Propositional logic (Ch. 7)
Announcements

Writing 3 due after break: think about project
Logic

Last time we introduced entailment, which is the environment version of “implies”

We can check this by recursively generating the complete truth table and seeing if every time KB is True, $\alpha$ is also True (then $\text{KB} \models \alpha$)

This time we will use multiple rules to reach entailment, much like a proof follows from a premises
Logic: definitions

We say that two sentences are equivalent if they both contain the same models:

\[ \alpha \equiv \beta \iff (\text{if and only if}) \ M(\alpha) = M(\beta) \]

... or alternatively...

\[ \alpha \equiv \beta \iff \alpha \vdash \beta \ \text{AND} \ \beta \vdash \alpha \]

This is the sentence version of \( \iff \) (the boolean “iff” operator)
Logic: definitions

A tautology is a statement that is always True.

For example: “It is raining or it is not raining”
Logically: \( P \lor \neg P \)

A sentence is valid, if it is true in every model (the truth table makes a tautology).

A sentence is satisfiable if it has at least one model that makes it true (one T in truth table).
We can use validity to say: $\alpha \models \beta$ iff the sentence $(\alpha \rightarrow \beta)$ is valid

... or alternatively...

$\alpha \models \beta$ iff the sentence $(\alpha \lor \neg \beta)$ is not satisfiable

This second version is basically contradiction (technically called contrapositive)

You assume the opposite ($\neg \beta$) to reach a conclusion that this is impossible with $\alpha$
Logic: inference

We have these rules for inference:
1. Any logically equivalent statements (e.g. $\alpha \iff \beta$) → know top
   $\alpha \Rightarrow \beta \land \beta \Rightarrow \alpha$ → can deduce bottom

2. Modus ponens: $\alpha \Rightarrow \beta, \alpha$ (top is two sentences) → $\beta$

3. And-elimination: $\alpha \land \beta$ → $\alpha$

We repeatedly apply these rules until we reach the statement we desire.
For example consider the following KB: 
\[ A \land B \land C, \quad B \Rightarrow D \land E \]
We can deduce D by:
1. And elimination on first (KB1) 
   \[ B \]
2. Modus ponens with KB2 and 1. 
   \[ D \land E \]
3. And elimination on 2. 
   \[ D \]
You try it! Deduce D:
\neg A, \neg A \iff \neg (C \lor \neg D)
Logic: inference

You try it! Deduce D:
\[ \neg A, \quad \neg A \iff \neg (C \lor \neg D) \]

1. Equivalence of “iff” in KB2
\[ \neg A \Rightarrow \neg (C \lor \neg D) \land \neg (C \lor \neg D) \Rightarrow \neg A \]

2. And elimination on 1.
\[ \neg A \Rightarrow \neg (C \lor \neg D) \]

3. Modus ponens with KB1 and 2.
\[ \neg (C \lor \neg D) \]

4. De Morgan’s equivalence
\[ \neg C \land D \]

5. And-elim. 4. \[ D \]
Logic: inference

Truth tables grow exponentially in the number of symbols.

Using these logic rules, we can ignore irrelevant sentences and the number of symbols has no direct effect on runtime.

We could do a search on the inference space, where each action is to try and apply an inference rule.
Logic: inference

For example (mindsweep):

Game rules (one of them): (one surround is B)

\[ P1, 3, 1 \Rightarrow (P1, 2, B \land \neg P2, 2, B \land \neg P2, 3, B) \]
\[ \lor (\neg P1, 2, B \land P2, 2, B \land \neg P2, 3, B) \]
\[ \lor (\neg P1, 2, B \land \neg P2, 2, B \land P2, 3, B) \]

KB from current game state:

\[ P1, 1, 1 \land P1, 2, 1 \land P1, 3, 1 \land P2, 1, 2 \land P2, 3, 2 \]

Let's use inference to deduce P2,2,B
1. Use And-elimination on KB state to get: 
   \( P1, 3, 1 \)

2. Use modus ponens with above and rules:
   \[
   (P1, 2, B \land \neg P2, 2, B \land \neg P2, 3, B) \\
   \lor (\neg P1, 2, B \land P2, 2, B \land \neg P2, 3, B) \\
   \lor (\neg P1, 2, B \land \neg P2, 2, B \land P2, 3, B)
   \]

3. ... Uh oh... We are stuck
   These set of rules are not complete (from last time we know we can deduce this)
Resolution is when two complementary literals cancel each other out:

\[ P \lor Q, \quad \neg P \]

\[ Q \]

Generally speaking, you have to merge the two sentences without the complementary ones:

\[ A \lor B \lor \neg G, \quad X \lor Y \lor G \]

\[ A \lor B \lor X \lor Y \]

Unlike our previous inferences, resolution is complete (for any \( \alpha \) & \( \beta \) can tell whether \( \alpha \models \beta \))
Logic: resolution

Assume KB is: $A \Rightarrow \neg B, \quad B$, Entails not A?

