Artificial Intelligence **(Harvard CS182, Fall 2015)**

Lecture Notes on Propositional Logic

Alexander Rush

Contents

1 Introduction

One of the central goals of artificial intelligence is to be able to model human reasoning. We would expect a reasoning agent to be able to start with a set of knowledge and be able to draw conclusions about the statements relating to this input. This task seems quite far from our previous two units on search and constraint satisfaction.

In this unit we will explicitly discuss the task of reasoning. We introduce a very simple form of a logic known as propositional logic, and show how it can be used to encode basic sets of knowledge, known as knowledge bases. We then show how to use this knowledge base to reason about new statements. To do this we will transform the logical representation into a form that again allows us to use the tools and language of search. We finish with a brief survey of the area of planning, which uses a logical representation to reason about sequences of events.

2 A Running Example: Medical Diagnosis

Consider the following flowchart distributed to help people with flu-like symptoms to perform self-diagnostics.

Department of Veterans Affairs

Name ....

If you took your temperature, write it down: .... degrees F

Date: .... Time: .... am / pm

Phone number of nurse advice line or health provider

....

Do I have the flu? What care should I get?

A self-assessment flowchart

This self-assessment flowchart gives influenza (flu) symptoms. It may help you decide to stay home if you have some of these symptoms. It also lists when to get care right away and when to call your health care provider. Your local health department may have additional advice on what to do if you have flu symptoms. **This chart is not a substitute for medical care.**

Q. Do you now or in past few days:

- __ feel like you have a fever?
- __ have shaking or chills?
- __ feel sweaty?

Yes to one or more → **Q. Do you have a cough or sore throat?**

No → You probably do not have the flu.

Q. Do you have a cough or sore throat?

Yes → You may have the flu.

No → You probably do not have the flu.

You may have the flu.

(Other flu symptoms can be aches, a stuffy or runny nose, fatigue, diarrhea, or nausea.)

→ **Q. Do you have any of these?**

- __ Trouble breathing or shortness of breath
- __ New pain or pressure in the chest
- __ Confusion
- __ Vomiting
- __ Dizziness
- __ Less urine (pee) produced
- __ Seizures
- __ A fever or cough/sore throat that went away and came back

Yes → Get care right away

No → **Q. Do you have any of these?**

Get care right away

Call your health care provider or, if it seems like an emergency, call 911 or go to an emergency room.

Q. Do you have any of these?

- __ Age 18 and under on chronic aspirin treatment
- __ Age 65 and older
- __ Lung disease (like asthma)
- __ Heart disease
- __ Diabetes
- __ Severe obesity
- __ A transplanted organ
- __ HIV/AIDS
- __ Kidney failure (dialysis)
- __ Liver disease
- __ Sickle cell disease
- __ Brain illness (such as traumatic brain injury or TBI)
- __ Spinal cord injury
- __ Nerve and muscle disease (such as multiple sclerosis)
- __ Treatment with 20 mg or more prednisone per day

Q. Are you pregnant?

Yes to any of these → Call your health care provider

No → **Q. Is there someone at home to take care of you?**

Q. Is there someone at home to take care of you?

No → Call your health care provider

Yes → Stay home

Call your health care provider

He or she may help you get care at home, prescribe medications, or assess whether you should come in for a medical visit or hospital stay.

Stay home

- Rest
- Drink fluids
- Take fever-reducing drugs, such as acetaminophen (Tylenol) or ibuprofen (Advil or Motrin)
- Take other medicines as prescribed by your health care provider

NOTE: *Stay home for several days until you feel better and have been without a fever for at least 24 hours without taking fever-reducing drugs.*

(Brand names included in this information are provided as examples only.)

You probably do not have the flu.

Be sure to stay healthy, wash your hands, cover your coughs, and stay away from people who are ill if you are able. If you begin to worry about your health, call the nurse advice line or health care provider.

If we treat the state of a patient and the recommended action as a set of possible worlds, this chart can be seen as a knowledge base in propositional logic which will be the main focus of this note.

