CS 188: Artificial Intelligence

Propositional Logic II (cont.) + First order Logic

Any questions about previous logic lectures?

Slides mostly from Stuart Russell

University of California, Berkeley

Pacman's knowledge base: Transition model

How does each ***state variable*** at each time gets its value?

- Here we care about location variables, e.g., At_3,3_17

A state variable X gets its value according to a ***successor-state axiom***

- X_t ⇔ [X_t-1 ∧ ¬(some action_t-1 made it false)] v
  [¬X_t-1 ∧ (some action_t-1 made it true)]

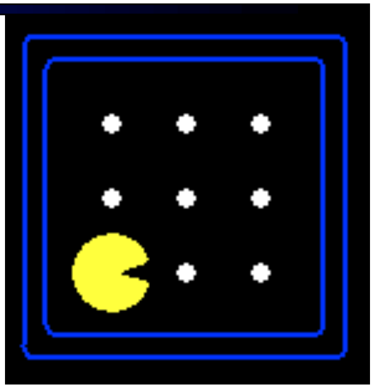
For Pacman location:

- At_3,3_17 ⇔ [At_3,3_16 ∧ ¬((¬Wall_3,4 ∧ N_16) v (¬Wall_4,3 ∧ E_16) v …)]
  v [¬At_3,3_16 ∧ ((At_3,2_16 ∧ ¬Wall_3,3 ∧ N_16) v
  (At_2,3_16 ∧ ¬Wall_3,3 ∧ E_16) v …)]

Food_3,3_17 ⇔ ***??***

Lec 7, Slide 20

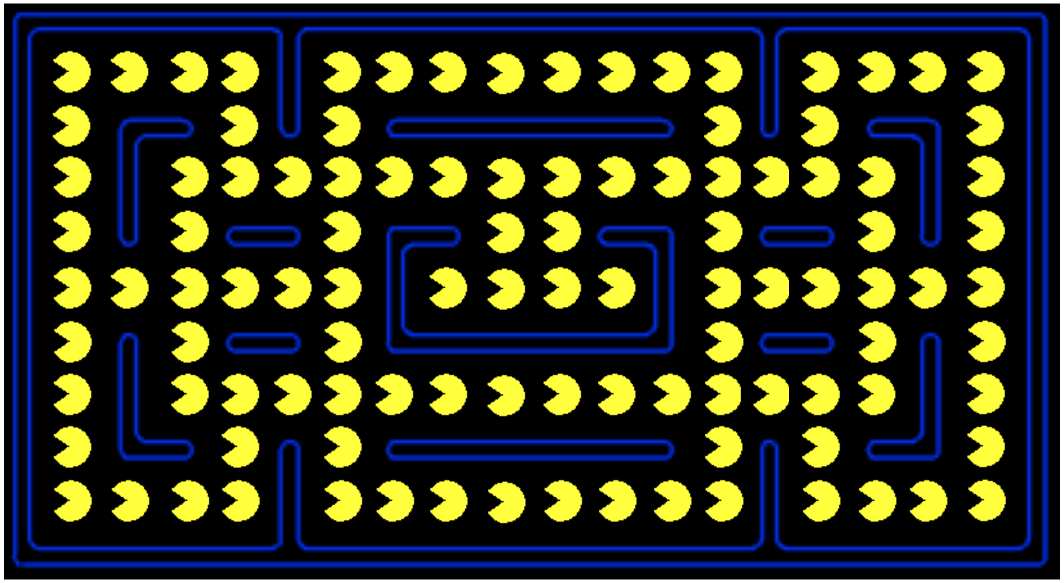
Reminder: Partially observable Pacman



- Basic question: where am I?
- Variables:
  - Wall_0,0, Wall_0,1, …
  - Blocked_W_0, Blocked_N_0, …, Blocked_W_1, …
  - W_0 , N_0, …, W_1, …
  - At_0,0_0 , At_0,1_0, …, At_0,0_1 , …
- Sensor model:
  - Blocked_W_0 $\Leftrightarrow$ ((At_1,1_0 $\wedge$ Wall_0,1) v
    (At_1,2_0 $\wedge$ Wall_0,2) v
    (At_1,3_0 $\wedge$ Wall_0,3) v ….)
- Map: where are the walls
- Initial state: Pacman definitely somewhere
- Domain constraints: e.g. only one action per timestep
- Transition model: how state variables change (or don't)

