RESOLUTION AND FIRST ORDER LOGIC

Introduction

- Last week we talked about logic.
- In particular we talked about why logic would be useful.
- We covered propositional logic — the simplest kind of logic.
- We talked about proof using the rules of natural deduction.
- This week we will look at some other aspects of proof.
- We will also look at a more expressive kind of logic.

More on proof

- One of the good things about natural deduction is that it is easy to understand.
  - Proofs are often intuitive
- However, there is lots to decide:
  - Which sentence to use
  - Which rule to apply
- Can be hard to program a system to use it.
- Q: How to make it easier?

Horn clauses

- A: Restrict the language
  - *Horn clauses*
- A Horn clause is:
  - An atomic proposition; or
  - A conjunction of atomic propositions $\Rightarrow$ atomic proposition
- For example:

$$C \wedge D \Rightarrow B$$

- KB = *conjunction* of *Horn clauses*
- For example:

$$C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$$

- Modus ponens is then:

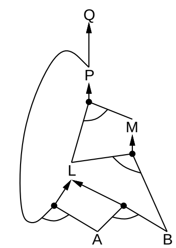
$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \;\Rightarrow\; \beta}{\beta}$$

- For Horn clauses, modus ponens is all you need
  - Complete
- Can be used with *forward chaining* or *backward chaining*.
- These algorithms are very natural and run in *linear* time

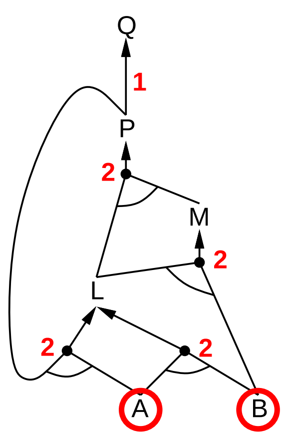
Forward chaining

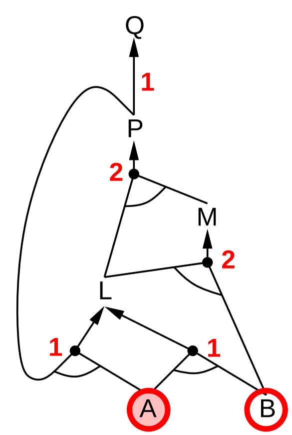
- Idea: fire any rule whose premises are satisfied in the *KB*, add its conclusion to the *KB*, until query is found

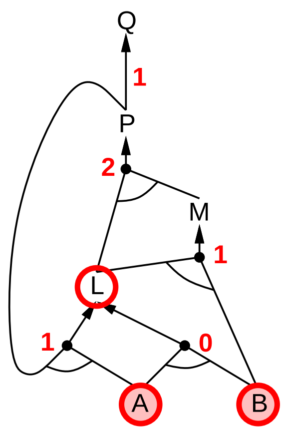
$P \;\Rightarrow\; Q$
$L \wedge M \;\Rightarrow\; P$
$B \wedge L \;\Rightarrow\; M$
$A \wedge P \;\Rightarrow\; L$
$A \wedge B \;\Rightarrow\; L$
A
B

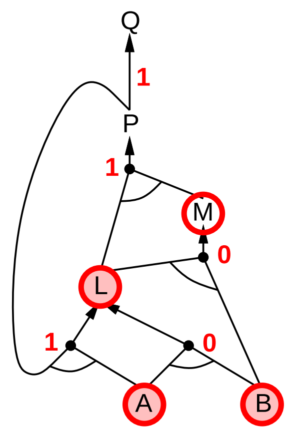


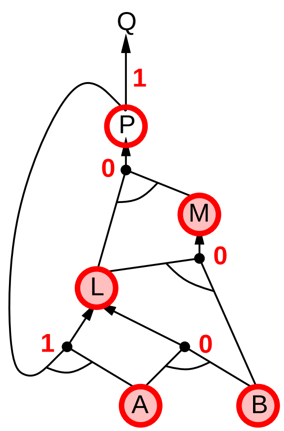
- How does this work?