For simplicity let's cut down the number of possible symptoms to around 10. In English, we can roughly interpret these statements as saying:

- You may have the flu only if you have a cough in addition to a fever or are sweating.
- If you may have the flu and you have confusion or vomiting then call the emergency room.
- If you may have the flu but don't think it's an emergency, and you are over 65, then call the doctor.

It allows us to ask and answer formal questions about the structure of the world. For instance we might wonder:

- If I may have the flu does this mean that I am sweaty?

When we return to this example, we will see algorithms that allow us to answer these types of queries.

3 Propositional Logic

3.1 The Logic Framework

In this class we'll be considering a simple variant of **formal logic**. In this framework we assume that there is an underlying state of the world which is completely described by a **model**. For now, we will treat the model m abstractly as some member of the set of all possible world $\mathcal{M}$. In the next section we will explicitly define a particular model.

Logic allows us to make statements about the world, which we will call sentence α. Under some worlds the statement may evaluate to true, and in others it may evaluate to false. In the first case we say that the model **satisfies** the sentence. We use the notation $\mathcal{M}(\alpha)$ to specify all models that satisfy a given sentence.

We can also consider the relationship between two logical sentences α and β. If the statement α implies that β is true, we say α **entails** β and write this as $\alpha \models \beta$.

The entailment relationship can be defined set-theoretically using models. Formally

$$\alpha \models \beta \text{ iff } \mathcal{M}(\alpha) \subset \mathcal{M}(\beta)$$

that is α entails β is all models satisfied by α are also satisfied by β.

In practice we will very rarely able to enumerate all the possible states of the world in order to check entailment relations. However this provides an important vocabulary for thinking about the underlying assumptions of formal logic.

3.2 Syntax and Semantics of Propositional Logic

Now we look at specific variant of logic known as propositional logic. In propositional logic, we define a model as a mapping from atomic variables to true or false. That is we assume world consists of a set of proposition variables, and a model simply specifies whether each specific proposition is true or false in the current world, i.e. $m : \mathcal{P} \mapsto \{\top, \bot\}$.

Next we define a language we constructing sentences α about our propositional world. The following chart describes each of the symbols in the logic and their informal meaning.

Name	Syntax	Description
literal	P, Q	Reference to underlying propositions in $\mathcal{P}$
true	$\top$	Symbol for true
false	$\bot$	Symbol for false
not	$\neg$	$\neg P$ is true iff P is false in m
conjunction/and	$\wedge$	$P \wedge Q$ is true iff P and Q is true in m
disjunction/or	$\vee$	$P \vee Q$ is true iff P or Q is true in m
conditional	$\Rightarrow$	
biconditional	$\Leftrightarrow$	

AIMA also gives the full specification for the grammar, order of operations, and truth table of these symbols. It generally follows the standard mathematical standards for these symbols. For instance here is the definition of conditional

P	Q	P$\Rightarrow$ Q
$\bot$	$\bot$	
$\bot$	$\top$	
$\top$	$\bot$	
$\top$	$\top$	

3.3 Knowledge Base

Using a logical formalism we can define a knowledge base (KB) as a collection of conjoined ($\wedge$) sentences from the formalism. Implicitly our knowledge base KB defines the set of worlds $\mathcal{M}(KB)$ of interest.

Once we have a knowledge base, we can issue **queries** α consisting of logical statements about the knowledge base. In particular we will be interested in whether the knowledge base entails the question, i.e. $KB \models \alpha$.

The simplest way to do this would be to use **model checking**. Here we utilize the theory of entailment and check whether $\mathcal{M}(KB) \subset \mathcal{M}(\alpha)$. We could do this by constructing the two sets of models and confirming a subset relation, or by finding a contradiction.

