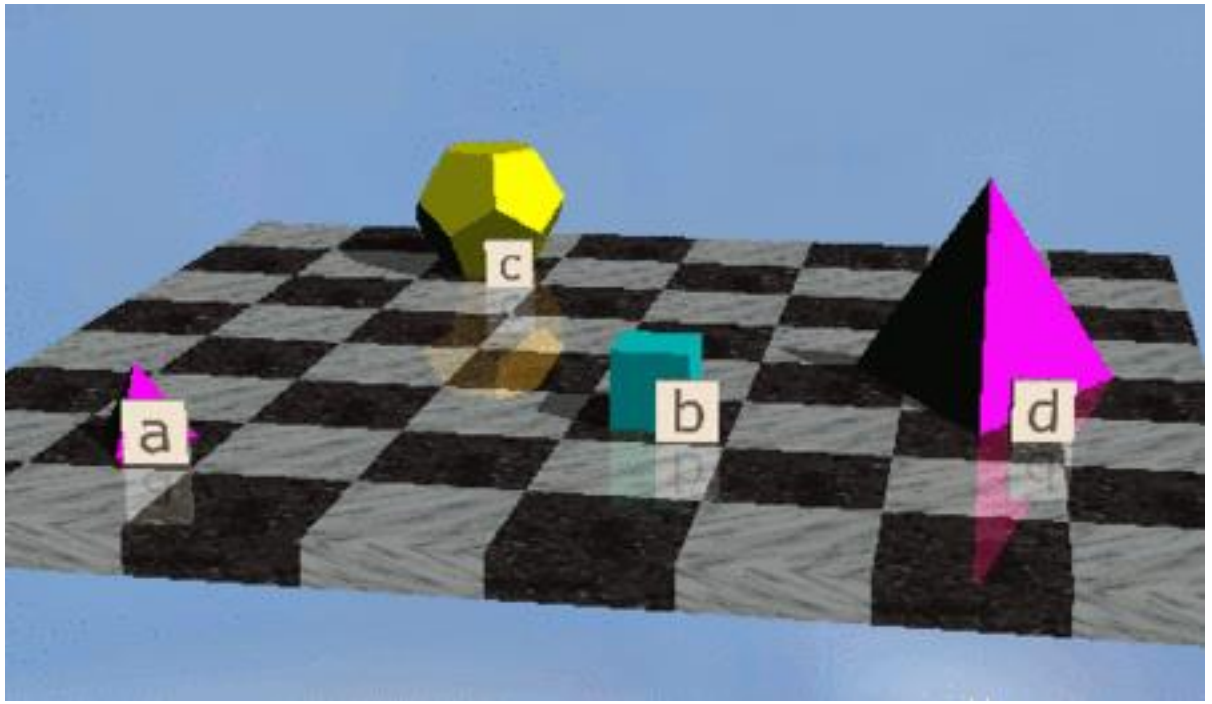
40417: Artificial Intelligence

First-Order Logic

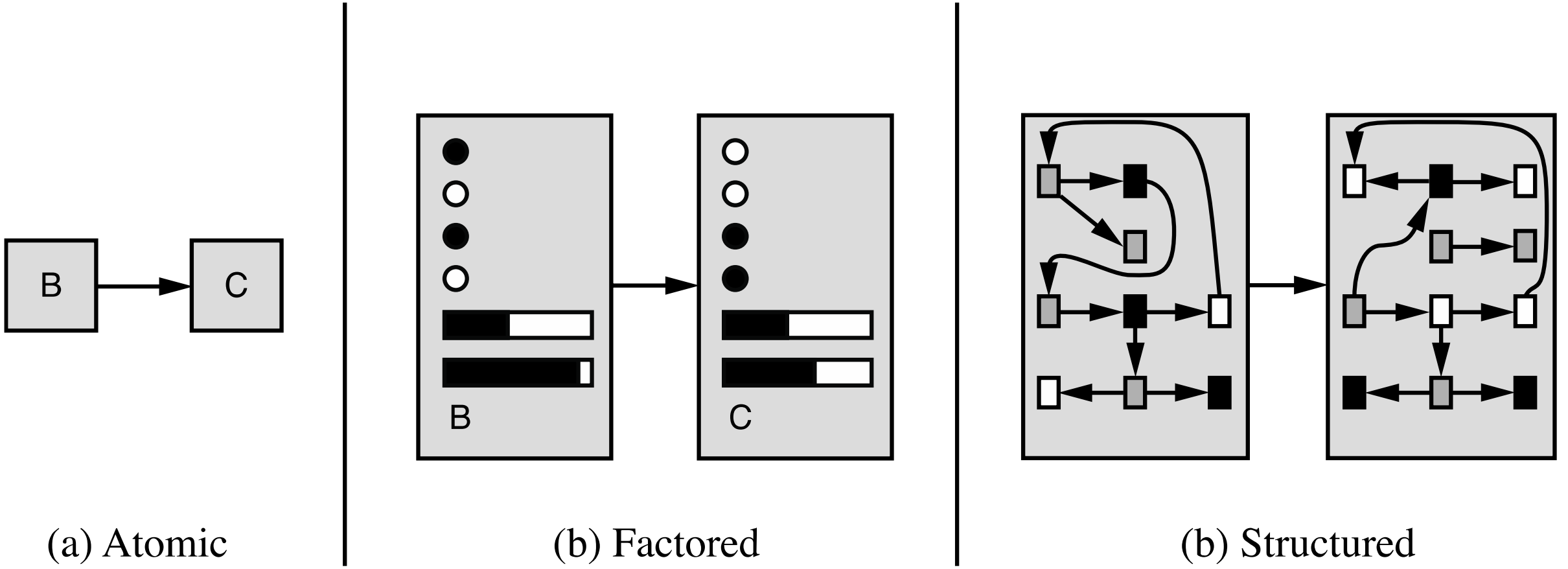


Instructor: Gholamreza Ghassem-Sani

Sharif University of Technology

[These slides were created by Stuart Russell and Dawn Song for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Spectrum of representations



Search, game-playing

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

