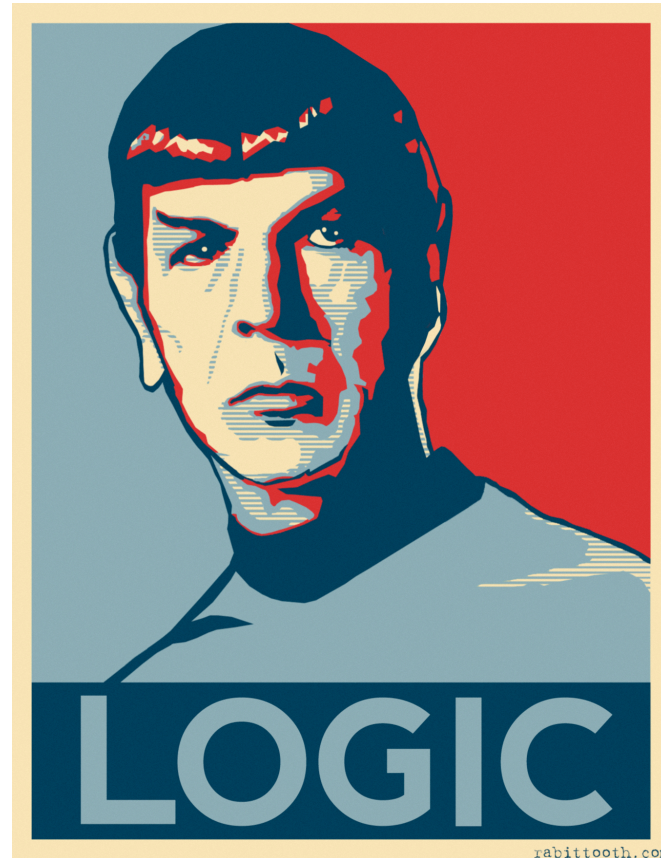


# Ve492: Introduction to Artificial Intelligence

## PL Agents & First Order Logic

---



Paul Weng

UM-SJTU Joint Institute

Slides adapted from <http://ai.berkeley.edu>, AIMA, UM, CMU

---

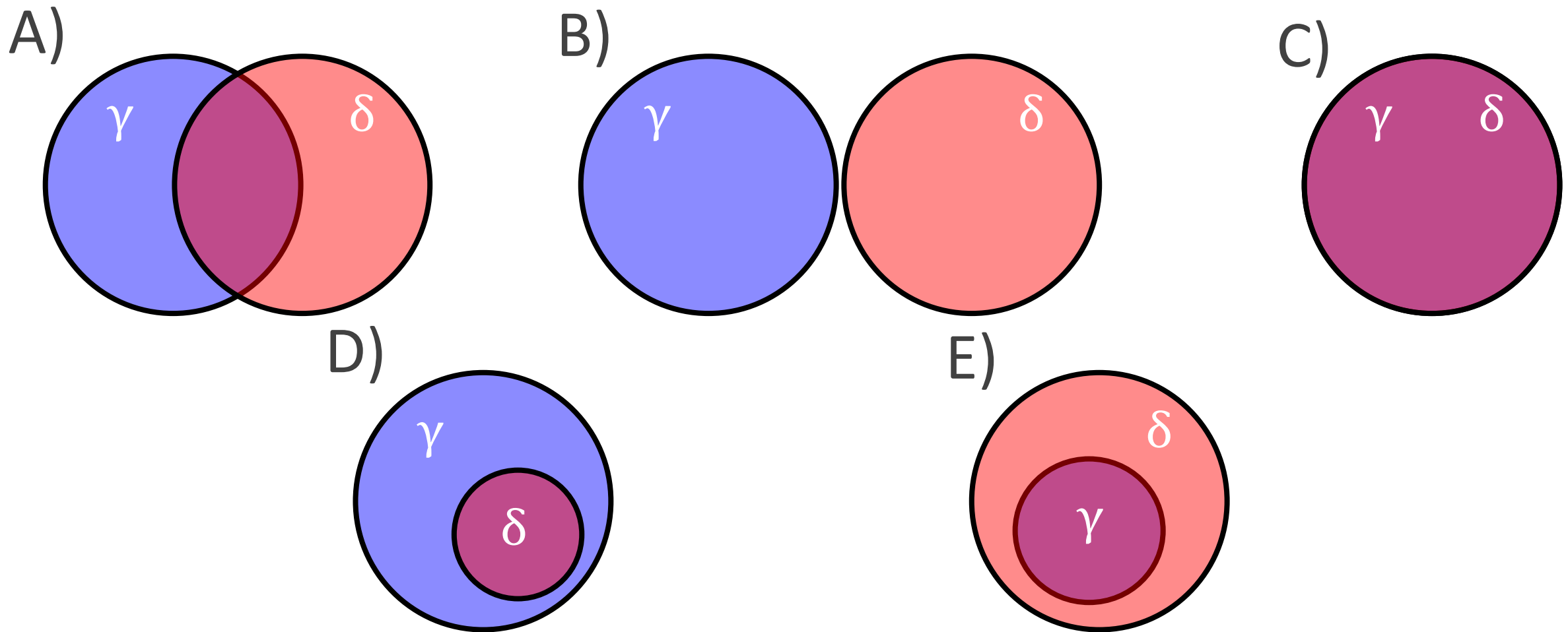
# Today

---

- ❖ Recap of logical agents and propositional logic (PL)
- ❖ Implementing a logical agent using PL
- ❖ First-order logic

# Quiz: Entailment

The regions below visually enclose the set of models that satisfy the respective sentences  $\gamma$  or  $\delta$ . For which of the following diagrams does  $\gamma$  entail  $\delta$ ? Select all that apply.



