# CS 331: Artificial Intelligence
# Propositional Logic 2

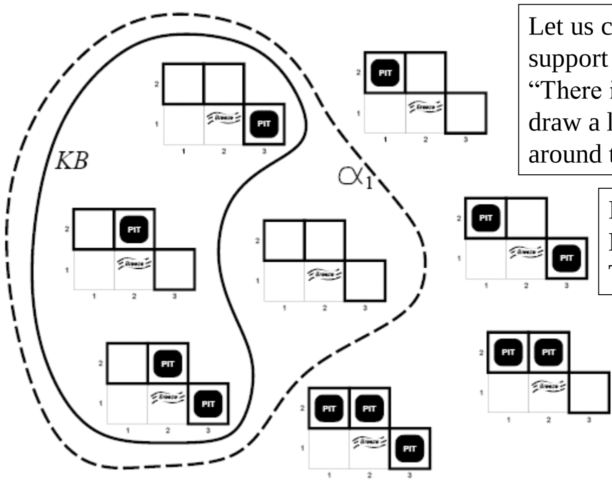
1

## Review of Last Time

- |= means "logically follows"
- $|\!-_i$ means "can be derived from"
- If your inference algorithm derives only things that follow logically from the KB, the inference is sound
- If everything that follows logically from the KB can be derived using your inference algorithm, the inference is complete

2

# Entailment Applied to the Wumpus World




Let us consider the models that support the conclusion $\alpha_1$ = "There is no pit in [1,2]." We draw a line marked with $\alpha_1$ around these models

In every model in which KB is true, $\alpha_1$ is also true. Therefore KB $\models \alpha_1$



# Inference: Model Checking

- Suppose we want to know if KB $\models \neg P_{1,2}$?
- In the 3 models in which KB is true, $\neg P_{1,2}$ is also true

| $B_{1,1}$ | $B_{2,1}$ | $P_{1,1}$ | $P_{1,2}$ | $P_{2,1}$ | $P_{2,2}$ | $P_{3,1}$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | KB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| false | false | false | false | false | false | false | true | true | true | true | false | false |
| false | false | false | false | false | false | true | true | true | false | true | false | false |
| : | : | : | : | : | : | : | : | : | : | : | : | : |
| false | true | false | false | false | false | false | true | true | false | true | true | false |
| false | true | false | false | false | false | true | true | true | true | true | true | true |
| false | true | false | false | false | true | false | true | true | true | true | true | true |
| false | true | false | false | false | true | true | true | true | true | true | true | true |
| false | true | false | false | true | false | false | true | false | false | true | true | false |
| : | : | : | : | : | : | : | : | : | : | : | : | : |
| true | true | true | true | true | true | true | false | true | true | false | true | false |

## Complexity

- If the KB and $\alpha$ contain n symbols in total, what is the time complexity of the truth table enumeration algorithm?

- Space complexity is O(n) because the actual algorithm uses DFS

5

## The really depressing news

- Every known inference algorithm for propositional logic has a **worst-case** complexity that is exponential in the size of the input

You can't handle the truth!

- But some algorithms are more efficient **in practice**

6

# Logical equivalence

- Intuitively: two sentences $\alpha$ and $\beta$ are logically equivalent (i.e. $\alpha \equiv \beta$ ) if they are true in the same set of models
- Formally: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$
- Can prove this with truth tables

7

# Standard Logic Equivalences

$$
\begin{array}{rcll}
(\alpha \wedge \beta) & \equiv & (\beta \wedge \alpha) & \text{commutativity of } \wedge \\
(\alpha \vee \beta) & \equiv & (\beta \vee \alpha) & \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) & \equiv & (\alpha \wedge (\beta \wedge \gamma)) & \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) & \equiv & (\alpha \vee (\beta \vee \gamma)) & \text{associativity of } \vee \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \vee \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) & \equiv & (\neg\alpha \vee \neg\beta) & \text{de Morgan} \\
\neg(\alpha \vee \beta) & \equiv & (\neg\alpha \wedge \neg\beta) & \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) & \equiv & ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) & \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) & \equiv & ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) & \text{distributivity of } \vee \text{ over } \wedge
\end{array}
$$

In the above, $\alpha$, $\beta$, and $\gamma$ are arbitrary sentences of propositional logic

8

## Validity

- A sentence is valid if it is true in all models
- E.g. $P \vee \neg P$ is valid
- Valid sentences = Tautologies
- Tautologies are vacuous

**Deduction theorem**

For any sentences $\alpha$ and $\beta$, $\alpha \models \beta$ iff the sentence $(\alpha \Rightarrow \beta)$ is valid

9

## Satisfiability

- A sentence is satisfiable if it is true in some model.
- A sentence is unsatisfiable if it is true in no models
- Determining the satisfiability of sentences in propositional logic was the first problem proved to be NP-complete
- Satisfiability is connected to validity:

  $\alpha$ is valid iff $\neg\alpha$ is unsatisfiable
- Satisfiability is connected to entailment:

  $\alpha \models \beta$ iff the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable (proof by contradiction)

10

## Exercise

- Is the following sentence valid?

$$(A \Rightarrow B) \vee (\neg A \Rightarrow \neg B)$$



## Proof methods

How do we prove that $\alpha$ can be entailed from the KB?

