Artificial Intelligence - 2017
Symbolic AI - Applications in Planning
Knowledge Based Agents

Prepared By;
Vijani Supeshala Piyawardana | BSc (Hons) Computer Science |
Demonstrator - COMP3004L, COMP3008L - UCD |
NSBM Green University Town

-

Knowledge

- What makes us intelligent
- Allow us to disentangle ambiguous pieces of information expressed in natural language
- Makes us to develop new pieces of knowledge

Knowledge based agents (also known as Model Based Reflex Agents)

- Agents act based on knowledge
- Requires knowledge in first place
- Have a Knowledge-Base (KB)
    - something made up of sentences (different from natural language sentences)
- Conclusions will be generated by inference starting from KB
    - Conclusions - new knowledge, not previously explicitly expressed
    - Inference - answer should follow from what is known
      (ex: mathematical theorem which follows by, well defined and unambiguous steps - you may have experience in proving pythagoras theorem, many trigonometric theorems)

Paradigm

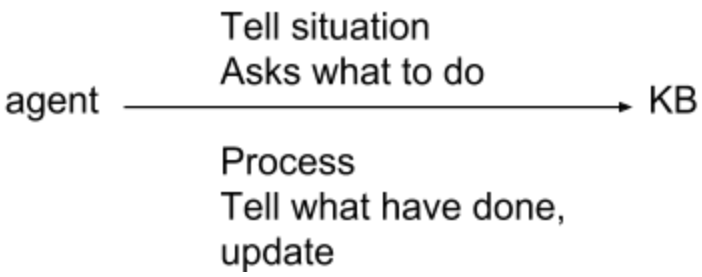
- An AGENT tells the KB what is the situation is or its PERCEPTIONS and ASKs what to do.
- Process will be done
- Then the AGENT tells the KB what it has done, to UPDATE the KB
- Open-ended paradigm

Agent

- It should be able to REPRESENT knowledge in KB
    - Initial knowledge is TELLed to it, when it is booted.
- It should be able to REASON leading to actions. (infer based on its perceptions and the KB)
- It should be able to LEARN

Knowledge based agents : act based on knowledge ---> requires knowledge.
Has a KB (Knowledge Base) ----> made up of sentences
Conclusions generated by inference starting from KB.

Inference : answer should follow what is known.



Logic

- To draw conclusions from the KB, you need to formalise the process
- To do that, need to define an unambiguous language
- That language should have well defined syntax to write sentences in that language
- You have to find a way to attach meaning to sentences, semantics.

Syntax

- Means that a sentence is well formed in a particular language.
  - x + 4 = y
  - A & B & C == true
- To know what is correct/what is wrong > you need to establish rules.
- The fact, a sentence is syntactically correct does not say anything about its meaning, or truth.

Semantics

- Goes with the meaning of a sentence
- Will attach a truth value to a sentence according to some rules.
- We want to be able to infer truth values to new sentences according to the KB
- The semantics will in fact deal with all possible worlds, that is, a true sentence will be true everywhere.

Entailment

- We draw conclusions based on entailment, the idea that a sentence follows logically from another one or set of ones.
- a □ b
- a entails b if and only if, in every model in which a is true, b is also true.
- The truth of b is contained in the truth of a.

Checking Entailment

That can be checked, for instance, by enumerating all cases.
Or by deriving conclusions based on the KB.
We need an inference algorithm.
Desirable properties:
– soundness
– completeness

Soundness and Completeness of an inference algorithm

Soundness: the inference algorithm if only derives sentences that are entailed. This is evidently highly desirable. Without this, there is nothing to be concluded.

Completeness: an inference algorithm is complete if it can derive any sentence that is entailed.
If we have this, it is pretty powerful.

Logic and Formalism

Syntax and semantics based on rules.
The actual grounding in a specific world, or model, is up to us and depends on whether the rules we encode are reasonable in the world we are trying to model.
That is: this is maths. Whether the conclusions we draw from the maths are sensible or not depends on whether the maths is an OK model in the first place.

Logic and the real world

If the KB is true in the real world, then every inference made by a sound algorithm is also true.
This is something that excited some minds. In fact it works as such in some limited environments.
The real problem is that very few sentences are in fact true in the real world in the first place.