3.4 Example: Diagnostics

Now let us return to our diagnostics example. Recall that the diagram tells us the following relationship:

- You may have the flu only if you have a cough in addition to a fever or are sweating.
- If you may have the flu and you have confusion or vomiting then call the emergency room.
- If you may have the flu but don't think it's an emergency, and you are over 65, then call the doctor.

As noted above: If we treat the state of a patient and the recommended action as a set of possible worlds, this chart can be seen as a knowledge base in propositional logic. Here's what it looks like:

$$
\begin{aligned}
(((\textsc{Fever} \lor \textsc{Sweaty}) \land \textsc{Cough})) &\Leftrightarrow \textsc{MayHaveFlu}) \land \\
((\textsc{MayHaveFlu} \land (\textsc{Confusion} \lor \textsc{Vomiting}) &\Rightarrow \textsc{Emergency}) \land \\
((\textsc{MayHaveFlu} \land \neg\textsc{Emergency} \land \textsc{Over65}) &\Rightarrow \textsc{CallDoctor})
\end{aligned}
$$

The knowledge base contains 9 propositions:

Propositions	$\mathcal{M}(KB)_1$	$\mathcal{M}(KB)_2$	$\mathcal{M}(KB)_3$	$\dots$
Cough	$\bot$	$\top$	$\top$	
Sweaty	$\bot$	$\top$	$\bot$	
Fever	$\bot$	$\bot$	$\top$	
Confusion	$\bot$	$\bot$	$\bot$	
Vomiting	$\bot$	$\bot$	$\bot$	
Over65	$\bot$	$\bot$	$\bot$	
MayHaveFlu	$\bot$	$\top$	$\top$	
Emergency	$\bot$	$\bot$	$\bot$	
CallDoctor	$\bot$	$\bot$	$\bot$	

Question 1 *How many possible worlds exist, what is the size of* $|\mathcal{M}|$*?*

We could in theory enumerate all models that satisfy this KB, however we will see that there are easier ways to do reasoning.

As an example consider a query α we might issue to this knowledge base,

- If I may have the flu does this imply that I am sweaty?

$$\textsc{MayHaveFlu} \Rightarrow \textsc{Sweaty}$$

Model checking method would enumerate all models that are consistent with the KB and the query. If $M(\alpha) \subset \mathcal{M}(KB)$ then it is entailed. Conversely, if we can find a model that is in $\mathcal{M}(KB)$ but violates the sentence they we can conclude the opposite.

Question 2 *Does $KB \models \alpha$?*

4 Inference

As an alternative to model-based reasoning we can instead perform reasoning based on **inferences**. Starting from a knowledge base KB we will attempt to transform the logical representation to find our goal sentence α. We do this by applying a set of classical inference **rules**.

An inference rule takes a set of premises and returns a new sentence as a conclusion:

$$\frac{\text{premise1} \quad \text{premise2}}{\text{conclusion}}$$

We can extend the knowledge by applying inference rules to the current data. For instance the simplest inference **modus ponens** or implication elimination:

$$\frac{\alpha \Rightarrow \beta \quad \alpha}{\beta}$$

Another is simply eliminating a conjuntion:

$$\frac{\alpha \wedge \beta}{\alpha}$$

We can also express DeMorgan's laws as inferences

$$\frac{\neg(\alpha \wedge \beta)}{\neg\alpha \vee \neg\beta}$$

4.1 Search and Inference

Using these rules we can define a search problem. This brings us back to our trusty search formalism

Name	Type	Description
State		
Initial state		
Actions		
Result		
Goal		

There are two desirable properties of inference rules.

Definition 1 *A rule is **sound** if it is truth preserving, i.e. it does not change the set of models.*

Definition 2 *A set of rules is **complete** if they are guaranteed to find a solution, when used with a complete search algorithm*

AIMA gives a set of sound rules for propositional logic and an informal assertion that they are complete. However there is much simpler cleaner inference strategy for dealing with propositional logic.

5 Resolution

Instead of searching for rules that infer our query, resolution will instead try to find a **proof by contradiction**. We will first assume the negation of our query, and then try to search for a contradiction.

