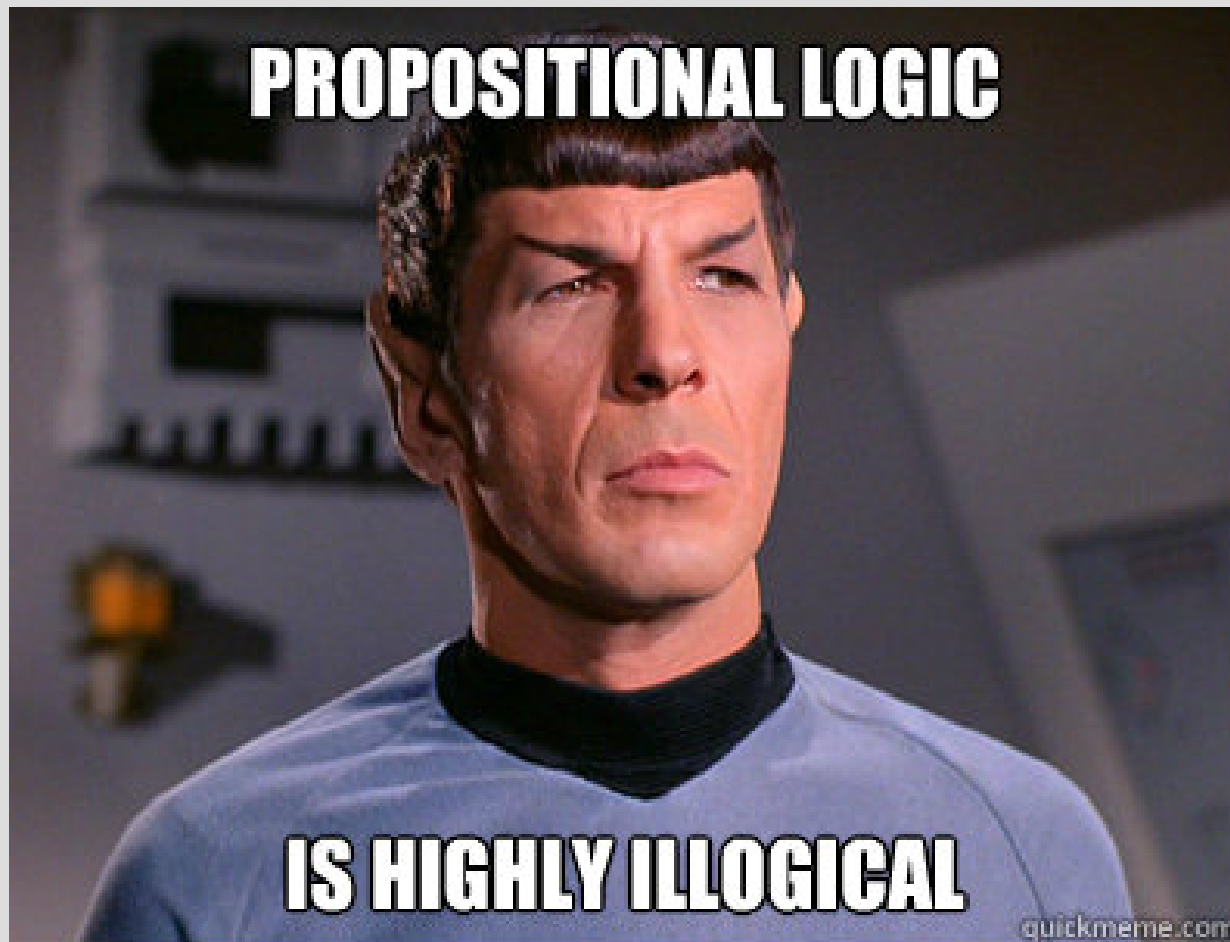


Propositional logic (Ch. 7)



Announcements

Midterm graded

-will probably adjust a bit when put on moodle

HW 3 due Sunday

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

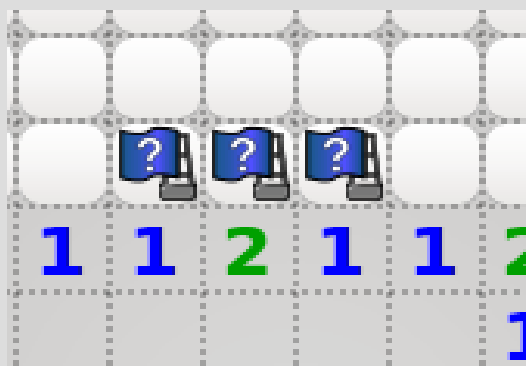
One example of a simple rule:

The 1 in corner marks
flag as a mine

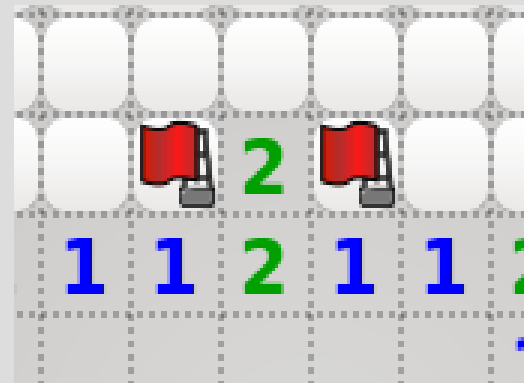


Another rule:

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



safe
→



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

Each single piece of the knowledge base is a sentence involving at least one symbol

A model is a valid assignment of symbols, a “possible outcome” of the environment