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

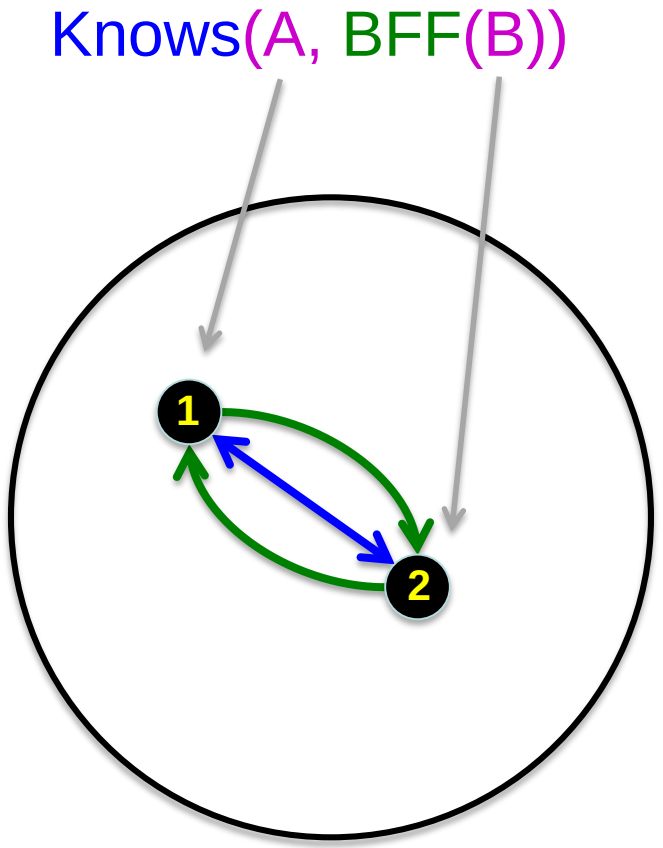
Expressive power

- Rules of chess:
  - 100,000 pages in propositional logic
  - 1 page in first-order logic
- Rules of Pacman:
  - $\forall t\ \text{Alive}(t) \Leftrightarrow$

    $[\text{Alive}(t\text{-}1) \wedge \neg \exists\ g,x,y\ [\text{Ghost}(g) \wedge \text{At}(\text{Pacman},x,y,t\text{-}1) \wedge \text{At}(g,x,y,t\text{-}1)]]$

