

## Reasoning

- You meet three natives of an island of truth-tellers and liars: Aiden, Beatrice and Celia.
- You approach Aiden and ask how many of the trio are truth-tellers, but can't hear his response over the eruption of a nearby volcano.
- After the lava subsides, you ask Beatrice what Aiden said, and her reply is:
  - "He said that one of us is a truth-teller and two of us are liars."
- To which Celia replies:
  - "Don't believe Bea. She's lying."
- What conclusions can you draw about who is telling the truth?
- More importantly, how do you figure out the solution?

## Knowledge-Based Agents

- **Knowledge-based agents** represent, store and process information in order to make logical inferences about the world
- Involves a **knowledge base (KB)**, made up of a series of **sentences** about the world
- Reasoning is the act of obtaining new information given a set of facts
  - modeled on mathematical reasoning
    - drawing certain conclusions from a set of facts
    - more facts → strictly more conclusions
  - different from modeling plausible reasoning
    - conclusions are drawn from a set of evidence
    - new evidence can modify conclusions

## Reasoning Example

- Minesweeper (a nice NP-complete problem)
  - Assuming that the top-left corner of the grid is [0,0], what can we infer about:
    - position [3,1]?
    - position [4,4]?
    - position [0,5]?
  - Some conclusions can be drawn immediately, some need a few intermediate steps, while others can't be inferred without further information
  - **First step:** how do you even represent this knowledge?



## The Language of Knowledge

- Knowledge bases and reasoning must start with a formalism for expression knowledge
- Definitions:
  - **model** = the representation of a world. Like a virtual environment in which certain facts are true and extrapolations can be made
  - **sentence** = an assertion about the world
  - **entailment** = the logical assertion that sentence  $\beta$  must be true in all possible models  $m$  in which sentence  $\alpha$  is true
    - written as  $\alpha \models \beta$
    - spoken as " $\alpha$  entails  $\beta$ "
    - Example: "Bob hit the ball over the fence"  
"The ball was hit over the fence"

## More on Models

- Models are possible configurations of the world, but they do not necessarily match the facts of the KB
- The models that do not contradict the KB form something similar to the belief states described in the section on searching
- Other sentences form their own collections of conforming models
  - task is to find a statement  $\alpha$  such that  $KB \models \alpha$
- Idea is that if KB reflects real world, then inferences made from KB also reflect true statements about the real world.

## Inference Algorithms

- Before getting into representations of the world, consider the techniques for deriving new truths from current ones
- An inference algorithm can have two characteristics:
  - **sound** = only derives entailed sentences (what's the easiest way to find sound inferences?)
  - **complete** = derives all entailed sentences (again, what's the easiest way to achieve this?)
- Want to have algorithms that achieve both, of course

## Propositional Logic

- **Propositional logic** is one form of expressing knowledge about the world in terms of atomic and complex sentences
- Sentences are in **Backus-Naur form** (BNF)
- Made up of:
  - symbols: P, Q, R, True, False
  - negations (NOT):  $\neg$
  - conjunctions (AND):  $\wedge$
  - disjunctions (OR):  $\vee$
  - implications:  $\Rightarrow$  (e.g. *apple  $\Rightarrow$  fruit*)
  - biconditionals:  $\Leftrightarrow$  (e.g. *dark  $\Leftrightarrow$  no light*)
- Parenthesis also used to remove any ambiguity regarding precedence of operators

## PL Connectives

- Truth table values (review):

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

- Minesweeper example:
  - $P_{x,y}$ : # of bombs near  $[x,y]$       $B_{x,y}$ : existence of bomb at  $[x,y]$
  - $KB = \{ P_{2,2} = 1, B_{2,1} = \text{true}, \dots \}$
  - $m_1 = \{ B_{3,1} = \text{false} \}$
  - Therefore  $KB \models m_1 \rightarrow$  but how?