Grounding

- The actual grounding in a specific world, or model is upto us, and depends on whether the rules we encode are reasonable in the world we are trying to model.
- If the KB is true in real world, then every inference made by a sound algorithm is also true.
- The process of figuring out that something is true in the real world in the first place is incredibly complex and is ultimately a philosophical question humans have been debating for millennia.
- Most of what we know is based on inference derived from observed facts. As stubborn as facts might be, they aren't always stubborn in confirming laws.
- Think of black swans.
- That is why logic didn't have quite the impact that researchers hoped, and why we need fuzzy reasoning techniques.

Propositional Logic

Objects are called prepositions.
Proposition is a statement that is true or false, but not both.

Syntax: there are atomic sentences, composed of single symbols, proposition symbols.
The actual symbols we use are arbitrary (though some mnemonic value helps), and can be true or false.
Complex sentences are built from simpler ones (ultimately atomic symbols) using some logical connectives:

¬	not
^	and
V	or
⇨	implies
⇔	biconditional

Single atomic symbols (possibly negated) are sometimes called literals.

Sentence

AtomicSentence | ComplexSentence

AtomicSentence
True | False | Symbol

Symbol
P | Q | R | …

ComplexSentence
¬ Sentence
Sentence ^ Sentence
Sentence v Sentence
Sentence ⇨ Sentence
Sentence ⇔ Sentence

Example

¬ P v Q v R ⇨ S

A ^ (B v ¬C)

Precedence is (highest to lowest)
¬ ^ v ⇨ ⇔

Parenthesis : The definition of the syntax would always require parenthesis, but in some cases they can be omitted, when the meaning is clear based on precedence (or there is ambiguity over truth-preserving sentences)

In propositional logic, a model simply fixes the truth value -true or false- for every proposition symbol.
E.g.:
m = {P = true, Q = true, R = false}

Once the symbols are pinned down, all we need to know is how to interpret the connectives. We need their truth tables.

Meaning of Connectives : Truth Table

A	¬A
False	True

True	False

A	B	A ^ B
False	False	False
False	True	False
True	False	False
True	True	True

A	B	A v B
False	False	False
False	True	True
True	False	True
True	True	True

A	B	A ^ B
False	False	False
False	True	False
True	False	False
True	True	True

A	B	A ⇨ B
False	False	True
False	True	True
True	False	False
True	True	True

A	B	A ⇔ B
False	False	True
False	True	False
True	False	False
True	True	True

Implies :
If the antecedent is true we imply the consequent is true. Otherwise we make no claim.
Ex : If it is raining then there are clouds in the sky.

Inference

One way to do it is to enumerate all the models and check what is true according to the semantics of the language.
This is possible if the world is finite, and practical if it is not too big.
If you have n symbols, there are 2^n combinations of truth values (models). It is only a matter of checking them all.
A number of them may turn out to be sound according to the rules.

Inference algorithm (for finite and small problems)

By enumerating all models and check what is true according to the semantics.

- Enumerate all combinations (n symbols ---> 2^n of combinations) of truth values for all ground symbols.
- Add to that, sentences from KB with their associated truth values based on symbols.
- Pick those models which all rules are true.
- Check our new sentence is true in all those models.
- If so, it's true...

Sound and Complete

The algorithm is:
sound (it directly implements the definition of entailment), and
complete (it only has to enumerate a finite number of models to reach the conclusion).
As said, that doesn't mean it is tractable.

Equivalence

Logical equivalence
two sentences a and b are logically equivalent if they are true in the same set of models. •

a ≡ b

This is true if and only if:

a □ b and b □ a

(a entails b and b entails a) ----> same as; a implies b and b implies a

(For all combinations of truth values a and b have the same truth value)

Logical Equivalence in Propositional Logic

A relationship between two statements or sentences (prepositions) in prepositional logic. Two sentences a and b are said to be logically equivalent if they have identical truth values. Informally, a and b are equivalent if whenever a is true, b is true and vise versa. (iff : if and only if)

Symbol for Logical Equivalence a ⇔ b or a ≡ b

Validity

A sentence is valid if it is true in all models.

E.g. A v ¬A

These are also called tautologies.

All valid sentences are, obviously, logically equivalent.

Inference mechanisms are typically embodied by a valid sentence.