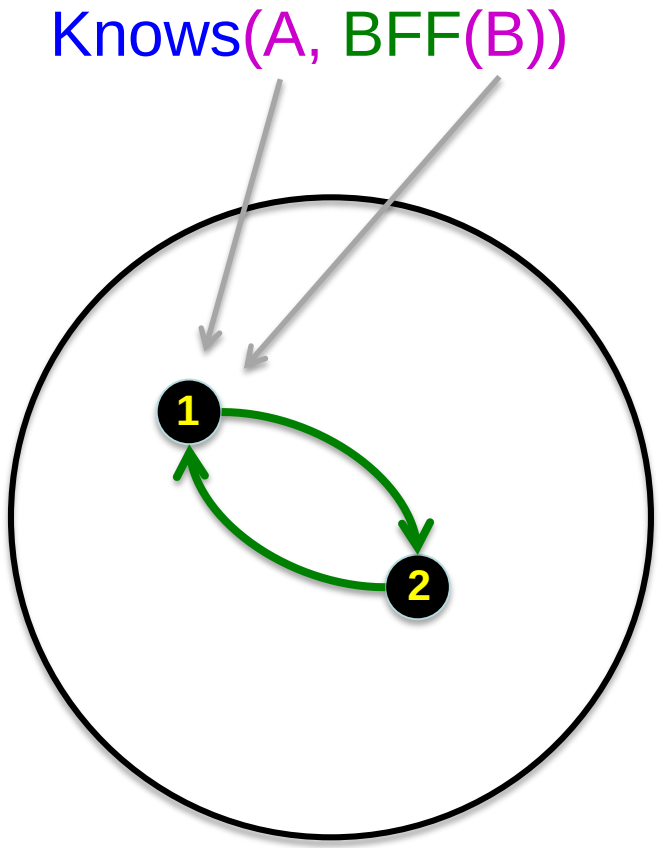
Possible worlds

- 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)



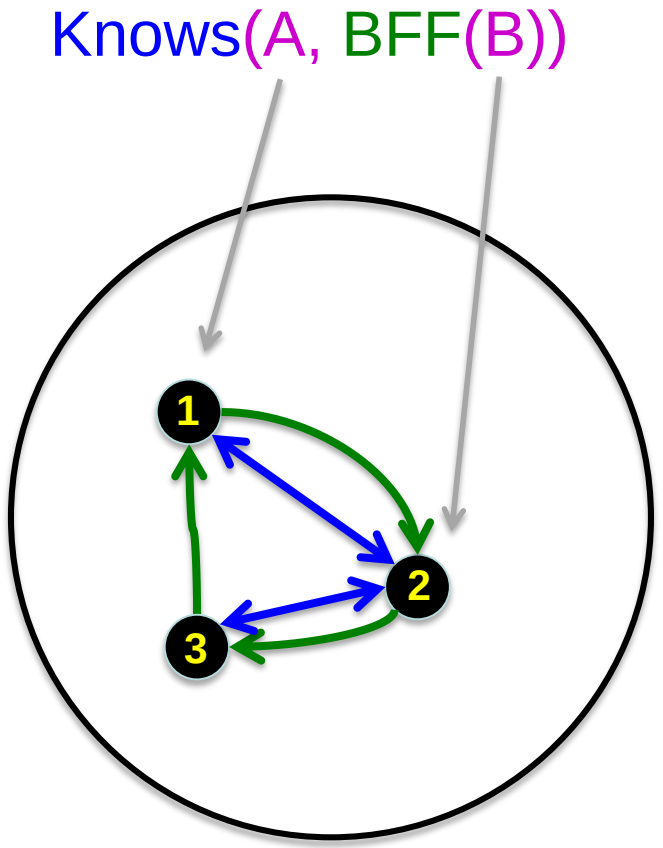
Possible worlds

- 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)



