Reasoning and Declarative Problem Solving, 2003, Cambridge University Press

Michael Gelfond and Yulia Kahl, Knowledge representation, Reasoning and the Design of Intelligent Agents: The Answer Set Programming Approach, 2014, Cambridge University Press

THANK YOU