---

# Recap: Logical Agent

---

## KB

- ❖ Collection of sentences representing facts and rules we know about the world

## Sentence

- ❖ Logical statement
- ❖ Composition of logic symbols and operators

## Model vs Possible World

- ❖ Complete assignment of symbols to True/False

## Query

- ❖ Sentence we want to know if it is *provably* True, *provably* False, or *unsure*.

# Recap: Logical Agent

---

## Satisfy

- ❖ Input: *model*, *sentence*
- ❖ Does *model* *satisfy* *sentence*?
- ❖ Is this *sentence* true in this *model*?
- ❖ **PL-TRUE**

## Entailment

- ❖ Input: *sentence1*, *sentence2*
- ❖ If I know *sentence1* holds, then do I know *sentence2* holds?
- ❖ Each model that satisfies *sentence1* must also satisfy *sentence2*
- ❖ How to compute entailment?
  - ❖ Model checking, e.g., **TT-ENTAILS**
  - ❖ Theorem proving

# Recap: Logical Agent

---

## Valid

- ❖ Input: *sentence*
- ❖ Is *sentence* true in all possible models?

## Satisfiable

- ❖ Input: *sentence*
- ❖ Can find at least one model that satisfies this *sentence*?  
(We often want to know what that model is)
- ❖ Is it possible to make *sentence* true?
- ❖ **DPLL** (efficient SAT solver)

# Vocabulary: Propositional Logic

---

## Literal

- ❖ Atomic sentence: True, False, Symbol,  $\neg$ Symbol

## Clause

- ❖ Disjunction of literals:  $A \vee B \vee \neg C$

## Conjunctive Normal Form (CNF)

- ❖ Conjunction of clauses:  $(A \vee B \vee \neg C) \wedge (\neg A \vee C \vee \neg D)$

## Definite clause

- ❖ Disjunction of literals, *exactly one* is positive
- ❖  $\neg A \vee B \vee \neg C$

## Horn clause

- ❖ Disjunction of literals, *at most one* is positive
- ❖ All definite clauses are Horn clauses

# Implementing a Logical Agent

- ❖ TELL initial knowledge of agent
  - ❖ Initial state:  $\neg P_{1,1} , \neg W_{1,1}$
  - ❖ “Physics” of the world:  $\forall_{i,j} W_{i,j} , \neg(W_{i,j} \wedge W_{i',j'})...$
  - ❖ Encode all these facts in PL; not easy!
- ❖ How to make decisions?
  - ❖ Fully-based on PL
  - ❖ Hybrid



# Hybrid Example: Wumpus World

---

**function** HYBRID-WUMPUS-AGENT(*percept*) **returns** an *action*

**inputs:** *percept*, a list, [*stench*, *breeze*, *glitter*, *bump*, *scream*]  
**persistent:** *KB*, a knowledge base, initially the atemporal “wumpus physics”  
*t*, a counter, initially 0, indicating time  
*plan*, an action sequence, initially empty

TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))  
TELL the *KB* the temporal “physics” sentences for time *t*  
*safe*  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \text{OK}_{x,y}^t) = \text{true}\}$   
**if** ASK(*KB*, *Glitter*<sup>*t*</sup>) = *true* **then**  
    *plan*  $\leftarrow [\text{Grab}] + \text{PLAN-ROUTE}(\text{current}, \{[1,1]\}, \text{safe}) + [\text{Climb}]$   
**if** *plan* is empty **then**  
    *unvisited*  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, L_{x,y}^{t'}) = \text{false} \text{ for all } t' \leq t\}$   
    *plan*  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \text{unvisited} \cap \text{safe}, \text{safe})$   
**if** *plan* is empty and ASK(*KB*, *HaveArrow*<sup>*t*</sup>) = *true* **then**  
    *possible\_wumpus*  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg W_{x,y}) = \text{false}\}$   
    *plan*  $\leftarrow \text{PLAN-SHOT}(\text{current}, \text{possible\_wumpus}, \text{safe})$   
**if** *plan* is empty **then** // no choice but to take a risk  
    *not\_unsafe*  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg \text{OK}_{x,y}^t) = \text{false}\}$   
    *plan*  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \text{unvisited} \cap \text{not\_unsafe}, \text{safe})$   
**if** *plan* is empty **then**  
    *plan*  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \{[1, 1]\}, \text{safe}) + [\text{Climb}]$   
*action*  $\leftarrow \text{POP}(\text{plan})$   
TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))  
*t*  $\leftarrow t + 1$   
**return** *action*