Possible worlds

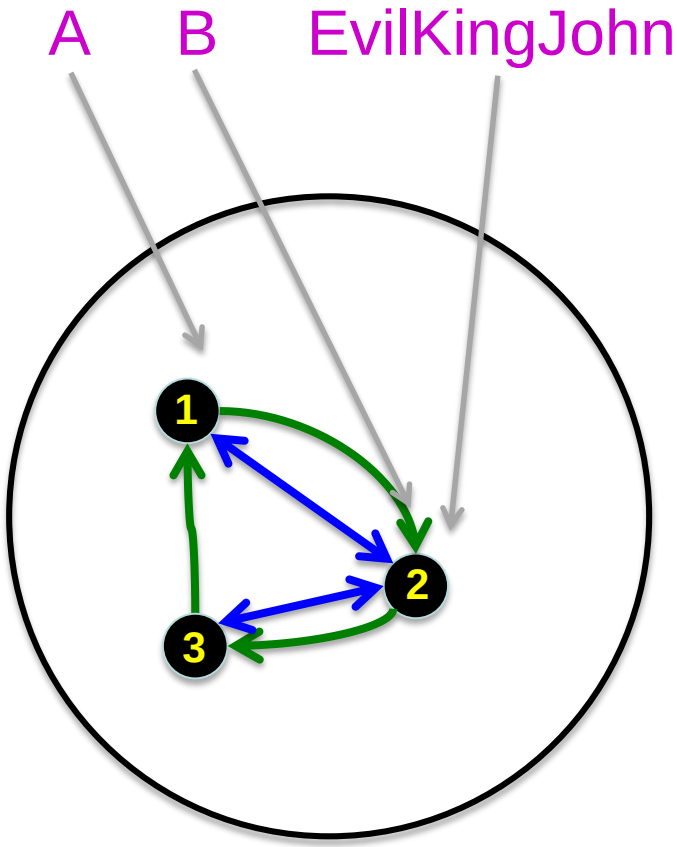
- A possible world for FOL consists of:
  - A non-empty set of objects
  - For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
  - For each k-ary function in the language, a mapping from k-tuples of objects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



How many possible worlds?

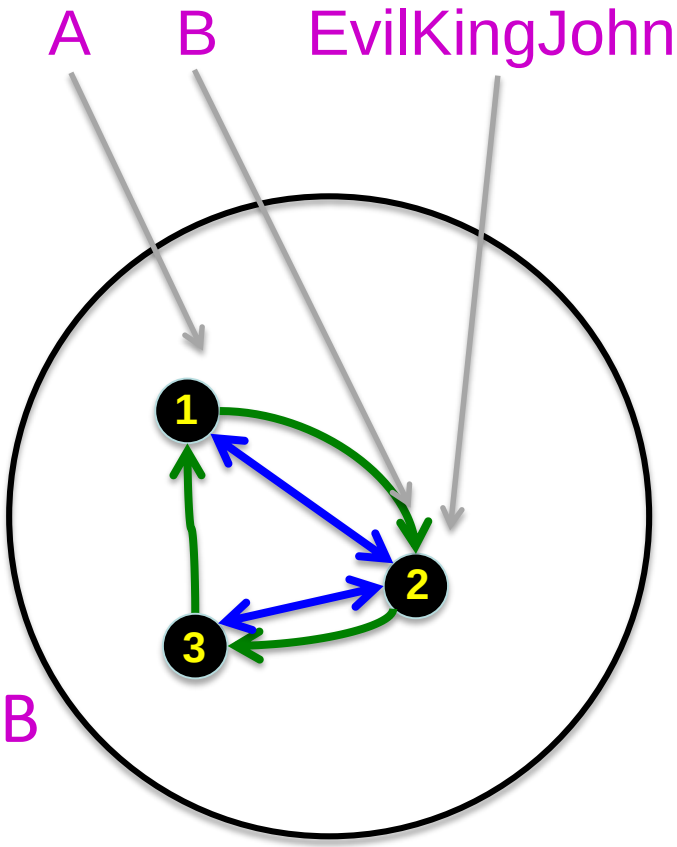
Syntax and semantics: Terms

- 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
      - BFF(EvilKingJohn) -> BFF(**2**) -> **3**
  - A logical variable, e.g., x
    - (more later)



