Ve492: Introduction to Artificial Intelligence

PL Agents & First Order Logic

Paul Weng

UM-SJTU Joint Institute

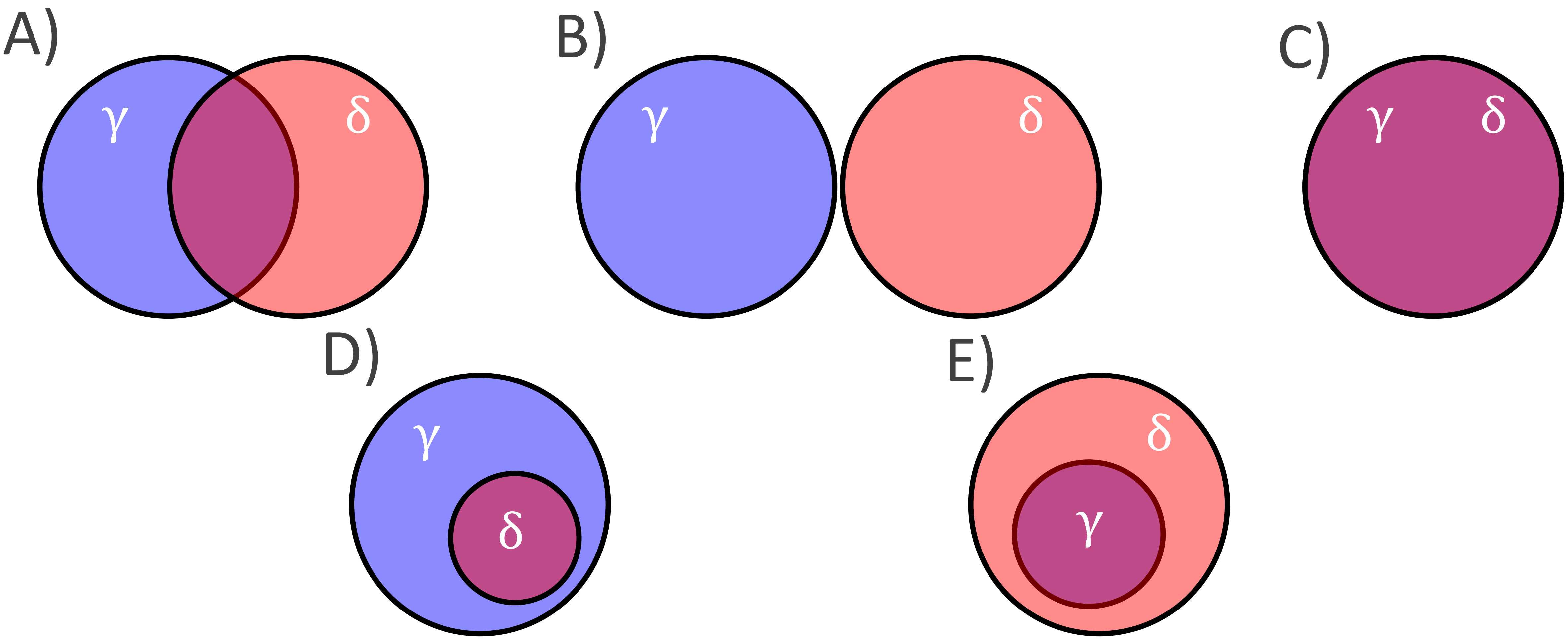
Slides adapted from http://ai.berkeley.edu, AIMA, UM, CMU

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 γ or δ. For which of the following diagrams does γ entail δ? 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, ¬Symbol

Clause

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

Conjunctive Normal Form (CNF)