---

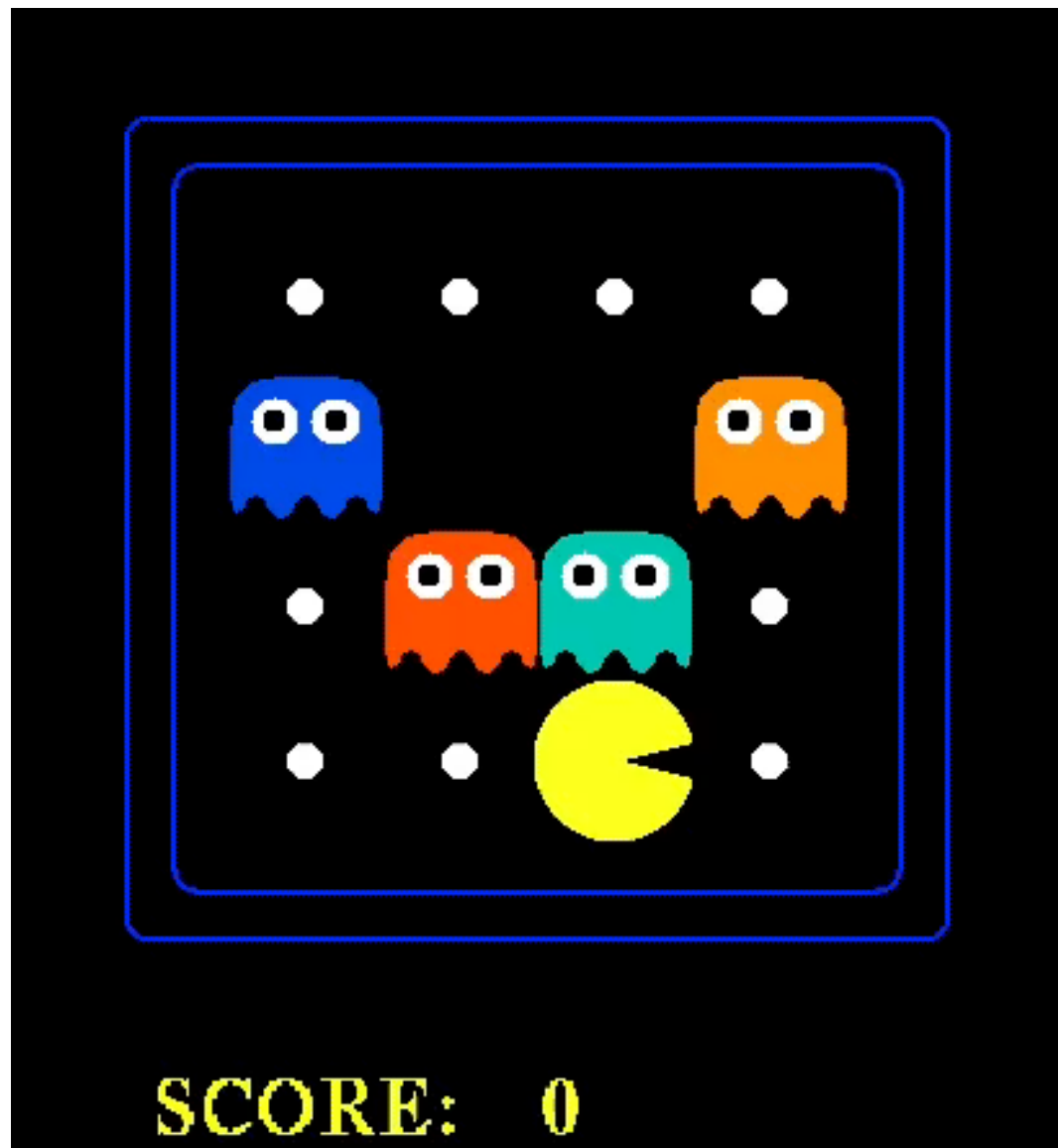
# PL-based Example

- ❖ Initial knowledge requires transition model
- ❖ How to encode the agent's location? Is it sufficient to add  $L_{i,j}$  for all  $i$  and  $j$ ?
  - ❖ We need  $L_{i,j}^t$  for all  $i, j, t$  !
  - ❖ Symbols that depend on time are called **fluents**
- ❖ We need symbols for actions:
  - ❖  $Forward^t, TurnLeft^t, \dots$
- ❖ Transition model (successor-state axioms) expressed for all  $t$ :
  - ❖  $F^{t+1} \iff (F^t \wedge \neg ActionCausesNotF^t) \vee ActionCausesF^t$
  - ❖ E.g.,  $L_{1,1}^{t+1} \iff (L_{1,1}^t \wedge (\neg Forward^t \vee Bump^{t+1})) \vee$   
 $(L_{1,2}^t \wedge (South^t \wedge Forward^t)) \vee$   
 $(L_{2,1}^t \wedge (West^t \wedge Forward^t))$

# PL-based Example

- ❖ Construct a sentence that includes
  - ❖ Initial state, domain knowledge
  - ❖ Transition model for all  $t = 1, \dots, T$
  - ❖ Axioms about the world (e.g., preconditions and action exclusion)
  - ❖ Goal state:  $\text{HaveGold}_T \wedge \text{ClimbOut}_T$
- ❖ Give the sentence to SAT solver
  - ❖ If not satisfiable increment  $T$ , and repeat
- ❖ Extract plan by choosing action at timestep  $t$  if corresponding fluent is true
- ❖ **Limitation:** only works with fully observable problem

# Pacman as a Logical Agent



---

# First Order Logic

---



**KEEP  
CALM  
AND  
USE  
FIRST-ORDER  
LOGIC**

# Pros and Cons of Propositional Logic

- ❖ Propositional logic is **declarative**: pieces of syntax correspond to facts
- ❖ Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- ❖ Propositional logic is **compositional**:  
e.g., meaning of  $B_{1,1} \wedge P_{1,2}$  is derived from meaning of  $B_{1,1}$  and of  $P_{1,2}$
- ❖ Meaning in propositional logic is **context-independent** (unlike natural language, where meaning depends on context)
- ❖ Propositional logic has very limited expressive power (unlike natural language)  
e.g., cannot say “pits cause breezes in adjacent squares”  
except by writing one sentence for each square

