RESOLUTION AND FIRST ORDER LOGIC
In the last class 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.
Logic and proof

• Need to be clear that logics exist separate from any proof method that they use.
• Can have one logic with many proof methods.
• Those same methods may work for many logics.
• So far we have looked at one logic (propositional logic) and one proof system (natural deduction).
• We will look at:
  – More proof systems for propositional logic
  – Another logic.
• New proof systems:
  – Forward chaining
  – Backward chaining
  – Resolution

• New logic
  – Predicate logic
New proof systems

• 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:
  \[ C \land D \Rightarrow B \]

• KB = *conjunction* of *Horn clauses*

• For example:
  \[ C \land (B \Rightarrow A) \land (C \land D \Rightarrow B) \]
• Modus ponens is then:

\[
\frac{\alpha_1, \ldots, \alpha_n, \ \alpha_1 \land \cdots \land \alpha_n}{\beta} \Rightarrow \beta
\]

• Sometimes called “generalized modus ponens”.
• For Horn clauses, modus ponens is all you need
  – Complete
• Can be used with \textit{forward chaining} or \textit{backward chaining}.
• These algorithms are very natural and run in \textit{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

\[ P \Rightarrow Q \]
\[ L \land M \Rightarrow P \]
\[ B \land L \Rightarrow M \]
\[ A \land P \Rightarrow L \]
\[ A \land B \Rightarrow L \]
\[ A \]
\[ B \]

- How does this work?
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)
        end
    end
    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 $a_1 \land \ldots \land a_k \Rightarrow b$ is false in $m$
     Then $a_1 \land \ldots \land a_k$ 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 $\alpha$
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
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:

\[
\frac{\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n}{\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n}
\]

where \( \ell_i \) and \( m_j \) are complementary literals.

• Eh?
• As an example, here:

![Diagram]

• We might resolve:

\[
\frac{P_{1,3} \lor P_{2,2}, \neg P_{2,2}}{P_{1,3}}
\]

• So, if we know \(P_{1,3} \lor P_{2,2}\) 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 disjunctions of literals*

  *clauses*

• Such as:

  \[(A \lor \neg B) \land (B \lor \neg C \lor \neg D)\]

• Have to convert sentences to CNF.
Example: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
1. Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.

$$(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$$

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

$$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1})$$

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

$$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$$

4. Apply distributivity law ($\lor$ over $\land$):

$$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$$
Resolution example

- \( KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \)
  \[ \alpha = \neg P_{1,2} \]

- First we have to convert the \( KB \) into conjunctive normal form.
- That is what we just did (here’s one I made earlier):
  \[
  \neg P_{2,1} \lor B_{1,1} \\
  \neg B_{1,1} \lor B_{P_{1,2}} \lor P_{2,1} \\
  \neg P_{1,2} \lor B_{1,1} \\
  \neg B_{1,1}
  \]

- 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 (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:

\[ \neg P_{2,1} \lor B_{1,1}, \neg B_{1,1} \]

\[ \frac{\neg P_{2,1} \lor B_{1,1}, \neg B_{1,1}}{\neg P_{2,1}} \]
• Similarly we might infer:

\[
\frac{-P_{1,2} \lor B_{1,1}, \ -B_{1,1}}{P_{1,2}}
\]

and

\[
\frac{P_{1,2}, \ -P_{1,2}}{\bot}
\]

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 ∧ ¬α
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 $B_{1,1} \land P_{1,2}$ 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, Barack Obama, 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.
• The line between functions and relations is sometimes confusing.
• This is a relation:

\[ PresidentOf(UnitedStates, BarakObama) \]

which is currently true.
• This is a function:

\[ PresidentOf(UnitedStates) \]

which currently returns the value \textit{BarackObama}. 
## Logics in general

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>facts + degree of truth</td>
<td>degree of belief</td>
</tr>
<tr>
<td></td>
<td></td>
<td>known interval value</td>
</tr>
</tbody>
</table>
Syntax of FOL: Basic elements

- Constants: $\text{KingJohn, 2, UCB, ...}$
- Predicates: $\text{Brother, >, ...}$
- Functions: $\text{Sqrt, LeftLegOf, ...}$
- Variables: $x, y, a, b, ...$
- Connectives: $\land, \lor, \neg, \Rightarrow, \Leftrightarrow$
- Equality: $=$
- Quantifiers: $\forall, \exists$

- Predicates express *relations* between things.
**Atomic sentences**

Atomic sentence = \( \text{predicate}(term_1, \ldots, term_n) \)

or \( term_1 = term_2 \)

Term = \( \text{function}(term_1, \ldots, term_n) \)

or constant or variable

E.g., \( \text{Brother}(KingJohn, RichardTheLionheart) \)

\[ > (\text{Length}(\text{LeftLegOf}(Richard)), \text{Length}(\text{LeftLegOf}(KingJohn))) \]
• The brothers we are talking about:
Complex sentences

- Complex sentences are made from atomic sentences using connectives

\[
\neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2
\]

