Propositional Logic

We need to know some rules for making correct inferences

Syntax, what the wffs look like

- True
- False
- Proposition symbol: Capital letter first, represents a proposition, e.g.
  - W13 means there is a wumpus in room [1,3]
  - FacingEast means the agent is facing East
- A wff in parentheses is also a wff
- A wff preceded by ¬ (not) is also a wff
  - Symbol or ¬Symbol is called a literal
  - X and ¬X are Complementary literals
- Two wffs joined by ∧ (and) is also a wff: Conjunction
- Two wffs joined by ∨ (or) is also a wff: Disjunction
- Two wffs joined by ⇒ is also a wff: Implication or If
- Also ⇐, A ⇐ B means exactly B ⇒ A
- Two wffs joined by ⇔ is also a wff: If-and-only-if (iff)
  - A ⇔ B means that both A ⇒ B and B ⇒ A are true

Semantics

- A model gives truth values to every symbol
  - If three symbols are in use, there are 8 possible models
- Semantics must show how to find truth of wff in any model
  - A simple recursive set of rules
  - Or maybe truth tables

Wumpus world again

- Symbols
  - Pij means there is a pit in room [i,j]
  - Wij means there is a wumpus in room [i,j]
  - Bij means you would feel a breeze in room [i,j]
  - Sij means you would smell a stench in room [i,j]
  - Lij (L = location) means the agent is in room [i,j]
- It is a very small world, so we could write all the rules explicitly
  - ¬P00 - there is no pit in the starting room, that is R1
  - B00 ⇔ P10 ∨ P01, that is R2
  - B01 ⇔ P00 ∨ P11 ∨ P02, that is R3
  - …
  - B33 ⇔ P23 ∨ P32, that is R17
  - S00 ⇔ W10 ∨ W01, that is R18
  - S01 ⇔ W00 ∨ W11 ∨ W02, that is R19
  - …
  - S33 ⇔ W23 ∨ W32, that is R33
- And also for the facts that the agent learned at the beginning
  - ¬B00, that is R34

B01, that is R35

...

- A very simple inference algorithm

  e.g. want to know if P10 is false

  Check every possible model one-by-one, $2^{\text{number of rules}}$ models

  Only in a few of them are all the rules true

  In all of those P10 is false

  Thereby we can deduce that there is no pit in room [1,0]

$A \equiv B$ is true if and only if both $A \models B$ and $B \models A$ are true

The formulae are equivalent. $\equiv$ makes a claim about wffs, $\Leftrightarrow$ is part of them
Equivalent wffs can be substituted for one another

$A \wedge B \equiv B \wedge A$ - Commutativity

$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ - Associativity

$\neg\neg A \equiv A$

$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$ - Equivalence elimination

$A \Rightarrow B \equiv B \vee \neg A$

$A \Rightarrow B \equiv \neg B \Rightarrow \neg A$ - Contrapositive

... and a whole bunch more

A wff is Valid if it is true in all possible models - Tautology
A wff is Satisfiable if it is true in at least one possible model

$A \models B$ is true if and only if $A \Rightarrow B$ is valid - The deduction theorem

Theorem proving

- Never mind about checking all possible models
- Apply inference rules to every fact already in the knowledge base

Inference rules

- Horizontal lines

  If you know all the things above the line, you can deduce the thing below the line

- $A \Rightarrow B$, A ______ B - Modus Ponens
- $A \wedge B$ ______ A - And elimination
- Equivalence rules too

Example

- Given

  $\neg$P00 - there is no pit in the starting room, that is R1

  B00 $\Leftrightarrow$ P10 $\vee$ P01, that is R2

  B01 $\Leftrightarrow$ P00 $\vee$ P11 $\vee$ P02, that is R3

  $\neg$B00, that is R34

  B01, that is R35

- We want to prove $\neg$P01

  Equivalence elimination to R2:

  (B00 $\Rightarrow$ P10 $\vee$ P01) $\wedge$ ((P10 $\vee$ P01) $\Rightarrow$ B00), that is Rn

  And elimination to Rn:

(P10 ∨ P01) ⇒ B00, that is Rn+1

Contrapositive to Rn+1:

¬B00 ⇒ ¬(P10 ∨ P01), that is Rn+2

Modus Ponens on Rn+2 and R34:

¬(P10 ∨ P01), that is Rn+3

DeMorgan's rule on Rn+3:

¬P10 ∧ ¬P01, that is Rn+4

And elimination on Rn+4:

¬P01, Q.E.D.

