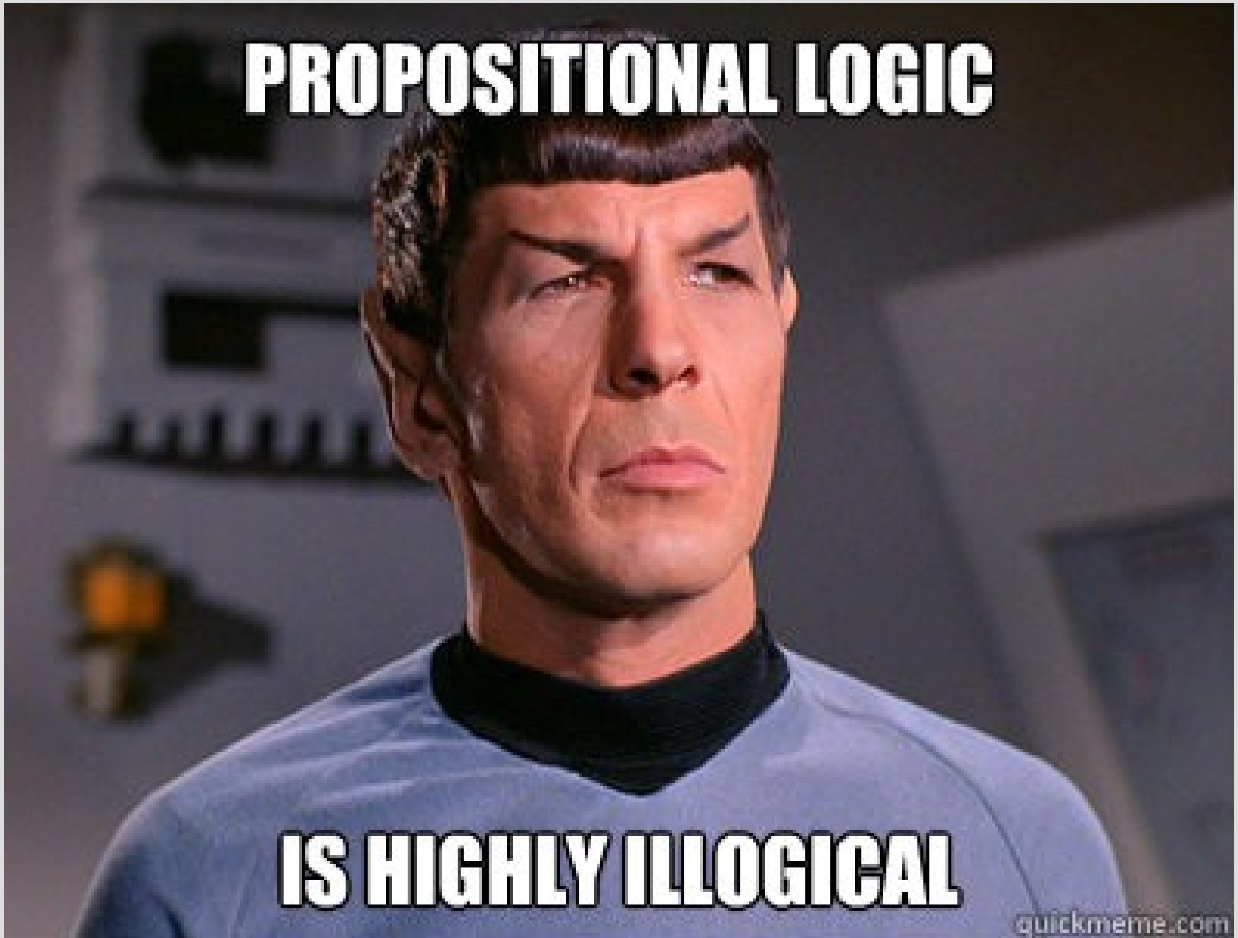
Propositional logic (Ch. 7)



Representing knowledge

So far we have looked at algorithms to find goals via search, where we are provided with all the knowledge and possibly a heuristic

With CSP we saw how to apply inference to rules to find the goal

Now we will expand more on that and fully represent a knowledge base that will store the rules/constraints and what we see/deduce

Logic

Minesweep?

http://minesweeperonline.com/

Write down any "deductions/rules" you find!

