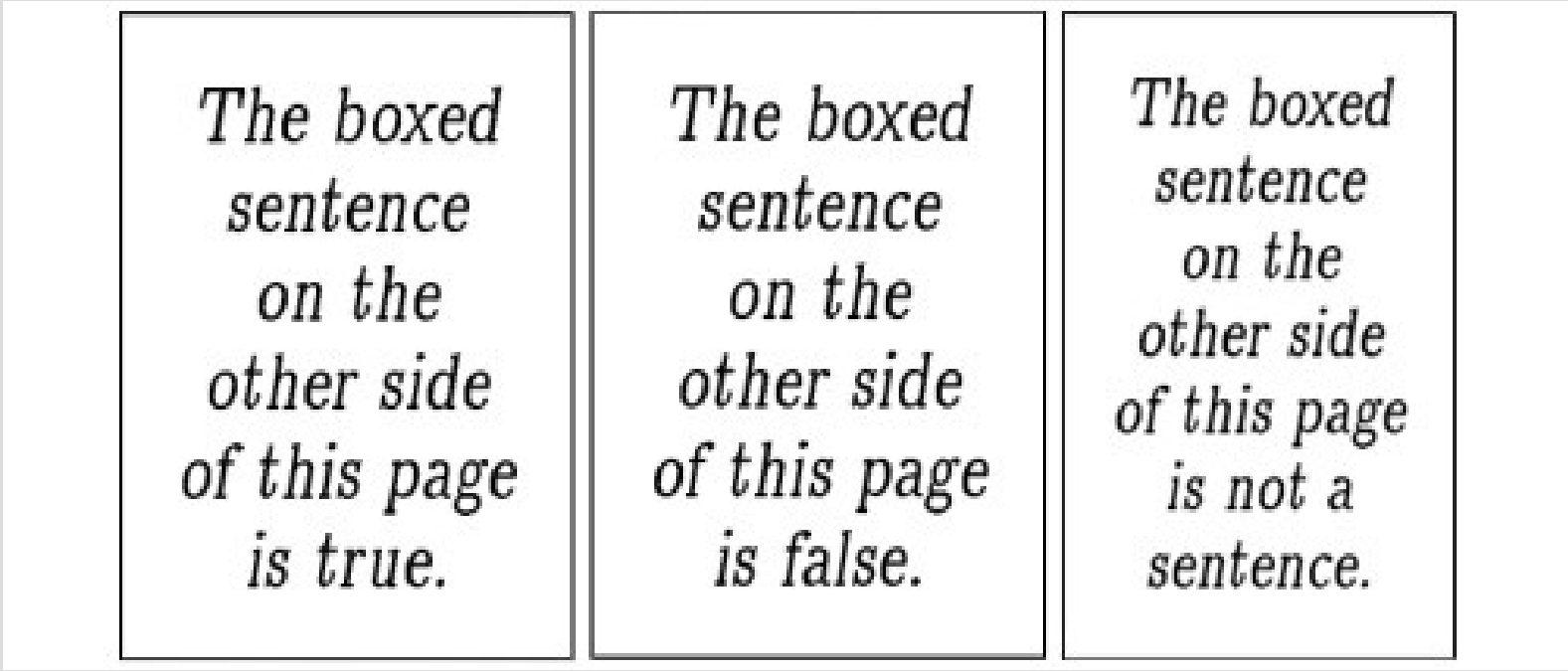
Propositional logic (Ch. 7)



Logic: definitions

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

$\alpha \equiv \beta$ iff (if and only if) $M(\alpha) = M(\beta)$

... or alternatively...

$\alpha \equiv \beta$ iff $\alpha \models \beta$ AND $\beta \models \alpha$

This is the sentence version of $\Leftrightarrow$ (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: $P \vee \neg P$

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)

Logic: inference

We can use validity to say:

$\alpha \models \beta$ iff the sentence $(\alpha \rightarrow \beta)$ is valid

... or alternatively...