## Inferencing

- To infer that  $m_1$  is true from the previous example, one approach is to explore all possible values for the unknown P and B quantities, and see which combinations cause the statements in KB to always be true
- New statements can also be generated through **logical equivalences** (Figure 7.11 in Russell & Norvig)
- Of note are the following:
  - **Contraposition:**  $(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$
  - **Implication elimination:**  $(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$
  - **DeMorgan:**  $\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$   
 $\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$
- These rules can help determine entailment by exploring the space of possible propositions, but all inferencing algorithms are NP-complete

## Other Inferencing Terms

- **Validity** = a sentence is **valid** if it is true across all models
- **Satisfiability** = a sentence is **satisfiable** if it is true in some model
- **Contradiction** = proof of a sentence by the refutation of its opposite



## Inference Rules

- Inference rules are written with the following notation:

$$\frac{\langle \text{known statements} \rangle}{\langle \text{statement to prove} \rangle}$$

- Example #1: **Modus Ponens**  $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$

- Example #2: **And-Elimination**  $\frac{\alpha \wedge \beta}{\alpha}$

- Example #3: **Resolution Rule** 
$$\frac{l_1 \vee \dots \vee l_k, m_1 \vee \dots \vee m_n, l_i \Leftrightarrow \neg m_j}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \dots \vee l_k, m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \dots \vee m_n}$$

## Resolution

- With these inference rules available and a set of initial propositions, proving a given statement becomes a basic search problem.
- Two starting points for proof:
  - Show that  $KB \models \alpha$ , or that  $KB \wedge \neg \alpha$  is true for all models
  - Try to show that KB does not entail  $\alpha$ , or that  $KB \wedge \neg \alpha$  is unsatisfiable (contradiction approach)
- Two basic resolution approaches:
  - **Bottom-up:** start with the basic facts and gradually build a larger knowledge base by combining these facts with Modus Ponens and the other inference rules (also called **forward chaining**)
  - **Top-down:** start with the sentence to prove, and use the propositions and inference rules to decompose it to its fundamental KB facts. Depth-first search, may require backtracking (also called **backward chaining**)

## Resolution (cont'd)

- In general, forward chaining is both sound and complete, whereas backward chaining needs an adapted version of iterative deepening
- Another useful adaptation: **Horn clauses**
  - most typical knowledge bases are built on literals in **Conjunctive Normal Form (CNF)** that imply a single literal
    - e.g. "if it is Wednesday and 2pm then I have class"
  - this form fits more easily with the forward- and backward-chaining algorithms described earlier
  - deciding entailment with Horn clauses can be done in time that is linear to the size of the KB

**Knowledge Base:**

```

P ⇒ Q
A ⇒ B
C ∧ D ⇒ P
B ∧ F ⇒ D
B ⇒ C
A
F
    
```

## Example: Forward Chaining

- Given set of facts at the right, determine if P (CSC384 is popular) is true, given the following terms:

- E = CSC384 is easy
- I = CSC384 is interesting
- F = CSC384 is fun
- U = CSC384 is useful
- R = CSC384 is a required course
- S = Steve teaches CSC384

**Knowledge Base:**

```

E ∧ I ⇒ P
F ∧ U ⇒ P
R ∧ S ⇒ U
S ⇒ F
E
R
S
    
```

- Proof database is built as follows:

```

{} Line 5 → {E} Line 6 → {E, R} Line 7 → {E, R, S} Line 4 → {E, R, S, F}
Line 3 {E, R, S, F, U} Line 2 {E, R, S, F, U, P}
    
```

## Example: Backward Chaining

- Given same KB as before, search for proof to P.
- Substitute clauses until low-level fact are reached

```

{P}
→ {E ∧ I}
→ {I}
→ failed, backtracking
    
```

```

{P}
→ {F ∧ U}
→ {S ∧ U}
→ {U}
→ {R ∧ S}
→ {S}
→ {} → True, entailment proven
    
```

**Knowledge Base:**

```

E ∧ I ⇒ P
F ∧ U ⇒ P
R ∧ S ⇒ U
S ⇒ F
E
R
S
    
```

## Limitations of PL

- In order to describe the Minesweeper domain, one has to write propositions that describe every possible layout for bombs and number values
  - e.g.  $P_{3,2} \wedge B_{2,1} \Rightarrow \neg B_{3,1}$
  - $P_{3,2} \wedge B_{2,1} \Rightarrow \neg B_{4,1}$
  - etc.
- This leads to the generation of way too many propositions
- Need the ability to generalize
  - universal quantifiers in logic form ( $\forall, \exists$ )
  - variable values in program form