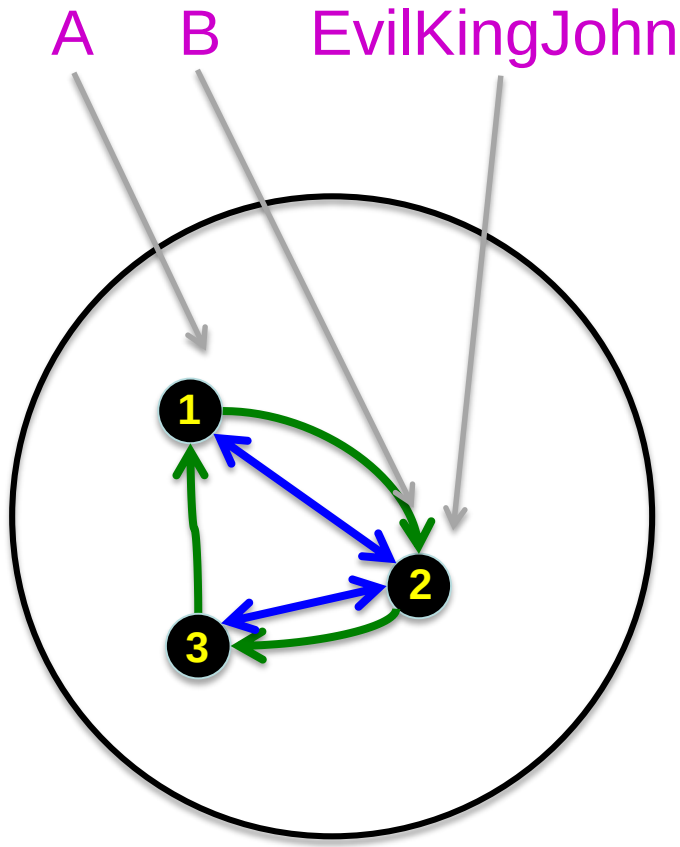
Syntax and semantics: Atomic sentences

- An atomic sentence is an elementary proposition (cf symbols in PL)
  - A predicate symbol with terms as arguments, 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**



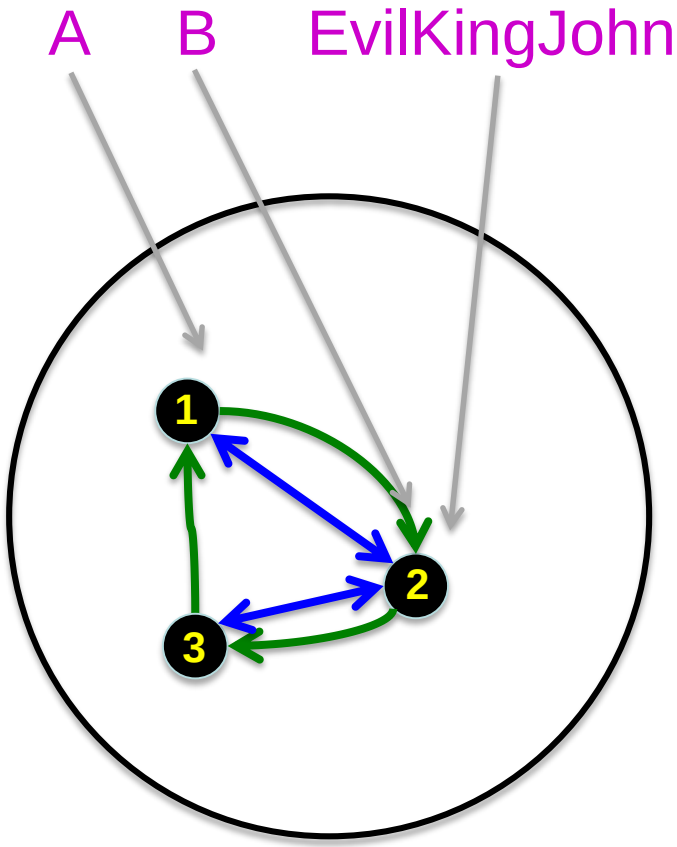
Syntax and semantics: Complex sentences

- 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.,
  - $\forall$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.,
  - $\exists$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**



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
  - $\forall x\ \text{Person}(x) \Rightarrow \exists y\ \text{Person}(y) \wedge \text{Knows}(x,y)$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
  - $\forall$n Person(n) $\Rightarrow$ Knows(n,Obama)
  - Everyone (every person) knows President Obama

- Common mistake: using $\wedge$ as the main connective with $\forall$:
  - $\forall$n Person(n) $\wedge$ Knows(n,Obama)
  - Everyone is a Person and everyone knows President Obama

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
  - $\exists n$ Person(n) $\wedge$ Knows(n,Obama)
  - There is someone (some person) who knows President Obama

- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:
  - $\exists n$ Person(n) $\Rightarrow$ Knows(n,Obama)
  - If there is someone (some person) he/she knows President Obama
  - is true even if there is not any person!