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

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: $\vee_{i,j} W_{i,j}$, $\neg(W_{i,j} \wedge W_{i',j'})$…
  - 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
  $safe \leftarrow \{[x, y] \;:\; \text{ASK}(KB, OK^t_{x,y}) = true\}$
  if $\text{ASK}(KB, Glitter^t) = true$ **then**
    $plan \leftarrow [Grab] + \text{PLAN-ROUTE}(current, \{[1,1]\}, safe) + [Climb]$
  if *plan* is empty **then**
    $unvisited \leftarrow \{[x, y] \;:\; \text{ASK}(KB, L^{t'}_{x,y}) = false \text{ for all } t' \leq t\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap safe, safe)$
  if *plan* is empty and $\text{ASK}(KB, HaveArrow^t) = true$ **then**
    $possible\_wumpus \leftarrow \{[x, y] \;:\; \text{ASK}(KB, \neg W_{x,y}) = false\}$
    $plan \leftarrow \text{PLAN-SHOT}(current, possible\_wumpus, safe)$
  if *plan* is empty **then** // no choice but to take a risk
    $not\_unsafe \leftarrow \{[x, y] \;:\; \text{ASK}(KB, \neg OK^t_{x,y}) = false\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap not\_unsafe, safe)$
  if *plan* is empty **then**
    $plan \leftarrow \text{PLAN-ROUTE}(current, \{[1, 1]\}, safe) + [Climb]$
  $action \leftarrow \text{POP}(plan)$
  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:
  - $Forward^{t}, TurnLeft^{t}, \ldots$
- Transition model (successor-state axioms) expressed for all t:
  - $F^{t+1} \Leftrightarrow (F^{t} \land \neg ActionCausesNotF^{t}) \lor ActionCausesF^{t}$
  - E.g., $L_{1,1}^{t+1} \Leftrightarrow (L_{1,1}^{t} \land (\neg Forward^{t} \lor Bump^{t+1})) \lor$

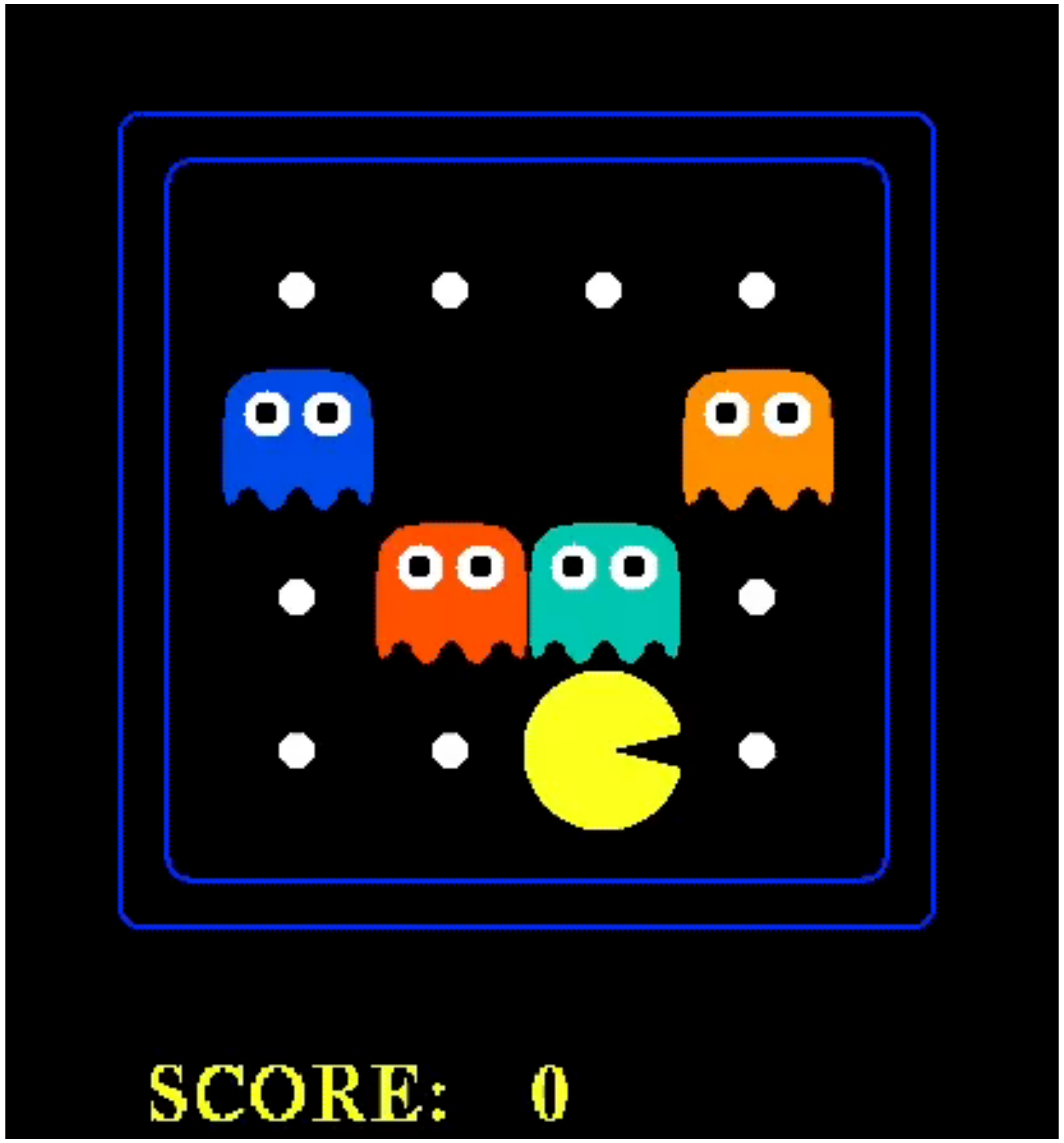
$$(L_{1,2}^{t} \land (South^{t} \land Forward^{t})) \lor$$