$\alpha \models \beta$ iff $(\alpha \text{ AND } \neg \beta)$ 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 α

Logic: inference

We have these rules for inference:

1. Any logically equivalent statements

   (e.g. $\dfrac{\alpha \iff \beta}{\alpha \Rightarrow \beta \wedge \beta \Rightarrow \alpha}$) ← know top ← can deduce bottom

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

3. And-elimination: $\dfrac{\alpha \wedge \beta}{\alpha}$

We repeatedly apply these rules until we reach the statement we desire

Logic: inference

For example consider the following KB:

$A \wedge B \wedge C, \qquad B \Rightarrow D \wedge E$

We can deduce D by:

1. And elimination on first (KB1)

   B

2. Modus ponens with KB2 and 1.

   $D \wedge E$

3. And elimination on 2.

   D

Logic: inference

You try it! Deduce D:

$$\neg A, \qquad \neg A \iff \neg(C \vee \neg D)$$

Logic: inference

You try it! Deduce D:

$\neg A, \quad \neg A \iff \neg(C \vee \neg D)$

1. Equivalence of “iff” in KB2

$\neg A \Rightarrow \neg(C \vee \neg D) \wedge \neg(C \vee \neg D) \Rightarrow \neg A$

2. And elimination on 1.

$\neg A \Rightarrow \neg(C \vee \neg D)$

3. Modus ponens with KB1 and 2.

$\neg(C \vee \neg D)$

4. De Morgan's equivalence

$\neg C \wedge D$ → 5. And-elim. 4. D

Logic: inference

You try it! Deduce C:

$A \vee B, \qquad A \Rightarrow C, \qquad B \Rightarrow C$

Logic: inference

You try it! Deduce C:

$A \vee B, \qquad A \Rightarrow C, \qquad B \Rightarrow C$

Start with:

$A \vee B$

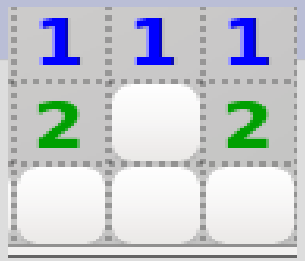
... but we actually get stuck here

We know we can apply either $A \Rightarrow C$ or $B \Rightarrow C$, but not which one

This is a limitation of our rules so far, they are ***not complete***!

Logic: inference

For example (mindsweep):



Game rules (one of them): (adjacent is bomb)

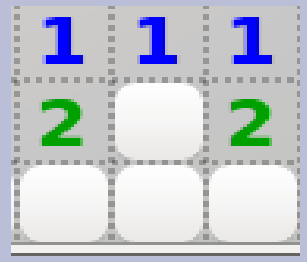
$$P1,3,1 \Rightarrow (P1,2,B \wedge \neg P2,2,B \wedge \neg P2,3,B) \vee (\neg P1,2,B \wedge P2,2,B \wedge \neg P2,3,B) \vee (\neg P1,2,B \wedge \neg P2,2,B \wedge P2,3,B)$$

KB from current game state:

$$P1,1,1 \wedge P1,2,1 \wedge P1,3,1 \wedge P2,1,2 \wedge P2,3,2$$

Let's use inference to deduce P2,2,B

Logic: inference



1. Use And-elimination on KB state to get:
   $P1,3,1$
2. Use modus ponens with above and rules:
   $(P1,2,B \wedge \neg P2,2,B \wedge \neg P2,3,B)$
   $\vee(\neg P1,2,B \wedge P2,2,B \wedge \neg P2,3,B)$
   $\vee(\neg P1,2,B \wedge \neg P2,2,B \wedge P2,3,B)$

3. ... Uh oh... We are stuck

These set of rules are not complete (from last time we know we can deduce this)

Logic: inference

You can represent all propositional logic with truth tables and brute force solve

This grows exponentially in the number of symbols (linearly by number of sentences)

Using these logic rules, we can can ignore irrelevant sentences (the runtime is bounded by symbol connectivity, not number of sentences)

Logic: resolution

Resolution is when two complementary literals cancel each other out:

$$\frac{P \vee Q, \qquad \neg P}{Q}$$

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

$$\frac{A \vee B \vee \neg G, \qquad X \vee Y \vee G}{A \vee B \vee X \vee Y}$$

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

Logic: resolution

Assume KB: $A \Rightarrow \neg B, \quad B$, Entails (not A)?
First, change to ORs: $\neg A \vee \neg B, \quad B$

There are two ways to use inference:

