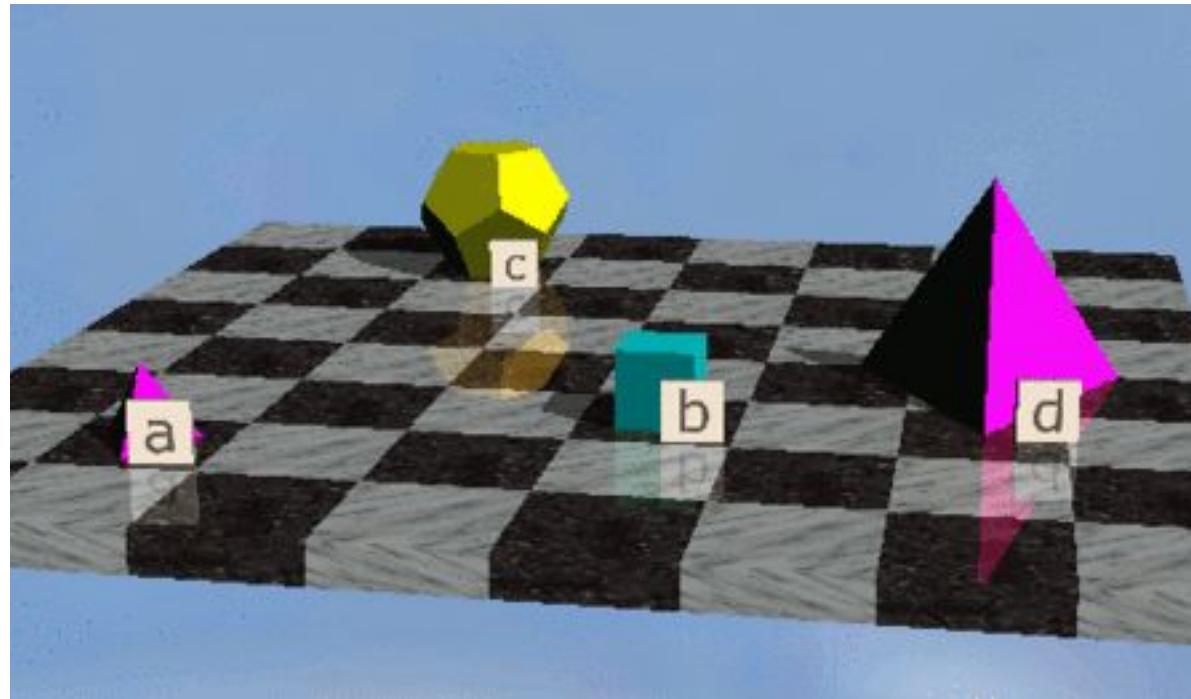


CS 188: Artificial Intelligence

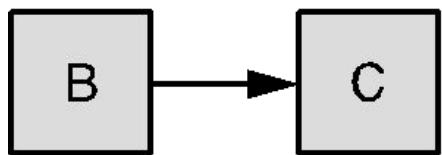
First-Order Logic



Instructors: Stuart Russell and Dawn Song

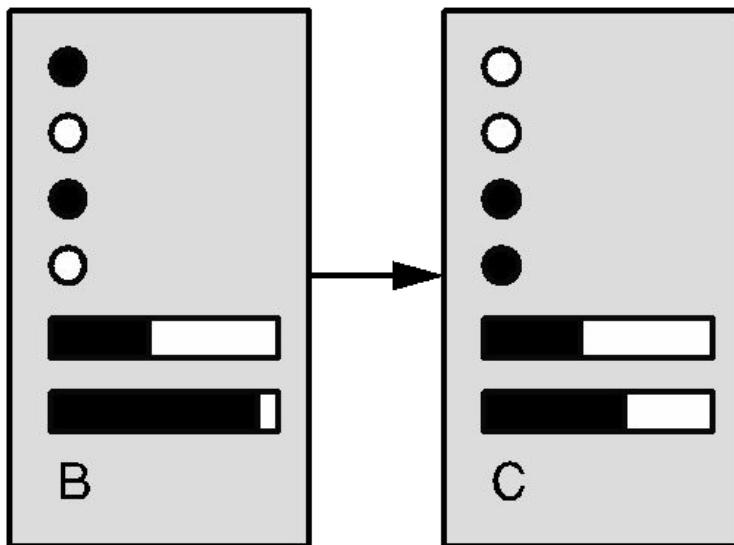
University of California, Berkeley

Spectrum of representations



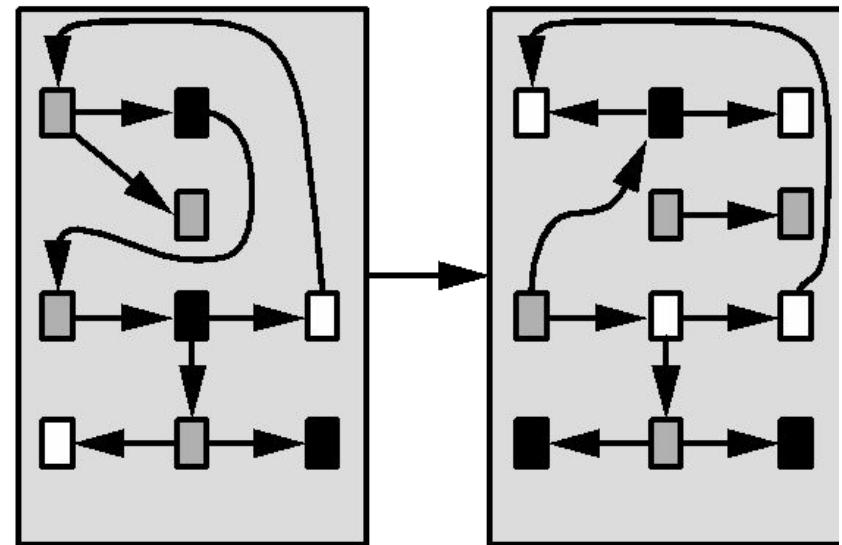
(a) Atomic

**Search,
game-playing**



(b) Factored

**CSPs, planning,
propositional logic,
Bayes nets, neural nets**



(b) Structured

**First-order logic,
databases,
probabilistic programs**

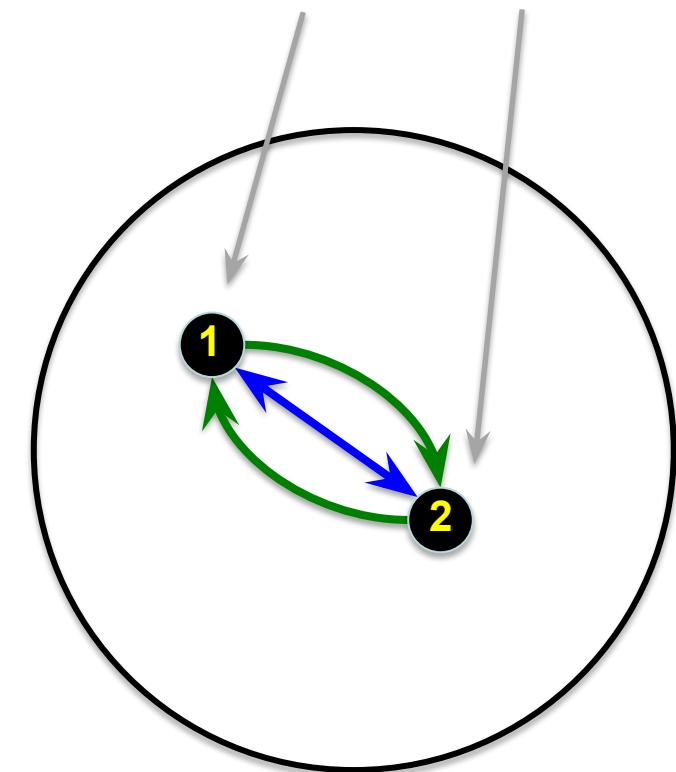
Expressive power

- Rules of chess:
 - 100,000 pages in propositional logic
 - 1 page in first-order logic
- Rules of Pacman:
 - $\forall t \text{Alive}(t) \Leftrightarrow$
 $[\text{Alive}(t-1) \wedge \neg \exists g,x,y [\text{Ghost}(g) \wedge \text{At}(\text{Pacman},x,y,t-1) \wedge \text{At}(g,x,y,t-1)]]$

Possible worlds

- A possible world for FOL consists of:
 - A non-empty set of objects
 - For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
 - For each k-ary function in the language, a mapping from k-tuples of objects to objects
 - For each constant symbol, a particular object (can think of constants as 0-ary functions)

