

# Artificial Intelligence

## Propositional Logic

Instructor: Vincent Conitzer

# Logic and AI

- Would like our AI to have **knowledge about the world**, and **logically draw conclusions** from it
- Search algorithms generate successors and evaluate them, but do not “understand” much about the setting
- Example question: is it possible for a chess player to have 8 pawns and 2 queens?
  - Search algorithm could search through tons of states to see if this ever happens, but...

# A story

- You roommate comes home; he/she is completely wet
- You know the following things:
  - Your roommate is wet
  - If your roommate is wet, it is because of rain, sprinklers, or both
  - If your roommate is wet because of sprinklers, the sprinklers must be on
  - If your roommate is wet because of rain, your roommate must not be carrying the umbrella
  - The umbrella is not in the umbrella holder
  - If the umbrella is not in the umbrella holder, either you must be carrying the umbrella, or your roommate must be carrying the umbrella
  - You are not carrying the umbrella
- Can you conclude that the sprinklers are on?
- Can AI conclude that the sprinklers are on?

# Knowledge base for the story

- RoommateWet
- RoommateWet => (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)
- RoommateWetBecauseOfSprinklers => SprinklersOn
- RoommateWetBecauseOfRain => NOT(RoommateCarryingUmbrella)
- UmbrellaGone
- UmbrellaGone => (YouCarryingUmbrella OR RoommateCarryingUmbrella)
- NOT(YouCarryingUmbrella)

# Syntax

- What do well-formed sentences in the knowledge base look like?
- A **BNF grammar**:
- $Symbol \rightarrow P, Q, R, \dots, RoommateWet, \dots$
- $Sentence \rightarrow True \mid False \mid Symbol \mid NOT(Sentence) \mid (Sentence \text{ AND } Sentence) \mid (Sentence \text{ OR } Sentence) \mid (Sentence \Rightarrow Sentence)$
- We will drop parentheses sometimes, but formally they really should always be there

# Semantics

- A **model** specifies which of the proposition symbols are true and which are false
- Given a model, I should be able to tell you whether a sentence is true or false
- **Truth table** defines semantics of operators:

a	b	NOT(a)	a AND b	a OR b	a => b
false	false	true	false	false	true
false	true	true	false	true	true
true	false	false	false	true	false
true	true	false	true	true	true

- Given a model, can compute truth of sentence recursively with these

# Caveats

- $\text{TwoIsAnEvenNumber} \text{ OR } \text{ThreeIsAnOddNumber}$   
is true (not exclusive OR)
- $\text{TwoIsAnOddNumber} \Rightarrow \text{ThreeIsAnEvenNumber}$   
is true (if the left side is false it's always true)

*All of this is assuming those symbols are assigned their natural values...*

# Tautologies

- A sentence is a **tautology** if it is true for any setting of its propositional symbols

P	Q	P OR Q	NOT(P) AND NOT(Q)	(P OR Q) OR (NOT(P) AND NOT(Q))
false	false	false	true	true
false	true	true	false	true
true	false	true	false	true
true	true	true	false	true

- $(P \text{ OR } Q) \text{ OR } (\text{NOT}(P) \text{ AND } \text{NOT}(Q))$  is a tautology

# Is this a tautology?

- $(P \Rightarrow Q) \text{ OR } (Q \Rightarrow P)$

# Logical equivalences

- Two sentences are **logically equivalent** if they have the same truth value for every setting of their propositional variables

P	Q	P OR Q	NOT(NOT(P) AND NOT(Q))
false	false	false	false
false	true	true	true
true	false	true	true
true	true	true	true

- P OR Q and NOT(NOT(P) AND NOT(Q)) are logically equivalent
- Tautology = logically equivalent to True