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

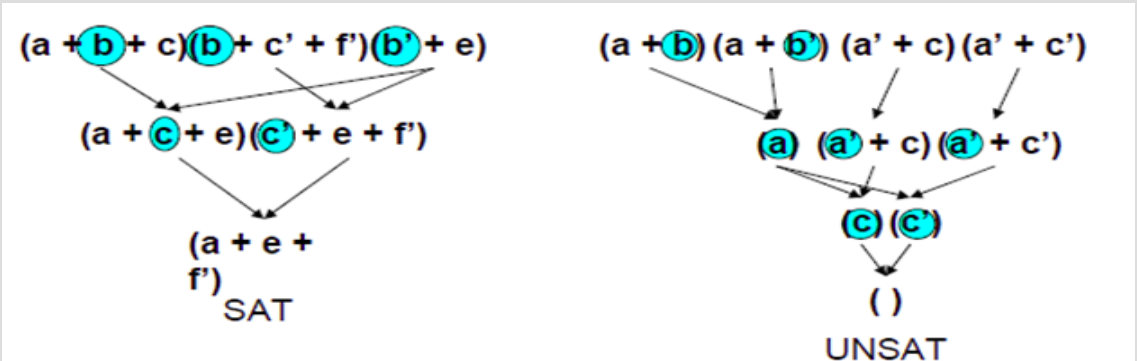
2. Use contradiction (see earlier slide):
   1. $KB \wedge \neg(\neg A) \equiv (\neg A \vee \neg B) \wedge (B) \wedge (A)$
   2. $(\neg A \vee \neg B) \wedge (B) \wedge (A) \equiv (\neg A) \wedge (A)$
   3. $(\neg A) \wedge (A) \equiv False$

Logic: resolution

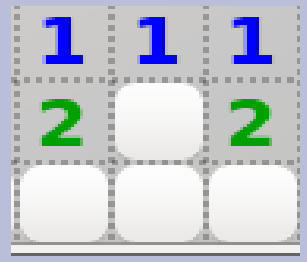
The algorithmic way is to use contradiction

1. Cancel out any literals possible and generate new rules
2. Repeat 1 until:
    1. (entails) A “blank” sentence derived involving the contradiction (or child)
    2. (not entails) No more possible resolutions

(book fig. 7.13 is better)



Logic: resolution



Back to minesweep!

$$P1,3,1 \Rightarrow (P1,2,B \wedge \neg P2,2,B \wedge \neg P2,3,B) \vee (\neg P1,2,B \wedge P2,2,B \wedge \neg P2,3,B) \vee (\neg P1,2,B \wedge \neg P2,2,B \wedge P2,3,B)$$

Need to FOIL right hand side (yuck)

$$\neg P1,3,1 \vee ((P1,2,B \vee \neg P1,2,B) \wedge (\neg P2,2,B \vee \neg P1,2,B) \wedge (\neg P2,3,B \vee \neg P1,2,B) \wedge (P1,2,B \vee P2,2,B) \wedge (\neg P2,2,B \vee P2,2,B) \wedge (\neg P2,3,B \vee \neg P2,2,B) \wedge (P1,2,B \vee \neg P2,3,B) \wedge (\neg P2,2,B \vee P2,3,B) \wedge (\neg P2,3,B \vee P2,3,B)) \vee (\neg P1,2,B \wedge \neg P2,2,B \wedge P2,3,B)$$

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

$$(\neg P1,3,1 \vee P1,2,B \vee P2,2,B \vee P2,3,B) \wedge (P1,3,1) \wedge (\neg P1,2,B) \wedge (\neg P2,3,B)$$

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: (For example: $(A \vee B \vee C) \wedge (\neg B \vee \neg D)$)
1. Negations (“not”) right next to symbol
2. Format: (sentence of ORs) AND (more ORs)

Logic: resolution

AND, OR and “not” are 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. $(A \vee \neg B) \wedge (B \vee C)$ has B conflicting)

Logic: resolution

Algorithm: (using contrapositive)

1. List clauses in CNF with (KB AND ¬ α)
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 “KB entails α”
7. Repeat 2 until no new clauses added
8. return “KB does not entail α”

Logic: resolution

Run this algorithm for both α and β:

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

α = $\neg A \vee B$, KB entails α?

β = $\neg A \wedge B$, KB entails β?

Logic: resolution

Run this algorithm for both α and β:

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

α = $\neg A \vee B$, KB entails α?

Entails!

β = $\neg A \wedge B$, KB entails β?