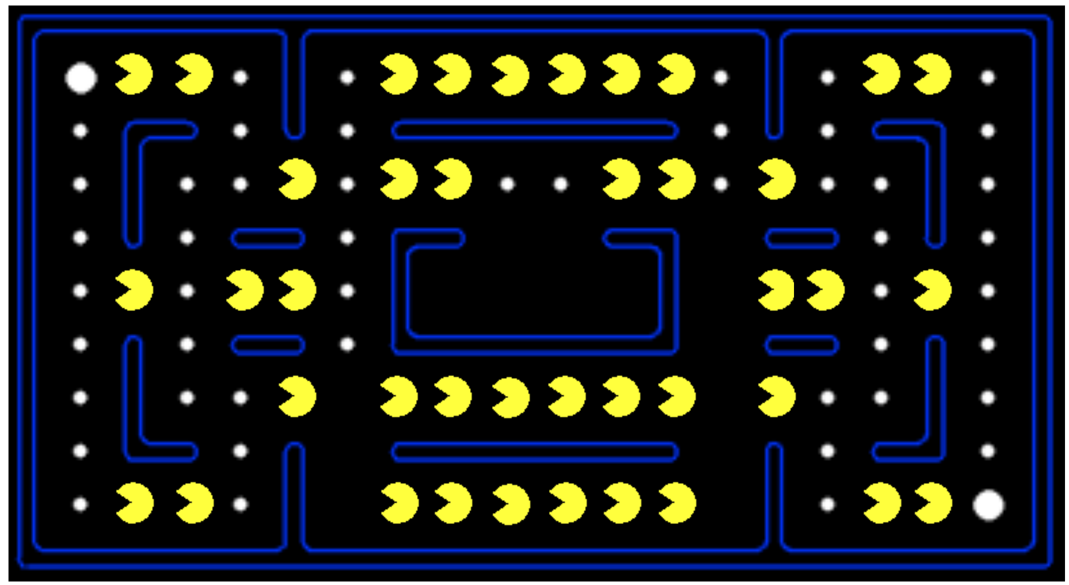
Localization demo

- Percept 
- Action
- Percept
- Action
- Percept
- Action
- Percept

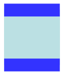


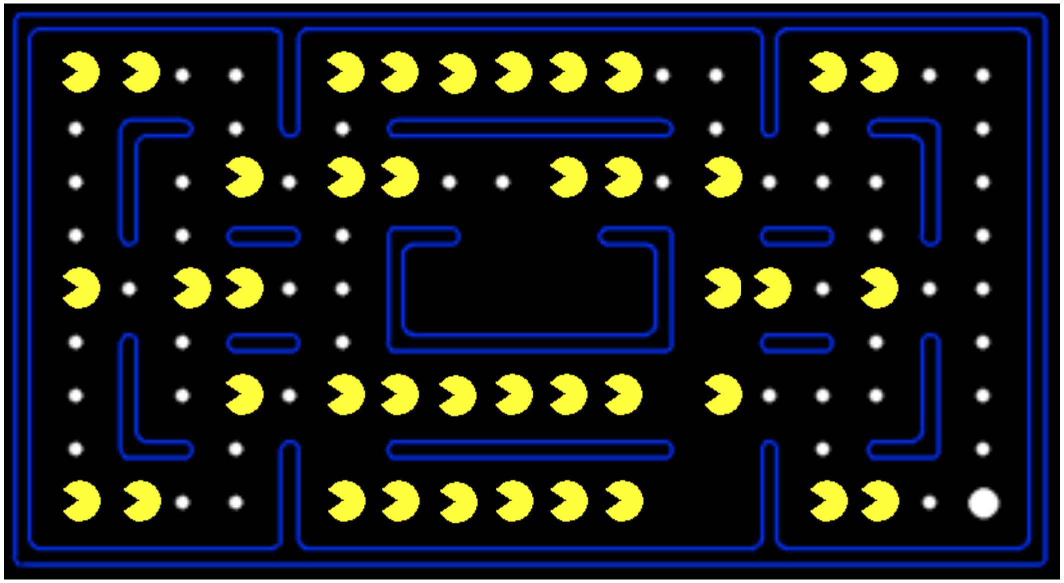
Localization demo

- Percept 
- Action *WEST*
- Percept
- Action
- Percept
- Action
- Percept

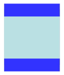


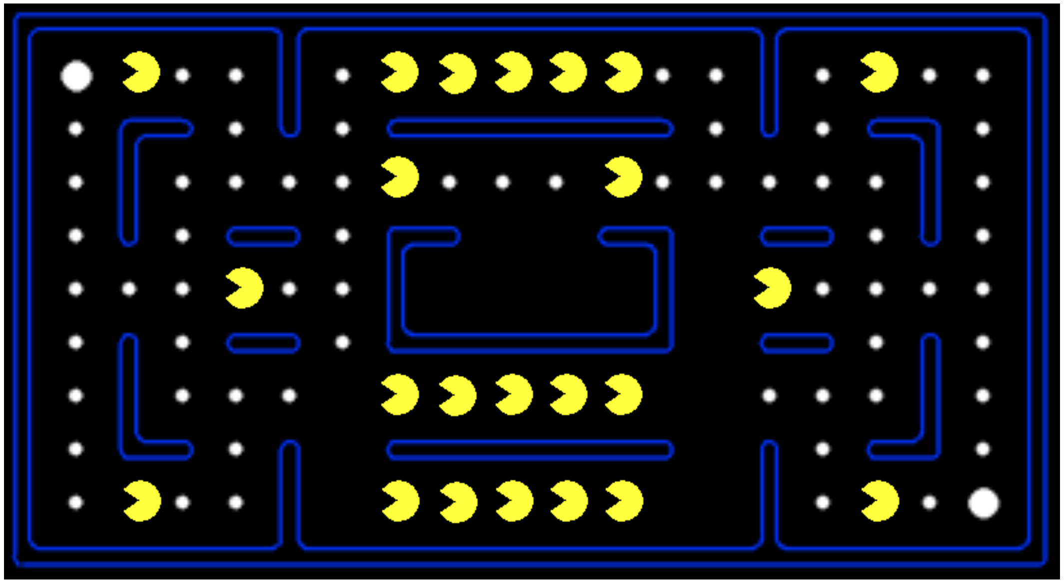