Formally, define any sentence α as **satisfiable** if there exists a model for which it is true, i.e. $\mathcal{M}(\alpha)$ is not empty.

$$\alpha \models \beta \text{ iff } (\alpha \wedge \neg\beta) \text{ is unsatisfiable}$$

For **resolution**, for a query α and knowledge base KB, we first construct $KB \wedge \neg\alpha$ and then search for a proof of unsatisfiability.

5.1 Resolution for Propositional Logic

Our aim for resolution will be to produce a contradiction of the form $p \wedge \neg p$, where p is a proposition. We will do this by finding smaller and smaller disjunctive clauses until we discover a contradiction or run out of inference rules to apply.

Consider a sentence consisting of a disjunction of propositions $l_1 \vee \ldots \vee l_k$. We call this a **clause**. If we also know that any of the propositions are false, i.e. $\neg l_i$, we can remove it from the list, producing a smaller disjunctive list. This is called a **unit resolution**:

$$\frac{l_1 \vee \ldots \vee l_k \quad \neg l_i}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k} \text{ where } l_i = \neg m.$$

However we don't need to just do unit resolutions, if there is a another clause $m_1 \ldots m_n$ that contains the negation of any proposition in l then we can resolve the two clauses. This is called **general resolution**:

It turns that this is the only inference rule actually necessary. It turns out with just resolution we have the two properties we need:

- Soundness: The resolution rule is truth preserving
- Completeness: It is always able to find a proof.

But what about all the other elements besides disjunction ($\vee$) and negation ($\neg$) ? Well it turns out we can apply a transformation that removes these elements from the problem entirely.

5.2 Conjunctive normal form

Conjunctive normal formal (CNF) is a general representation for propositional logic, that consists only of conjoined clauses with propositions or negative propositions. While much more difficult to read than general propositional logic, it can represent all possible propositional sentences.

To convert a sentence in propositional logic to CNF we apply the following 4 transfomations.

1. Biconditional Elimination: Replace $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Conditional Elimination: Replace $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
3. Move $\neg$ inward. Demorgan's rules

$$\neg(\alpha \wedge \beta) \triangleq (\neg\alpha \vee \neg\beta)$$
$$\neg(\alpha \vee \beta) \triangleq (\neg\alpha \wedge \neg\beta)$$

4. Distribute $\vee$ over $\wedge$.

$$\alpha \vee (\beta \wedge \gamma) \triangleq ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$$

The final form of the KB will now be a conjunction of disjunctive clauses. That is a conjunction of phrases each containing only $\vee$ and $\neg$ applied to atomic propositions.

Example Original

$$
\begin{aligned}
(((\textsc{Fever} \vee \textsc{Sweaty}) \wedge \textsc{Cough}) &\Leftrightarrow \textsc{MayHaveFlu}) \wedge \\
((\textsc{MayHaveFlu} \wedge (\textsc{Confusion} \vee \textsc{Vomiting}) &\Rightarrow \textsc{Emergency}) \wedge \\
((\textsc{MayHaveFlu} \wedge \neg\textsc{Emergency} \wedge \textsc{Over65} &\Rightarrow \textsc{CallDoctor})
\end{aligned}
$$

Step 1 Biconditional

$$
\begin{aligned}
\textsc{MayHaveFlu} &\Rightarrow ((\textsc{Fever} \vee \textsc{Sweaty}) \wedge \textsc{Cough}) \wedge \\
((\textsc{Fever} \vee \textsc{Sweaty}) \wedge \textsc{Cough}) &\Rightarrow \textsc{MayHaveFlu} \wedge \\
(\textsc{MayHaveFlu} \wedge (\textsc{Confusion} \vee \textsc{Vomiting}) &\Rightarrow \textsc{Emergency} \wedge \\
(\textsc{MayHaveFlu} \wedge \neg\textsc{Emergency} \wedge \textsc{Over65}) &\Rightarrow \textsc{CallDoctor}
\end{aligned}
$$