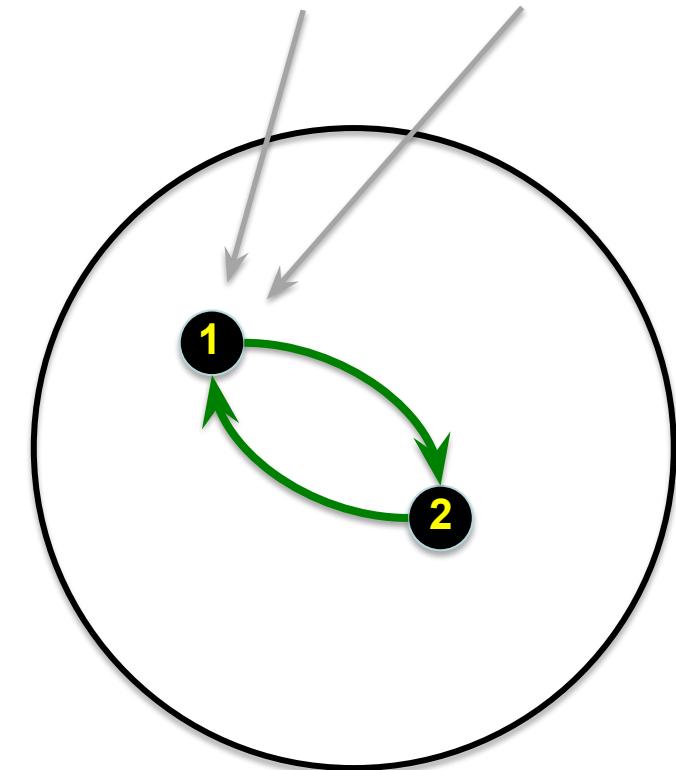
Knows(A, BFF(B))



Possible worlds

- A possible world for FOL consists of:
 - A non-empty set of objects
 - For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
 - For each k-ary function in the language, a mapping from k-tuples of objects to objects
 - For each constant symbol, a particular object (can think of constants as 0-ary functions)

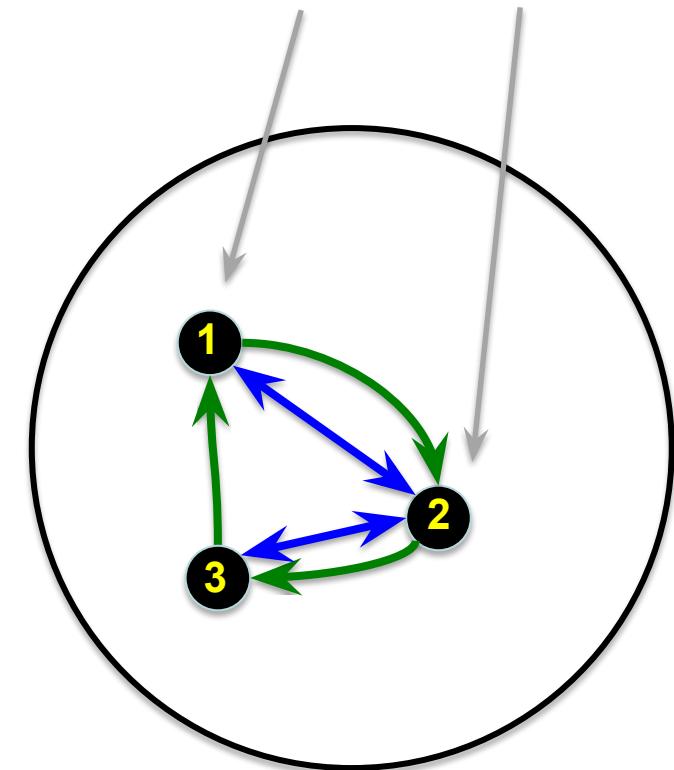
Knows(A, BFF(B))



Possible worlds

- A possible world for FOL consists of:
 - A non-empty set of objects
 - For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
 - For each k-ary function in the language, a mapping from k-tuples of objects to objects
 - For each constant symbol, a particular object (can think of constants as 0-ary functions)