$$(L_{2,1}^{t} \land (West^{t} \land Forward^{t}))$$

PL-based Example

- Construct a sentence that includes
  - Initial state, domain knowledge
  - Transition model for all $t = 1, \ldots, T$
  - Axioms about the world (e.g., preconditions and action exclusion)
  - Goal state: $\mathrm{HaveGold}_T \wedge \mathrm{ClimbOut}_T$
- 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

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

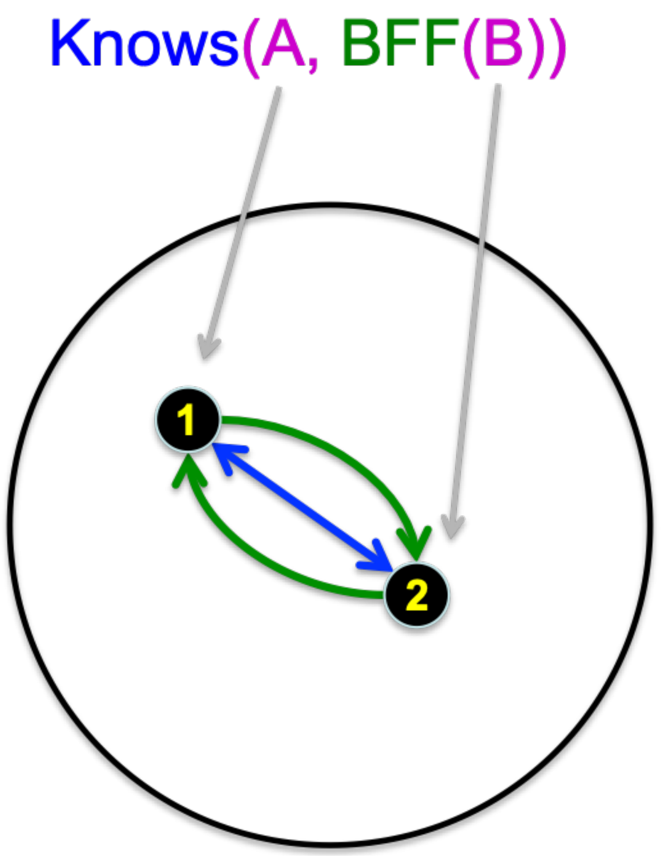
Pros and Cons of Propositional Logic

- Rules of Chess:
  - 100,000 pages in propositional logic
  - 1 page in first-order logic

- Rules of Wumpus World:
  - $\forall x, y \; \mathrm{Breezy}([x, y]) \Leftrightarrow \exists a, b \; \mathrm{Adjacent}([a, b], [x, y]) \wedge \mathrm{Pit}([a, b])$
  - $\forall x, y, a, b \quad \mathrm{Adjacent}([x, y], [a, b]) \Leftrightarrow [a, b] \in \{[x+1, y], [x-1, y], [x, y+1], [x, y-1]\}$