Step 2 Conditional

Step 3 De Morgan's Laws

$$
\begin{aligned}
(\neg\textsc{MayHaveFlu} &\vee ((\textsc{Fever} \vee \textsc{Sweaty}) \wedge \textsc{Cough})) \wedge \\
((\neg\textsc{Fever} \wedge \neg\textsc{Sweaty}) \vee \neg\textsc{Cough}) &\vee \textsc{MayHaveFlu}) \wedge \\
((\neg\textsc{MayHaveFlu} \vee (\neg\textsc{Confusion} \wedge \neg\textsc{Vomiting}) &\vee \textsc{Emergency}) \wedge \\
((\neg\textsc{MayHaveFlu} \vee \textsc{Emergency} \vee \neg\textsc{Over65} &\vee \textsc{CallDoctor})
\end{aligned}
$$

Step 4 Redistibute

$$\begin{aligned}
(\neg\textsc{MayHaveFlu} \lor \textsc{Fever} \lor \textsc{Sweaty}) \land \\
(\neg\textsc{MayHaveFlu} \lor \textsc{Cough}) \land \\
(\neg\textsc{Fever} \lor \textsc{Cough} \lor \textsc{MayHaveFlu}) \land \\
(\neg\textsc{Sweaty} \lor \textsc{Cough} \lor \textsc{MayHaveFlu}) \land \\
(\neg\textsc{Confusion} \lor \neg\textsc{MayHaveFlu} \lor \textsc{Emergency}) \land \\
(\neg\textsc{Vomiting} \lor \neg\textsc{MayHaveFlu} \lor \textsc{Emergency}) \land \\
(\neg\textsc{MayHaveFlu} \lor \textsc{Emergency} \lor \neg\textsc{Over65} \lor \textsc{CallDoctor})
\end{aligned}$$

One application of general resolution:

$$\frac{\neg\textsc{MayHaveFlu} \lor \textsc{Fever} \lor \textsc{Sweaty} \quad (\neg\textsc{Sweaty} \lor \textsc{Cough} \lor \textsc{MayHaveFlu})}{}$$

5.3 Resolution Algorithm

This brings us to the full resolution algorithm for inference.

- Create a new KB by adding $\neg\alpha$, i.e. $KB \land \neg\alpha$
- Convert to CNF.
- Repeatedly apply resolution.
  - If there are no new clauses to resolve, the KB does not entail α.
  - If there an empty clause is produce, we have a contradiction and KB does entail α.

We can show that given enough time this algorithm will either find a contradiction or find no new clauses to resolve. This gives a simple complete algorithm propositional logic.

However, there is a big *but* here. We have not discussed at all the complexity of this approach. This problem of determining whether a propositional statement is satisfiable is known as SAT. If you took CS-121 you know that a variety of this problem 3-SAT is perhaps the most famous problem in computer science. This is one of Karp's original 21 problems, and one of the first problems to be proven NP-Complete. Therefore for general logic we can't hope to find efficient solutions. That being said there are many cases where variants of resolution and goal directed search are able to find solutions to real-world versions.

6 Planning

So far we have discussed logic in the context of inference in a static world. In practice though the underlying model may be in constant flux. In planning we explicitly incorporate **actions** that change the underlying model. In many ways this is similar to the actions that occur in search.

However, planning will utilize propositional logic to explictly declare exactly which actions are applicable in each state and how they update the state of the world.

The upside of this extra information is that it will allow us to use general purpose planners to perform search. These planners can take advantage of the logical representation to generate effective heuristics for the planning search problem without requiring domain knowledge.

6.1 Closed-World Assumption

Before discussing planning, we need to make one change to our current assumptions about the semantics of propositional logic, known as the closed-world assumption. So far, we have been assuming that all propositions in the world are explicitly included in the knowledge base. In the closed-world assumption we allow there to be arbitrary additional propositions, but that they are assumed to be false in all models .

