AUTOMATED REASONING

Agostino Dovier

Università di Udine
CLPLAB

Udine, December 2016

Knowledge Representation

INTRODUCTION

Knowledge representation is one of the most important subareas of artificial intelligence. If we want to design an entity (a machine or a program) capable of behaving intelligently in some environment, then we need to supply this entity with sufficient knowledge about this environment. To do that, we need an unambiguous language capable of expressing this knowledge, together with some precise and well understood way of manipulating sets of sentences of the language which will allow us to draw inferences, answer queries, and to update both the knowledge base and the desired program behavior.

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (PDF available on-line)

Introduction

> Expressing information in declarative sentences is far more modular than expressing it in segments of computer programs or in tables. Sentences can be true in a much wider context than specific programs can be used. The supplier of a fact does not have to understand much about how the receiver functions or how or whether the receiver will use it. The same fact can be used for many purposes, because the logical consequences of collections of facts can be available. [McC59]

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (PDF available on-line)

This idea has been further developed by many researchers with various backgrounds and interests. First, the classical logic of predicate calculus served as the main technical tool for the representation of knowledge. It has a well-defined semantics and a well-understood and powerful inference mechanism, and it proved to be sufficiently expressive for the representation of mathematical knowledge.

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (PDF available on-line)

It was soon realized, however, that for the representation of *commonsense knowledge*, this tool is inadequate. The difficulty is rather deep and related to the so-called "monotonicity" of theories based on predicate calculus. A logic is called *monotonic* if the addition of new axioms to a theory based on it never leads to the loss of any theorems proved in this theory. Commonsense reasoning is nonmonotonic: new information constantly forces us to withdraw previous conclusions. This observation has led to the development and investigation of new logical formalisms, *nonmonotonic logics*.

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (PDF available on-line)

Socrate is a man

All men are mortal

Therefore Socrate is mortal

Socrate is a man
All men are mortal
Therefore Socrate is mortal

Socrate is a man
All men are mortal
Therefore Socrate is mortal

Let us model this statement using f.o.l.

Socrate is a man

$$\texttt{man}(\texttt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\texttt{man}(X) \rightarrow \texttt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)` One should know that a dog is not a man (or one can inference that using Complete World Assumption).

Observe that

$$(1),(2) \models \texttt{mortal}(\texttt{socrate})$$

Let us model this statement using f.o.l.
Socrate is a man

$$\texttt{man}(\texttt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\texttt{man}(X) \rightarrow \texttt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)` One should know that a dog is not a man (or one can inference that using Complete World Assumption).
Observe that

$$(1),(2) \models \texttt{mortal}(\texttt{socrate})$$

INTRODUCTION
THE SYLLOGISM

Let us model this statement using f.o.l.
Socrate is a man

$$\texttt{man}(\texttt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\texttt{man}(X) \rightarrow \texttt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)` One should know that a dog is not a man (or one can inference that using Complete World Assumption).
Observe that

$$(1), (2) \models \texttt{mortal}(\texttt{socrate})$$

Let us model this statement using f.o.l.
Socrate is a man

$$\texttt{man}(\texttt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\texttt{man}(X) \rightarrow \texttt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If rex is a dog we have that mortal(rex) but not that man(rex) One should know that a dog is not a man (or one can inference that using Complete World Assumption).
Observe that

$$(1), (2) \models \texttt{mortal}(\texttt{socrate})$$

Let us model this statement using f.o.l.
Socrate is a man

$$\texttt{man}(\texttt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\texttt{man}(X) \rightarrow \texttt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)`. One should know that a dog is not a man (or one can inference that using Complete World Assumption).
Observe that

$$(1),(2) \models \texttt{mortal}(\texttt{socrate})$$

Let us model this statement using f.o.l.
Socrate is a man

$$\texttt{man}(\texttt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\texttt{man}(X) \rightarrow \texttt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)` One should know that a dog is not a man (or one can inference that using Complete World Assumption).
Observe that

$$(1), (2) \models \texttt{mortal}(\texttt{socrate})$$

Let us model this statement using f.o.l.
Socrate is a man

$$\mathtt{man}(\mathtt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\mathtt{man}(X) \rightarrow \mathtt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)` One should know that a dog is not a man (or one can inference that using Complete World Assumption).