# Famous logical equivalences

- $(a \text{ OR } b) \equiv (b \text{ OR } a)$  *commutatitvity*
- $(a \text{ AND } b) \equiv (b \text{ AND } a)$  *commutatitvity*
- $((a \text{ AND } b) \text{ AND } c) \equiv (a \text{ AND } (b \text{ AND } c))$  *associativity*
- $((a \text{ OR } b) \text{ OR } c) \equiv (a \text{ OR } (b \text{ OR } c))$  *associativity*
- $\text{NOT}(\text{NOT}(a)) \equiv a$  *double-negation elimination*
- $(a \Rightarrow b) \equiv (\text{NOT}(b) \Rightarrow \text{NOT}(a))$  *contraposition*
- $(a \Rightarrow b) \equiv (\text{NOT}(a) \text{ OR } b)$  *implication elimination*
- $\text{NOT}(a \text{ AND } b) \equiv (\text{NOT}(a) \text{ OR } \text{NOT}(b))$  *De Morgan*
- $\text{NOT}(a \text{ OR } b) \equiv (\text{NOT}(a) \text{ AND } \text{NOT}(b))$  *De Morgan*
- $(a \text{ AND } (b \text{ OR } c)) \equiv ((a \text{ AND } b) \text{ OR } (a \text{ AND } c))$  *distributivity*
- $(a \text{ OR } (b \text{ AND } c)) \equiv ((a \text{ OR } b) \text{ AND } (a \text{ OR } c))$  *distributivity*

# Inference

- We have a knowledge base of things that we know are true
  - RoommateWetBecauseOfSprinklers
  - RoommateWetBecauseOfSprinklers => SprinklersOn
- Can we conclude that SprinklersOn?
- We say SprinklersOn is **entailed** by the knowledge base if, for every setting of the propositional variables for which the knowledge base is true, SprinklersOn is also true

RWBOS	SprinklersOn	Knowledge base
false	false	false
false	true	false
true	false	false
true	true	true

- SprinklersOn is entailed!

# Simple algorithm for inference

- Want to find out if sentence  $a$  is entailed by knowledge base...
- *For every possible setting of the propositional variables,*
  - *If knowledge base is true and  $a$  is false, return false*
- *Return true*
- Not very efficient:  $2^{\text{\#propositional variables}}$  settings

# Inconsistent knowledge bases

- Suppose we were careless in how we specified our knowledge base:
- $\text{PetOfRoommateIsABird} \Rightarrow \text{PetOfRoommateCanFly}$
- $\text{PetOfRoommateIsAPenguin} \Rightarrow \text{PetOfRoommateIsABird}$
- $\text{PetOfRoommateIsAPenguin} \Rightarrow \text{NOT}(\text{PetOfRoommateCanFly})$
- $\text{PetOfRoommateIsAPenguin}$
- **No** setting of the propositional variables makes all of these true
- Therefore, technically, this knowledge base implies **anything**
- $\text{TheMoonIsMadeOfCheese}$

# Reasoning patterns

- Obtain new sentences directly from some other sentences in knowledge base according to **reasoning patterns**
- If we have sentences  $a$  and  $a \Rightarrow b$ , we can correctly conclude the new sentence  $b$ 
  - This is called **modus ponens**
- If we have  $a \text{ AND } b$ , we can correctly conclude  $a$
- All of the logical equivalences from before also give reasoning patterns

# Formal proof that the sprinklers are on

- 1) RoommateWet
- 2) RoommateWet  $\Rightarrow$  (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)
- 3) RoommateWetBecauseOfSprinklers  $\Rightarrow$  SprinklersOn
- 4) RoommateWetBecauseOfRain  $\Rightarrow$  NOT(RoommateCarryingUmbrella)
- 5) UmbrellaGone
- 6) UmbrellaGone  $\Rightarrow$  (YouCarryingUmbrella OR RoommateCarryingUmbrella)
- 7) NOT(YouCarryingUmbrella)
- 8) YouCarryingUmbrella OR RoommateCarryingUmbrella (*modus ponens on 5 and 6*)
- 9) NOT(YouCarryingUmbrella)  $\Rightarrow$  RoommateCarryingUmbrella (*equivalent to 8*)
- 10) RoommateCarryingUmbrella (*modus ponens on 7 and 9*)
- 11) NOT(NOT(RoommateCarryingUmbrella)) (*equivalent to 10*)
- 12) NOT(NOT(RoommateCarryingUmbrella))  $\Rightarrow$  NOT(RoommateWetBecauseOfRain) (*equivalent to 4 by contraposition*)
- 13) NOT(RoommateWetBecauseOfRain) (*modus ponens on 11 and 12*)
- 14) RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers (*modus ponens on 1 and 2*)
- 15) NOT(RoommateWetBecauseOfRain)  $\Rightarrow$  RoommateWetBecauseOfSprinklers (*equivalent to 14*)
- 16) RoommateWetBecauseOfSprinklers (*modus ponens on 13 and 15*)
- 17) SprinklersOn (*modus ponens on 16 and 3*)