Localization demo

- Percept 
- Action *WEST*
- Percept 
- Action
- Percept
- Action
- Percept



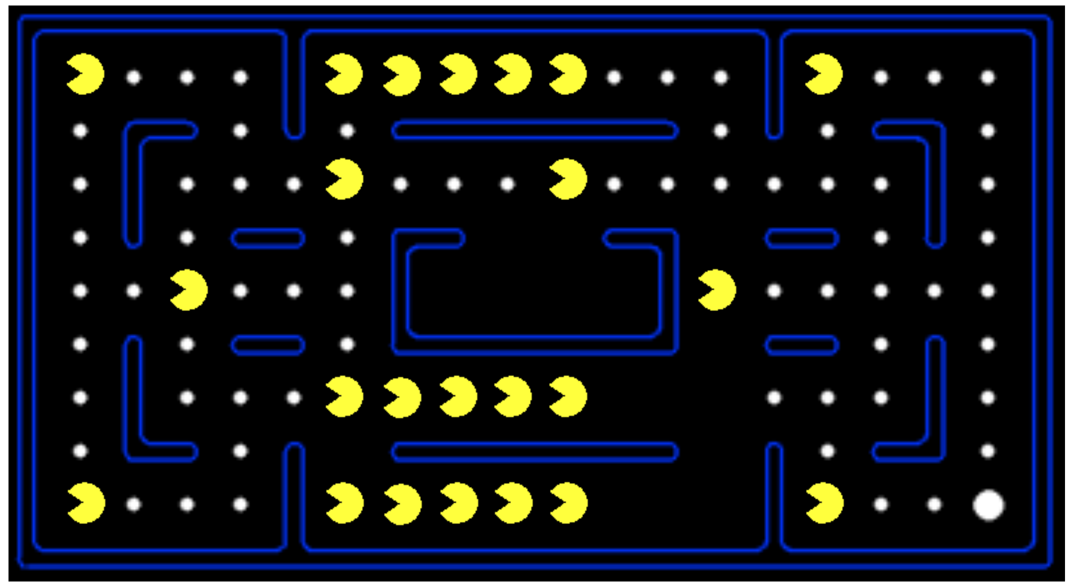
Localization demo

- Percept 
- Action *WEST*
- Percept 
- Action *WEST*
- Percept
- Action
- Percept



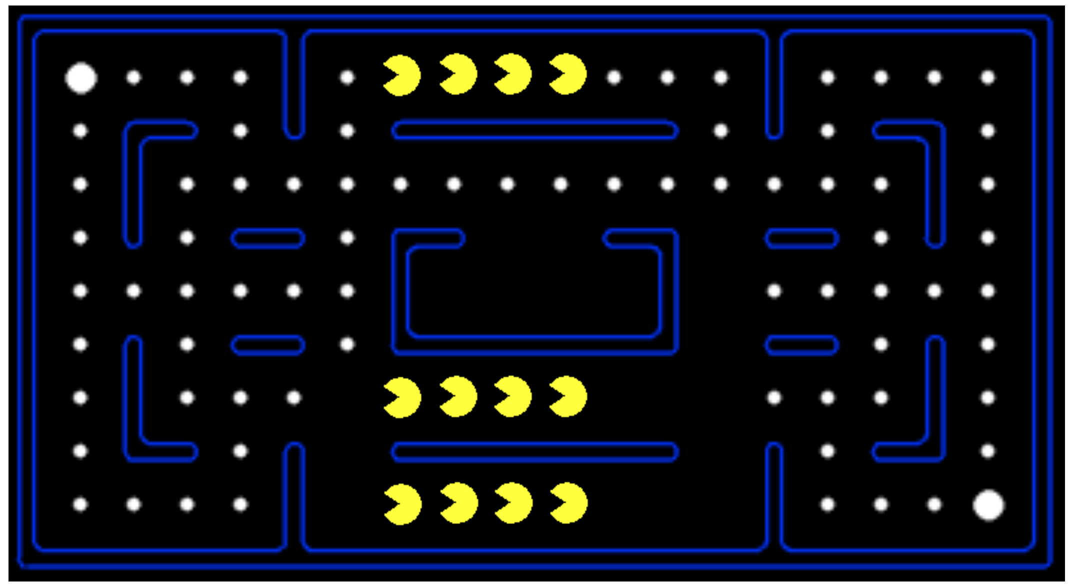
Localization demo

- Percept 
- Action *WEST*
- Percept 
- Action *WEST*
- Percept 
- Action
- Percept