A	¬A	A V ¬A
False	True	True
True	True	True

Satisfiability

- A sentence is satisfiable if it is true in some model.
- If a is true in a model m, we say that "m satisfies a" or "m is a model of a".
- You can check this by enumerating models until you find one where the sentence is true (so long as you have finite models).
- Many problems can be mapped to satisfiability, e.g. search.

Satisfiability and Validity (Obviously the two are connected.**)**

a is valid iff ¬a is unsatisfiable.
Proving something is valid by proving its negation is unsatisfiable (is exactly the same as proving a theorem ab absurdum.)
This is also called proof by refutation or by contradiction.
You assume something is false and show this leads necessarily to a clash with the axioms.

Patterns : inference rules : always true

Can be used to reach conclusions without enumerating models.

Reasoning Patterns

(a ⇨ b, a) implies b (Modus ponens)
(a ^ b) implies a (And-elimination)
It can be shown that these are always true (valid for all models).
As a consequence, they can be used to reach conclusions without enumerating models.

Modes ponens

(a ⇨ b , a) implies b

a ⇨ b

a

---

b

(a ⇨ b) ^ a ⇨ b

Simplification

(a ^ b) implies a

a ^ b

---

a

a ^ b ⇨ a

Patterns and equivalences

There are many known equivalences, which can all be used to draw conclusions like Modus Ponens and And-elimination.

Equivalences : known equivalences to draw conclusions like modes ponens.

a ^ b ≡ b ^ a	commutative
(a ^ b) ^ c ≡ a ^ (b ^ c)	associative
¬ (¬a) ≡ a	involution
a ⇨ b ≡ ¬a ⇨ ¬b	
¬(a ^ b) ≡ ¬a v ¬b	de morgan's

Proof as search

By applying chains of equivalences starting from known facts (axioms, sentences in KB) you can prove other sentences.
It is a search problem which you can use equivalence at a time to modify the state you are in.
Can start from KB and process until find the sentence or
Start from the sentence and reduce it to the KB.

Proof and Complexity

Prove by expansion of equivalences is complex. (generally exponential)
More efficient than enumerating models.
Because you can ignore much of the world. (If you need to prove something about a and b, you don't need to use any sentence in KB which does not contain them.)

Propositional Logic is nice and simple. Works in some very well defined domains. (circuit design, theorem proving, etc)
Each atomic fact requires a separate unique symbol.
    If there are n people and m locations, representing the fact that some person moved from one location to another requires nm^2 separate symbols.
Not good for representing knowledge in more flexible form;
    To mirror human reasoning
    To embody the unrestricted real world
The answer is :- First Order Logic

Propositional Logic is based on facts that may or may not hold.
First Order Logic makes a different ontological commitment which is rich. (objects, relations among them that may or may not hold.)

Advantage of FOL :
FOL is more like the natural language, it can easily used to demonstrate the real world's scenario's.

First Order Logic (First Order Predicate Calculus / Predicate Logic)

This is a much more powerful language than propositional logic, though still limited.
E.g. you can't easily express fuzzy belief in it, or temporal succession, etc., and it is still far from the expressiveness of natural languages.
But, among other things, it allows one to say things like "areas adjacent to a rubbish bin are smelly" without having to enumerate all rubbish bins and, for each one of them, all adjacent areas..

Natural Languages

They are a very complex instrument indeed.
They probably are more of a means of communication than of representation, although the battle on the relationship between language and thought rages on.
E.g. how many words to Eskimos (or skiers) use for snow? How many do you have in Sinhala? Does this shape thought, or is it merely an environmental accident of no consequence?
An early thrust to understand natural language through logic eventually ebbed, and now heuristics and, especially data rule.

What are the things in Languages

Objects: people, numbers, houses, ideas, theories, Gianluca Pollastri, soccer, rice and curry.
Relations: good, spicy, tall, big (unary); brother of, bigger than, owns, occurred after (n-ary).
Functions (notice there is one output): father of, best friend, third goal of, beginning of.

More simply, one can think of most sentences as being built of objects, properties and relations.
Ex:

Evil King John ruled England in 1200.
Objects: John, England, 1200
Properties: Evil, King.
Relations: ruled

So, what is First Order Logic?

- It centres around objects and relations.
- It allows us to state general properties about the universe without enumeration.
- While propositional logic is based on facts which may or may not hold, first order logic makes a different ontological commitment, that is: there are objects, and relations among them which themselves may or may not hold.