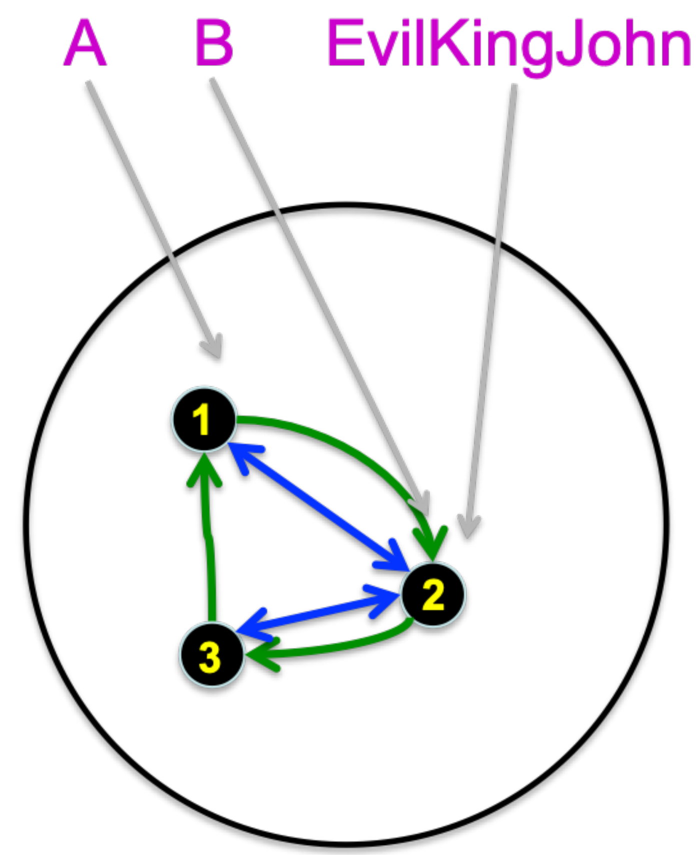
First-Order Logic

- 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



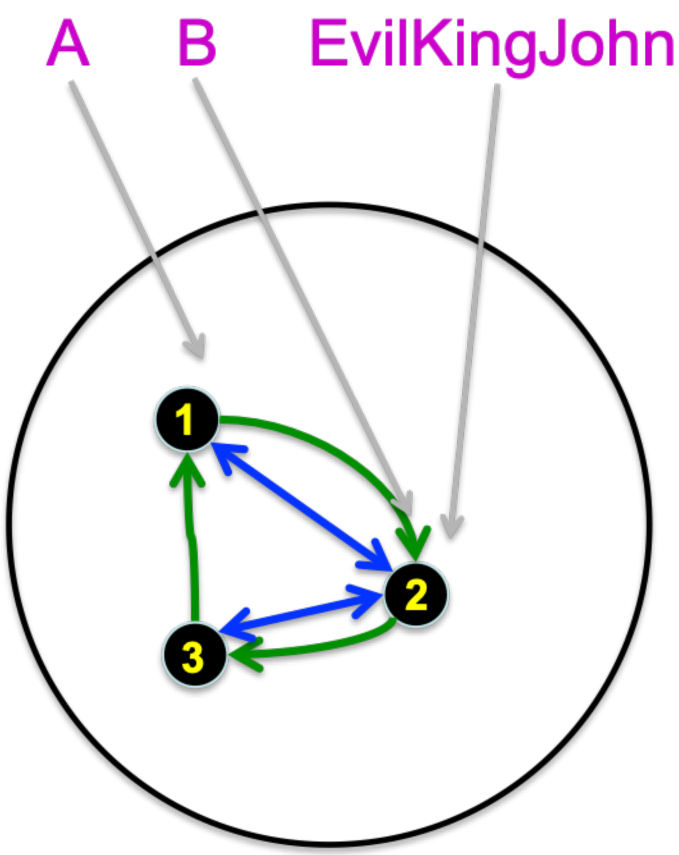
Syntax and Semantics: Terms

- 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) -> BFF(2) -> 3
  - a variable, e.g., x