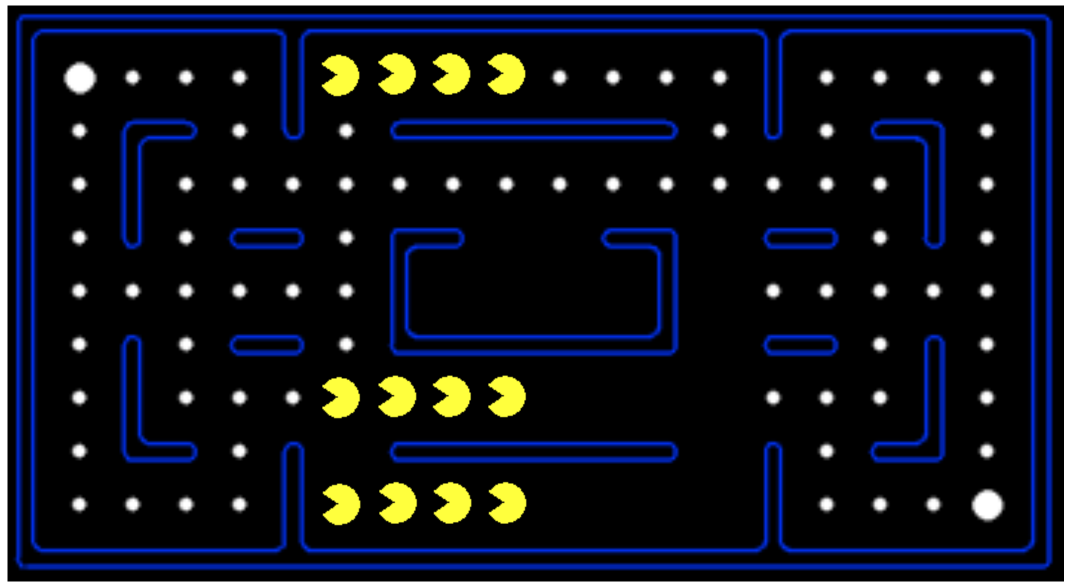
Localization demo

- Percept 
- Action *WEST*
- Percept 
- Action *WEST*
- Percept 
- Action *WEST*
- Percept



Localization demo

- Percept 
- Action *WEST*
- Percept 
- Action *WEST*
- Percept 
- Action *WEST*
- Percept



Example: Mapping from a known relative location

- Without loss of generality, call the initial location 0,0
- The percept tells Pacman which actions work, so he always knows where he is
  - "Dead reckoning"
- Initialize the KB with **PacPhysics** for *T* time steps, starting at 0,0
- Run the Pacman agent for *T* time steps
  - At each time step
    - Update the KB with previous action and new percept facts
    - For each wall variable Wall_x,y
      - If Wall_x,y is entailed, add to KB
      - If ¬Wall_x,y is entailed, add to KB
    - Choose an action
- The wall variables constitute the map

Mapping demo

- Percept 
- Action *NORTH*
- Percept 
- Action *EAST*
- Percept 
- Action *SOUTH*
- Percept 

Example: Simultaneous localization and mapping

- Often, dead reckoning won't work in the real world
  - E.g., sensors just count the ***number*** of adjacent walls (0,1,2,3 = 2 bits)
- Pacman doesn't know which actions work, so he's "lost"
  - So if he doesn't know where he is, how does he build a map???
- Initialize the KB with **PacPhysics** for *T* time steps, starting at 0,0
- Run the Pacman agent for *T* time steps
  - At each time step
    - Update the KB with previous action and new percept facts
    - For each x,y, add either Wall_x,y or ¬Wall_x,y to KB, if entailed
    - For each x,y, add either At_x,y_t or ¬At_x,y_t to KB, if entailed
    - Choose an action

Resolution (briefly)

clauses

Reminder of conjunctive normal form: $(A \vee B) \wedge (A \vee \neg C \vee D) \wedge (C \vee \neg B) \wedge (B)$

- Every CNF clause can be written as
  - Conjunction of symbols $\Rightarrow$ disjunction of symbols
  - $A \vee B \vee \neg C \vee \neg D \quad = \quad C \wedge D \Rightarrow A \vee B$
- The resolution inference rule takes two such clauses and infers a new one by **resolving** complementary symbols:
- Example: $A \wedge B \wedge C \Rightarrow \mathbf{U} \vee V$

$$\frac{\begin{array}{l} A \wedge B \wedge C \Rightarrow \mathbf{U} \vee V \\ D \wedge E \wedge \mathbf{U} \Rightarrow X \vee Y \end{array}}{A \wedge B \wedge C \wedge D \wedge E \Rightarrow V \vee X \vee Y}$$