Beyond First Order

- Notice that you can go further and have logics than make assertions about relations in general (higher order logic).
- Or you could add time, at which point facts and relations become dependent on it (temporal logic).
- Or you can make a different epistemological commitment and have more than true and false (probabilistic and fuzzy logic).

First Order Logic : Elements

A model for FO logic has objects in it: it's domain is the set of objects it contains.
Then there are relations, which are simply tuples over the objects.
Ex:
if you have the objects:

Richard the Lionheart (in short: Richard),
King John (in short: John),
the left leg of Richard,
the left leg of John,
the crown,

then the relation "brothers" could be:

{
(Richard,John),
(John, Richard)
}

The "on head" relation may be:

{
(crown, John)
}

You can think of some relations as functions, when one object is related to exactly one other object through it.
Ex:

"left leg" maps each person to his/her left leg, of which there is typically only one.
(Richard) -> Richard's left leg
(John) -> John's left leg

Strictly speaking, the functions should be total.
That is: there should be a mapping for each object in the model's domain, according to the function.
So, the crown has a left leg.
You can have a dummy left leg object to deal with this.
As long as one makes no assertions about the left legs of things that have no left legs, it doesn't really matter..

Elements of First Order Logic

First order logic is build upon propositional logic and allows more flexible and compact representation of knowledge. Centers around objects and relations. States general properties about the world without enumeration.

Predicate logic / First Order Logic includes a richer ontology / elements:
The main elements are the symbols that stand for objects, relations and functions.

- objects : things with individual identities. (terms)
- properties : of objects to distinguish from other objects. (unary predicates on terms)
- relations : hold among sets of objects. (n-ary predicates on terms)
- functions : subset of relations, only one value for any given input. (mappings from terms to other terms)

Ex:

Richard, John; Brother, OnHead, Person, King, Crown; LeftLeg.

- Objects : students, lectures, companies, cars,...
- Properties : blue, oval, even, large, …
- Relations : brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, …
- Functions : father-of, best-friend, second-half, one-more-than,...

Allows more flexible and compact representation of knowledge

Move(x, y, z) for person x moved from location y to z.

First Order Semantics

You have to relate sentences to models to figure out truth.
This generally entails saying what names refer to in the real world.

Ex:

Richard refers to king Richard the Lionheart;
Brother is the brotherhood relation;
OnHead is the relation that holds between the crown and king John;
Crown, Person, King refer to objects who are crowns, people and kings; etc.

This is just one interpretation. You could map Crown to Richard Lionheart, LeftLeg to King John, etc.
This way there are 25 interpretations just involving the symbols John and Richard.
You can even have more than one name referring to the same entity..
As entailment, validity, etc. have to be checked for all models and there can be enormous (or unbounded: think integers) numbers of interpretations, you cannot really figure things out by enumeration.

Knowledge Engineering and First Order Logic

Knowledge engineering, in brief, is the process of gathering and formalising necessary axioms in the KB so that queries can be run successfully.
In general, this may be a fairly difficult, expensive and time consuming exercise in real-world domains.
This is in fact one of the reasons for the stagnation of the field after some initial excitement, and for the shifting of focus towards automated learning in AI.

1. **Identify the task.**

You have to figure out the range of questions that might be possibly asked.
This depends on what is required, but also on what is possible
(e.g. what sensors you have, what FO logic supports).

2. Assemble the relevant knowledge.

This is where it gets sticky. You have to understand the domain.
If it is a complex one, this may require interviewing experts, understanding what needs to be modelled and what can be left out, etc.
This is known as knowledge acquisition.

3. Decide on a vocabulary.

This, again, may be a hard choice, and there may be many more or less equivalent solutions. You have to decide who is a predicate, a function, a constant. The result of this, which depends on style, will be the ontology (theory of nature of existence).

4. Encode.

Here you finally write down the rules based on the choices you have made before. There will be a general step, in which you encode the broad ontology, and a specific step in which you encode specific atomic sentences about the problem instance.

5. Ask queries.
Here you finally get to use the KB and solve unanswered questions, whose answers may or may not be added to it. If anything goes wrong (as it will), you will have to:

6. Debug.
The typical problem is that you will get answers that are correct for the knowledge as it is written, but the answers will make it clear that you haven't written the right way.. This involves rewriting some rules and, if things are more seriously wrong, perhaps go back all the way to defining the ontology.