# Pros and Cons of Propositional Logic

---

## ❖ Rules of Chess:

- ❖ 100,000 pages in propositional logic
- ❖ 1 page in first-order logic

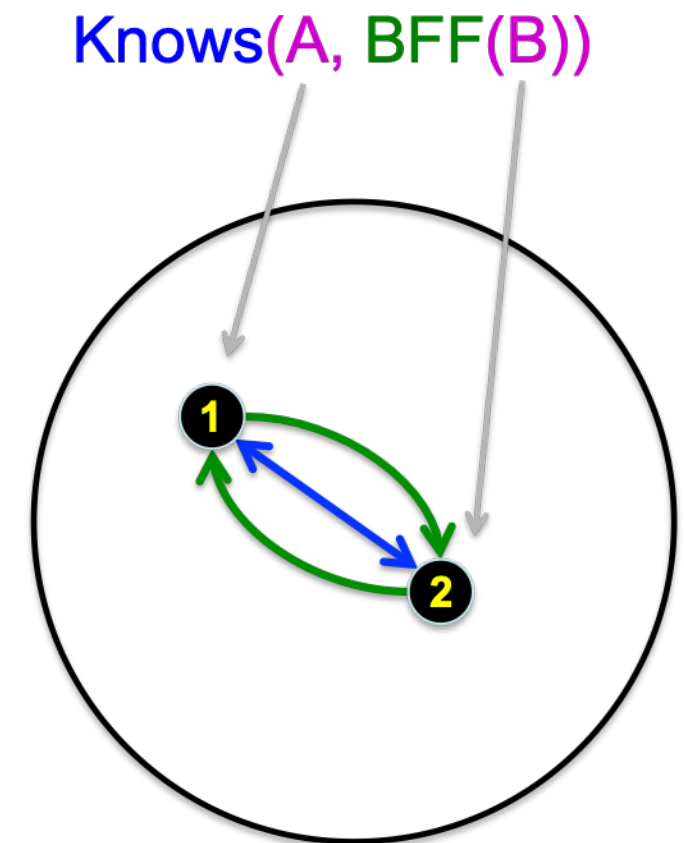
## ❖ Rules of Wumpus World:

$$\forall x, y \text{ Breezy}([x, y]) \Leftrightarrow \exists a, b \text{ Adjacent}([a, b], [x, y]) \wedge \text{Pit}([a, b])$$

$$\forall x, y, a, b \quad \text{Adjacent}([x, y], [a, b]) \Leftrightarrow [a, b] \in \{[x + 1, y], [x - 1, y], [x, y + 1], [x, y - 1]\}$$

# First-Order Logic

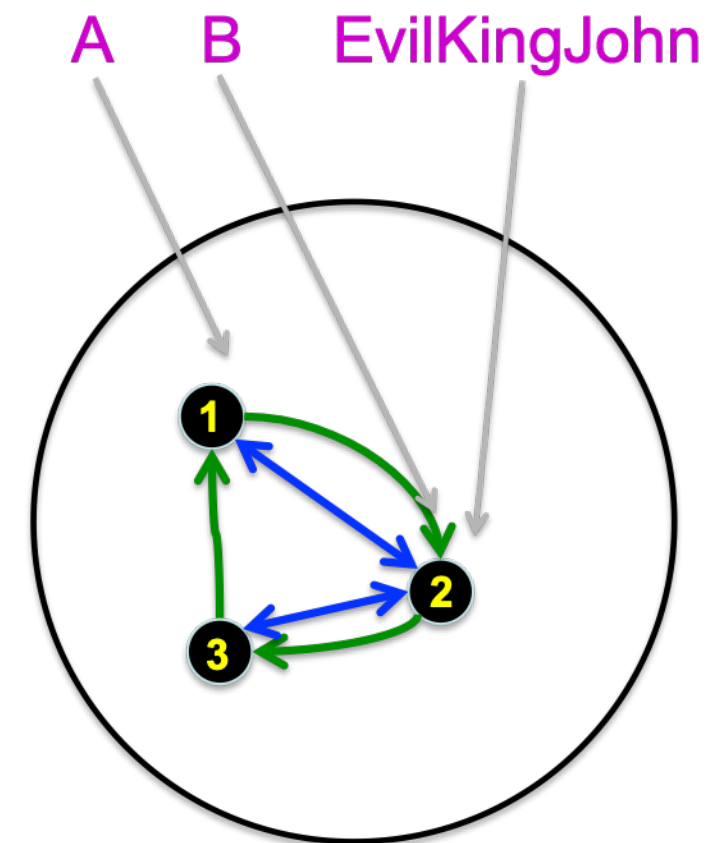
- ❖ Whereas propositional logic assumes world contains facts, first-order logic assumes the world contains:
  - ❖ Objects: people, integers, body parts, JI courses, events, dates...
  - ❖ Constants: Donald Trump, 127, Ve492, French revolution
  - ❖ Relations: knows, is prime, is US president, prerequisite, occurred after, ...
  - ❖ Functions: best friend forever (BFF), successor, left leg of, end of, ...
- ❖ These define possible worlds