Observe that

$$(1),(2) \models \mathtt{mortal}(\mathtt{socrate})$$

Let us model this statement using f.o.l.
Socrate is a man

$$\mathtt{man}(\mathtt{socrate}) \tag{1}$$

All men are mortal

$$\forall X\,(\mathtt{man}(X) \rightarrow \mathtt{mortal}(X)) \tag{2}$$

Let's have a look on the direction of the implication. If `rex` is a dog we have that `mortal(rex)` but not that `man(rex)` One should know that a dog is not a man (or one can inference that using Complete World Assumption).
Observe that

$$(1), (2) \models \mathtt{mortal}(\mathtt{socrate})$$

If we discovered that

$$\texttt{man}(\texttt{agostino}) \tag{3}$$

we have that

$$(1),(2),(3) \models \texttt{mortal}(\texttt{socrate})$$

and

$$(1),(2),(3) \models \texttt{mortal}(\texttt{agostino})$$

If, augmenting the premises, theorems that were true remains true (and possibly we have new theorems) we are in a monotonic setting.

If we discovered that

$$\texttt{man}(\texttt{agostino}) \tag{3}$$

we have that

$$(1),(2),(3) \models \texttt{mortal}(\texttt{socrate})$$

and

$$(1),(2),(3) \models \texttt{mortal}(\texttt{agostino})$$

If, augmenting the premises, theorems that were true remains true (and possibly we have new theorems) we are in a monotonic setting.

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

INTRODUCTION

NON MONOTONIC SETTINGS

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

INTRODUCTION

NON MONOTONIC SETTINGS

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

You want to cross the railway.
You can cross safely if the train is not there.
If the barriers are horizontal the train is there.
The barriers are vertical.
You'll probably cross
New info: If there is a power failure the barriers are always horizontal or always vertical.
Would you cross now?
New info: There is a power failure.
Would you cross now?
New info: You hear the train noise.
Would you cross now?

In commonsense reasoning we commonly (sometimes implicitly) use normative statements of the kind:

$$A\text{'s are normally } B\text{'s}$$

that admit exceptions.

Suppose that a reasoning agent has the following knowledge about birds: birds typically fly and penguins are non flying birds. He also knows that Tweety is a bird. Suppose now that the agent is hired to build a cage for Tweety, and he leaves off the roof on the grounds that he does not whether or nor Tweety can fly. It would be reasonable for us to view this argument as invalid and to refuse the agent's product. This would be not the case if Tweety could not fly for some reason (unknown to the agent), and we refused to pay for the bird cage because had unnecessarily put a roof on it. [McCarthy, 1959]

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

INTRODUCTION

COMMONSENSE REASONING

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird flies, unless it is abnormal.
Any penguin is a bird.
Any songbird is a bird.
Any penguin is abnormal.
Put the roof for any flying bird.
tweety is a bird.
Build the cage for tweety
tweety is a songbird

(pingu is a penguin)

Any bird that is not abnormal flies (for $\leftarrow$, think of mosquitos)

$$\forall X\,(\mathtt{bird}(X) \wedge \neg\mathtt{ab}(X) \rightarrow \mathtt{flies}(X)) \tag{4}$$

Any penguin is a bird. Any songbird is a bird.

$$\forall X\,(\mathtt{penguin}(X) \rightarrow \mathtt{bird}(X)) \tag{5}$$

$$\forall X\,(\mathtt{songbird}(X) \rightarrow \mathtt{bird}(X)) \tag{6}$$

Any penguin is abnormal. Put the roof for any flying bird.

$$\forall X\,(\mathtt{penguin}(X) \rightarrow \mathtt{ab}(X)) \tag{7}$$

$$\forall X\,(\mathtt{bird}(X) \wedge \mathtt{flies}(X) \rightarrow \mathtt{put\_roof\_cage}(X)) \tag{8}$$

tweety is a bird.

$$\mathtt{bird}(\mathtt{tweety}) \tag{9}$$

$$(4), (5), (6), (7), (8), \mathtt{CWA} \models \mathtt{put\_roof\_cage}(\mathtt{tweety})$$

Any bird that is not abnormal flies (for $\leftarrow$, think of mosquitos)

$$\forall X\,(\mathtt{bird}(X) \wedge \neg \mathtt{ab}(X) \rightarrow \mathtt{flies}(X)) \tag{4}$$