For instance in the medical diagnosis example, we might pose a closed world query of the form:

$$\text{ISDOCTOR} \lor \text{COUGHING}$$

Which would not be implied since ISDOCTOR is not in the knowledge-base and is assumed to be false and COUGHING may or may not be true.

We can take this a step further and make **lifted** literals for an occupation. Say for any occupation a is the following statement implied:

$$\text{IS}(\text{A}) \lor \text{COUGHING}$$

Don't let the function notation confuse you here, we can think of there being many different propositions $\text{IS}(\mathit{Doctor})$, $\text{IS}(\mathit{Lawyer})$, $\text{IS}(\mathit{Policeman})$, etc. Each of these is just an atomic proposition like ISDOCTOR. Since none of them are seen, they are assumed to be $\perp$.

6.2 Planning As Search

A planning problem fits under our standard definition of search. Note though that unlike in the last section, we are not trying to tell if a query is entailed from a knowledge base.

Instead in planning are state will consist of complete models. We start with an explicit model of the world, and at each step we apply actions that update the model until we reach a model that satisfy the goal condition. The ACTIONS function and the goal test will be based on whether the model satisfies a logical condition.

Name	Type	Description
State	$\mathcal{M}$	A model specified with the closed world assumption
Initial state		An explicit model, specified as a conjunction of literals.
Actions		The list of actions that match the explicit pre-conditions (see below)
Result		A model transformed by the actions effects (see below).
Goal		Does the model satisfy a logical goal condition?

6.3 Actions

The core of planning is a set of actions which modify the current state of the world. AIMA uses the example of an airline logistics system that moves goods from one city to the next. As part of the system there may be an action like the following:

$$\begin{array}{ll} Action \ (& \textsc{Load}(c,p,a), \\ & \text{Precond} : \textsc{At}(c,a) \wedge \textsc{At}(p,a) \wedge \textsc{Cargo}(c) \wedge \textsc{Plane}(p) \wedge \textsc{Airport}(a) \\ & \text{Effect} : \neg\textsc{At}(c,a) \wedge \textsc{In}(c,p)) \end{array}$$

where there is an implied for all c, p, a.

We read this as follows: for cargo c, plane p, and airport a, if cargo is at airport a and plane at airport a, then this action can be applied to take remove cargo from airport a and put it on plane p.

If we have defined our state of the world to be

$$\textsc{Cargo}(Elephant) \wedge \textsc{Plane}(767) \wedge \textsc{At}(Elepant, JFK) \wedge \textsc{At}(767, JFK)$$

Then this action could be applied to transform the state to

$$\textsc{Cargo}(Elephant) \wedge \textsc{Plane}(767) \wedge \textsc{In}(Elepant, 767)\ \textsc{At}(767, JFK)$$

In general we interpret actions to mean if the current model state satisfies the precondition α, thenthe action is **applicable** and we can update the state model with effect β.

In the planning language described in AIMA, PDDL, all states, effects and goal tests are assumed to be conjunctions of (possibly negated) literals. This makes it quite simple to match the preconditions and goal states as well as keep track of the world.

6.4 World Updates

Let us now be a bit more explicit about what it means to update the world. The effect gives a list of literals that must occur in the new world.

Since we are using the closed-world assumption, the current state can be represented as simply a list of positive literals (since all others are assumed to be false). Each positive effect is simply added to the list, and negative effects are removed. In practice this can be implemented as a set operation where the result of an action is:

$$\textsc{Result}(s, a) = (s \setminus \textsc{Del}(a)) \cup \textsc{Add}(a)$$

Additionally, because actions are generic functions over all arguments of their input, it is required that every variable have some precondition. This effectively **grounds** the action by the current model. Without this condition there could be potentially unbounded actions where say any lifted literal say $\textsc{At}(Elepant, JFK)$, $\textsc{At}(Elepant, OHare)$, or even $\textsc{At}(Elepant, Elephant)$ could be produced as an effect.