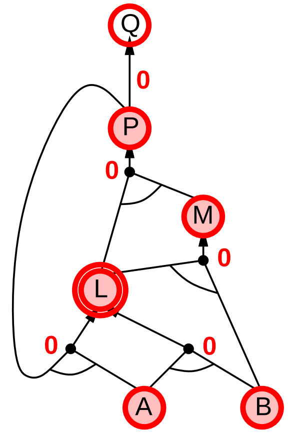


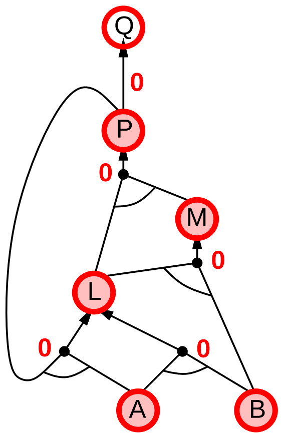


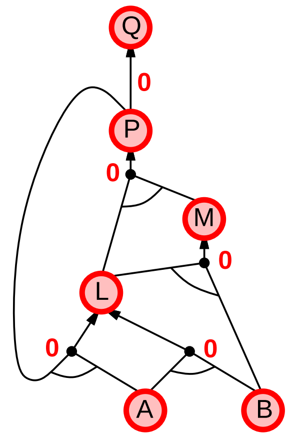

Q
1
P
2
M
1
L
1
0
A
B


Q
1
P
1
M
0
L
1
0
A
B


Q
1
P
0
M
0
L
1
0
A
B


Q
0
P
0
M
0
L
0
0
A
B





function PL-FC-ENTAILS?(*KB*, *q*) **returns** *true* or *false*
  inputs: *KB*, the knowledge base, *q* the query
  local variables: *count*, a table, indexed by clause,
      initially the number of premises
    inferred, table of symbols, initially all *false*
    agenda, list of symbols, initially whole *KB*

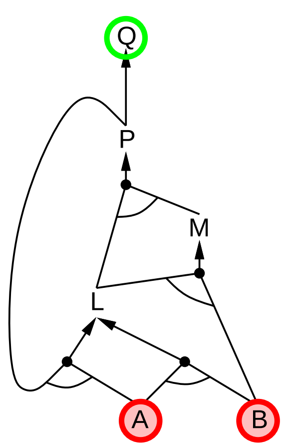
  while *agenda* is not empty **do**
    p ← POP(*agenda*)
    unless *inferred*[*p*] **do**
      inferred[*p*] ← *true*
      for each Horn clause *c* in whose premise *p* appears **do**
        decrement *count*[*c*]
        if *count*[*c*] = 0 **then do**
          if HEAD[*c*] = *q* **then return** *true*
          PUSH(HEAD[*c*], *agenda*)
  return *false*

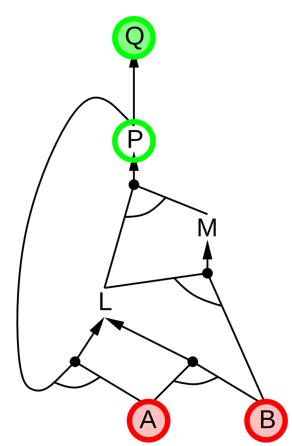
Proof of completeness

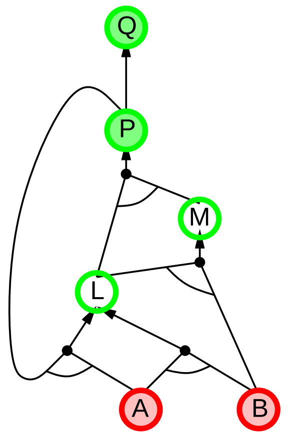
- FC derives every atomic sentence that is entailed by *KB*
  1. FC reaches a *fixed point* where no new atomic sentences are derived
  2. Consider the final state as a model m, assigning true/false to symbols
  3. Every clause in the original *KB* is true in m
     Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in m
     Then $a_1 \wedge \ldots \wedge a_k$ is true in m and b is false in m
     Therefore the algorithm has not reached a fixed point!
  4. Hence m is a model of *KB*
  5. If $KB \models q$, q is true in *every* model of *KB*, including m
- *General idea*: construct any model of *KB* by sound inference, check α