Any penguin is a bird. Any songbird is a bird.

$$\forall X\,(\mathtt{penguin}(X) \rightarrow \mathtt{bird}(X)) \tag{5}$$

$$\forall X\,(\mathtt{songbird}(X) \rightarrow \mathtt{bird}(X)) \tag{6}$$

Any penguin is abnormal. Put the roof for any flying bird.

$$\forall X\,(\mathtt{penguin}(X) \rightarrow \mathtt{ab}(X)) \tag{7}$$

$$\forall X\,(\mathtt{bird}(X) \wedge \mathtt{flies}(X) \rightarrow \mathtt{put\_roof\_cage}(X)) \tag{8}$$

tweety is a bird.

$$\mathtt{bird}(\mathtt{tweety}) \tag{9}$$

$$(4), (5), (6), (7), (8), \mathtt{CWA} \models \mathtt{put\_roof\_cage}(\mathtt{tweety})$$

Any bird that is not abnormal flies (for $\leftarrow$, think of mosquitos)

$$\forall X\,(\mathtt{bird}(X) \wedge \neg \mathtt{ab}(X) \rightarrow \mathtt{flies}(X)) \tag{4}$$

Any penguin is a bird. Any songbird is a bird.

$$\forall X\,(\mathtt{penguin}(X) \rightarrow \mathtt{bird}(X)) \tag{5}$$

$$\forall X\,(\mathtt{songbird}(X) \rightarrow \mathtt{bird}(X)) \tag{6}$$

Any penguin is abnormal. Put the roof for any flying bird.

$$\forall X\,(\mathtt{penguin}(X) \rightarrow \mathtt{ab}(X)) \tag{7}$$

$$\forall X\,(\mathtt{bird}(X) \wedge \mathtt{flies}(X) \rightarrow \mathtt{put\_roof\_cage}(X)) \tag{8}$$

tweety is a bird.

$$\mathtt{bird}(\mathtt{tweety}) \tag{9}$$

$$(4), (5), (6), (7), (8), \mathtt{CWA} \models \mathtt{put\_roof\_cage}(\mathtt{tweety})$$

If we knew that

$$\texttt{songbird(tweety)} \tag{10}$$

it still holds that

$$(4), (5), (6), (7), (8), (9), (10), \texttt{CWA} \models \texttt{put\_roof\_cage(tweety)}$$

Let us repeat with pingu. We know that is is a bird:

$$\texttt{bird(pingu)} \tag{11}$$

Then

$$(4)\text{–}(11), \texttt{CWA} \models \texttt{put\_roof\_cage(pingu)}$$

If we now discover that

$$\texttt{penguin(pingu)} \tag{12}$$

We'll have that:

$$(4)\text{–}(11), (12), \texttt{CWA} \not\models \texttt{put\_roof\_cage(pingu)}$$

If we knew that

$$\texttt{songbird}(\texttt{tweety}) \tag{10}$$

it still holds that

$$(4),(5),(6),(7),(8),(9),(10),\texttt{CWA} \models \texttt{put\_roof\_cage}(\texttt{tweety})$$

Let us repeat with pingu. We know that is is a bird:

$$\texttt{bird}(\texttt{pingu}) \tag{11}$$

Then

$$(4)\text{–}(11),\texttt{CWA} \models \texttt{put\_roof\_cage}(\texttt{pingu})$$

If we now discover that

$$\texttt{penguin}(\texttt{pingu}) \tag{12}$$

We'll have that:

$$(4)\text{–}(11),(12),\texttt{CWA} \not\models \texttt{put\_roof\_cage}(\texttt{pingu})$$

Default negation and stable models allow us to deal with commonsense reasoning!

Encode the previous example in ASP.