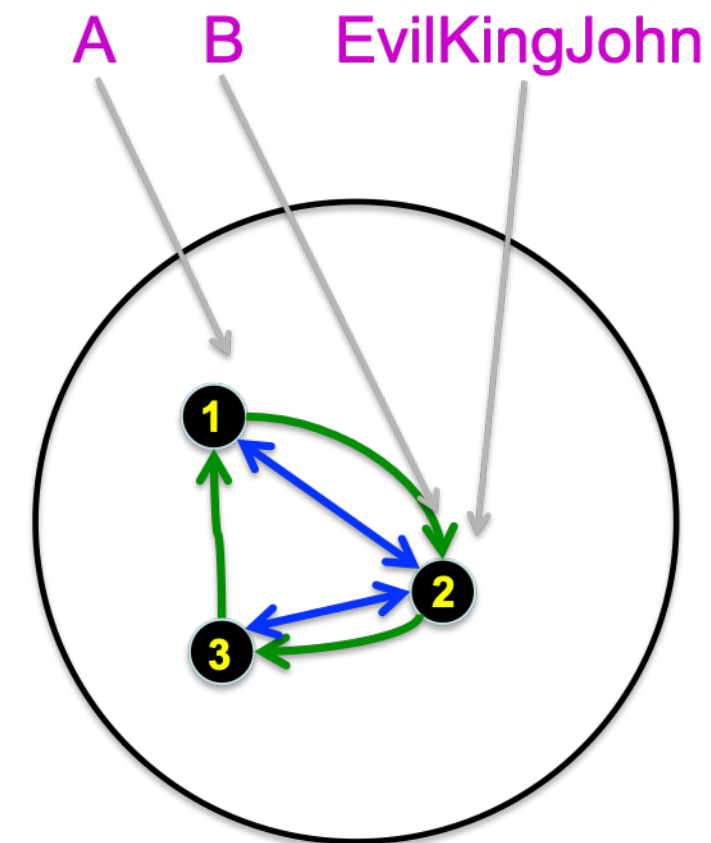
# Syntax and Semantics: Terms

- ❖ A term refers to an object; it can be:
  - ❖ a constant symbol, e.g.,  $A$ ,  $B$ ,  $EvilKingJohn$ 
    - ❖ The possible world fixes these referents
  - ❖ a function symbol with terms as arguments, e.g.,  $BFF(EvilKingJohn)$ 
    - ❖ The possible world specifies the value of the function, given the referents of the terms
      - ❖  $BFF(EvilKingJohn) \rightarrow BFF(2) \rightarrow 3$
  - ❖ a variable, e.g.,  $x$



# Syntax and Semantics: Atomic Sentences

- ❖ An atomic sentence is an elementary proposition (cf symbols in PL)
  - ❖ A predicate symbol with terms as arguments, e.g.,  $\text{Knows}(\text{A}, \text{BFF}(\text{B}))$ 
    - ❖ True iff the objects referred to by the terms are in the relation referred to by the predicate
    - ❖  $\text{Knows}(\text{A}, \text{BFF}(\text{B})) \rightarrow \text{Knows}(1, \text{BFF}(2)) \rightarrow \text{Knows}(1, 3) \rightarrow \text{F}$
  - ❖ An equality between terms, e.g.,  $\text{BFF}(\text{BFF}(\text{BFF}(\text{B}))) = \text{B}$ 
    - ❖ True iff the terms refer to the same objects
    - ❖  $\text{BFF}(\text{BFF}(\text{BFF}(\text{B}))) = \text{B} \rightarrow \text{BFF}(\text{BFF}(\text{BFF}(2))) = 2 \rightarrow \text{BFF}(\text{BFF}(3)) = 2 \rightarrow \text{BFF}(1) = 2 \rightarrow 2 = 2 \rightarrow \text{T}$



# Syntax and Semantics: Complex Sentences

- ❖ Sentences with logical connectives

- ❖  $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \Rightarrow \beta, \alpha \Leftrightarrow \beta$

- ❖ Sentences with universal or existential quantifiers, e.g.,

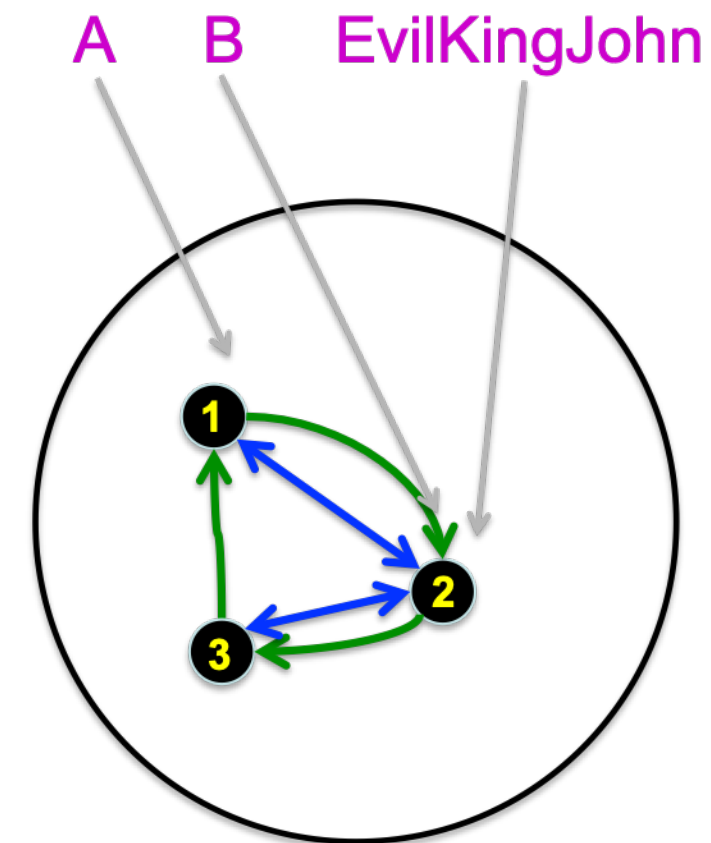
- ❖  $\forall x \text{ Knows}(x, \text{BFF}(x))$