Does not entail

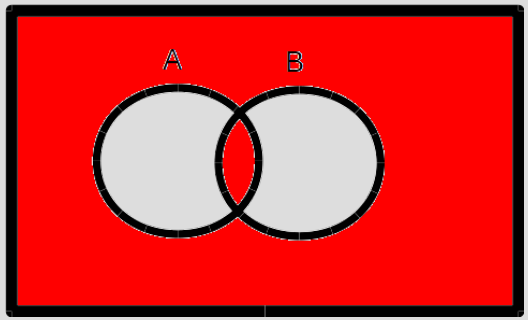
Logic: resolution

Consider these sentences: $(\neg A \vee B) \wedge (A \vee \neg B)$

They each have complementary literals for resolution (i.e. (not A) in first and A in second)

However, if you “resolve” these A’s you get: $B \vee \neg B$, which is a tautology (not helpful)

Think of the Venn diagram:
(cannot reduce to single var)



Logic: wrap-up

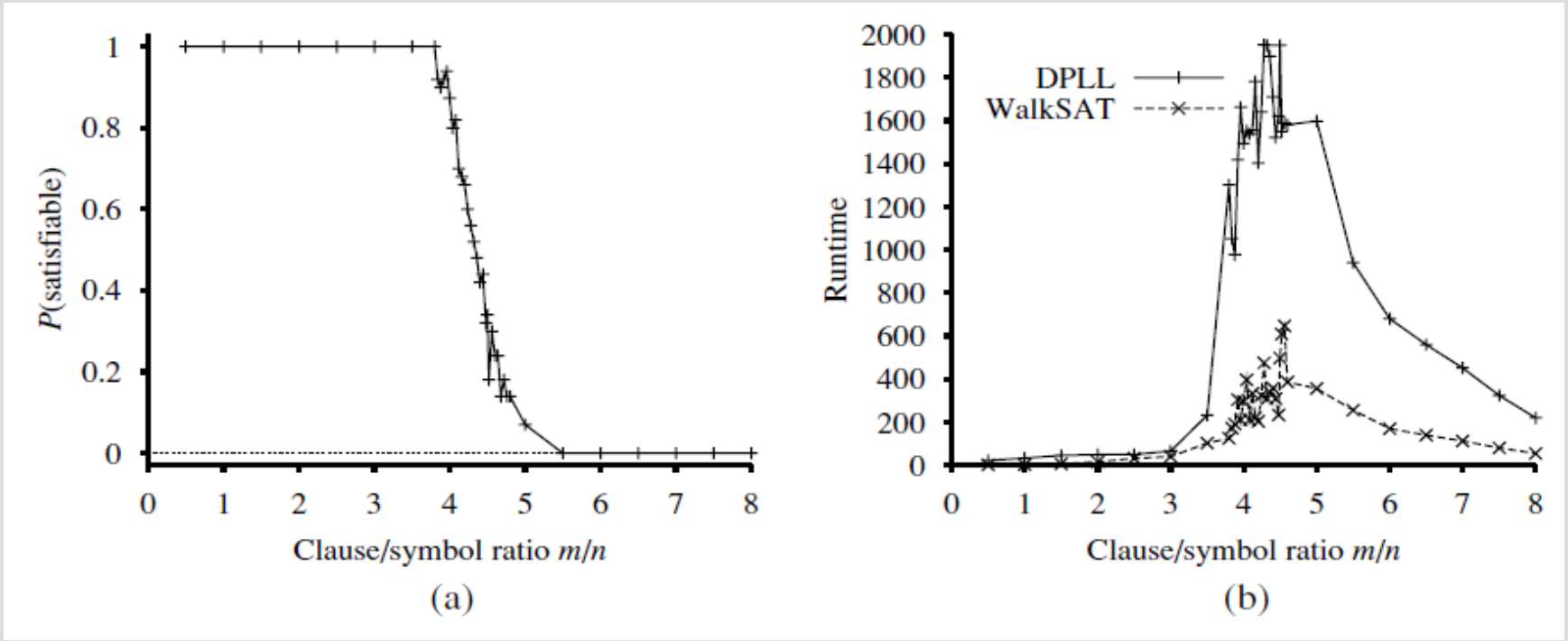
There are “local search” hill-climbing versions of solving propositional logic

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

Searches are much faster than logical thinking, so we should use for straightforward parts

For propositional logic, the default branching factor is 2 (true or false) but a single inference can reduce this factor to 1 for multiple depths

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 next)