General Problem with logic

- In order to use it, you have to come up with an ontology.
- If your problem is sufficiently complex, this ontology will cover a big chunk of the world.
- People have debated about categories of objects and taxonomies for millennia. .
- It is difficult enough to come up with a set of classes for your code, where you can define very well where the ropes are. Imagine doing it for an unpredictable environment (e.g. one in which a robot moves).
- What makes knowledge particularly hard to organise is the fact that most facts in the natural world are not true or false.
- While you can somehow contrive to add degrees of belief to FO, this is, well.. contrived.

Planing

If you want to build an intelligent agent, it has to be able plan what to do.
The problem with planning, as with everything in AI, is knowledge and heuristics.
Ex:
If you want to have a book and do not know you need to buy it (or borrow it) in order to have it, think of all the actions you will have to try before you succeed.

If, however, you know this:

Buy(x) ⇨ Have(x)

Then you can do a single (backward) unification step from the goal Have(Book1234) to get the necessary action and have solved your planning problem.

Planning often involves multiple goals,
Ex: buying multiple books, or delivering multiple packages across an area.
When you hear "multiple", immediately think exponential.

In order to solve problems with multiple goals efficiently you have to have a way to partition the problem into parcels you can tackle (you may have to readjust things when you recompose the parts into one, but this should be abundantly offset by the gains).
Again, this is about having good heuristics

This is just an action sequence which when executed in the initial state of the system leads to a state which satisfies the goal.

Planing : Find a set of actions to achieve a goal. (finding one such sequence)

Make some assumptions;

- Time is a sequence of instants
- Actions have no duration
- Actions have deterministic outcomes

Ex : Spare Tire Problem

Formalizing Planning

You can represent states of the world using logic.
Ex:

- Poor ^ Unknown, At(Plane1, Melbourne) ^ At(Plane2, Sydney)
- (propositional vs. FO)

Assume that states are function-free literals.
Let us also make a closed-world assumption, that is: states not mentioned in preconditions are assumed false.

A goal (generally multiple) can be represented by a conjunction of (positive) literals,
Ex. Rich ^ Famous or At(P2, Tahiti).

A goal is assumed satisfied if we reach a state that contains it,
Ex: Rich ^ Famous ^ Miserable.

An action is specified in terms of the preconditions that must hold before it can be executed and the effects that ensue when it is executed.
For example, an action for flying a plane from one location to another is:

Action(Fly(p, from, to),
PRECOND : At(p,from) ^ Plane(p) ^ Airport(from) ^ Airport(to)
EFFECT : ¬At(p,from) ^ At(p,to)

This is an action schema in that you get different action by instantiating the variables p, to and from.

All parameters (variables) in the precondition must appear in the action parameter list.
The effect is a new state the action results into.

Planning Semantics

Any action is applicable whenever the preconditions are met.
Typically, in FO, this implies substitution.

Ex: if we know:
At(P1,JFK) ^ At(P2,SFO) ^ Plane(P1) ^ Plane(P2) ^ Airport(JFK) ^ Airport(SFO),
then we meet the precondition by substituting
{p/ P1,from/JFK,to/SFO} (or {p/P2,from/SFO,to/JFK})
At this point Fly(P1,JFK,SFO) is possible.

Results

If you act with action a when in state s, then the resulting state s':
contains all that was in s
plus the positive effects of a
minus any negative literals in the effects of a, if they happened to be in s

After Fly(P1,JFK,SFO) the state becomes
[At(P1,JFK) ^ At(P2,SFO) ^ Plane(P1) ^ Plane(P2) ^ Airport(JFK) ^ Airport(SFO)] + [¬At(p,from) ^ At(p,to)],
which is:
At(P1,SFO) ^ At(P2,SFO) ^ Plane(P1) ^ Plane(P2) ^ Airport(JFK) ^ Airport(SFO)

Solving a Planning Problem

In brief, this is just an action sequence which, when executed in the initial state of the system, leads to a state which satisfies (contains) the goal.
With this formalisation, planning means finding one such sequence (possibly with some cost constraints on it).