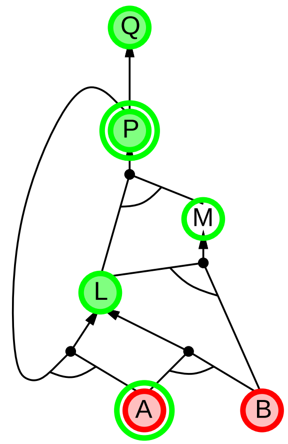
Backward chaining

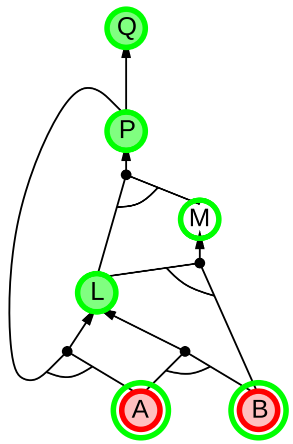
- Idea: work backwards from the query q
  - to prove q by BC,
  - check if q is known already, or
  - prove by BC all premises of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  1. has already been proved true, or
  2. has already failed

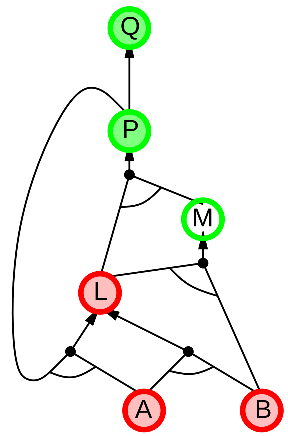


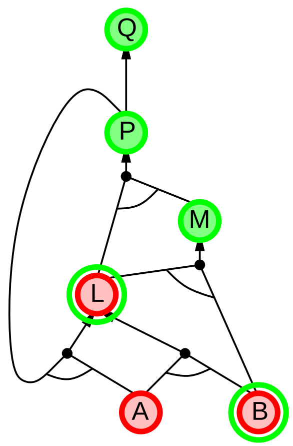


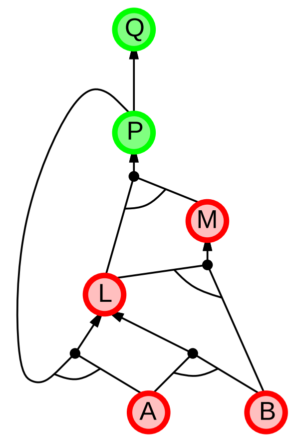

Q
P
M
L
A
B


Q
P
M
L
A
B


Q
P
M
L
A
B


Q
P
M
L
A
B





Forward v. backward chaining

- FC is *data-driven*, cf. automatic, unconscious processing
  - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal
- BC is *goal-driven*, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be *much less* than linear in size of KB

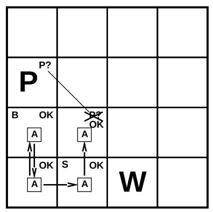
Resolution

- Resolution is another proof system.
  - Sound and complete for propositional logic.
- Just one inference rule:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  where ℓ_i and m_j are complementary literals.
- Eh?