1. Model checking e.g. check that $\alpha$ is true in all models in which KB is true
2. Inference rules

# Inference Rules

1. Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

2. And-Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

These are both sound inference rules. You don't need to enumerate models now

13

# Other Inference Rules

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

All of the logical equivalences can be turned into inference rules e.g.

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

# Example

Given the following KB, can we prove ¬R?

KB:

$P \Rightarrow \neg(Q \vee R)$
$P$

Proof:

$\neg(Q \vee R)$ by Modus Ponens

$\neg Q \wedge \neg R$ by De Morgan's Law

$\neg R$ by And-Elimination



# Proofs

- A sequence of applications of inference rules is called a proof
- Instead of enumerating models, we can search for proofs
- Proofs ignore irrelevant propositions
- 2 methods:
  - Go forward from initial KB, applying inference rules to get to the goal sentence
  - Go backward from goal sentence to get to the KB

## In-class Exercise

| | |
|---|---|
| If it is October, there will not be a football game at OSU | |
| If it is October and it is Saturday, I will be in Corvallis | |
| If it doesn't rain or if there is a football game, I will ride my bike to OSU | |
| Today is Saturday and it is October | |
| If I am in Corvallis, it will not rain | |

Can you prove that I will ride my bike to OSU?



## Monotonicity

- Proofs only work because of monotonicity
- Monotonicity: the set of entailed sentences can only increase as information is added to the knowledge base
- For any sentences $\alpha$ and $\beta$,

  if KB $\models \alpha$ then KB $\wedge\ \beta \models \alpha$

# Resolution

- An inference rule that is sound and complete
- Forms the basis for a family of complete inference procedures
- Here, complete means refutation completeness: resolution can refute or confirm the truth of any sentence with respect to the KB

19

# Resolution

- Here's how resolution works ($\neg l_2$ and $l_2$ are called complementary literals):

$$\frac{l_1 \vee l_2, \quad \neg l_2 \vee l_3}{l_1 \vee l_3}$$

- Note that you need to remove multiple copies of literals (called factoring) i.e.

$$\frac{l_1 \vee l_2, \quad \neg l_2 \vee l_1}{l_1}$$

- If $l_i$ and $m_j$ are complementary literals, the full resolution rule looks like:

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

## In-class Exercise

KB

| |
|---|
| Person $\Rightarrow$ Mortal |
| Socrates $\Rightarrow$ Person |

Can we show that :

KB $\models$ (Socrates $\Rightarrow$ Mortal)?



## Conjunctive Normal Form

- Resolution only applies to sentences of the form $l_1 \vee l_2 \vee \ldots \vee l_k$
- This is called a disjunction of literals
- It turns out that every sentence of propositional logic is logically equivalent to a conjunction of disjunction of literals
- Called Conjunctive Normal Form or CNF
  e.g. $(l_1 \vee l_2 \vee l_3 \vee l_4) \wedge (l_5 \vee l_6 \vee l_7 \vee l_8) \wedge \ldots$
- k-CNF sentences have exactly k literals per clause
  e.g. A 3-CNF sentence would be $(l_1 \vee l_2 \vee l_3) \wedge (l_4 \vee l_5 \vee l_6) \wedge (l_7 \vee l_8 \vee l_9)$

## Recipe for Converting to CNF

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$
3. Move $\neg$ inwards using:
   - $\neg(\neg\alpha) \equiv \alpha$ (double-negation elimination)
   - $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$ (De Morgan's Law)
   - $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$ (De Morgan's Law)
4. Apply distributive law $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

23

## Exercise

- Convert the following sentence to CNF.

$$(B \vee C) \Rightarrow D$$

24

# A resolution algorithm

To prove KB |= $\alpha$, we show that (KB $\wedge \neg\alpha$) is unsatisfiable
(Remember that $\alpha$ |= $\beta$ iff the sentence ($\alpha \wedge \neg\beta$) is unsatisfiable)

The algorithm:
1. Convert (KB $\wedge \neg\alpha$) to CNF
2. Apply resolution rule to resulting clauses. Each pair with complementary literals is resolved to produce a new clause which is added to the KB
3. Keep going until
   – There are no new clauses that can be added ( meaning KB $\not\models \alpha$)
   – Two clauses resolve to yield the empty clause ( meaning KB |= $\alpha$ )

The empty clause is equivalent to false because a disjunction is true only if one of its disjuncts is true

25

# Exercise

- Suppose the KB contains the following sentences in CNF.

  1. $\neg C \vee E$
  2. $\neg P \vee E$
  3. $\neg E \vee \neg R$
  4. $\neg A \vee \neg P \vee E$
- Does $KB \models \neg R$?

26

# Resolution Pseudocode

**function** PL-RESOLUTION($KB, \alpha$) **returns** *true* or *false*

    *clauses* ← the set of clauses in the CNF representation of $KB \land \neg\alpha$
    *new* ← { }
    **loop do**
        **for each** $C_i$, $C_j$ **in** *clauses* **do**
            *resolvents* ← PL-RESOLVE($C_i, C_j$)
            **if** *resolvents* contains the empty clause **then return** *true*
            *new* ← *new* ∪ *resolvents*
        **if** *new* ⊆ *clauses* **then return** *false*
        *clauses* ← *clauses* ∪ *new*



# Things you should know

- Understand the syntax and semantics of propositional logic
- Know how to do a proof in propositional logic using inference rules
- Know how to convert arbitrary sentences to CNF
- Know how resolution works