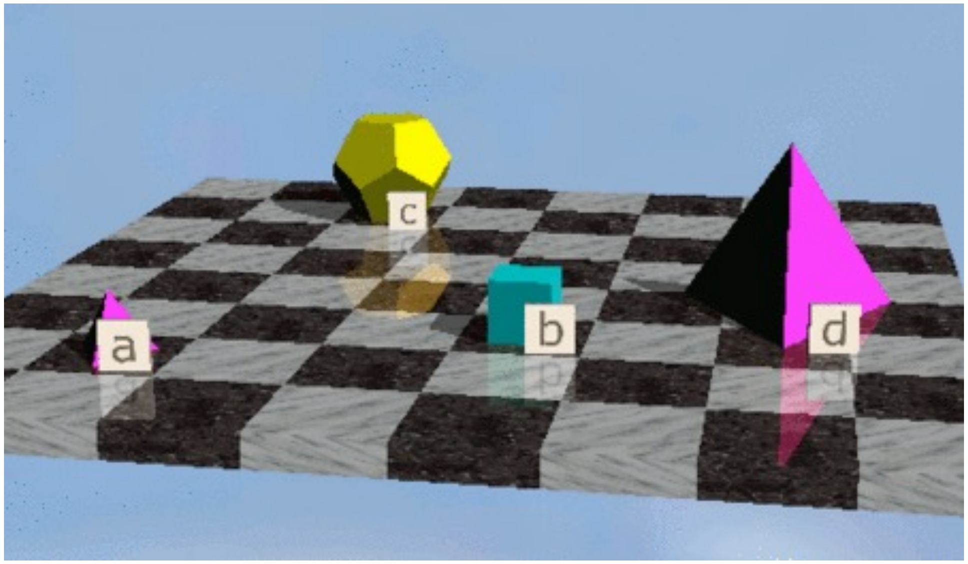
- Sentence unsatistfiable iff repeated resolution produces $() \Rightarrow ()$
- Resolution is complete for propositional logic, but exp-time

Summary

- Logical inference computes entailment relations among sentences
- Theorem provers apply inference rules to sentences
  - Forward chaining applies modus ponens with definite clauses; linear time
  - Resolution is complete for PL but exponential time in the worst case
- SAT solvers based on DPLL provide incredibly efficient inference
- Logical agents can do localization, mapping, SLAM, planning (and many other things) just using one generic inference algorithm on one knowledge base