- As an example, here:



- We might resolve:

$$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$

- So, if we know $P_{1,3} \vee P_{2,2}$ and $\neg P_{2,2}$ then we can conclude $P_{1,3}$

- Only issue — resolution only works for KB in *conjunctive normal form*
- *conjunction* of $\underbrace{\textit{disjunctions} \text{ of } \textit{literals}}_{\textit{clauses}}$
- Such as:

$$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$$

- Have to convert sentences to CNF.
- Example: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

Resolution example

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
  $\alpha = \neg P_{1,2}$
- First we have to convert the *KB* into conjunctive normal form.
- That is what we just did (here's one I made earlier):

$$\begin{array}{l} \neg P_{2,1} \vee B_{1,1} \\ \neg B_{1,1} \vee B_{P_{1,2}} \vee P_{2,1} \\ \neg P_{1,2} \vee B_{1,1} \\ \neg B_{1,1} \end{array}$$

- To this we add the negation of the thing we want to prove.

$$P_{1,2}$$

- Resolution works by repeatedly combining these formulae together until we get nothing (or the empty set).
- This represents the contradiction.
- When we find this we can conclude the negation of the thing we added to the *KB*.
  - This is just the thing we want to prove.
- So we might combine:

$$\frac{\neg P_{2,1} \vee B_{1,1}, \qquad \neg B_{1,1}}{\neg P_{2,1}}$$

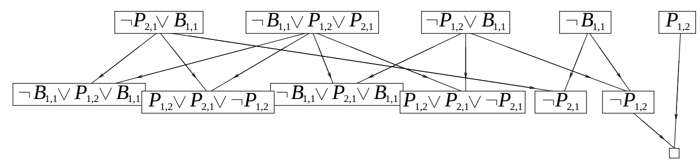
- Similarly we might infer:

$$\frac{\neg P_{1,2} \vee B_{1,1}, \qquad \neg B_{1,1}}{P_{1,2}}$$

and

$$\frac{P_{1,2} \qquad \neg P_{1,2}}{\perp}$$

thus finding the contradiction and concluding the proof.
- Many of the possible inferences are summarised by:



function PL-RESOLUTION(*KB*, α) **returns** *true* or *false*
  inputs: *KB*, the knowledge base, a sentence in propositional logic
  α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new ← { }
  loop do
    for each C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

In favor 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*
  - 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

Against propositional logic

- Propositional logic has very limited expressive power
  - Unlike natural language
- For example, cannot say:

  "pits cause breezes in adjacent squares"

  except by writing one sentence for each square.

First order logic

- Whereas propositional logic assumes world contains *facts*, *first-order logic* (like natural language) assumes the world contains:
  - *Objects*: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
  - *Relations*: red, round, bogus, prime, multistoried . . ., *brother of*, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .

    Relations are statements that are true or false.
  - *Functions*: father of, best friend, third inning of, one more than, end of . . .

    Functions return values.

- On the subject of brothers

Logics in general

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Syntax of FOL: Basic elements

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic sentences

Atomic sentence $=$ $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term $=$ $function(term_1, \ldots, term_n)$
or *constant* or *variable*

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

- More brothers:


Complex sentences

- Complex sentences are made from atomic sentences using connectives

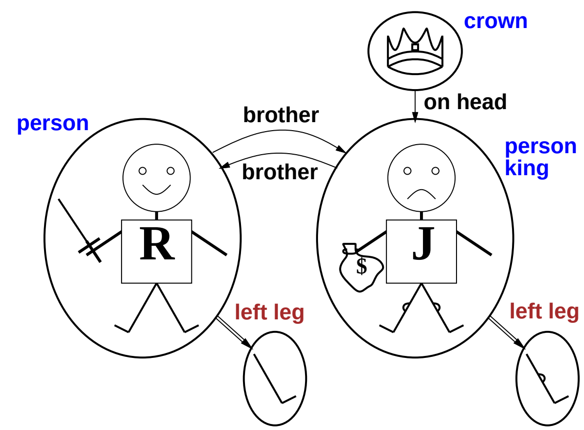
$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg{>}(1,2)$

Truth in first-order logic

- Sentences are true with respect to a *model* and an *interpretation*
- Model contains ≥ 1 objects (*domain elements*) and relations among them
- Interpretation specifies referents for:
  - constant symbols $\rightarrow$ objects
  - predicate symbols $\rightarrow$ relations
  - function symbols $\rightarrow$ functional relations
- An atomic sentence $predicate(term_1, \ldots, term_n)$ is true iff the objects referred to by $term_1, \ldots, term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Truth example

- Consider the interpretation in which
  - *Richard* $\rightarrow$ Richard the Lionheart
  - *John* $\rightarrow$ the evil King John
  - *Brother* $\rightarrow$ the brotherhood relation
- Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model.