- ❖ True in world  $w$  iff true in all extensions of  $w$  where  $x$  refers to an object in  $w$

- ❖  $x \rightarrow 1: \text{Knows}(1, \text{BFF}(1)) \rightarrow \text{Knows}(1, 2) \rightarrow \text{T}$

- ❖  $x \rightarrow 2: \text{Knows}(2, \text{BFF}(2)) \rightarrow \text{Knows}(2, 3) \rightarrow \text{T}$

- ❖  $x \rightarrow 3: \text{Knows}(3, \text{BFF}(3)) \rightarrow \text{Knows}(3, 1) \rightarrow \text{F}$



# Syntax and Semantics: Complex Sentences

- ❖ Sentences with logical connectives

- ❖  $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \Rightarrow \beta, \alpha \Leftrightarrow \beta$

- ❖ Sentences with universal or existential quantifiers, e.g.,

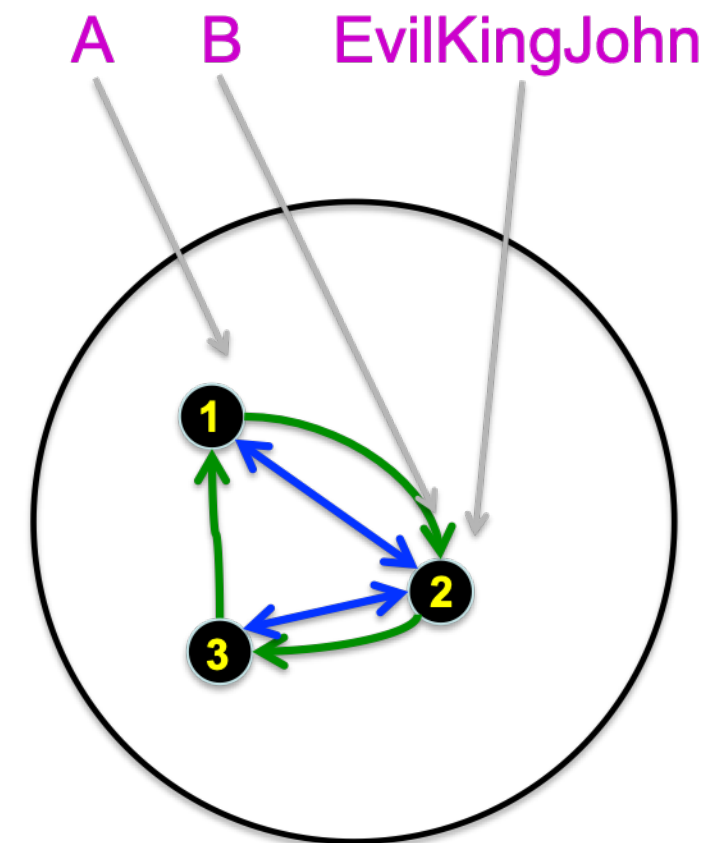
- ❖  $\exists x \text{ Knows}(x, \text{BFF}(x))$

- ❖ True in world  $w$  iff true in some extension of  $w$  where  $x$  refers to an object in  $w$

- ❖  $x \rightarrow 1: \text{Knows}(1, \text{BFF}(1)) \rightarrow \text{Knows}(1, 2) \rightarrow \text{T}$

- ❖  $x \rightarrow 2: \text{Knows}(2, \text{BFF}(2)) \rightarrow \text{Knows}(2, 3) \rightarrow \text{T}$

- ❖  $x \rightarrow 3: \text{Knows}(3, \text{BFF}(3)) \rightarrow \text{Knows}(3, 1) \rightarrow \text{F}$



# Syntax of First Order Logic

- ❖ Sentence  $\rightarrow$  AtomicSentence | ComplexSentence
- ❖ AtomicSentence  $\rightarrow$  Predicate | Predicate(Term, ...)
  - | Term = Term
- ❖ Term  $\rightarrow$  Function(Term, ...) | Constant | Variable
- ❖ ComplexSentence  $\rightarrow$  (Sentence) |  $\neg$  Sentence
  - | Sentence  $\wedge$  Sentence
  - | Sentence  $\vee$  Sentence
  - | Sentence  $\Rightarrow$  Sentence
  - | Sentence  $\Leftrightarrow$  Sentence
  - | Quantifier variable,... Sentence
- ❖ Quantifier  $\rightarrow \forall$  |  $\exists$
- ❖ Constant  $\rightarrow$  A | X\_1 | John | ...
- ❖ Variable  $\rightarrow$  a | x | s | ...
- ❖ Predicate  $\rightarrow$  True | False | Even | Raining | NeighborOf | Loves | ...
- ❖ Function  $\rightarrow$  Successor | Temperature | Mother | LeftLeg | ...

