Logic Programming Solution to the Frame Problem

Vladimir Lifschitz

University of Texas at Austin

June 11, 2012

What We Will Talk About

This talk is about the formula

$$\frac{R(x,s)\ :\ M\,R(x,f(x,s))}{R(x,f(x,s))},$$

called *the frame default*.

We will trace its story from 1980, when it first appeared in print, until 2001, when it inspired this piece of code from the program called *the RCS/USA Advisor*:

```
h(value(W,X),T1) :- next(T,T1),
                    signal(X),
                    is_wire(W),
                    h(value(W,X),T),
                    not nh(value(W,X),T1).
```

What We Will NOT Talk About

This is not an attempt to describe the history of research on the frame problem. We will only discuss one line of work.

Some of the other important contributions are discussed in these monographs:

- Erik Mueller, *Commonsense Reasoning*
- Raymond Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*
- Erik Sandewall, *Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems*
- Murray Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*

What is the Frame Problem?

This is the problem of describing a dynamic domain without explicitly specifying which conditions are *not* affected by an action.

- Initially, Alice is in the room, and Bob is not:

$$In_0(Alice), \quad \neg In_0(Bob).$$

- Bob enters the room:

$$Enter(Bob).$$

- Any person who enters will be in the room:

$$\forall x(Enter(x) \rightarrow In_1(x)).$$

We would like to conclude:

$$In_1(Alice), \quad In_1(Bob).$$

The second formula logically follows, but the first does not.

(McCarthy)

Default Logic and the Frame Default (1980)

A *default theory* is defined by a set of axioms and *defaults* of the form

$$\frac{F_1 \ \cdots \ F_m \ : \ M\, G_1 \ \cdots \ M\, G_n}{H}$$

("derive the conclusion H from the premises $F_1, \ldots, F_m$ if the justifications $G_1, \ldots, G_n$ can be consistently assumed").

Example:

$$\frac{In_0(x) \ : \ M\, In_1(x)}{In_1(x)}, \qquad \frac{\neg In_0(x) \ : \ M \neg In_1(x)}{\neg In_1(x)}.$$

Generalization:

$$\frac{R_0(x) \ : \ M\, R_1(x)}{R_1(x)}.$$

The frame default:

$$\frac{R(x,s) \ : \ M\, R(x, f(x,s))}{R(x, f(x,s))}.$$

(Reiter)

Minimizing Abnormality (1986)

An *abnormality theory* is a set of axioms that include the special predicate Ab. In *minimal models*, the extent of Ab cannot be made smaller without violating the axioms. Minimality can be expressed by the second-order formula called *circumscription*.

Another solution to the frame problem:

$$R_0(x) \wedge \neg Ab(x) \rightarrow R_1(x).$$

Example: moving and painting blocks.

Reasoning about minimal models can be sometimes verified using a proof checker for second-order logic.

Circumscription can be sometimes rewritten as a first-order formula.

The Yale Shooting Scenario: unintended minimal models.

(McCarthy; VL, Rabinov, Doherty, Łukaszewicz and Szałas; Hanks and McDermott)

Explanation Closure Axioms (1987)

Default logic and circumscripton are general-purpose languages for representing defaults.

Alternative: assuming that the effects of actions are described by axioms in a special syntactic form, generate "frame axioms" by a syntactic transformation.

Effect axiom:

$$\forall x(\mathit{Enter}(x) \rightarrow \mathit{In}_1(x)).$$

Explanation closure:

$$\forall x(\mathit{In}_1(x) \land \neg \mathit{In}_0(x) \rightarrow \mathit{Enter}(x)).$$

(Haas, Schubert, Pednault, Reiter)

Hard Times for the Frame Default

By 1987,

- no attempt had been made to use the frame default for describing a specific domain,
- a close relative of the frame default had been carefully tested and found inadequate,
- there had been no attempt to automate reasoning in default logic,
- an entirely different approach to the frame problem had been proposed.

Default Logic is Related to Negation in Prolog (1987)

Prolog program:

```
p(a).
p(c) :- p(a), \+ p(b).
```

Query:

```
?- p(X).
X = a ;
X = c.
```

Declarative explanation: understand the rule

```
p(c) :- p(a), \+ p(b)
```

as the default

$$\frac{p(a)\ :\ M\neg p(b)}{p(c)}.$$

(Bidoit and Froidevaux; Gelfond; Marek and Truszczyński)

Answer Sets of A Logic Program (1990)

Rule

$$L_0 \leftarrow L_1, \ldots, L_m, \textit{not}\ L_{m+1}, \ldots, \textit{not}\ L_n,$$

where each L_i is a literal, positive (A) or negative ($\neg A$), has the same meaning as the default

$$\frac{L_1\ \cdots\ L_m\ :\ M\,\overline{L_{m+1}}\ \cdots\ M\,\overline{L_n}}{L_0}.$$

The defaults

$$\frac{In_0(x)\ :\ M\,In_1(x)}{In_1(x)}, \qquad \frac{\neg In_0(x)\ :\ M\,\neg In_1(x)}{\neg In_1(x)}$$

can be written as

$$\begin{aligned} In_1(x) &\leftarrow In_0(x), \textit{not}\ \neg In_1(x), \\ \neg In_1(x) &\leftarrow \neg In_0(x), \textit{not}\ In_1(x). \end{aligned}$$

This syntax emphasizes similarities between the frame default and the calculus of events.

(Gelfond and VL; Kowalski and Sergot)

The Role of Literals

The new syntax calls for the use of *literals*, rather than arbitrary formulas, as axioms, premises, justifications, and conclusions.