Models for FOL: Lots!

- Entailment in propositional logic can be computed by enumerating models
- We *can* enumerate the FOL models for a given KB vocabulary:
  - For each number of domain elements n from 1 to ∞
  - For each k-ary predicate P_k in the vocabulary
  - For each possible k-ary relation on n objects
  - For each constant symbol C in the vocabulary
  - For each choice of referent for C from n objects . . .
- Computing entailment by enumerating FOL models is not easy!

Decidability

- In fact, it is worse than "not easy".
- Is there any procedure that we can use, that will be guaranteed to tell us, in a finite amount of time, whether a FOL formula is, or is not, valid?
- The answer is no.
- FOL is for this reason said to be *undecidable*.

Universal quantification

- $\forall \langle variables \rangle \ \langle sentence \rangle$
- Everyone at Brooklyn College is smart:

$$\forall x \ At(x, BC) \Rightarrow Smart(x)$$

- $\forall x \ P$ is true in a model m iff P is true with x being *each* possible object in the model
- *Roughly* speaking, equivalent to the conjunction of instantiations of P

$$\begin{aligned} & (At(KingJohn, BC) \Rightarrow Smart(KingJohn)) \\ \wedge \ & (At(Richard, BC) \Rightarrow Smart(Richard)) \\ \wedge \ & (At(BC, BC) \Rightarrow Smart(BC)) \\ \wedge \ & \ldots \end{aligned}$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

$$\forall x \ At(x, BC) \wedge Smart(x)$$

means "Everyone is at Brooklyn College and everyone is smart"

Existential quantification

- $\exists \langle variables \rangle \ \langle sentence \rangle$
- Someone at City College is smart:

$$\exists x \ At(x, City) \wedge Smart(x)$$

- $\exists x \ P$ is true in a model m iff P is true with x being *some* possible object in the model
- *Roughly* speaking, equivalent to the disjunction of instantiations of P:

$$\begin{aligned} & (At(KingJohn, City) \wedge Smart(KingJohn)) \\ \vee \ & (At(Richard, City) \wedge Smart(Richard)) \\ \vee \ & (At(Stanford, City) \wedge Smart(Stanford)) \\ \vee \ & \ldots \end{aligned}$$

A common mistake to avoid (2)

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \;\; At(x, City) \;\Rightarrow\; Smart(x)$$

is true if there is anyone who is not at City College!

Properties of quantifiers

- $\forall x \;\; \forall y$ is the same as $\forall y \;\; \forall x$ (why?)
- $\exists x \;\; \exists y$ is the same as $\exists y \;\; \exists x$ (why?)
- $\exists x \;\; \forall y$ is *not* the same as $\forall y \;\; \exists x$
- $\exists x \;\; \forall y \;\; Loves(x, y)$
  "There is a person who loves everyone in the world"
- $\forall y \;\; \exists x \;\; Loves(x, y)$
  "Everyone in the world is loved by at least one person"
- Quantifier duality: each can be expressed using the other

$$\forall x \;\; Likes(x, IceCream) \qquad \neg\exists x \;\; \neg Likes(x, IceCream)$$
$$\exists x \;\; Likes(x, Broccoli) \qquad \neg\forall x \;\; \neg Likes(x, Broccoli)$$

Fun with sentences

- Brothers are siblings

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$
- "Sibling" is symmetric

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$
- "Sibling" is symmetric
  $\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$
- "Sibling" is symmetric
  $\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$
- One's mother is one's female parent

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$
- "Sibling" is symmetric
  $\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$
- One's mother is one's female parent
  $\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y))$