---

# Let's Have Fun with FOL!

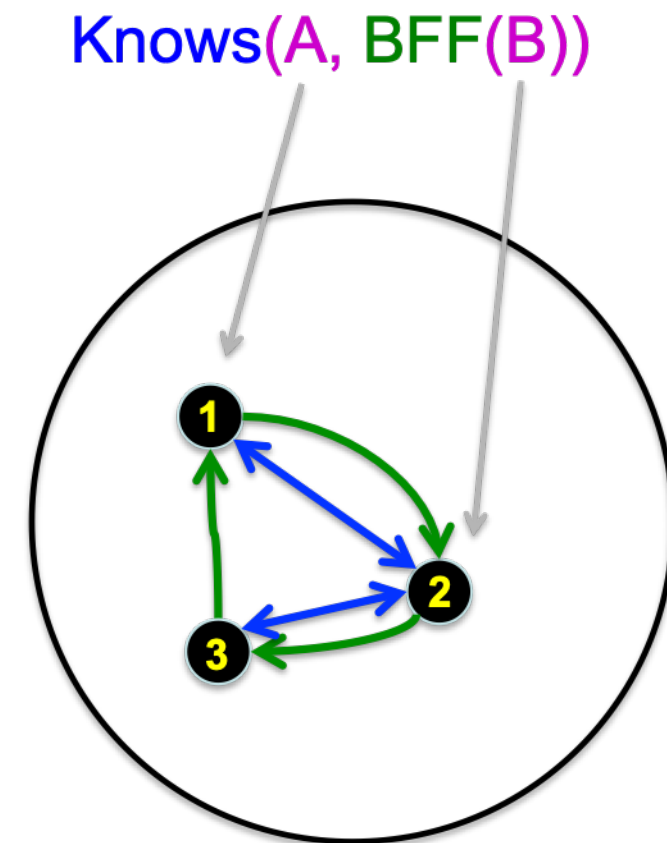
---

## ❖ Translate

- ❖ Everybody loves somebody
- ❖ Everybody's looking for something
- ❖ Some of them want to use you
- ❖ Some of them want to get used by you
- ❖ All greedy kings are evil
- ❖ Some greedy kings are evil

# Models and Interpretations in FOL

- ❖ Given a set of objects, a model is defined by an interpretation:
  - ❖ Which object each constant refers to?
  - ❖ How to define each relation?
  - ❖ How to define each function?



# Let's Formalize Natural Numbers

---

- ❖ Objects =  $\mathbb{O}$
- ❖ Constant: 0
- ❖ Function:  $S : \mathbb{N} \rightarrow \mathbb{N}$
- ❖ Predicates:  $\text{NatNum} : \mathbb{O} \rightarrow \mathbb{B}$ 
  - ❖  $\text{NatNum}(0)$
  - ❖  $\forall n \text{ NatNum}(n) \Rightarrow \text{NatNum}(S(n))$
- ❖ Addition:  $+ : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 
  - ❖  $\forall n \text{ NatNum}(n) \Rightarrow +(n, 0) = n$
  - ❖  $\forall n, m \text{ NatNum}(n) \wedge \text{NatNum}(m) \Rightarrow +(n, S(m)) = S(+(n, m))$



# Quiz: FOL on $\mathbb{N}$

❖ Choose the correct FOL sentence for “Any square number is not a prime.”

1.  $\exists n \exists m \ n = m \times m \Rightarrow \neg \text{Prime}(n)$

2.  $\forall n \exists m \ n = m \times m \Rightarrow \neg \text{Prime}(n)$

3.  $\exists n \exists m \ (n = m \times m) \wedge (\neg \text{Prime}(n))$

4.  $\forall n \exists m \ (n = m \times m) \wedge (\neg \text{Prime}(n))$

---

# Tarski's World

---

- ❖ Book + software

- ❖ <https://web.stanford.edu/group/cslipublications/cslipublications/site/1575864843.shtml>

- ❖ Open source version:

- ❖ <https://courses.cs.washington.edu/courses/cse590d/03sp/tarski/tarski.html>

# Let's Formalize Wumpus World

---

## ❖ Objects:

- ❖ Wumpus
- ❖ Right, Left, Forward, Shoot, Grab, Release, Climb
- ❖  $\mathbb{N}$  for location and time
- ❖ ...

## ❖ Functions:

- ❖ Turn(Right)
- ❖ ...

## ❖ Predicates:

- ❖ Breezy( $[x, y]$ ), Pit( $[a, b]$ ), Adjacent( $[a, b]$ ,  $[x, y]$ ), At( $[x, y]$ ,  $t$ ), Action( $a$ ,  $t$ )
- ❖ West( $t$ ), East( $t$ ), North( $t$ ), South( $t$ )
- ❖ ...