## First-Order Logic

- Where propositional logic deals with the storage and manipulation of facts, first-order logic deals with objects and the relations between them
  - ontological** commitments = a logic's view of reality, or how the world is made up (different in PL and FOL)
  - epistemological** commitments = a logic's state of knowledge about certain facts (in PL and FOL, can be *True*, *False* or *No Opinion*)
- Similar to propositional logic in that it contains constants, connectives and sentences, but it introduces three key concepts
  - Predicates** = functions that characterize objects
  - Variables** = symbols that bind to matching objects
  - Quantifiers** = generalizing operators for the predicates

## First-Order Logic (cont'd)

- Also called **first-order calculus**
  - strong enough to formalize all of mathematics, and is therefore strong enough to represent the world we use
- Introduces new concepts
  - Terms**: an object without its own name. Not a function, more like a dereferenced pointer value.
    - e.g. Father(Steve), Mayor(Toronto)
    - a term with no variables is a **ground term**
  - Atomic sentences**: basic facts, stated in terms of relations between objects
    - e.g. Brother(Chip, Dale)
  - Universal quantifiers** ( $\forall$ ): Declares rule for all cases
    - e.g. "all fathers are men" =  $\forall x \text{ Father}(x) \Rightarrow \text{Man}(x)$
  - Existential quantifiers** ( $\exists$ ): Declares rule for at least one case
    - e.g. "I have a headache" =  $\exists x \text{ Headache}(x) \Rightarrow \text{Has}(\text{Steve}, x)$

## FOL Example

- Family member domain
  - $\forall m,c$   $\text{Mother}(c) = m \Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m,c)$
  - $\forall w,c$   $\text{Husband}(h,w) \Leftrightarrow \text{Male}(h) \wedge \text{Married}(h,w)$
  - $\forall x$   $\text{Male}(x) \Leftrightarrow \neg \text{Female}(x)$
  - $\forall p,c$   $\text{Parent}(p,c) \Leftrightarrow \text{Child}(c,p)$
  - $\forall x,y$   $\text{Sibling}(x,y) \Leftrightarrow (x \neq y) \wedge \exists p \text{Parent}(p,x) \wedge \text{Parent}(p,y)$
  - $\forall a,c$   $\text{Ancestor}(a,c) \Leftrightarrow (a = c) \vee (\exists p \text{Parent}(p,c) \wedge \text{Ancestor}(a,p))$
- How would you express cousins, or nephews?
- Always be careful to anticipate the case when two variables might bind to each other by accident (the Sibling case, for example)

## Expressing Minesweeper

- Back to the explosive domain
  - $\forall x,y,a,b$   $\text{Adjacent}([x,y], [a,b]) \Leftrightarrow$   
 $[a,b] \in \{[x,y+1], [x+1,y+1], [x+1,y],$   
 $[x+1, y-1], [x,y-1], [x-1,y-1],$   
 $[x-1,y], [x-1,y+1]\}$
  - $\forall x,y,a,b,c,d$   $\text{BombValue}(a,b) = 1 \wedge \text{Adjacent}([a,b], [x,y])$   
 $\wedge \text{Adjacent}([a,b], [c,d]) \wedge \text{HasBomb}(c,d)$   
 $\Rightarrow \neg \text{HasBomb}(x,y)$

## Inference with FOL

- Inferring sentences is similar to that of PL, except that in addition to Modus Ponens and the various other operations, a new available action is the **binding** of terms to variables in the KB
- Forward- and backward-chaining algorithms operate in the same way, but with extra operations available to generate statements in the search
  - See Sections 9.3 and 9.4 in Russell & Norvig for more details on these modified algorithms
- Prolog's interpreter does much the same thing, using depth-first backward chaining

## Problems with FOL

- Since FOL definitions can be recursive, sentence definitions can lead to infinite inference loops
- Computationally, this formalism can lead to redundant or repeated calculations
  - lookup tables & shortcut mechanisms used to circumvent this
  - also known as memoization
- Unlike propositional logic, first order logic is undecidable
  - given an arbitrary formula P, there is no provable decision procedure for determining the validity of P