6.5 Full Example:

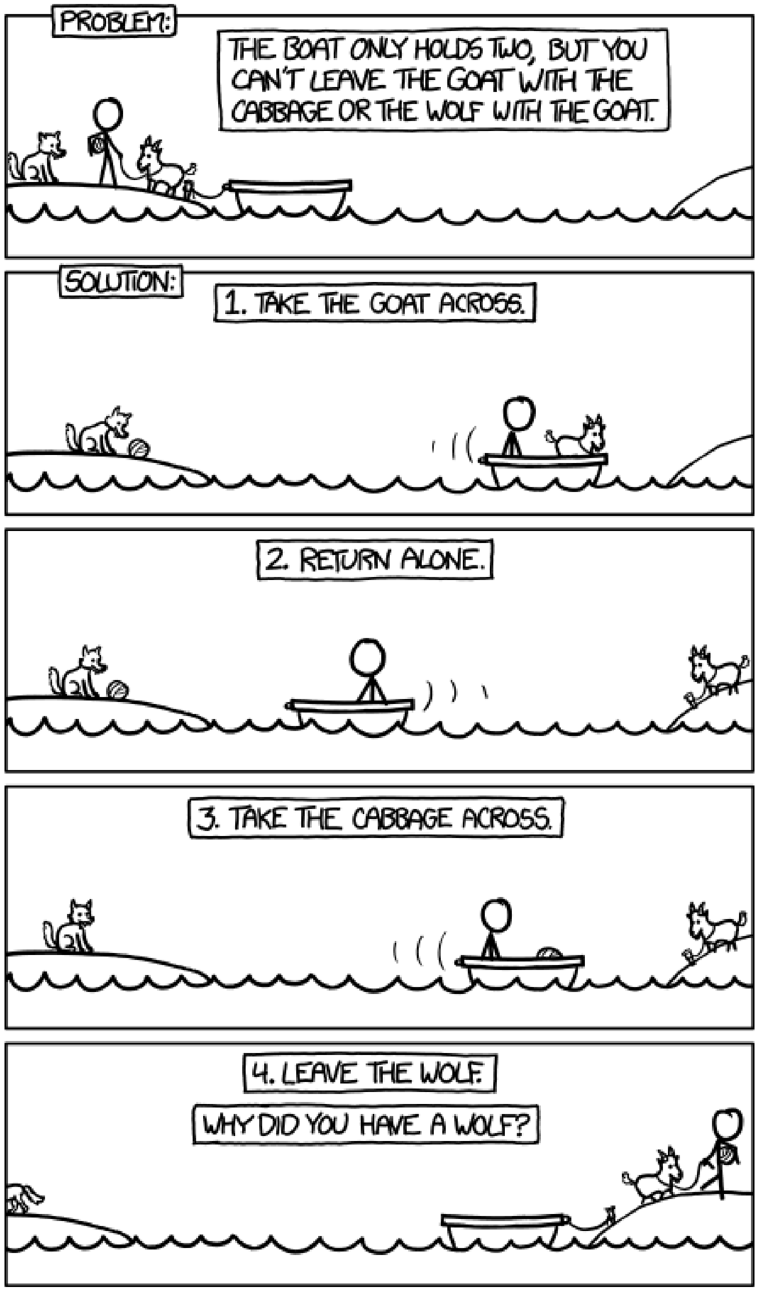
As a simple example of planning, we consider the famous wolf, goat, and cabbage problem dating from the 9th century. The problem states:

> A man has to cross a stream in a boat that can hold himself and only one other object. He needs to transport a wolf (lion, jackal), a goat (sheep), and a cabbage (bundle of hay, pumpkin). He must be sure that when he is out in the boat the wolf does not eat the goat and the goat does not eat the cabbage.