CS 188: Artificial Intelligence

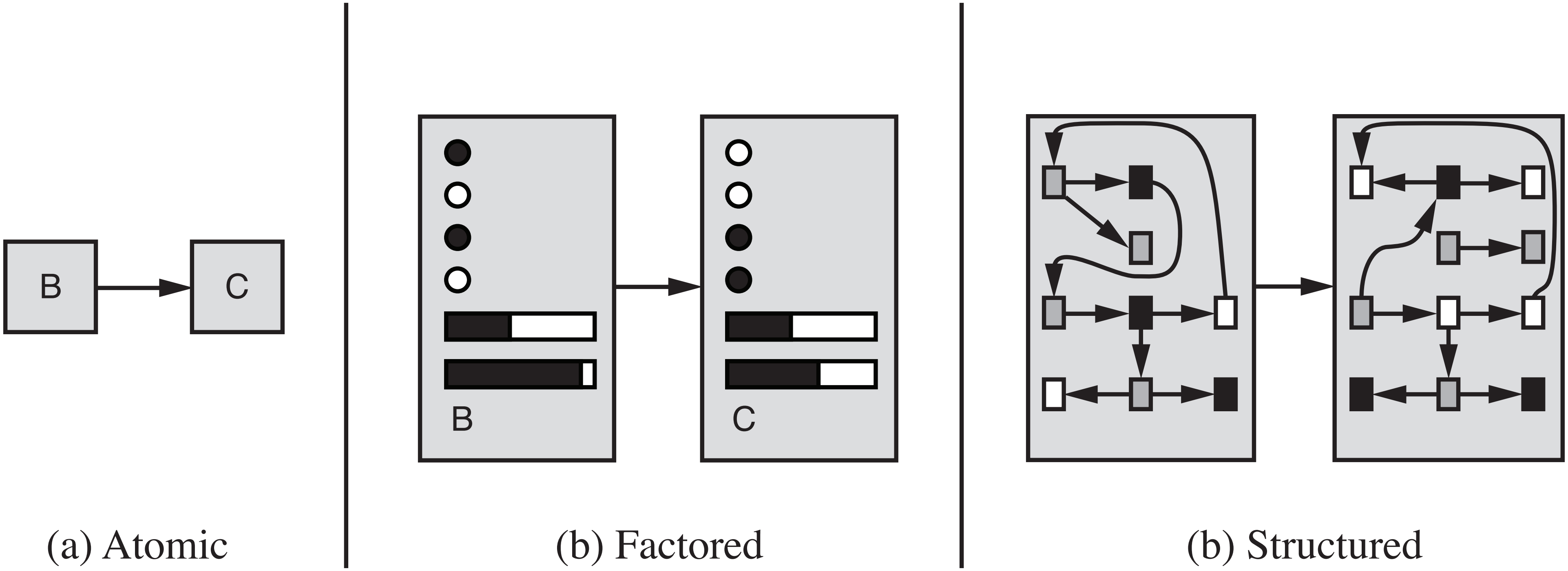
First-Order Logic



Slides mostly from Stuart Russell

University of California, Berkeley

Spectrum of representations



Search, game-playing

Planning, propositional logic, Bayes nets

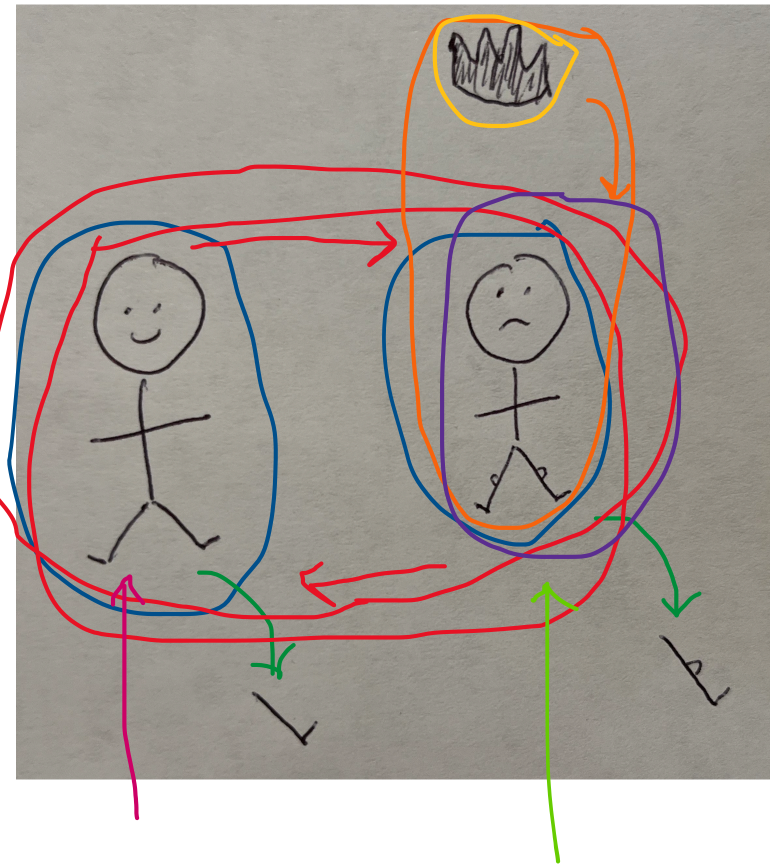
First-order logic, databases, logic programs, probabilistic programs

Expressive power

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

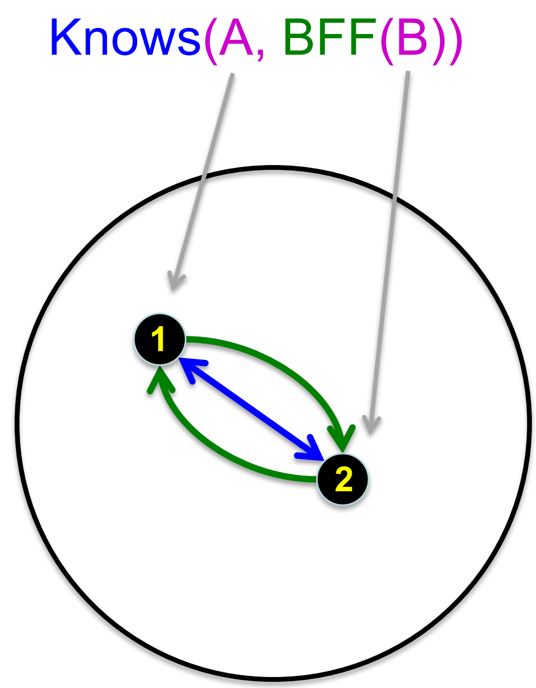
Possible worlds

- A possible world of five objects:
  - "left leg" unary function (arity is # arguments)
  - "on head" binary relation
  - "brother" binary relation
  - "person" unary relation
  - "king" unary relation
  - "crown" unary relation
  - "John" constant (0-ary function)
  - "Richard" constant (0-ary function)
- If a function/relation/constant is mentioned
- World must have object(s) plus definitions of those functions/relations/constants