A man once had to travel with a wolf, a goat and a cabbage. He had to take good care of them, since the wolf would like to taste a piece of goat if he would get the chance, while the goat appeared to long for a tasty cabbage. After some traveling, he suddenly stood before a river. This river could only be crossed using the small boat laying nearby at a shore. The boat was only good enough to take himself and one of his loads across the river. The other two subjects/objects he had to leave on their own. How must the man row across the river back and forth, to take himself as well as his luggage safe to the other side of the river, without having one eating another? (and what is the minimum number of crossings?)

```
time(1..n).
place(left;right;boat).
object(man;goat;cabbage;wolf).

%%% In any time, any object is exactly in one place.
1 { on(T,O,P) : place(P) } 1 :- time(T), object(O).

%%% INITIAL STATE
on(1,goat,left).     on(1,cabbage,left).
on(1,wolf,left).     on(1,man,left).

%%%    IF (goat and cabbage) OR (goat and wolf) are in place P,
%%%    THEN the man is in P
on(T,man,P) :- on(T,goat,P), on(T,cabbage,P), time(T), place(P).
on(T,man,P) :- on(T,goat,P), on(T,wolf,P),    time(T), place(P).

%%% Boat effect (it is important where the motion has started)
on(T+2,O,right):- on(T+1,O,boat), on(T,O,left),  time(T), object(O).
on(T+2,O,left) :- on(T+1,O,boat), on(T,O,right), time(T), object(O).
```

```
time(1..n).
place(left;right;boat).
object(man;goat;cabbage;wolf).

%%% In any time, any object is exactly in one place.
1 { on(T,O,P) : place(P) } 1 :- time(T), object(O).

%%% INITIAL STATE
on(1,goat,left).     on(1,cabbage,left).
on(1,wolf,left).     on(1,man,left).

%%%    IF (goat and cabbage) OR (goat and wolf) are in place P,
%%%    THEN the man is in P
on(T,man,P) :- on(T,goat,P), on(T,cabbage,P), time(T), place(P).
on(T,man,P) :- on(T,goat,P), on(T,wolf,P),    time(T), place(P).

%%% Boat effect (it is important where the motion has started)
on(T+2,O,right):- on(T+1,O,boat), on(T,O,left),  time(T), object(O).
on(T+2,O,left) :- on(T+1,O,boat), on(T,O,right), time(T), object(O).
```

The wolf, the goat, and the cabbage

```
time(1..n).
place(left;right;boat).
object(man;goat;cabbage;wolf).

%%% In any time, any object is exactly in one place.
1 { on(T,O,P) : place(P) } 1 :- time(T), object(O).

%%% INITIAL STATE
on(1,goat,left).    on(1,cabbage,left).
on(1,wolf,left).    on(1,man,left).

%%%    IF (goat and cabbage) OR (goat and wolf) are in place P,
%%%    THEN the man is in P
on(T,man,P) :- on(T,goat,P), on(T,cabbage,P), time(T), place(P).
on(T,man,P) :- on(T,goat,P), on(T,wolf,P),    time(T), place(P).

%%% Boat effect (it is important where the motion has started)
on(T+2,O,right):- on(T+1,O,boat), on(T,O,left),  time(T), object(O).
on(T+2,O,left) :- on(T+1,O,boat), on(T,O,right), time(T), object(O).
```

The wolf, the goat, and the cabbage

```
time(1..n).
place(left;right;boat).
object(man;goat;cabbage;wolf).

%%% In any time, any object is exactly in one place.
1 { on(T,O,P) : place(P) } 1 :- time(T), object(O).

%%% INITIAL STATE
on(1,goat,left).    on(1,cabbage,left).
on(1,wolf,left).    on(1,man,left).

%%%   IF (goat and cabbage) OR (goat and wolf) are in place P,
%%%   THEN the man is in P
on(T,man,P) :- on(T,goat,P), on(T,cabbage,P), time(T), place(P).
on(T,man,P) :- on(T,goat,P), on(T,wolf,P),    time(T), place(P).

%%% Boat effect (it is important where the motion has started)
on(T+2,O,right):- on(T+1,O,boat), on(T,O,left),  time(T), object(O).
on(T+2,O,left) :- on(T+1,O,boat), on(T,O,right), time(T), object(O).
```