# Reasoning about penguins

- 1)  $\text{PetOfRoommateIsABird} \Rightarrow \text{PetOfRoommateCanFly}$
- 2)  $\text{PetOfRoommateIsAPenguin} \Rightarrow \text{PetOfRoommateIsABird}$
- 3)  $\text{PetOfRoommateIsAPenguin} \Rightarrow \text{NOT}(\text{PetOfRoommateCanFly})$
- 4)  $\text{PetOfRoommateIsAPenguin}$
- 5)  $\text{PetOfRoommateIsABird}$  (*modus ponens on 4 and 2*)
- 6)  $\text{PetOfRoommateCanFly}$  (*modus ponens on 5 and 1*)
- 7)  $\text{NOT}(\text{PetOfRoommateCanFly})$  (*modus ponens on 4 and 3*)
- 8)  $\text{NOT}(\text{PetOfRoommateCanFly}) \Rightarrow \text{FALSE}$  (*equivalent to 6*)
- 9)  $\text{FALSE}$  (*modus ponens on 7 and 8*)
- 10)  $\text{FALSE} \Rightarrow \text{TheMoonIsMadeOfCheese}$  (*tautology*)
- 11)  $\text{TheMoonIsMadeOfCheese}$  (*modus ponens on 9 and 10*)

# Getting more systematic

- Any knowledge base can be written as a single formula in **conjunctive normal form (CNF)**
  - CNF formula: (... OR ... OR ...) AND (... OR ...) AND ...
  - ... can be a symbol  $x$ , or  $\text{NOT}(x)$  (these are called **literals**)
  - Multiple facts in knowledge base are effectively ANDed together

RoommateWet  $\Rightarrow$  (RoommateWetBecauseOfRain  
OR RoommateWetBecauseOfSprinklers)

becomes

(NOT(RoommateWet) OR  
RoommateWetBecauseOfRain OR  
RoommateWetBecauseOfSprinklers)

# Converting story problem to conjunctive normal form

- RoommateWet
  - RoommateWet
- RoommateWet => (RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers)
  - NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers
- RoommateWetBecauseOfSprinklers => SprinklersOn
  - NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
- RoommateWetBecauseOfRain => NOT(RoommateCarryingUmbrella)
  - NOT(RoommateWetBecauseOfRain) OR NOT(RoommateCarryingUmbrella)
- UmbrellaGone
  - UmbrellaGone
- UmbrellaGone => (YouCarryingUmbrella OR RoommateCarryingUmbrella)
  - NOT(UmbrellaGone) OR YouCarryingUmbrella OR RoommateCarryingUmbrella
- NOT(YouCarryingUmbrella)
  - NOT(YouCarryingUmbrella)

# Unit resolution

- **Unit resolution:** if we have

- $I_1 \text{ OR } I_2 \text{ OR } \dots \text{ OR } I_k$

and

- $\text{NOT}(I_i)$

we can conclude

- $I_1 \text{ OR } I_2 \text{ OR } \dots I_{i-1} \text{ OR } I_{i+1} \text{ OR } \dots \text{ OR } I_k$

- Basically modus ponens

# Applying resolution to story problem

- 1) RoommateWet
- 2) NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers
- 3) NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
- 4) NOT(RoommateWetBecauseOfRain) OR NOT(RoommateCarryingUmbrella)
- 5) UmbrellaGone
- 6) NOT(UmbrellaGone) OR YouCarryingUmbrella OR RoommateCarryingUmbrella
- 7) NOT(YouCarryingUmbrella)
- 8) NOT(UmbrellaGone) OR RoommateCarryingUmbrella (6,7)
- 9) RoommateCarryingUmbrella (5,8)
- 10) NOT(RoommateWetBecauseOfRain) (4,9)
- 11) NOT(RoommateWet) OR RoommateWetBecauseOfSprinklers (2,10)
- 12) RoommateWetBecauseOfSprinklers (1,11)
- 13) SprinklersOn (3,12)

# Limitations of unit resolution

- $P \text{ OR } Q$
- $\text{NOT}(P) \text{ OR } Q$
- Can we conclude  $Q$ ?