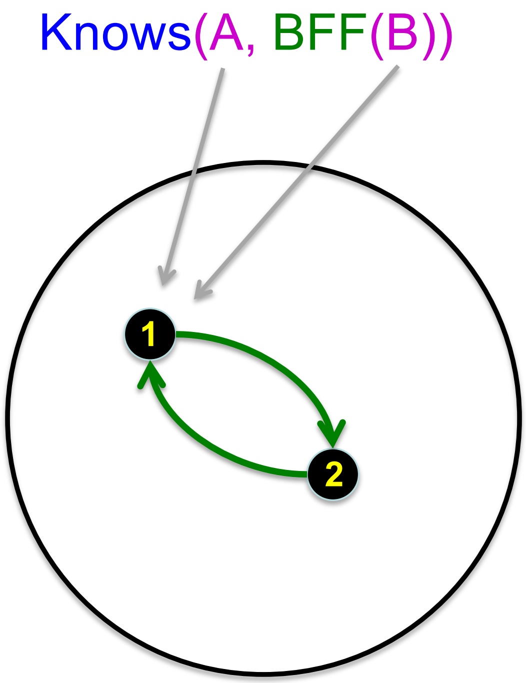
Possible worlds

- 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)

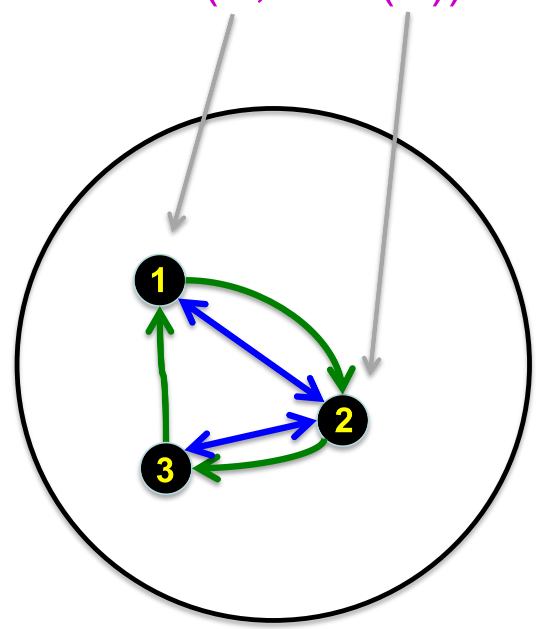


Possible worlds

- 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)

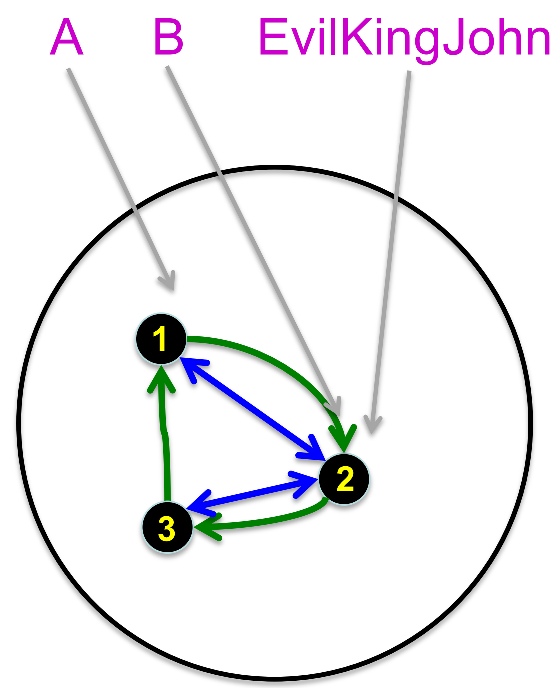
How many possible worlds?

Knows(A, BFF(B))



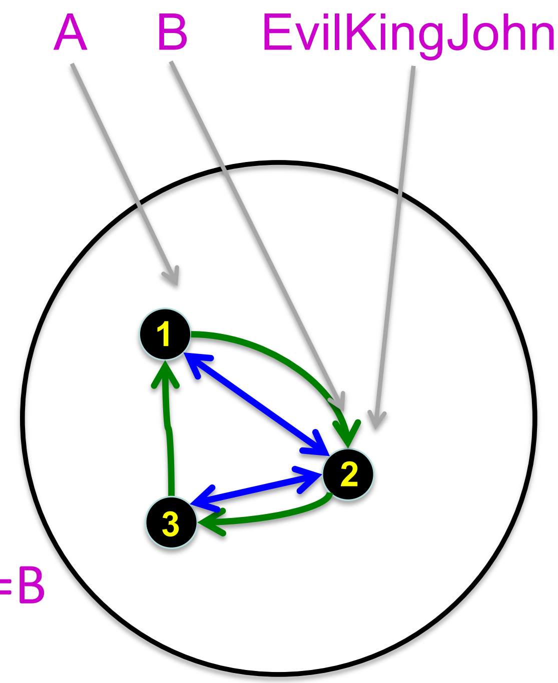
Syntax and semantics: Terms

- A term is something that 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)