Instead of the axiom

$$\forall x(\mathit{Enter}(x) \rightarrow \mathit{In}_1(x))$$

we use the rule

$$\mathit{In}_1(x) \leftarrow \mathit{Enter}(x),$$

which corresponds to a default without justifications:

$$\frac{\mathit{Enter}(x)}{\mathrm{In}_1(x)}.$$

Prolog as an Implementation of Default Logic

Default theory as a logic program:

$$
\begin{array}{rl}
& \mathit{In}_0(\mathit{Alice}). \quad \neg \mathit{In}_0(\mathit{Bob}). \quad \mathit{Enter}(\mathit{Bob}). \\
\mathit{In}_1(x) & \leftarrow \mathit{Enter}(x). \\
\mathit{In}_1(x) & \leftarrow \mathit{In}_0(x), \mathit{not}\ \neg \mathit{In}_1(x). \\
\neg \mathit{In}_1(x) & \leftarrow \neg \mathit{In}_0(x), \mathit{not}\ \mathit{In}_1(x).
\end{array}
$$

In the syntax of Prolog:

```
 in0(alice).   out0(bob).   enter(bob).
 in1(X) :- enter(X).
 in1(X) :- in0(X), \+ out1(X).
out1(X) :- out0(X), \+ in1(X).
```

Query:

```
?- in1(X).
X = bob ;
X = alice.
```

Limitations of Prolog

The default theory

$$\frac{:\ \mathsf{M}\,p}{p} \qquad \frac{:\ \mathsf{M}\,\neg p}{\neg p}$$

has two extensions, and the logic program

$$\begin{aligned} p &\leftarrow \mathit{not}\ \neg p, \\ \neg p &\leftarrow \mathit{not}\ p \end{aligned}$$

has two answer sets: $\{p\}$ and $\{\neg p\}$.

In the syntax of Prolog:

```
p :- \+ np.
np :- \+ p.
```

Query:

```
?- p.
ERROR: Out of local stack
```

Planning as Satisfiability (1992)

Generating a plan of a given length
that solves a given planning problem

can be sometimes reduced

to finding a truth assignment satisfying a given set of clauses.

Then it may be possible to find a plan by running a SAT solver.

(Kautz and Selman)

Answer Set Solvers (1997)

Computational methods used in the design of efficient SAT solvers can be applied to the problem of generating answer sets of a logic program. Some of the existing answer set solvers are as sophisticated and efficient as the fastest SAT solvers available today.

Program	Output of answer set solver

`p(a).`
`p(c) :- p(a), not p(b).`
	`Answer: 1`
	`  Answer set: p(a) p(c)`

`p :- not np.`
`np :- not p.`
	`Answer: 1`
	`  Answer set: np`
	`Answer: 2`
	`  Answer set: p`

(Aalto U., U. of Kentucky, Vienna U. of Technology, U. of Calabria, U. of Texas, U. of Genova, Hong Kong U. of Science and Technology, Guizhou U., Texas Tech U., Ohio State U., U. of Potsdam, Catholic U. of Louvain, U. of Bath, U. of Angers, Corunna U., Kodak Research Labs)

Planning as Generating an Answer Set (1997)

When effects of actions are described by a logic program, planning problems can be sometimes solved by running an answer set solver.

A discrete, linear model of time is used. The rules

$$
\begin{aligned}
In_1(x) &\leftarrow Enter(x)\\
In_1(x) &\leftarrow In_0(x), \mathit{not}\ \neg In_1(x)
\end{aligned}
$$

would become

$$
\begin{aligned}
Holds(In(x), t_1) \leftarrow\ & Next(t, t_1),\\
& Person(x),\\
& Occurs(Enter(x), t)
\end{aligned}
$$

$$
\begin{aligned}
Holds(In(x), t_1) \leftarrow\ & Next(t, t_1),\\
& Person(x),\\
& Holds(In(x), t),\\
& \mathit{not}\ \neg Holds(In(x), t_1)
\end{aligned}
$$

(Dimopoulos, Nebel, and Koehler)

The Frame Default in the RCS/USA Advisor (2001)

Toy example:

$$\begin{aligned} \mathit{Holds}(\mathit{In}(x), t_1) \leftarrow\ & \mathit{Next}(t, t_1), \\ & \mathit{Person}(x), \\ & \mathit{Holds}(\mathit{In}(x), t), \\ & \mathit{not}\ \neg\mathit{Holds}(\mathit{In}(x), t_1) \end{aligned}$$

RCS/USA Advisor:

```
h(value(W,X),T1) :- next(T,T1),
                    signal(X),
                    is_wire(W),
                    h(value(W,X),T),
                    not nh(value(W,X),T1).
```

(Texas Tech University and United Space Alliance)

Actions with Indirect Effects

Some effects of an action can be derived from its other effects.

If Bob is in the room, and he holds his cell phone in his hand, then the cell phone is in the room also. The action $\mathit{Enter}(\mathit{Bob})$ affects the truth value of $\mathit{In}(\mathit{Bob})$ directly, and it may also affect the truth value of $\mathit{In}(\mathit{Phone})$ indirectly.

The frame default is applicable in the presence of actions with indirect effects. Generating explanation closure axioms becomes more difficult.

(Baral, Gelfond, Lin, McCain, Turner, VL)

Answer Set Programming

Answer set programming is the use of logic programs under the answer set semantics for knowledge representation, and the use of answer set solvers for search. (Book: Chitta Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*.) ASP grew out of interaction between

- traditional logic programming and deductive databases,
- default logic and other systems of nonmonotonic reasoning,
- design and use of satisfiability solvers.

$$\begin{tikzcd}
\text{LP} \arrow[dr] & \text{NMR} \arrow[d] & \text{SAT} \arrow[dl] \\
 & \text{ASP} &
\end{tikzcd}$$

(The names of all contributors wouldn't fit on one slide.)