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$
- "Sibling" is symmetric
  $\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$
- One's mother is one's female parent
  $\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y))$
- A first cousin is a child of a parent's sibling

Fun with sentences

- Brothers are siblings
  $\forall x, y \;\; Brother(x, y) \;\Rightarrow\; Sibling(x, y)$
- "Sibling" is symmetric
  $\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$
- One's mother is one's female parent
  $\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y))$
- A first cousin is a child of a parent's sibling

$$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow$$
$$\exists\, p, ps \;\; Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$$

Equality

- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object
  E.g., $1 = 2$ and $\forall x \;\; \times(Sqrt(x), Sqrt(x)) = x$ are satisfiable
  $2 = 2$ is valid
- E.g., definition of (full) *Sibling* in terms of *Parent*:
  $\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; [\neg(x = y) \wedge \exists\, m, f \;\; \neg(m = f) \wedge$
  $Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]$

Interacting with FOL KBs

- Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at $t = 5$:
- $Tell(KB, Percept([Smell, Breeze, None], 5))$
  $Ask(KB, \exists\, a \;\; Action(a, 5))$
- Does *KB* entail any particular actions at $t = 5$?
- Answer: $Yes, \; \{a/Shoot\}$ $\quad\leftarrow$ *substitution* (binding list)
- Given a sentence S and a substitution σ, $S\sigma$ denotes the result of plugging σ into S

- For example:

  $S = Smarter(x, y)$

  $\sigma = \{x/Hillary, y/Bill\}$

  $S\sigma = Smarter(Hillary, Bill)$

- $Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Knowledge base for the wumpus world

- "Perception"

  $\forall b, g, t \;\; Percept([Smell, b, g], t) \;\Rightarrow\; Smelt(t)$

  $\forall s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$

- Reflex

  $\forall t \;\; AtGold(t) \;\Rightarrow\; Action(Grab, t)$

- Reflex with internal state: do we have the gold already?

  $\forall t \;\; AtGold(t) \land \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$