- How is it done?

  We've got a search tree

  States are sets of wffs that we already know to be true

  The axioms are at the root,

  The inference rules let us work out the next states

  It is a big tree,

  a lot of wffs are true, and

  there are a lot of inference rules

  Or a slightly different kind of search tree,

  where the axioms are just taken for granted

  and nodes only contain newly deduced facts

  This example is Monotonic:

  Once you discover something is true, it stays true for ever

Resolution

- A-lot-of-ORs ∨ X, Another-lot-of-ORs ∨ ¬X

  ______ A-lot-of-ORs ∨ Another-lot-of-ORs

- A Clause is a bunch of literals ORed together: X ∨ ¬Something ∨ Cat

  Resolution works on clauses

- Remember when the agent first moved to [0,1]

  The percepts are S01 but not B01. Can make a new fact:

  ¬B01, that is Rn+5

  R3, which was B01 ⇔ P00 ∨ P11 ∨ P02, is equivalent to

  ¬B01 ⇔ ¬(P00 ∨ P11 ∨ P02)

  So with Rn+3, ¬B01, we get

  ¬(P00 ∨ P11 ∨ P02)

  De Morgan's law gives

  ¬P00 ∧ ¬P11 ∧ ¬P02

  And elimination gives us all three of

  ¬P00, which we already knew

  ¬P11, that is Rn+6

  ¬P02, that is Rn+7

  The equivalence rule for ⇔ applied to R3 gives

  (B01 ⇒ P00 ∨ P11 ∨ P02) ∧ (P00 ∨ P11 ∨ P02 ⇒ B01)

  Then using R35, which is B01, modus ponens gives

  P00 ∨ P11 ∨ P02, that is Rn+8

  Now the literal ¬P11 from Rn+6 resolves with P11 from Rn+8

  P00 ∨ P02, that is Rn+9

And ¬P00 from R1 resolves with P00 from Rn+9

P02, that is Rn+10

Using resolution we have deduced that there is a fatal pit in [0,2]

Resolution algorithm

- If a knowledge base is in Conjunctive Normal Form (CNF)
  Then resolution can tell us everything
- Resolution algorithms work through proof by contradiction

  To prove that KB ⊨ Conjecture, we prove that
  KB ∧ ¬Conjecture is unsatisfiable, or impossible
- Start with KB ∧ ¬Conjecture converted to CNF
- Apply resolution, where possible, to pairs of clauses
  Each time, the result is a new clause, which is added to KB
- In the end, either
  There is nothing left to resolve
  therefore KB does not entail Conjecture, or
  The resolution of two clauses produces nothing (i.e. False)
  therefore KB does entail Conjecture
  this can only happen when two contradictions
  e.g. X and ¬X are resolved
- Example, start with KB = just two rules, R2 and R34

  KB = (B00 ⇔ P10 ∨ P01) ∧ (¬B00)

  Want to prove ¬P10, so convert KB ∧ P10 into CNF

  ¬B00 ∨ P10 ∨ P01 - clause 1
  ¬P01 ∨ B00 - clause 2
  ¬P10 ∨ B00 - clause 3
  ¬B00 - clause 4
  P10 - clause 5

  Resolve clause 1 and clause 2 around P01
  ¬B00 ∨ P10 ∨ B00 - clause 6, ≡ True
  Resolve clause 1 and clause 2 around B00
  P10 ∨ P01 ∨ ¬P01 - clause 7, ≡ True
  Resolve clause 1 and clause 3 around P01
  ¬B00 ∨ P10 ∨ B00 - clause 8, ≡ True
  Resolve clause 1 and clause 3 around B00
  P10 ∨ P01 ∨ ¬P01 - clause 9, ≡ True
  Resolve clause 2 and clause 4 around B00
  ¬P01 - clause 10
  Resolve clause 3 and clause 4 around B00
  ¬P10 - clause 11
  Resolve clause 5 and clause 11 around P01
  yields nothing
  Therefore ¬P10 is true