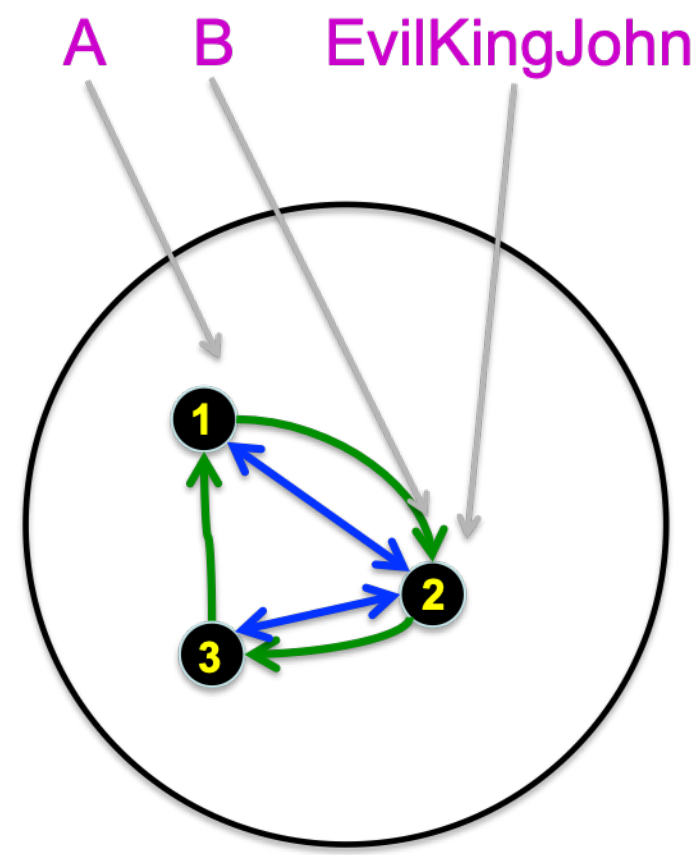
Syntax and Semantics: Atomic Sentences

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



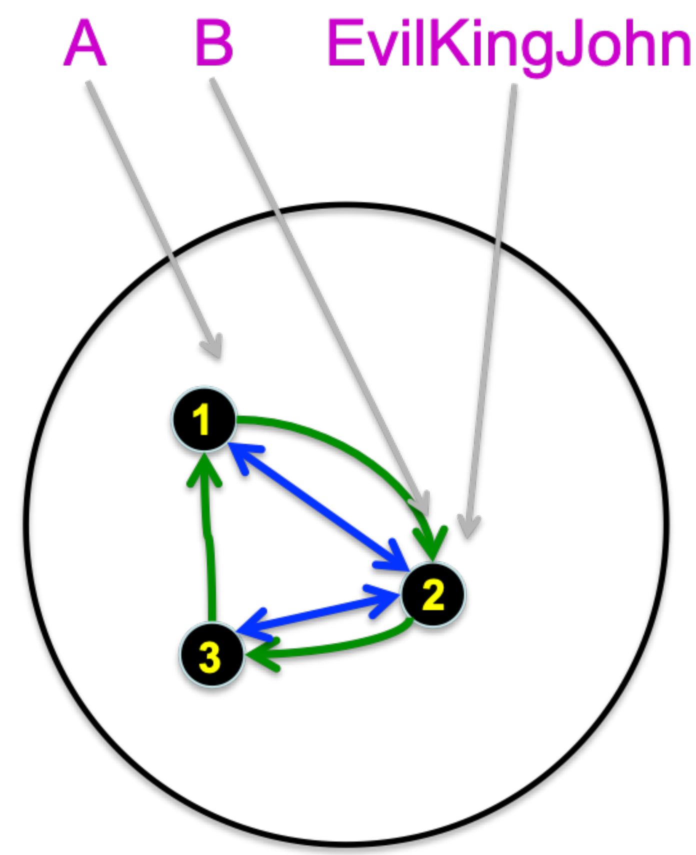
Syntax and Semantics: Complex Sentences

- 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.,
  - ∀x Knows(x, BFF(x))
  - True in world w iff true in all extensions of w where x refers to an object in w
  - x -> 1: Knows(1,BFF(1)) -> Knows(1,2) -> T
  - x -> 2: Knows(2,BFF(2)) -> Knows(2,3) -> T
  - x -> 3: Knows(3,BFF(3)) -> Knows(3,1) -> 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.,
  - ∃x Knows(x, BFF(x))
  - True in world w iff true in some extension of w where x refers to an object in w
  - x -> 1: Knows(1,BFF(1)) -> Knows(1,2) -> T
  - x -> 2: Knows(2,BFF(2)) -> Knows(2,3) -> T
  - x -> 3: Knows(3,BFF(3)) -> Knows(3,1) -> F