- $Holding(Gold, t)$ cannot be observed $\Rightarrow$ keeping track of change is essential


```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```


Deducing hidden properties

- Properties of locations:

  $\forall x, t \;\; At(Agent, x, t) \land Smelt(t) \;\Rightarrow\; Smelly(x)$

  $\forall x, t \;\; At(Agent, x, t) \land Breeze(t) \;\Rightarrow\; Breezy(x)$

- Squares are breezy near a pit.
- *Diagnostic* rule—infer cause from effect

  $\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x, y)$

- *Causal* rule—infer effect from cause

  $\forall x, y \;\; Pit(x) \land Adjacent(x, y) \;\Rightarrow\; Breezy(y)$

- Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy
- *Definition* for the *Breezy* predicate:

  $\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \land Adjacent(x, y)]$

Proof in FOL

- Proof in FOL is similar to propositional logic; we just need an extra set of rules, to deal with the quantifiers.
- FOL *inherits* all the rules of PL.
- To understand FOL proof rules, need to understand *substitution*.
- The most obvious rule, for $\forall$-E.

  Tells us that if everything in the domain has some property, then we can infer that any *particular* individual has the property.

$$\frac{\vdash \forall x \cdot P(x);}{\vdash P(a)} \; \forall\text{-E} \quad \text{for any } a \text{ in the domain}$$

  Going from *general* to *specific*.
- If all Brooklyn College students are smart, then anyone in the class is smart.

- Example 1.

  Let's use $\forall$-E to get the Socrates example out of the way.

$$Person(s); \forall x \cdot Person(x) \Rightarrow Mortal(x) \vdash Mortal(s)$$

1.	$Person(s)$	Given
2.	$\forall x \cdot Person(x) \Rightarrow Mortal(x)$	Given
3.	$Person(s) \Rightarrow Mortal(s)$	2, $\forall$-E
4.	$Mortal(s)$	1, 3, $\Rightarrow$-E

- We can also go from the general to the slightly less specific!

$$\frac{\vdash \forall x \cdot P(x);}{\vdash \exists x \cdot P(x)} \; \exists\text{-I(1)} \quad \text{if domain not empty}$$

  Note the *side condition*.

  The $\exists$ quantifier *asserts the existence* of at least one object.

  The $\forall$ quantifier does not.
- So, while we can say "All unicorns have horns" irrespective of whether unicorns are real or not, we can only say "There's a unicorn living on my street whose name is Fred and he has a horn" if there is at least one unicorn.

- We can also go from the very specific to less specific.

$$\frac{\vdash P(a);}{\vdash \exists x \cdot P(x)} \; \exists\text{-I(2)}$$

- In other words once we have a concrete example, we can infer there exists something with the property of that example.
- If I find a student at City College who is smart, I can say "There is a smart student at City College".

- We often informally make use of arguments along the lines...
  1. We know somebody is the murderer.
  2. Call this person a.
  3. a must have been in the library with the lead pipe.
  4. ...

  (Here, a is called a *Skolem constant*.)


Thoralf Skolem

- We have a rule which allows this, but we have to be careful how we use it!

$$\frac{\vdash \exists x \cdot P(x);}{\vdash P(a)} \; \exists\text{-E} \quad a \text{ doesn't occur elsewhere}$$

- Here is an *invalid* use of this rule:

1.	$\exists x \cdot Boring(x)$	Given
2.	$Lecture(AI)$	Given
3.	$Boring(AI)$	1, $\exists$-E

- (The conclusion may be true, the argument isn't sound.)

- Another kind of reasoning:
  - Let a be arbitrary object.
  - ... (some reasoning) ...
  - Therefore a has property P
  - Since a was arbitrary, it must be that every object has property P.
- Common in mathematics:

  Consider a positive integer n ... so n is either a prime number or divisible by a smaller prime number ... thus every positive integer is either a prime number or divisible by a smaller prime number.

- If we are careful, we can also use this kind of reasoning:

$$\frac{\vdash P(a);}{\vdash \forall x \cdot P(x)} \forall\text{-I} \quad a \text{ is arbitrary}$$

- Here's an invalid use of this rule:

1.	$Boring(AI)$	Given
2.	$\forall x \cdot Boring(x)$	1, $\forall$-I

- An example:
  1. Everybody is either happy or rich.
  2. Simon is not rich.
  3. Therefore, Simon is happy.

  Predicates:
  – $H(x)$ means x is happy;
  – $R(x)$ means x is rich.
- Formalisation:

$$\forall x.H(x) \vee R(x); \neg R(Simon) \vdash H(Simon)$$

1.	$\forall x.H(x) \vee R(x)$	Given
2.	$\neg R(Simon)$	Given
3.	$H(Simon) \vee R(Simon)$	1, $\forall$-E
4.	$\neg H(Simon) \Rightarrow R(Simon)$	3, defn $\Rightarrow$
5.	$\neg H(Simon)$	As. |
6.	$R(Simon)$	4, 5, $\Rightarrow$-E |
7.	$R(Simon) \wedge \neg R(Simon)$	2, 6, $\wedge$-I |
8.	$\neg\neg H(Simon)$	5, 7, $\neg$-I
9.	$H(Simon) \Leftrightarrow \neg\neg H(Simon)$	PL axiom
10.	$(H(Simon) \Rightarrow \neg\neg H(Simon))$ $\wedge(\neg\neg H(Simon) \Rightarrow H(Simon))$	9, defn $\Leftrightarrow$
11.	$\neg\neg H(Simon) \Rightarrow H(Simon)$	10,$\wedge$-E
12.	$H(Simon)$	8, 11, $\Rightarrow$-E

Summary

- This lecture completes our treatment of logic.
- We have added some new proof techniques:
  – Forward chaining
  – Backward chaining
  – Resolution

  to our treatment of propositional logic; and
- Covered the basics of first order logic.
- There is plenty more to logic (a whole other chapter in the texbook) but we will look at other things next week.