Logic

One example of a simple rule:

The 1 in corner marks flag as a mine

Another rule:

The two can mark the two outer mines if flanked by ones

Logic

The goal is to simply tell the computer about the rules of the game

Then based on what it sees as it plays, it will automatically realize these “safe plays”

This type of reasoning is important in partially observable environments as the agent must often reason on new/unseen information

Logic: definitions

A <u>symbol</u> represents a part of the environment (e.g. a minesweep symbol might be if a cell has a mine or not), like math variables

will mostly call them variables

Each single piece of the knowledge base is a <u>sentence</u> involving at least one symbol

A <u>model</u> is an assignment of symbols, a "possible outcome" of the environment (typically we look at assignments that work)

Logic: definitions

Let’s consider a simple sentence:
“I’m happy if it is summer or the weekend”

In logic, this could be:

$$Summer \vee Weekend \Rightarrow Happy$$

... breaking this down into the terminology:

$$\underbrace{\underbrace{Summer}_{\text{symbol}} \vee \underbrace{Weekend}_{\text{symbol}} \Rightarrow \underbrace{Happy}_{\text{symbol}}}_{\text{sentence}}$$

One possible model would be:
Summer=false, Weekend=true, Happy=true

Logic: definitions

In our (current) logic, we allow 5 operations:

$\neg$ = logical negation (i.e. "not T" = F)

$\wedge$ = AND operation

$\vee$ = OR operation (Note: not XOR)

$\Rightarrow$ = "implies" operation

$\Longleftrightarrow$ = "if and only if" operation (iff)

The order of operations (without parenthesis) is top to bottom

Logic: definitions

Here are the truth tables:

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

And equivalent laws:

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of $\vee$

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of $\vee$

$\neg(\neg \alpha) \equiv \alpha$ double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$ contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination

$\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$ de Morgan

$\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$ de Morgan

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

Logic: definitions

We mentioned a symbol is P1,3,2 but a literal is either P1,3,2 or ¬P1,3,2

Two notes:

OR is not XOR (exclusive or), which is not the English "or" (e.g. ordering food)

"implies" only provides information if left hand side is true (e.g. F = cats can fly, B = cats are birds: F implies B is true...)

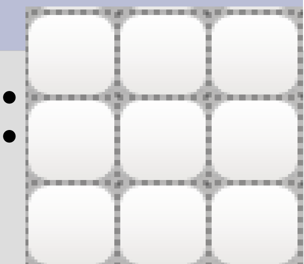
Logic: definitions

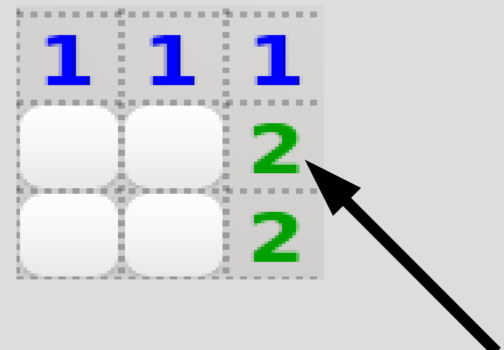
In propositional logic, a symbol is either true or false (as it represents a proposal of a “variable”)

If “m” is a model and is “α” a sentence, m satisfies α means α is true in m (also said as “m models α”)

Let M(α) be all models that satisfy α

Logic: example

For example, consider a 3x3 minesweep: 

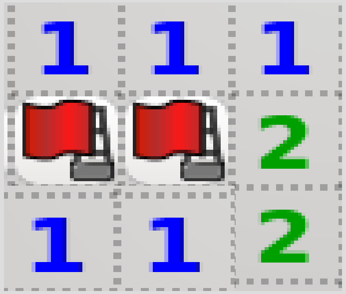
After the first play we have: 

Let us define P2,3,2 as the proposition that row 2, column 3 cell has value 2 (i.e. α=P2,3,2)

After playing the first move, we add to the knowledge base that this proposition is true (this representation has 10^9 states)

Logic: example

Here is one possible assignment:



This does **not** satisfy our proposition P2,3,2 as there are only two mines adjacent to row 2, column 3 cell

So the assignment does not represent our knowledge base (i.e. the picture not in M(α))

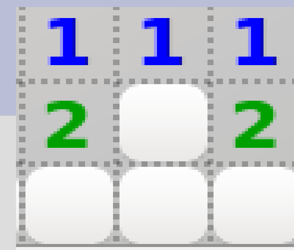
Logic: entailment

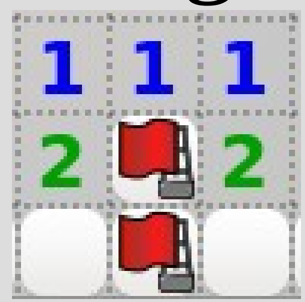
We say β <u>entails</u> α (β ⊨ α) if and only if every model with β true, α is also true
(similar to "implies" where β → α has if β=T, then α=T also)