More fun with sentences

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

Properties of quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$

- But, $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
  - $\exists x\ \forall y$ Loves(x,y) means "There is a person who loves everyone"
  - $\forall y\ \exists x$ Loves(x,y) means "Everyone is loved by at least one person"

- Quantifier duality: each can be expressed using the other
  - $\forall x$ Likes(x,IceCream) is equal to $\neg\exists x\ \neg$Likes(x,IceCream)
  - $\exists x$ Likes(x,IceCream) is equal to $\neg\forall x\ \neg$Likes(x,IceCream)

Using FOL - example

- The set domain:

1. $\forall s\ \mathit{Set}(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2 \quad \mathit{Set}(s_2) \wedge s = \mathit{Add}(x, s_2))$
2. $\neg\exists x, s \quad \mathit{Add}(x, s) = \{\}$
3. $\forall x, s\ x \in s \Leftrightarrow s = \mathit{Add}(x, s)$
4. $\forall x, s\ x \in s \Leftrightarrow \exists y, s_2\ (s = \mathit{Add}(y, s_2) \wedge (x = y \vee x \in s_2))$
5. $\forall s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
6. $\forall s_1, s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
7. $\forall x, s_1, s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
8. $\forall x, s_1, s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

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., $\forall x$ Knows(x,Obama) entails $\exists y \forall x$ 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 = $\forall x$ Knows(x,Obama)
  - Query = $\exists y \forall x$ Knows(x,y)
  - Answer = Yes, σ = {y/Obama}
  - Notation: $\alpha\sigma$ means applying substitution σ to sentence α
    - E.g., if α= $\forall x$ Knows(x,y) and σ = {y/Obama}, then $\alpha\sigma$ = $\forall x$ Knows(x,Obama)

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
    - $\forall x$ Knows(x,Obama) and Democrat(Feinstein)
      - Knows(Obama,Obama) and Knows(Feinstein,Obama) and Democrat(Feinstein)
      - Knows_Obama_Obama $\wedge$ Knows_Feinstein_Obama $\wedge$ Democrat_Feinstein
    - and $\forall x$ Knows(Mother(x),x)
      - Knows(Mother(Obama),Obama) and Knows(Mother(Mother(Obama)),Mother(Obama)) …….
  - 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), $\forall x$ Person(x) $\Rightarrow$ 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

Unification

- Can we find a substitution σ such that Knows(x, BFF(y)) match Knows(A, BFF(B))?
  - σ = {x/A, y/B} is such a substitution
  - Unify(α ,β) = σ if $\alpha\sigma = \beta\sigma$

α	β	σ
Knows(John, x)	Knows(John, Jane)	{x/Jane}
Knows(John, x)	Knows(y, OJ)	{x/OJ, y/John}
Knows(John, x)	Knows(y, Mother(y))	{y/John, x/Mother(John)}
Knows(John, x)	Knows(x, OJ)	{fail}
Knows(John, x)	Knows(z_{17}, OJ)	{z_{17}/John, x/OJ)}

- *Standardizing apart* eliminates overlap of variables, e.g., Knows(z_{17}, OJ)

Unification

- To unify Knows(John, x) and Knows(y, z),
  - σ = {y/John, x/z } or σ = {y/John, x/John, z/John}
- The first unifier is more general than the second.
- There is a single Most General Unifier (MGU) that is unique up to renaming of variables.
  - MGU = { y/John, x/z } and { y/John, z/x }

Unification algorithm

```
function UNIFY(x, y, θ=empty) returns a substitution to make x and y identical, or failure
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
      return UNIFY(ARGS(x), ARGS(y), UNIFY(OP(x), OP(y), θ))
  else if LIST?(x) and LIST?(y) then
      return UNIFY(REST(x), REST(y), UNIFY(FIRST(x), FIRST(y), θ))
  else return failure

function UNIFY-VAR(var, x, θ) returns a substitution
  if {var/val} ∈ θ for some val then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ for some val then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

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)