Logic: definitions

Side note:

A model is just any assignment of true/false to the variables

A model of a sentence is **only** assignments that make the sentence true

The models of a sentence are all possible true assignments

Logic: definitions

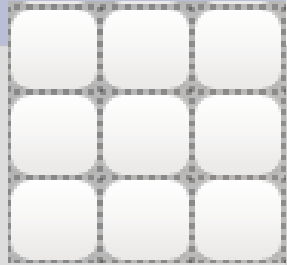
In propositional logic, a symbol is either true or false (as it represents a proposal of a “variable”)

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 ”)

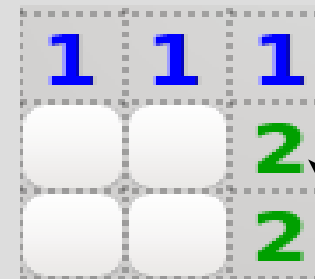
Let $M(\alpha)$ be all models of α

Logic: example

For example, consider a 3x3 minesweep:



After the first play we have:

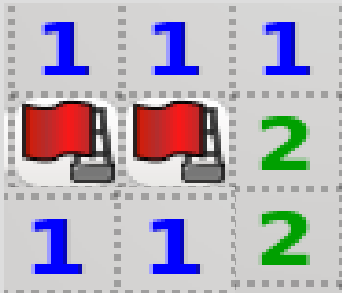


Let us define $P_{2,3,2}$ as the proposition that row 2, column 3 cell has value 2 (i.e. $\alpha = P_{2,3,2}$)

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 $P_{2,3,2}$ 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 $M(\alpha)$)

Logic: entailment

We say α entails β ($\alpha \models \beta$) if and only if every model with α true, β is also true

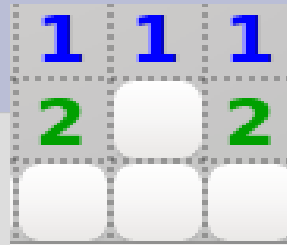
Another definition (mathy):

$\alpha \models \beta$ if and only if $M(\alpha)$ subset $M(\beta)$

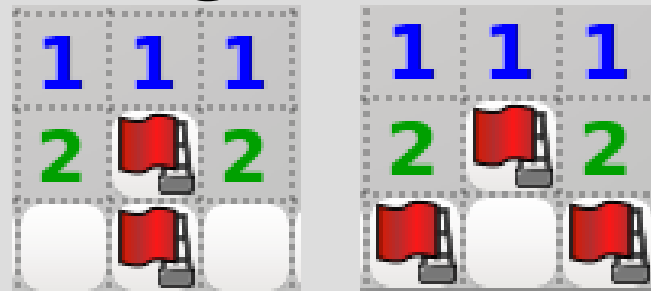
This means there are fewer models true with proposition α than β

Logic: entailment

Consider this example:



There are two valid configurations based on our knowledge base:



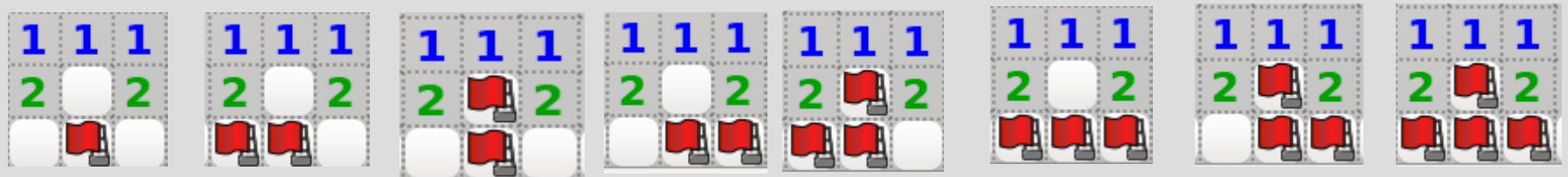
If we let $\alpha = \{\text{mine at } (2,2)\}$, then this can mean (if we also know the numbered cells):



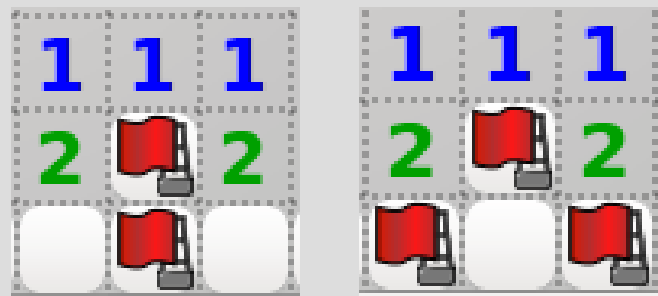
We can see that $M(\text{above}) \subset M(\alpha(\text{below}))$

Logic: entailment

However, if we let $\beta = \text{mine at } (3,2)$, we get:



$M(\text{knowledge base (KB)})$ is (again):



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

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

Logic syntax

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

\iff = “if and only if” operation (iff)

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

Logic syntax

We mentioned a symbol is $P_{1,3,2}$ but a literal is either $P_{1,3,2}$ or $\neg P_{1,3,2}$

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. $F = \text{cats can fly}$, $B = \text{cats are birds}$: F implies B is true...)

Logic syntax

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

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 (repeat)
2. Once all symbols are assigned, check if consistent, 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 $\alpha =$ “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}

	P	Q	not P	P \rightarrow Q
Consistent?	T	T	F	T
	T	F	F	F
	F	T	T	T
	F	F	T	T
	No! (top row)	F	F	T

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