We formulate this as a planning problem. (Generally the AIMA description requires that preconditions should only be positive or negative literals, for simplicity here we allow a negative conjunction).

$$\begin{array}{rcl}
\textit{Init} & & \textsc{On}(\text{Left}, \text{Wolf}) \land \textsc{On}(\text{Left}, \text{Goat}) \land \textsc{On}(\text{Left}, \text{Cabbage}) \land \textsc{On}(\text{Left}, \text{Boat}) \\
& & \land \textsc{Across}(\text{Left}, \text{Right}) \land \textsc{Across}(\text{Right}, \text{Left}) \\
\textit{Action} & (& \textsc{Move}(s, s', a), \\
& & \text{Precond} : \textsc{Across}(s, s') \land \textsc{On}(s, \text{Boat}) \land \textsc{On}(s, a) \land \\
& & \neg(\textsc{On}(s', \text{Wolf}) \land \textsc{On}(s', \text{Goat})) \land \neg(\textsc{On}(s', \text{Goat}) \land \textsc{On}(s', \text{Cabbage})), \\
& & \text{Effect} : \neg\textsc{On}(s, \text{Boat}) \land \neg\textsc{On}(s, a) \land \textsc{On}(s', \text{Boat}) \land \textsc{On}(s', a)) \\
\textit{Action} & (& \textsc{Move-Empty}(s, s'), \\
& & \text{Precond} : \textsc{OtherSide}(s, s') \land \textsc{On}(s, \text{Boat}) \land \\
& & \neg(\textsc{On}(s', \text{Wolf}) \land \textsc{On}(s', \text{Goat})) \land \neg(\textsc{On}(s', \text{Goat}) \land \textsc{On}(s', \text{Cabbage})), \\
& & \text{Effect} : \neg\textsc{On}(s, \text{Boat}) \land \textsc{On}(s', \text{Boat})) \\
\textit{Goal} & & \textsc{On}(\text{Right}, \text{Wolf}) \land \textsc{On}(\text{Right}, \text{Goat}) \land \textsc{On}(\text{Right}, \text{Cabbage})
\end{array}$$

XKCD finds this version of the problem a little silly. Figure **??** gives their interpretation. Here is the path that they take for this problem.


PROBLEM:
THE BOAT ONLY HOLDS TWO, BUT YOU CAN'T LEAVE THE GOAT WITH THE CABBAGE OR THE WOLF WITH THE GOAT.
SOLUTION:
1. TAKE THE GOAT ACROSS.
2. RETURN ALONE.
3. TAKE THE CABBAGE ACROSS.
4. LEAVE THE WOLF.
WHY DID YOU HAVE A WOLF?

$$
\begin{aligned}
s_0 &= \textsc{On}(\text{Left}, \text{Wolf}) \wedge \textsc{On}(\text{Left}, \text{Goat}) \wedge \textsc{On}(\text{Left}, \text{Cabbage}) \wedge \textsc{On}(\text{Left}, \text{Boat}) \\
a_0 &= \textsc{Move}(Left, Right, Goat) \\
s_1 &= \textsc{On}(\text{Left}, \text{Wolf}) \wedge \textsc{On}(\text{Right}, \text{Goat}) \wedge \textsc{On}(\text{Left}, \text{Cabbage}) \wedge \textsc{On}(\text{Right}, \text{Boat}) \\
a_1 &= \textsc{Move-Empty}(Right, Left) \\
s_1 &= \textsc{On}(\text{Left}, \text{Wolf}) \wedge \textsc{On}(\text{Right}, \text{Goat}) \wedge \textsc{On}(\text{Left}, \text{Cabbage}) \wedge \textsc{On}(\text{Left}, \text{Boat}) \\
a_2 &= \textsc{Move}(Left, Right, Cabbage) \\
s_2 &= \textsc{On}(\text{Left}, \text{Wolf}) \wedge \textsc{On}(\text{Right}, \text{Goat}) \wedge \textsc{On}(\text{Right}, \text{Cabbage}) \wedge \textsc{On}(\text{Right}, \text{Boat}) \\
a_2 &= \textsc{Move}(Right, Left, Cabbage)
\end{aligned}
$$