THE WOLF, THE GOAT, AND THE CABBAGE

```
time(1..n).
place(left;right;boat).
object(man;goat;cabbage;wolf).

%%% In any time, any object is exactly in one place.
1 { on(T,O,P) : place(P) } 1 :- time(T), object(O).

%%% INITIAL STATE
on(1,goat,left).    on(1,cabbage,left).
on(1,wolf,left).    on(1,man,left).

%%%    IF (goat and cabbage) OR (goat and wolf) are in place P,
%%%    THEN the man is in P
on(T,man,P) :- on(T,goat,P), on(T,cabbage,P), time(T), place(P).
on(T,man,P) :- on(T,goat,P), on(T,wolf,P),    time(T), place(P).

%%% Boat effect (it is important where the motion has started)
on(T+2,O,right):- on(T+1,O,boat), on(T,O,left),  time(T), object(O).
on(T+2,O,left) :- on(T+1,O,boat), on(T,O,right), time(T), object(O).
```

```
%%% IF someone is in boat, then the man must be on boat.
on(T,man,boat) :- on(T,O,boat), time(T), object(O).

%%% The boat contains from 0 to 2 objects
0 { on(T,O,boat) : object(O) } 2 :- time(T).
```

```
%%% INERTIA rules
:- on(T+1,O,left), on(T,O,right), time(T), object(O).
:- on(T+1,O,right), on(T,O,left), time(T), object(O).
```

```
%%% FINAL STATE (goal)
final :- on(n,goat,right), on(n,cabbage,right),
         on(n,wolf,right), on(n,man,right).
:- not final.

#show on/3.
```

THE WOLF, THE GOAT, AND THE CABBAGE

```
%%% IF someone is in boat, then the man must be on boat.
on(T,man,boat) :- on(T,O,boat), time(T), object(O).

%%% The boat contains from 0 to 2 objects
0 { on(T,O,boat) : object(O) } 2 :- time(T).


%%% INERTIA rules
:- on(T+1,O,left), on(T,O,right), time(T), object(O).
:- on(T+1,O,right), on(T,O,left), time(T), object(O).


%%% FINAL STATE (goal)
final :- on(n,goat,right), on(n,cabbage,right),
           on(n,wolf,right), on(n,man,right).
:- not final.

#show on/3.
```

THE WOLF, THE GOAT, AND THE CABBAGE

```
%%% IF someone is in boat, then the man must be on boat.
on(T,man,boat) :- on(T,O,boat), time(T), object(O).

%%% The boat contains from 0 to 2 objects
0 { on(T,O,boat) : object(O) } 2 :- time(T).


%%% INERTIA rules
:- on(T+1,O,left), on(T,O,right), time(T), object(O).
:- on(T+1,O,right), on(T,O,left), time(T), object(O).


%%% FINAL STATE (goal)
final :- on(n,goat,right), on(n,cabbage,right),
           on(n,wolf,right), on(n,man,right).
:- not final.

#show on/3.
```

From Life International, December 17, 1962

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets [sic].

FROM LIFE INTERNATIONAL, DECEMBER 17, 1962

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets [sic].

From Life International, December 17, 1962

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets [sic].

From Life International, December 17, 1962

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets [sic].

FROM LIFE INTERNATIONAL, DECEMBER 17, 1962

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarets [sic].

The zebra puzzle

```
%%% 1.       There are five houses.
house(1..5).

%%% 2. The Englishman lives in the red house.
i2 :- house(C), owner_comes_from(C,england), colored(C,red).

%%% 3.        The Spaniard owns the dog.
i3 :- house(C), owns_animal(C,dog), owner_comes_from(C,spain).

%%% 4. Coffee is drunk in the green house.
i4 :- house(C), colored(C,green), drinks(C,coffee).

%%% 5. The Ukrainian drinks tea.
i5 :- house(C),  owner_comes_from(C,ukraina), drinks(C,tea).
```

THE ZEBRA PUZZLE

```
%%% 6. The green house is immediately to the right of the ivory house.
i6 :- house(C1), house(C2), lefttoright(C1,C2),
       colored(C1,ivory), colored(C2,green).

%%% 7.  The Old Gold smoker owns snails.
i7 :- house(C), owns_animal(C,snail), smokes(C,oldgold).

%%% 8. Kools are smoked in the yellow house.
i8 :- house(C), colored(C,yellow), smokes(C,kools).

%%% 9. Milk is drunk in the middle house.
i9 :- house(C), middle(C), drinks(C,milk).

%%% 10. The Norwegian lives in the first house.
i10 :- house(C), first(C), owner_comes_from(C,norway).
```