Syntax of First Order Logic

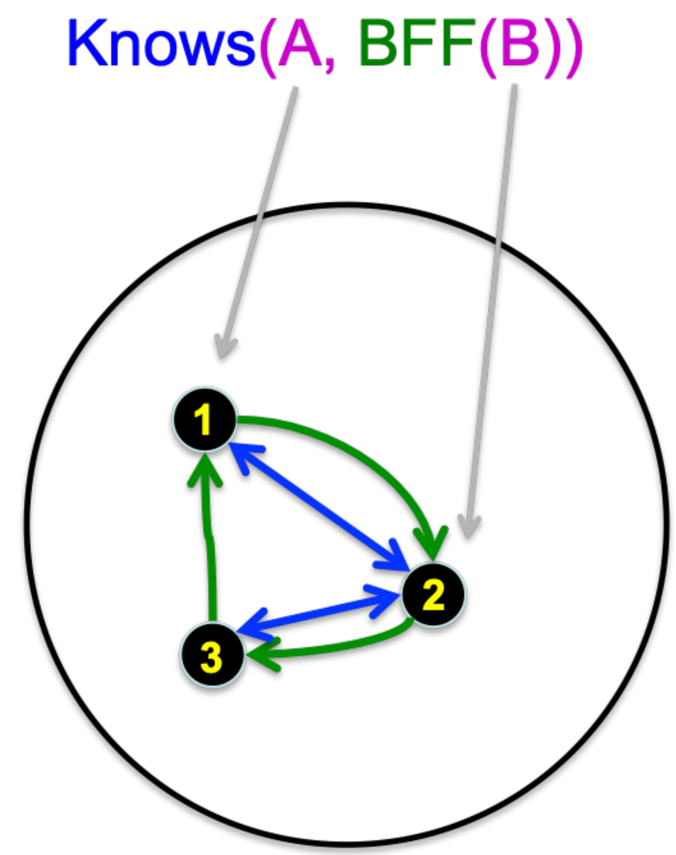
- Sentence → AtomicSentence | ComplexSentence
- AtomicSentence → Predicate | Predicate(Term, …)
  | Term = Term
- Term → Function(Term, …) | Constant | Variable
- ComplexSentence → (Sentence) | ¬ Sentence
  | Sentence ∧ Sentence
  | Sentence ∨ Sentence
  | Sentence ⇒ Sentence
  | Sentence ⇔ Sentence
  | Quantifier variable,… Sentence
- Quantifier → ∀ | ∃
- Constant → A | X_1 | John | …
- Variable → a | x | s | …
- Predicate → True | False | Even | Raining | NeighborOf | Loves |…
- Function → 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 : $\mathbb{N} \to \mathbb{N}$
- Predicates: NatNum : $\mathbb{O} \to \mathbb{B}$
  - NatNum(0)
  - $\forall n$ NatNum(n) $\Rightarrow$ NatNum(S(n))
- Addition: + : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$
  - $\forall n$ NatNum(n) $\Rightarrow$ +(n, 0) = n
  - $\forall n, m$ NatNum(n) $\wedge$ 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. $\exists n \exists m \;\; n = m \times m \Rightarrow \neg \mathrm{Prime}(n)$
2. $\forall n \exists m \;\; n = m \times m \Rightarrow \neg \mathrm{Prime}(n)$
3. $\exists n \exists m \;\; (n = m \times m) \land (\neg \mathrm{Prime}(n))$
4. $\forall n \exists m \;\; (n = m \times m) \land (\neg \mathrm{Prime}(n))$

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:
  - $\text{Breezy}([x, y]), \text{Pit}([a, b]), \text{Adjacent}([a, b], [x, y]), \text{At}([x, y], t), \text{Action}(a, t)$
  - $\text{West}(t), \text{East}(t), \text{North}(t), \text{South}(t)$
  - ...

Let's Formalize Wumpus World

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

Inference in FOL

- Entailment is defined exactly as for PL:
  - $\alpha \vDash \beta$ iff in every model where α is true, β is also true
  - E.g., ∀x Knows(x,Obama) entails ∃y∀x 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):
  - KB = ∀x Knows(x,Obama)
  - Query = ∃y∀x Knows(x,y)
  - Answer = Yes, {y/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
    - ∀x Knows(x,Obama) and Democrat(Hillary_Clinton)
      - Knows(Obama,Obama) and Knows(Hillary_Clinton,Obama) and Democrat(Hillary_Clinton)
      - K_O_O ∧ K_C_O ∧ D_C
    - and ∀x Knows(Mother(x),x)
    - Knows(Mother(Obama),Obama), Knows(Mother(Mother(Obama)),Mother(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 = Person(Socrates), $\forall$x Person(x) $\Rightarrow$ Mortal(x)
  - conclude Mortal(Socrates)
  - The general rule is a version of Modus Ponens:
    - Given $\alpha[x] \Rightarrow \beta[x]$ and α', where $\alpha'\sigma = \alpha[x]\sigma$ for some substitution σ conclude $\beta[x]\sigma$
      - σ is {x/Socrates}
    - Given Knows(x,Obama) and Knows(y,z) $\Rightarrow$ Likes(y,z)
      - σ is {y/x, z/Obama}, conclude Likes(x,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)
  - 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)