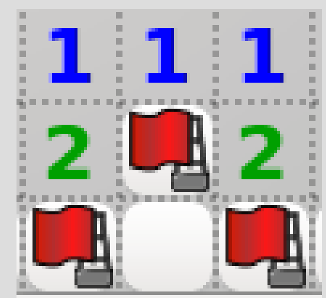
Another definition (mathy):
β ⊨ α if and only if M(β) subset M(α)

This means there are fewer models true with proposition β than α

Logic: entailment

Consider this example: 

There are two valid configurations based on our knowledge base: 

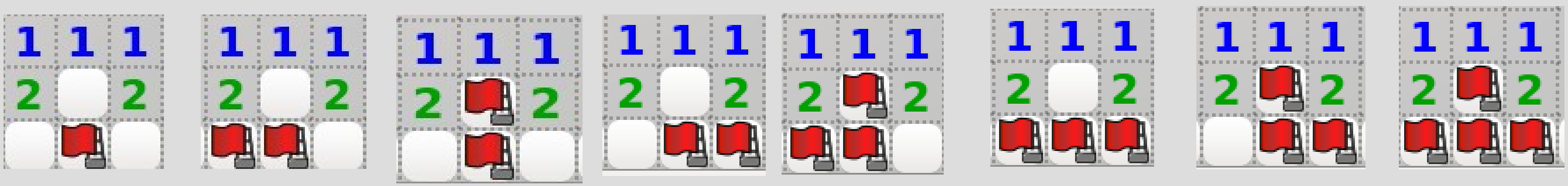


If we let α = {mine at (2,2)}, then this can mean (if we also know the numbered cells):

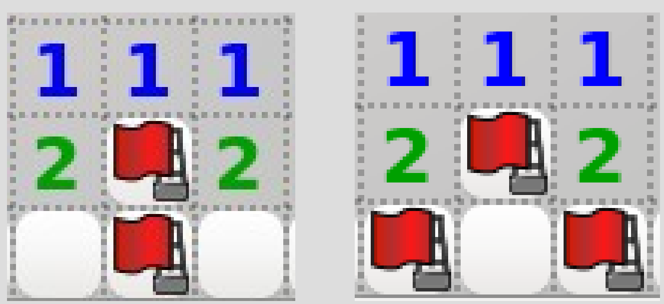
We can see that M(above) subset M(α(below))

Logic: entailment

However, if we let β = mine at (3,2), we get:



M(knowledge base (KB)) is (again):



This is not entailment, as this is not in M(β), thus KB $\not\models$ β (in other words “from the KB, you cannot conclude (3,2) is a mine”)

Logic: entailment

Entailment may seem like implies, but the scope they are working on is different

Implies <u>needs</u> to know if the values of the symbols in order to give T/F answer

$$Summer \lor Weekend \Rightarrow Happy$$

need to know one or both to make a statement about Happy

Entailment shares the "if... then..." thought process, but does not need values to deduce:

$$\left(Summer \lor Weekend \Rightarrow Happy\right) \models \left(Weekend \Rightarrow Happy\right)$$

Logic: model checking

Entailment can generate new sentences for our knowledge base(i.e. can add “mine at (2,2)”)

Model checking is when we write out all the actual models (as I did in the last example) then directly check entailment

This is exponential, and unfortunately this is very typical (although some are much worse exponential than others)

Logic: model checking

Model checking...

1. Preserves truth through inference
2. Is complete, meaning it can derive any sentence that is entailed (and in finite time)

The “complete” is important as some environments have an infinite number of possible sentences

Check model

We can make use model checking to make an inference algorithm, much the same way we modified DFS to do backtracking search

1. Enumerate possibilities on a symbol for all values (T/F) ANDed together... recursive call on next symbol

2. Once all symbols are assigned, check if inconsistent (KB=T, α=F), if not return false (all the way up tree due to recursive call)

Check model

Example: suppose our KB is “P implies Q”
We want to check α = “not P”

Try to use model checking to find if:
KB entails α

(1) Write this as a truth table
(2) Write this as a tree
(3) Which way is better? Why?

Check model

Example: suppose our KB is “P implies Q”

We want to check α = “not P”

Enumerate P: {P = true}, {P = false}

Enumerate Q: {P=T,Q=T}, {P=T,Q=F}, {P=F,Q=T}, {P=F,Q=F}

Consistent?

P	Q	not P	$P \rightarrow Q$
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	T

No! (top row)

“not P” is false when “P implies Q” is true