THE ZEBRA PUZZLE

```
%%% 11. The man who smokes Chesterfields lives in the house
%%%       next to the man with the fox
i11 :- house(C1), house(C2), next(C1,C2),
         smokes(C1,chesterfield), owns_animal(C2,fox).

%%% 12.  Kools are smoked in the house next to the house
%%%        where the horse is kept.
i12 :- house(C1), house(C2), next(C1,C2),
          smokes(C1,kools), owns_animal(C2,horse).

%%% 13. The Lucky Strike smoker drinks orange juice.
i13 :- house(C), smokes(C,luckystrike), drinks(C,orangejuice).

%%% 14.  The Japanese smokes Parliaments.
i14 :- house(C), owner_comes_from(C,japan), smokes(C,parliaments).

%%% 15. The Norwegian lives next to the blue house.
i15 :- house(C1), house(C2), next(C1,C2),
          colored(C1,blue), owner_comes_from(C2,norway).
```

```
hints :- i2 , i3, i4, i5,  i6, i7, i8, i9, i10, i11, i12, i13, i14, i15.
:- not hints.

%%%% AUX
lefttoright(C,C+1) :- house(C), house(C+1).
next(A,B) :- lefttoright(A,B).
next(A,B) :- lefttoright(B,A).
middle(3).
first(1).
```

Does it work?

```
%% ... each of the five houses is painted a different color,
%%% their inhabitants are of different national extractions,
%%% own different pets, drink different beverages and
%%% smoke different brands of American cigarets [sic].

country(japan). country(ukraina). country(norway). country(england). country(spain).
1{ owner_comes_from(A,B) : country(B)   }1 :- house(A).
1{ owner_comes_from(A,B) : house(A)   }1 :- country(B).

color(red). color(green). color(ivory). color(blue). color(yellow).
1{ colored(A,B) : color(B) }1 :- house(A).
1{ colored(A,B) : house(A) }1 :- color(B) .

beverage(coffee). beverage(milk). beverage(orangejuice). beverage(tea). beverage(water).
1{ drinks(A,B) : beverage(B) }1 :- house(A).
1{ drinks(A,B) : house(A) }1 :- beverage(B) .

cig(oldgold). cig(kools). cig(chesterfield). cig(luckystrike). cig(parliaments).
1{ smokes(A,B) : cig(B) }1 :- house(A).
1{ smokes(A,B) : house(A) }1 :- cig(B) .

animal(dog). animal(snail). animal(horse). animal(zebra). animal(fox).
1{ owns_animal(A,B) : animal(B) }1 :- house(A).
1{ owns_animal(A,B) : house(A) }1 :- animal(B) .
```

There are four people: Roberta, Thelma, Robin, and Pete.
Among them, they hold eight different jobs.
Each holds exactly two jobs.
The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.
The job of nurse is held by a male.
The husband of the chef is the telephone operator.
Roberta is not a boxer.
Pete has no education past the ninth grade.
Roberta, the chef, and the police officer went golfing together.
Question: Who holds which jobs?

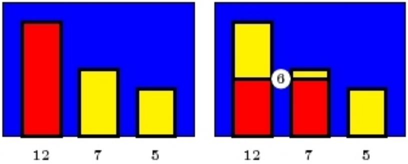
JOBS PUZZLE - WOS ET AL 1984

There are four people: Roberta, Thelma, Robin, and Pete.
Among them, they hold eight different jobs.
Each holds exactly two jobs.
The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.
The job of nurse is held by a male.
The husband of the chef is the telephone operator.
Roberta is not a boxer.
Pete has no education past the ninth grade.
Roberta, the chef, and the police officer went golfing together.
Question: Who holds which jobs?

Jobs puzzle - Wos et al 1984

There are four people: Roberta, Thelma, Robin, and Pete.
Among them, they hold eight different jobs.
Each holds exactly two jobs.
The jobs are: chef, guard, nurse, telephone operator, police officer (gender not implied), teacher, actor, and boxer.
The job of nurse is held by a male.
The husband of the chef is the telephone operator.
Roberta is not a boxer.
Pete has no education past the ninth grade.
Roberta, the chef, and the police officer went golfing together.
Question: Who holds which jobs?