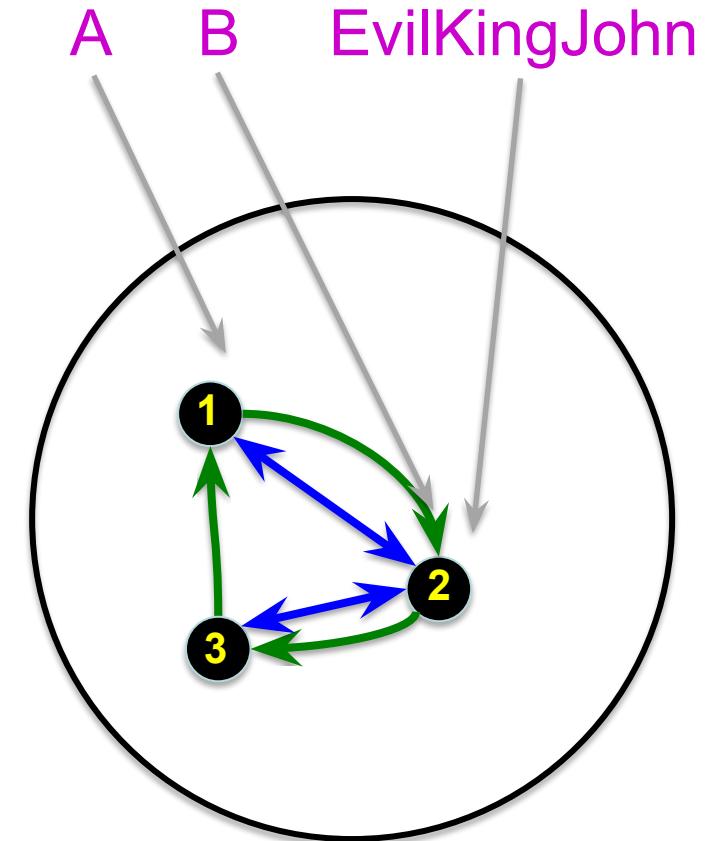
Knows(A, BFF(B))



How many possible worlds?

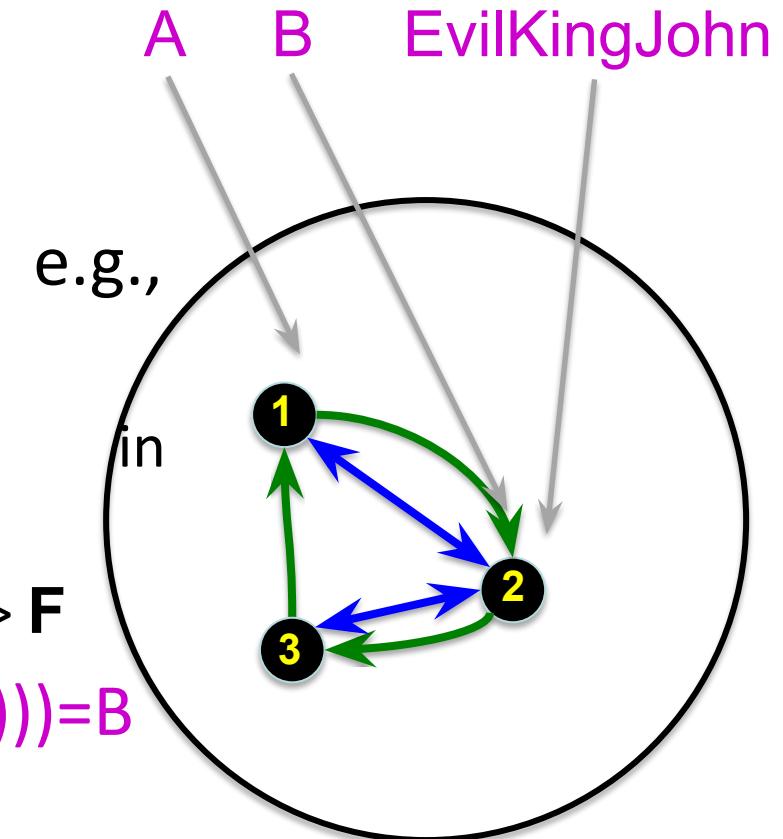
Syntax and semantics: Terms

- 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
 - $\text{BFF}(\text{EvilKingJohn}) \rightarrow \text{BFF}(2) \rightarrow 3$
 - A logical variable, e.g., x
 - (more later)