# Let's Formalize Wumpus World

## ❖ Physics of the world:

- ❖  $\forall x, y, a, b \quad \text{Adjacent}([x, y], [a, b]) \Leftrightarrow [a, b] \in \{[x + 1, y], [x - 1, y], [x, y + 1], [x, y - 1]\}$
- ❖  $\forall x, y \quad \text{Breezy}([x, y]) \Leftrightarrow \exists a, b \quad \text{Adjacent}([a, b], [x, y]) \wedge \text{Pit}([a, b])$
- ❖  $\forall x, y, t \quad \text{At}([x, y], t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}([x, y])$
- ❖  $\forall x, y, t \quad \text{At}([x, y], t) \Leftrightarrow \begin{aligned} &(\text{At}([x + 1, y], t - 1) \wedge \text{West}(t - 1) \wedge \text{Action}(\text{Forward}, t - 1)) \\ &\vee (\text{At}([x - 1, y], t - 1) \wedge \text{East}(t - 1) \wedge \text{Action}(\text{Forward}, t - 1)) \\ &\vee (\text{At}([x, y - 1], t - 1) \wedge \text{North}(t - 1) \wedge \text{Action}(\text{Forward}, t - 1)) \\ &\vee (\text{At}([x, y + 1], t - 1) \wedge \text{South}(t - 1) \wedge \text{Action}(\text{Forward}, t - 1)) \\ &\vee (\text{At}([x, y], t - 1) \wedge (\exists a \neg (a = \text{Forward}) \wedge \text{Action}(a, t - 1))) \\ &\vee (\text{At}([x, y], t - 1) \wedge \dots \end{aligned}$
- ❖ ...

# Inference in FOL

- ❖ Entailment is defined exactly as for PL:
  - ❖  $\alpha \models \beta$  iff in every model where  $\alpha$  is true,  $\beta$  is also true
  - ❖ E.g.,  $\forall x \text{ Knows}(x, \text{Obama})$  entails  $\exists y \forall x \text{ Knows}(x, y)$
- ❖ If asked “Do you know what time it is?”, it’s rude to say “Yes”
- ❖ Similarly, given an existentially quantified query, it’s polite to provide an answer in the form of a substitution (or binding) for the variable(s):
  - ❖  $\text{KB} = \forall x \text{ Knows}(x, \text{Obama})$
  - ❖  $\text{Query} = \exists y \forall x \text{ Knows}(x, y)$
  - ❖  $\text{Answer} = \text{Yes}, \{y/\text{Obama}\}$
- ❖ Applying the substitution should produce a sentence that is entailed by KB

# Inference in FOL: Propositionalization

- ❖ Convert  $(KB \wedge \neg \alpha)$  to PL, use a PL SAT solver to check (un)satisfiability
  - ❖ Trick: replace variables with ground terms, convert atomic sentences to symbols
    - ❖  $\forall x \text{ Knows}(x, \text{Obama})$  and  $\text{Democrat}(\text{Hillary\_Clinton})$ 
      - ❖  $\text{Knows}(\text{Obama}, \text{Obama})$  and  $\text{Knows}(\text{Hillary\_Clinton}, \text{Obama})$  and  $\text{Democrat}(\text{Hillary\_Clinton})$
      - ❖  $K\_O\_O \wedge K\_C\_O \wedge D\_C$
    - ❖ and  $\forall x \text{ Knows}(\text{Mother}(x), x)$ 
      - ❖  $\text{Knows}(\text{Mother}(\text{Obama}), \text{Obama})$ ,  $\text{Knows}(\text{Mother}(\text{Mother}(\text{Obama})), \text{Mother}(\text{Obama}))$ , ...
  - ❖ Real trick: for  $k = 1$  to infinity, use terms of function nesting depth  $k$
  - ❖ If entailed, will find a contradiction for some finite  $k$ ; if not, may continue for ever;  
**semidecidable**

# Inference in FOL: Lifted Inference

- ❖ Apply inference rules directly to first-order sentences, e.g.,
  - ❖ KB =  $\text{Person}(\text{Socrates}), \forall x \text{Person}(x) \Rightarrow \text{Mortal}(x)$
  - ❖ conclude  $\text{Mortal}(\text{Socrates})$
  - ❖ The general rule is a version of Modus Ponens:
    - ❖ Given  $\alpha[x] \Rightarrow \beta[x]$  and  $\alpha'$ , where  $\alpha'\sigma = \alpha[x]\sigma$  for some substitution  $\sigma$  conclude  $\beta[x]\sigma$ 
      - ❖  $\sigma$  is  $\{x/\text{Socrates}\}$
    - ❖ Given  $\text{Knows}(x, \text{Obama})$  and  $\text{Knows}(y, z) \Rightarrow \text{Likes}(y, z)$ 
      - ❖  $\sigma$  is  $\{y/x, z/\text{Obama}\}$ , conclude  $\text{Likes}(x, \text{Obama})$
- ❖ Examples: Prolog (backward chaining), Datalog (forward chaining), production rule systems (forward chaining), resolution theorem provers

# Gödel's Incompleteness Theorem

---

- ❖ For any logic and consistent KB beyond very simple, some true statements are unprovable.
  - ❖ “beyond very simple” means “capable of expressing the theory of numbers”, which requires the mathematical induction schema.
- ❖ Gödel showed how to express the statement, “This sentence is not provable.”
- The two difficult parts are to express, in logic:
  - ❖ “This sentence  $S$ ” (self-referentiality)
  - ❖  $\text{provable}(S)$
- The paradox of the sentence proves the theorem.



# Summary and Pointers

---

- ❖ FOL is a very expressive formal language
- ❖ Many domains of common-sense and technical knowledge can be written in FOL (see AIMA Ch. 12)
  - ❖ circuits, software, planning, law, network and security protocols, product descriptions, ecommerce transactions, geographical information systems, Google Knowledge Graph, Semantic Web, etc.
- ❖ Inference is semidecidable in general; many problems are efficiently solvable in practice
- ❖ Inference technology for logic programming is especially efficient (see AIMA Ch. 9)