" There are three barrels of capacity N (an even number), $N/2+1$, and $N/2-1$, respectively. At the beginning, the largest barrel is full of wine while the other two are empty. We wish to reach a state in which the two larger barrels contain the same amount of wine. The only permissible action is to pour wine from one barrel to another, until the latter is full or the former is empty. (you can't measure the wine flow or the barrels weight)."



THE THREE BARRELS

```
litri(0..12).
barrel(5;7;12).
time(0..t).

% One and only one pour at the time T
1{ pour(T,X,Y): barrel(X), barrel(Y), X != Y} 1 :- time(T), T < t.

% Initial state
contains(0,12,12).     contains(0,7,0).    contains(0,5,0).

% Final state (and condition)
goal :- contains(t,12,6), contains(t,7,6), contains(t,5,0).
:- not goal.
```

THE THREE BARRELS

```
litri(0..12).
barrel(5;7;12).
time(0..t).

% One and only one pour at the time T
1{ pour(T,X,Y): barrel(X), barrel(Y), X != Y} 1 :- time(T), T < t.

% Initial state
contains(0,12,12).     contains(0,7,0).    contains(0,5,0).

% Final state (and condition)
goal :- contains(t,12,6), contains(t,7,6), contains(t,5,0).
:- not goal.
```

THE THREE BARRELS

```
litri(0..12).
barrel(5;7;12).
time(0..t).

% One and only one pour at the time T
1{ pour(T,X,Y): barrel(X), barrel(Y), X != Y} 1 :- time(T), T < t.

% Initial state
contains(0,12,12).    contains(0,7,0).   contains(0,5,0).

% Final state (and condition)
goal :- contains(t,12,6), contains(t,7,6), contains(t,5,0).
:- not goal.
```

Pour effect:

```
contains(T+1,X,0) :-
    time(T),time(T+1),
    barrel(X),  barrel(Y), X != Y,
    litri(LX), litri(LY),
    %%%
    pour(T,X,Y),
    contains(T,X,LX),contains(T,Y,LY),
    Y-LY > LX.
contains(T+1,Y,LX+LY) :-
    time(T),time(T+1),
    barrel(X),  barrel(Y), X != Y,
    litri(LX), litri(LY),
    %%%
    pour(T,X,Y),
    contains(T,X,LX), contains(T,Y,LY),
    Y-LY > LX.
```

Pour effect:

```
contains(T+1,X,LX-(Y-LY)) :-
     time(T),time(T+1),
     barrel(X),  barrel(Y), X != Y,
     litri(LX), litri(LY),
     pour(T,X,Y),
     contains(T,X,LX),
     contains(T,Y,LY),
     Y-LY <= LX.
contains(T+1,Y,Y) :-
     time(T),time(T+1),
     barrel(X),  barrel(Y), X != Y,
     litri(LX), litri(LY),
     pour(T,X,Y),
     contains(T,X,LX),
     contains(T,Y,LY),
     Y-LY <= LX.
```

THE THREE BARRELS

Inertia:

```
contains(T+1,B,L) :-
     time(T),  time(T+1),
     barrel(X),  barrel(Y), barrel(B),
     litri(L),
     X != Y, X != B, Y != B,
     pour(T,X,Y),
     contains(T,B,L).
```

```
%%% You can't pour an empty barrel or in a full one
:- barrel(X), barrel(Y), X!=Y, time(T), pour(T,X,Y), contains(T,X,0).
:- barrel(X), barrel(Y), X!=Y, time(T), pour(T,X,Y), contains(T,Y,Y).

#show pour/3.
```

Exercise: encode (and solve) the similar problem from Die Hard movie:
`http://puzzles.nigelcoldwell.co.uk/twentytwo.htm`

Inertia:

```
contains(T+1,B,L) :-
     time(T),  time(T+1),
     barrel(X),  barrel(Y), barrel(B),
     litri(L),
     X != Y, X != B, Y != B,
     pour(T,X,Y),
     contains(T,B,L).

%%% You can't pour an empty barrel or in a full one
:- barrel(X), barrel(Y), X!=Y, time(T), pour(T,X,Y), contains(T,X,0).
:- barrel(X), barrel(Y), X!=Y, time(T), pour(T,X,Y), contains(T,Y,Y).

#show pour/3.
```

Exercise: encode (and solve) the similar problem from Die Hard movie:
http://puzzles.nigelcoldwell.co.uk/twentytwo.htm