The Horn clause

- Any clause with at most one positive literal is a Horn clause
  e.g. ¬A ∨ B ∨ ¬C ∨ ¬D

- A Horn clause that actually has a positive literal is a Definite clause
- A definite clause can be converted into an implication with all positives

  $A \wedge C \wedge D \Rightarrow B$
- With no positives, e.g. $\neg A \vee \neg C \vee \neg D$, they are Goal clauses
- Just one positive and nothing else is a Fact, e.g. X
- This is the basis of Logic Programming

An agent in the wumpus world

- A huge number of basic facts stating the rules of the world …
- A breeze somewhere means a neighbouring pit:

  B00 ⇔ P01 ∨ P10, and so on as before.
- A stench somewhere means a neighbouring wumpus:

  S00 ⇔ W01 ∨ W10, and so on as before.
- There is at least one wumpus:

  W00 ∨ W01 ∨ W02 ∨… ∨ W33
- There is at most one wumpus:

  For every possible pair of locations, at least one has no wumpus

  ¬W00 ∨ ¬W01

  ¬W00 ∨ ¬W02

  ¬W00 ∨ ¬W03

  …

  ¬W32 ∨ ¬W33
- Some things are Fluents: their truth value changes with time (# steps)
- The percepts are fluents:

  $Stench^3$ = we perceive a stench at time 3

  $Breeze^t$ = we perceive a breeze at time t

  $Bump^t$ = we moved forward at time t-1 but there was a wall in the way
- And some plain facts are fluents:

  $FacingEast^t$, $HaveBullet^t$, $WumpusAlive^t$, and so on

  $Loc23^t$ = were are in room [2,3] at time t
- Our observations tell us some facts:

  $Loc00^t \Rightarrow (Breeze^t \Leftrightarrow B00)$

  $Loc01^t \Rightarrow (Breeze^t \Leftrightarrow B01)$

  …

  $Loc00^t \Rightarrow (Stench^t \Leftrightarrow S00)$

  $Loc01^t \Rightarrow (Stench^t \Leftrightarrow S01$

  …
- The actions taken need to be represented too:

  $Forward^6$ = the action taken at time 6 is to move forward

  $TurnLeft^t$ = the action taken at time t is to turn left

  …
- Effect Axioms tell us what effects the different actions have:

  $Loc00^0 \wedge FacingEast^0 \wedge Forward^0 \Rightarrow Loc01^1 \wedge \neg Loc00^1$

  …

  an enormous number of rules like this
- We also ned to specify when fluents don’t change:

$Forward^7 \Rightarrow (HaveBullet^7 \Leftrightarrow HaveBullet^8)$

$TurnLeft^{22} \Rightarrow (WumpusAlive^{22} \Leftrightarrow WumpusAlive^{23})$

…

- And we initially know:

  $Loc00^0 \wedge HaveBullet^0 \wedge FacingEast^0 \wedge WumpusAlive^0$
- This is completely unmanageable

It is a bit better if we write axioms about fluents instead of actions

- A fluent (becomes) true if

  We do an action that makes it true,
  Or it was already true and we didn't do anything to make it false

  $HaveBullet^8 \Leftrightarrow HaveBullet^7 \wedge \neg Shoot^7$

  $$\begin{aligned} Loc00^4 \Leftrightarrow\ & Loc00^3 \wedge (\neg Forward^3 \vee Bump^4) \\ \vee\ & Loc10^3 \wedge (FacingSouth^3 \wedge Forward^3) \\ \vee\ & Loc01^3 \wedge (FacingWest^3 \wedge Forward^3) \end{aligned}$$

  …
- And we want a way to work out whether a room is safe to move into:

  $OK23^t \Leftrightarrow \neg P23 \wedge \neg(W23 \wedge WumpusAlive^t)$

  …
- That is still a huge knowledge base.