Example: cargo transport

- Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ Cargo(C1) ∧ Cargo(C2) ∧ Plane(P1) ∧ Plane(P2) ∧ Airport(JFK) A Airport(SFO))
- Goal (At (C1, JFK) ∧ At (C2, SFO))
- Action(Load(c, p, a),
- PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
- EFFECT: ¬At(c, a) ∧ In(c, p))
- Action(Unload(c, p, a),
- PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
- EFFECT: At(c, a) ∧ ¬In(c, p))
- Action(Fly(p, from, to),
- PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
- EFFECT: ¬At(p, from) ∧ At(p, to))

Example: solution

- (Obvious to humans..)
- [Load(C1,P1,SFO),Fly(P1,SFO,JFK), Unload(C1,P1,JFK), Load(C2,P2,JFK), Fly(P2,JFK,SFO), Unload(C2, P2, SFO)]

Example: The spare tire problem

Consider the problem of changing a flat tire. More precisely, the goal is to have a good spare tire properly mounted onto the car's axle, where the initial state has a flat tire on the axle and a good spare tire in the trunk.
To keep it simple, our version of the problem is a very abstract one, with no sticky lug nuts or other complications.
There are just four actions: removing the spare from the trunk, removing the flat tire from the axle, putting the spare on the axle, and leaving the car unattended overnight.
We assume that the car is in a particularly bad neighborhood, so that the effect of leaving it overnight is that the tires disappear.

Steps to solve the planning problem:

Representation of states : decompose world into logical conditions and represent a state as a conjunction of positive literals. Closed world assumption : states not mentioned in preconditions are assumed false.

Representation of goals : partially specified state, represent as a conjunction of positive ground literals. A propositional state s satisfies a goal g if s contains all atoms in g.

Representation of actions : represent in terms of preconditions and effects.

Precondition : a conjunction of function free positive literals stating what must be true in a state before the action can be executed. Must hold before it can be executed.

Effect : a conjunction of function free literals describing how the state changes when action is executed. Positive literal ¬P to be true while negative literal P to be false. Ensure when it is executed.

Now see how to solve spare tire problem:

01. Initial state (identify initial states in the problem definition)
    I. flat tire on the axle
    Ii. spare tire on the trunk

02. Action (identify actions)
    I. remove spare tire from the trunk
    Ii. remove flat tire from the axle
    Iii. put the spare tire on the axle

03. Specify the goal
    I have a good spare tire properly mounted to the car's axle.

1. Init (At(Flat,Axle) ^ At(Spare,Trunk))
2. Goal (At(Spare,Axle))
3. Action (Remove(Spare, Trunk),
   PRECOND : At(Spare, Trunk)
   EFFECT : ¬At(Spare, Trunk)^At(Spare, Ground))

   Action (Remove(Flat, Axle),
   PRECOND : At(Flat,Axle)
   EFFECT :¬At(Flat,Axle)^At(Flat,Ground))

   Action (PutOn (Spare, Axle),
   PRECOND : At(Spare,Ground)^At(Flat,Axle)
   EFFECT : ¬At(Spare,Ground)^At(Spare, Axle))

We can see that the final action is satisfied the goal.

The ADL description of the problem is shown in Figure 11.3.
Notice that it is purely propositional. It goes beyond STRIPS in that it uses a negated precondition, **¬At(Flat, Axle),** for the **PutOn(Spare, Axle)** action. This could be avoided by using **Clear(Axle)** instead, as we will see in the next example.

$Init(At(Flat, Axle) \wedge At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(Spare, Trunk),$
PRECOND: $At(Spare, Trunk)$
EFFECT: $\neg At(Spare, Trunk) \wedge At(Spare, Ground))$
$Action(Remove(Flat, Axle),$
PRECOND: $At(Flat, Axle)$
EFFECT: $\neg At(Flat, Axle) \wedge At(Flat, Ground))$
$Action(PutOn(Spare, Axle),$
PRECOND: $At(Spare, Ground) \wedge \neg At(Flat, Axle)$
EFFECT: $\neg At(Spare, Ground) \wedge At(Spare, Axle))$
$Action(LeaveOvernight,$
PRECOND:
EFFECT: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
$\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle))$

Figure 11.3 The simple spare tire problem.

Past Exam Questions

2016 AI Batch 4

Question 4 (a) In propositional logic, define what is meant by *logical equivalence*

(2 marks)

(b) By writing down the truth table for each expression, show that

$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$

(4 marks)

(c) What are the elements of First Order (FO) logic?