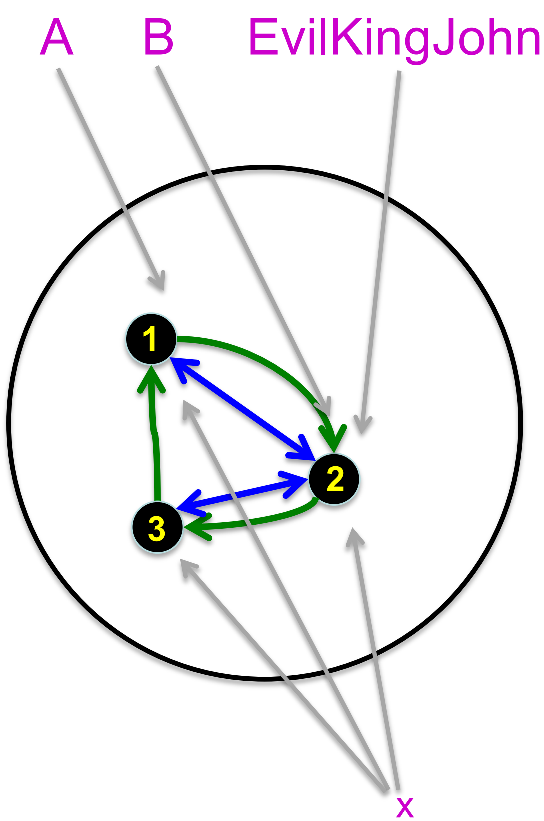
Syntax and semantics: Atomic sentences

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



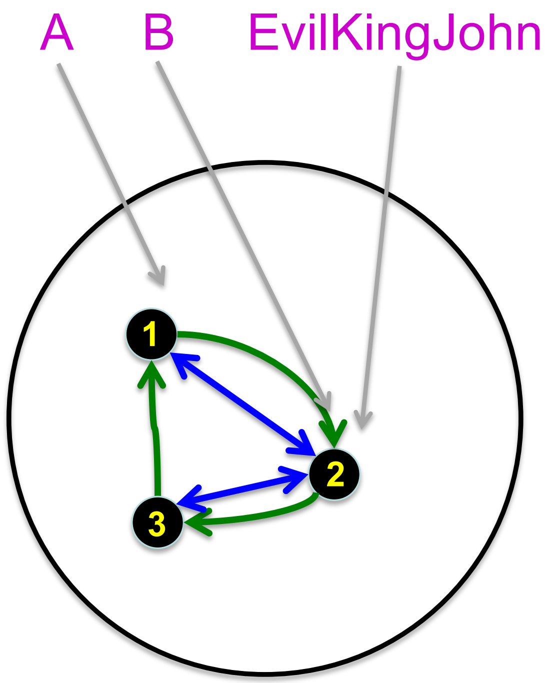
Syntax and semantics: Complex sentences

- 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.,
  - ∃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
  - $\forall n\ Person(n) \Rightarrow Knows(n,Obama)$
- There is someone that nobody else knows
  - $\exists s\ Person(s) \wedge \forall n\ (Person(n) \wedge \neg(n = s)) \Rightarrow \neg Knows(n,s)$
- Everyone knows someone
  - $\forall x\ Person(x) \Rightarrow \exists y\ Person(y) \wedge Knows(x,y)$
  - $\forall x\ (Person(x) \Rightarrow \exists y\ (Person(y) \wedge Knows(x,y)))$

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
  - $\forall x,y\ [(\exists\ n\ \text{Nationality}(x,n) \wedge \text{Nationality}(y,n)) \Rightarrow$

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

Conciseness of first order logic

- Pacman can't be in two places at once
  - FOL: $\forall\, x_1, y_1, x_2, y_2, t\ (At(x_1, y_1, t) \wedge At(x_2, y_2, t)) \Rightarrow (x_1 = x_2 \wedge y_1 = y_2)$
  - PL: ¬ (At_1,1_0 ∧ At_1,2_0) ∧ ¬ (At_1,1_0 ∧ At_1,3_0) ∧ ¬ (At_1,1_0 ∧ At_2,1_0) ∧ ¬ (At_1,1_0 ∧ At_2,2_0) ∧ ¬ (At_1,1_0 ∧ At_2,3_0) ∧ ¬ (At_1,1_0 ∧ At_3,1_0) ∧ ¬ (At_1,1_0 ∧ At_3,2_0) ∧ ¬ (At_1,1_0 ∧ At_3,3_0) ∧ …
  - And that's just if he's in the bottom left at the first timestep

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, $\sigma = \{y/Obama\}$
  - Notation: $\alpha\sigma$ means applying substitution σ to sentence α
    - E.g., if $\alpha = \forall x\ Knows(x,y)$ and $\sigma = \{y/Obama\}$, 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
    - $\exists$x Knows(x,Obama)
      - Knows(X_1,Obama)
      - Knows_X1_Obama
    - $\forall$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
    - $\forall$x Knows(Mother(x),x)
      - Knows(Mother(Obama),Obama) and Knows(Mother(Mother(Obama)),Mother(Obama)) …….
  - Real trick: for *k* = 1 to infinity:
    - Get a set of terms: constants, functions of constants, funcs of funcs of constants, … up to depth *k*
    - Propositionalize as if those are all the terms that exist
    - If a contradiction is found, halt; otherwise, continue
  - If FOL sentence is unsatisfiable, 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 $\forall$x Knows(x,Obama) and $\forall$y, z 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)