# (General) resolution

- **General resolution:** if we have
- $I_1 \text{ OR } I_2 \text{ OR } \dots \text{ OR } I_k$
- and
- $m_1 \text{ OR } m_2 \text{ OR } \dots \text{ OR } m_n$
- where for some  $i, j$ ,  $I_i = \text{NOT}(m_j)$
- we can conclude
- $I_1 \text{ OR } I_2 \text{ OR } \dots \text{ OR } I_{i-1} \text{ OR } I_{i+1} \text{ OR } \dots \text{ OR } I_k \text{ OR } m_1 \text{ OR } m_2$   
 $\text{OR } \dots \text{ OR } m_{j-1} \text{ OR } m_{j+1} \text{ OR } \dots \text{ OR } m_n$
- Same literal may appear multiple times; remove those

# Applying resolution to story problem (more clumsily)

- 1) RoommateWet
- 2) NOT(RoommateWet) OR RoommateWetBecauseOfRain OR RoommateWetBecauseOfSprinklers
- 3) NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
- 4) NOT(RoommateWetBecauseOfRain) OR NOT(RoommateCarryingUmbrella)
- 5) UmbrellaGone
- 6) NOT(UmbrellaGone) OR YouCarryingUmbrella OR RoommateCarryingUmbrella
- 7) NOT(YouCarryingUmbrella)
- 8) NOT(RoommateWet) OR RoommateWetBecauseOfRain OR SprinklersOn (2,3)
- 9) NOT(RoommateCarryingUmbrella) OR NOT(RoommateWet) OR SprinklersOn (4,8)
- 10) NOT(UmbrellaGone) OR YouCarryingUmbrella OR NOT(RoommateWet) OR SprinklersOn (6,9)
- 11) YouCarryingUmbrella OR NOT(RoommateWet) OR SprinklersOn (5,10)
- 12) NOT(RoommateWet) OR SprinklersOn (7,11)
- 13) SprinklersOn (1,12)

# Systematic inference?

- General strategy: if we want to see if sentence  $a$  is entailed, add  $\text{NOT}(a)$  to the knowledge base and see if it becomes inconsistent (we can derive a contradiction)
- CNF formula for modified knowledge base is **satisfiable** if and only if sentence  $a$  is not entailed
  - Satisfiable = there exists a model that makes the modified knowledge base true = modified knowledge base is consistent

# Resolution algorithm

- Given formula in conjunctive normal form, repeat:
- Find two clauses with complementary literals,
- Apply resolution,
- Add resulting clause (if not already there)
- If the **empty** clause results, formula is not satisfiable
  - Must have been obtained from  $P$  and  $\text{NOT}(P)$
- Otherwise, if we get stuck (and we will **eventually**), the formula is guaranteed to be satisfiable (proof in a couple of slides)

# Example

- Our knowledge base:
  - 1) RoommateWetBecauseOfSprinklers
  - 2) NOT(RoommateWetBecauseOfSprinklers) OR SprinklersOn
- Can we infer SprinklersOn? We add:
  - 3) NOT(SprinklersOn)
- From 2) and 3), get
  - 4) NOT(RoommateWetBecauseOfSprinklers)
- From 4) and 1), get empty clause

# If we get stuck, why is the formula satisfiable?

- Consider the final set of clauses  $C$
- Construct satisfying assignment as follows:
- Assign truth values to variables in order  $x_1, x_2, \dots, x_n$
- If  $x_j$  is the last chance to satisfy a clause (i.e., all the other variables in the clause came earlier and were set the wrong way), then set  $x_j$  to satisfy it
  - Otherwise, doesn't matter how it's set
- Suppose this fails (for the first time) at some point, i.e.,  $x_j$  must be set to true for one last-chance clause and false for another
- These two clauses would have resolved to something involving only up to  $x_{j-1}$  (not to the empty clause, of course), which must be satisfied
- But then one of the two clauses must also be satisfied - contradiction

# Special case: Horn clauses

- **Horn clauses** are implications with only positive literals
- $x_1 \text{ AND } x_2 \text{ AND } x_4 \Rightarrow x_3 \text{ AND } x_6$
- $\text{TRUE} \Rightarrow x_1$
- Try to figure out whether some  $x_j$  is entailed
- Simply follow the implications (modus ponens) as far as you can, see if you can reach  $x_j$
- $x_j$  is entailed if and only if it can be reached (can set everything that is not reached to false)
- Can implement this more efficiently by maintaining, for each implication, a count of how many of the left-hand side variables have been reached