(8 marks)

(d) Describe how planning problems can be solved using First Order logic?

(6 marks)

Answer

Question 4 (a) Two sentences a and b are logically equivalent if they are true in the same set of models

(2 marks)

(b)

A	B	$A \wedge B$	$\neg(A \wedge B)$
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

A	B	$\neg A$	$\neg B$	$(\neg A \vee \neg B)$
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Since last column of truth table is identical, the two expressions are equivalent.
(1.5 marks per table, 1 mark for recognising equivalence)

(4 marks)

(c) The elements are objects, properties and relations. (1 mark each for mentioning each element) (1.66 mark each for an example that shows an understanding for what they are)

(8 marks)

(d) Important things to mention

i. Represent states of the world using first-order logic – specifically, assume they are function-free literals.

ii. Write down a goal as a conjunction of literals

iii. Specify actions in terms of preconditions that must exist before they can be applied and effects that ensue when they are executed.

iv. Carry out a search from the initial state to the goal, where moving from state to state is carried out by applying the allowable actions in our knowledge-base i.e. those actions whose preconditions are true.

Again, student may not list the above so clearly, so need to be generous if the gist of the above is given. Students are likely to give an example of an action, or of a representation of a planning goal. They should get a mark for each such example. An example and a reasonably good explanation of the planning steps, should given full or nearly full marks.

(6 marks)

2015 AI Batch 3

Question 4 (a) Write down the truth table for the implies operator ($\Rightarrow$) in propositional logic.

(2 marks)

(b) If A is `true`, B is `false` and C is `true`, what is the truth value of the following expressions:

i. $\neg A \vee B \Rightarrow C$

ii. $A \Rightarrow (B \wedge C)$

(4 marks)

(c) What are the elements of First Order (FO) logic?

(8 marks)

(d) Describe how planning problems can be solved using First Order logic.

(6 marks)

2014 AI Batch 2

Question 4 (a) In propositional logic, define what is meant by *logical equivalence*

(2 marks)

(b) By writing down the truth table for each expression, show that

$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$

(4 marks)

(c) What are the elements of First Order (FO) logic?

(8 marks)

(d) Describe how planning problems can be solved using First Order logic?

(6 marks)

A	B	¬ A	¬ B	A ^ B	¬ (A ^ B)	¬ A v ¬ B
T	T	F	F	T	F	F
T	F	F	T	F	T	T
F	T	T	F	F	T	T
F	F	T	T	F	T	T

Therefore; ¬ (A ^ B) ≡ ¬ A v ¬ B

Useful Documents

https://www.cs.utexas.edu/~mooney/cs343/slide-handouts/fopc.4.pdf
http://aima.cs.berkeley.edu/2nd-ed/newchap11.pdf

Vocabulary

Disentangle - dis-entangle - පටලැවිල්ලනේ බේරනවා
Entangle - en-tangle - පටලවනවා
Tangle - අවුල
(Rapunzel tangled)
Ambiguous - බහු අර්ථ ඇති
Unambiguous - Un-ambiguous - තේරුම නිශ්චිත
Inference - අනුමානය, තීරණය
Infer - අනුමාන කරනවා, තීරණය කරනවා
Conclusion - නිගමනය, අවසානය
Explicitly - explicit-ly - explicit - පැහැදිලි, සවිස්තර
Paradigm - ක්රියාමාලාව
Open-ended - having no predetermined limit or boundary.

- (of a question) allowing the formulation of any answer, rather than a selection from a set of possible answers.

Draw - උකහා ගන්නවා, බැංකුවකින් මුදල් හැරගන්නවා (withdraw)
Formalise - give (something) legal or formal status, give a definite structure or shape to.
Infer - අනුමාන කරනවා
Semantic - අර්ථ විචාරය සම්බන්ධ
Entailment - An **entailment** is a deduction or implication, that is, something that follows logically from or is implied by something else. In logic, an **entailment** is the relationship between sentences whereby one sentence will be true if all the others are also true.

Antecedent - පරෙ පදය
consequent - පසුපදය
Inference - නිගමනය
Tracktable - හික්මවිය හැකි
Enumeration - ගණන් කිරීම
Stagnation - ඇණහිටීම