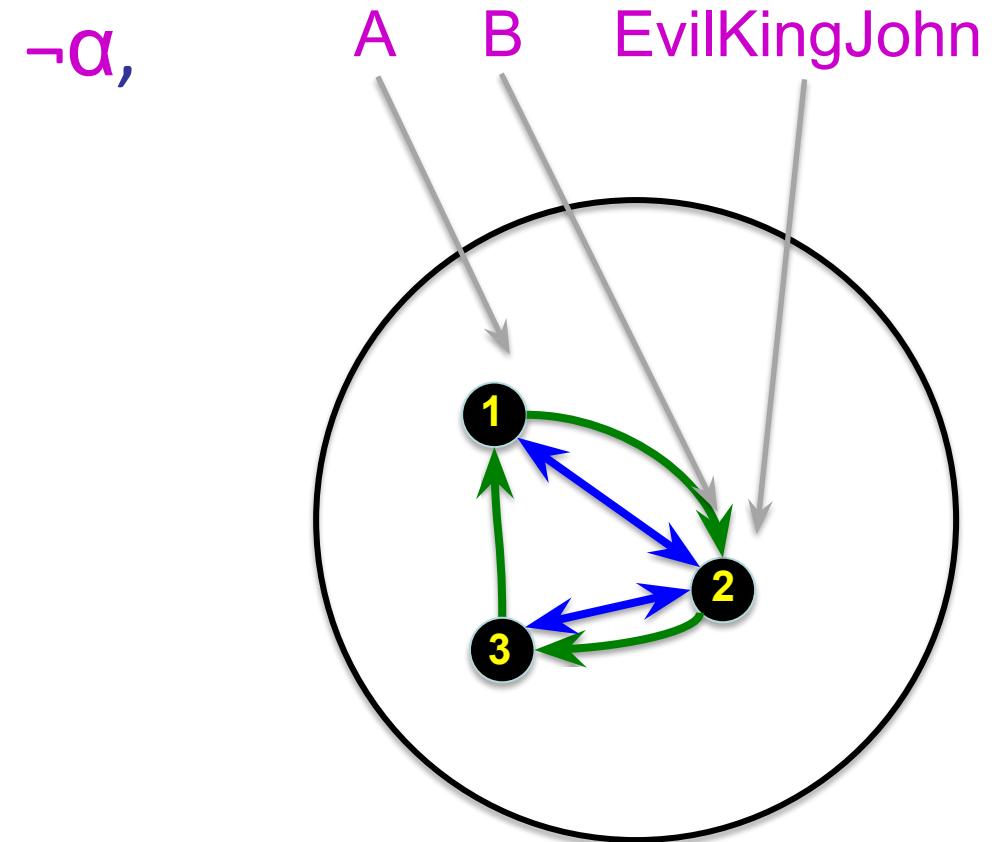
Syntax and semantics: Atomic sentences

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



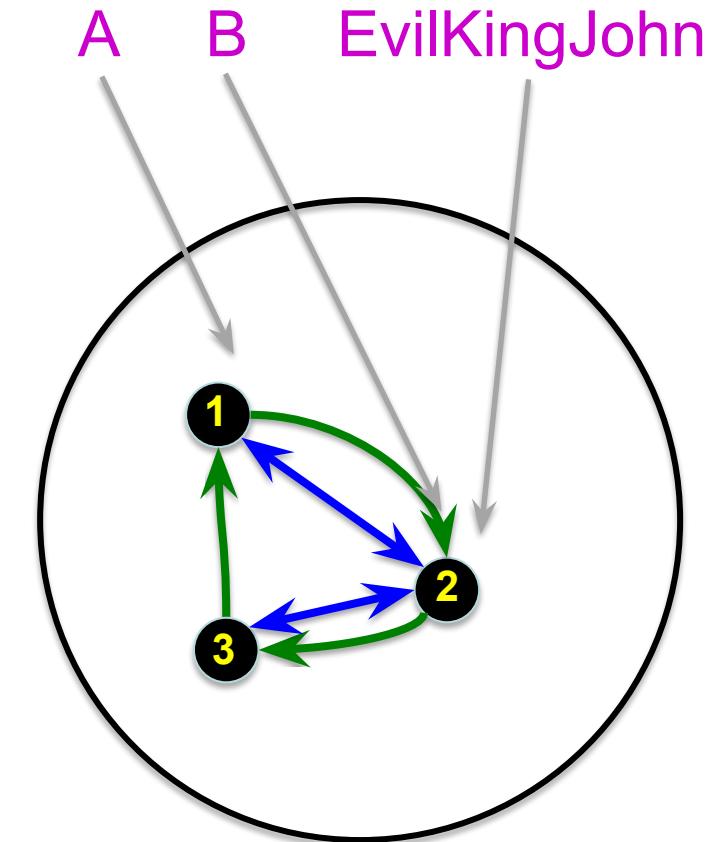
Syntax and semantics: Complex sentences

- 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
 - $x \rightarrow 1: \text{Knows}(1, \text{BFF}(1)) \rightarrow \text{Knows}(1, 2) \rightarrow T$
 - $x \rightarrow 2: \text{Knows}(2, \text{BFF}(2)) \rightarrow \text{Knows}(2, 3) \rightarrow T$
 - $x \rightarrow 3: \text{Knows}(3, \text{BFF}(3)) \rightarrow \text{Knows}(3, 1) \rightarrow F$