First, change to ORs: $\neg A \lor \neg B, \quad B$

There are two ways to use inference:

1. Directly: $\dfrac{\neg A \lor \neg B, B}{\neg A}$

2. Use contradiction (see earlier slide):
   1. $KB \land \neg (\neg A) \equiv (\neg A \lor \neg B) \land (B) \land (A)$
   2. $(\neg A \lor \neg B) \land (B) \land (A) \equiv (\neg A) \land (A)$
   3. $(\neg A) \land (A) \equiv False$
Logic: resolution

The algorithmic way is to use contradiction

1. Cancel out any literals possible and generate new rules
2. Repeat 1 until:
   1. (entails) A “blank” sentence exists (i.e. F)
   2. (not entails) No more possible resolutions

(book fig. 7.13 is better)
Logic: resolution

Back to minesweep!

\[P_1, 3, 1 \Rightarrow (P_1, 2, B \land \neg P_2, 2, B \land \neg P_2, 3, B)\]
\[\lor (\neg P_1, 2, B \land P_2, 2, B \land \neg P_2, 3, B)\]
\[\lor (\neg P_1, 2, B \land \neg P_2, 2, B \land P_2, 3, B)\]

Need to FOIL right hand side (yuck)

\[\neg P_1, 3, 1 \lor ((P_1, 2, B \lor \neg P_1, 2, B) \land (\neg P_2, 2, B \lor \neg P_1, 2, B) \land (\neg P_2, 3, B \lor \neg P_1, 2, B) \land (P_1, 2, B \lor P_2, 2, B) \land (\neg P_2, 2, B \lor P_2, 3, B) \land (\neg P_2, 3, B \lor P_2, 3, B))\]
\[\lor (\neg P_1, 2, B \land \neg P_2, 2, B \land P_2, 3, B)\]

And again (pull out only important term) (RED)

\[\neg P_1, 3, 1 \lor P_1, 2, B \lor P_2, 2, B \lor P_2, 3, B\]
\[\land (P_1, 3, 1) \land (\neg P_1, 2, B) \land (\neg P_2, 3, B)\]
Logic: resolution

Only thing left is P2,2,B (direct method)

We can conclude KB entails P2,2,B

However, to use resolution we need the sentences to be in Conjunctive normal form

This means:
1. Negations (“not”) right next to symbol
2. Format: (sentence of ORs) AND (more ORs)
AND, OR and “not” and fully expressive, so we lose no expressive power with “implies” and “iff” missing.

In the examples, I knew which parts were important to the problem and which were not.

An algorithmic way is to just brute force check all pairs of clauses that have a conflicting term (i.e. $(A \lor \lnot B) \land (B \lor C)$ has $B$ conflicting).
Logic: resolution

Algorithm: (using contrapositive)
1. List clauses in CNF with \((\text{KB AND } \neg \alpha)\)
2. For all \(p\) = pair of clauses
3. For all conflicts in pair
4. Add merged clause without conflict
5. If(merged clause is empty)
6. return “\(\text{KB entails } \alpha\)”
7. Repeat 2 until no new clauses added
8. return “\(\text{KB does not entail } \alpha\)”
Logic: resolution

Run this algorithm for both $\alpha$ and $\beta$:

$KB = (A \implies B) \land (B \iff C)$

$\alpha = \neg A \lor B$

$\beta = \neg A \land B$
Logic: resolution

Run this algorithm for both $\alpha$ and $\beta$:

$KB = (A \Rightarrow B) \land (B \Leftrightarrow C)$

$\alpha = \neg A \lor B$
Entails!

$\beta = \neg A \land B$
Does not entail (get to a point where you cannot reduce anymore)
Logic: wrap-up

There are “local search” hill-climbing versions of solving propositional logic.

These are useful if there are a large number of solutions available for it to find.

Otherwise there are some modifications we can make to our recursive truth-table method to improve performance (similar to how we improved DFS in CSP).
Logic: wrap-up

One major factor of propositional logic is how many symbols to sentences/clauses there are.

If there are too few sentences, it is easy to find the answer... too many and trivially fails.
Logic: wrap-up

Typically, to actually solve a full problem, (and not just one part) we need many more sentences to impose “obvious” rules, such as:

1. The state can only be in one place at one time (i.e. in mindsweep a cell cannot be both a “2” and a “4”)

2. Full search has a time component, and we must ensure by default no symbols are allowed to “change” between time steps
Logic: wrap-up

So far we have talked about pure inference to solve problems, but we can mix in search.

For example, we use inference to decide which cell to play next in minesweep, but we use another algorithm to actually move the mouse and click (abstraction).

Searches are much faster than logical thinking, so we should use them for straightforward parts.
Logic: wrap-up

As propositional are all only True or False proposals about the environment, we typically need all combinations of variables for all time.

This rapidly grows the problem exponentially, and makes larger problems not feasible.

This would not be the case if we had a more expressive form of logic (which we will talk about after spring break).