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
 - $x \rightarrow 1: \text{Knows}(1, \text{BFF}(1)) \rightarrow \text{Knows}(1, 2) \rightarrow T$
 - $x \rightarrow 2: \text{Knows}(2, \text{BFF}(2)) \rightarrow \text{Knows}(2, 3) \rightarrow T$
 - $x \rightarrow 3: \text{Knows}(3, \text{BFF}(3)) \rightarrow \text{Knows}(3, 1) \rightarrow F$



Fun with sentences

- Everyone knows President Obama
 - $\forall n \text{ Person}(n) \Rightarrow \text{Knows}(n, \text{Obama})$
- There is someone that everyone knows
 - $\exists s \text{ Person}(s) \wedge \forall n \text{ Person}(n) \Rightarrow \text{Knows}(n, s)$
- Everyone knows someone
 - $\forall x \text{ Person}(x) \Rightarrow \exists y \text{ Person}(y) \wedge \text{Knows}(x, y)$

More fun with sentences

- Any two people of the same nationality speak a common language
 - $\text{Nationality}(x,n)$ – x has nationality n
 - $\text{Speaks}(x,l)$ – x speaks language l
 - $\forall x,y (\exists n \text{ Nationality}(x,n) \wedge \text{Nationality}(y,n)) \Rightarrow (\exists l \text{ Speaks}(x,l) \wedge \text{Speaks}(y,l))$

Inference in FOL

- Entailment is defined exactly as for propositional logic:
 - $\alpha \models \beta$ (“ α entails β ”) iff in every world where α is true, β is also true
 - E.g., $\forall x \text{ Knows}(x, \text{Obama})$ entails $\exists y \forall x \text{ Knows}(x, y)$
- In FOL, we can go beyond just answering “yes” or “no”; given an existentially quantified query, return a **substitution** (or **binding**) for the variable(s) such that the resulting sentence is entailed:
 - KB = $\forall x \text{ Knows}(x, \text{Obama})$
 - Query = $\exists y \forall x \text{ Knows}(x, y)$
 - Answer = Yes, $\sigma = \{y/\text{Obama}\}$
 - Notation: $\alpha\sigma$ means applying substitution σ to sentence α
 - E.g., if $\alpha = \forall x \text{ Knows}(x, y)$ and $\sigma = \{y/\text{Obama}\}$, then $\alpha\sigma = \forall x \text{ Knows}(x, \text{Obama})$

Inference in FOL: Propositionalization

- Convert $(KB \wedge \neg a)$ 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 $\text{Democrat}(\text{Feinstein})$
 - $\text{Knows}(\text{Obama}, \text{Obama})$ and $\text{Knows}(\text{Feinstein}, \text{Obama})$ and $\text{Democrat}(\text{Feinstein})$
 - $\text{Knows_Obama_Obama} \wedge \text{Knows_Feinstein_Obama} \wedge \text{Democrat_Feinstein}$
 - and $\forall x \text{Knows}(\text{Mother}(x), x)$
 - $\text{Knows}(\text{Mother}(\text{Obama}), \text{Obama})$ and $\text{Knows}(\text{Mother}(\text{Mother}(\text{Obama})), \text{Mother}(\text{Obama})) \dots\dots$
 - Real trick: for $k = 1$ to infinity, use all possible terms of function nesting depth k
 - If entailed, will find a contradiction for some finite k (Herbrand); 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), $\forall x \text{ Person}(x) \Rightarrow \text{Mortal}(x)$
 - conclude Mortal(Socrates)
 - The general rule is a version of Modus Ponens:
 - Given $\alpha \Rightarrow \beta$ and α' , where $\alpha'\sigma = \alpha\sigma$ for some substitution σ , conclude $\beta\sigma$
 - 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

Summary, pointers

- FOL is a very expressive formal language
- Many domains of common-sense and technical knowledge can be written in FOL (see AIMA Ch. 10)
 - circuits, software, planning, law, taxes, 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)