E.g. \( \text{Sibling}(\text{KingJohn, Richard}) \Rightarrow \text{Sibling}(\text{Richard, KingJohn}) \)
\[
>(1, 2) \lor \leq (1, 2) \]
\[
>(1, 2) \land \neg > (1, 2)
\]
Truth in first-order logic

• Sentences are true with respect to a model and an interpretation (Remember that in propositional logic, these ideas were interchangeable)

• Model contains $\geq 1$ 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 $\text{predicate}(term_1, \ldots, term_n)$ is true iff the objects referred to by $term_1, \ldots, term_n$ are in the relation referred to by $\text{predicate}$
Models for FOL: Example

- Person R
- Person J
- Person king
- Person crown
- Brother R to J
- Left leg of R and J
- On head of crown
Truth example

- Consider the interpretation in which
  - *Richard* → Richard the Lionheart
  - *John* → the evil King John
  - *Brother* → 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.
- If the model is Northern Europe in the years 1166 to 1199, then the interpretation is true.
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 $\infty$
  - 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

- $\forall \langle \text{variables} \rangle \langle \text{sentence} \rangle$
- 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 $P$
  $\quad (At(KingJohn, BC) \Rightarrow Smart(KingJohn))$
  $\quad \land (At(Richard, BC) \Rightarrow Smart(Richard))$
  $\quad \land (At(BC, BC) \Rightarrow Smart(BC))$
  $\quad \land \ldots$
A common mistake to avoid

- Typically, \( \Rightarrow \) is the main connective with \( \forall \)
- Common mistake: using \( \land \) as the main connective with \( \forall \):
  \[
  \forall x \ At(x, BC) \land Smart(x)
  \]
  means “Everyone is at Brooklyn College and everyone is smart”
Existential quantification

- $\exists \langle \text{variables} \rangle \langle \text{sentence} \rangle$

- Someone at City College is smart:
  $$\exists x \ At(x, \text{City}) \land Smart(x)$$

- $\exists x \ P$ 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 $P$:
  $$(At(KingJohn, \text{City}) \land Smart(KingJohn))$$
  $$\lor (At(Richard, \text{City}) \land Smart(Richard))$$
  $$\lor (At(\text{City}, \text{City}) \land Smart(\text{City}))$$
  $$\lor \ldots$$
A common mistake to avoid (2)

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

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

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
  
  $\forall x \ Loves(x, IceCream) \iff \exists x \ \neg Loves(x, IceCream)$
  $\exists x \ Loves(x, Broccoli) \iff \forall x \ \neg Loves(x, Broccoli)$
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
  \[ \forall x, y \; \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) \]
- “Sibling” is symmetric
Fun with sentences

- Brothers are siblings
  \[ \forall x, y \ Brother(x, y) \Rightarrow Sibling(x, y) \]
- “Sibling” is symmetric
  \[ \forall x, y \ Sibling(x, y) \iff Sibling(y, x) \]
Fun with sentences

• Brothers are siblings
  \[ \forall x, y \ Brother(x, y) \Rightarrow Sibling(x, y) \]

• “Sibling” is symmetric
  \[ \forall x, y \ Sibling(x, y) \iff Sibling(y, x) \]

• One’s mother is one’s female parent
Fun with sentences

- Brothers are siblings
  \[ \forall x, y \quad \text{Brother}(x, y) \implies \text{Sibling}(x, y) \]

- “Sibling” is symmetric
  \[ \forall x, y \quad \text{Sibling}(x, y) \iff \text{Sibling}(y, x) \]

- One’s mother is one’s female parent
  \[ \forall x, y \quad \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)) \]
Fun with sentences

- Brothers are siblings
  \[ \forall x, y \quad \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) \]
- “Sibling” is symmetric
  \[ \forall x, y \quad \text{Sibling}(x, y) \iff \text{Sibling}(y, x) \]
- One’s mother is one’s female parent
  \[ \forall x, y \quad \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)) \]
- A first cousin is a child of a parent’s sibling
Fun with sentences

• Brothers are siblings
  \( \forall x, y \; \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) \)

• “Sibling” is symmetric
  \( \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x) \)

• One’s mother is one’s female parent
  \( \forall x, y \; \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)) \)

• A first cousin is a child of a parent’s sibling
  \( \forall x, y \; \text{FirstCousin}(x, y) \iff \exists p, ps \; \text{Parent}(p, x) \land \text{Sibling}(ps, p) \land \text{Parent}(ps, y) \)
Equality

• \textit{term}_1 = \textit{term}_2 \text{ is true under a given interpretation if and only if } \textit{term}_1 \text{ and } \textit{term}_2 \text{ 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 \textit{Parent}:
  
  \( \forall x, y \ Sibling(x, y) \iff [\neg (x = y) \land \exists m, f \ \neg (m = f) \land Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)] \)
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, $\{a/\text{Shoot}\} \leftarrow \text{substitution}$ (binding list)
- Given a sentence $S$ and a substitution $\sigma$, $S\sigma$ denotes the result of plugging $\sigma$ into $S$
• For example:

\[ S = \text{Smarter}(x, y) \]
\[ \sigma = \{x/\text{Hillary}, y/\text{Bill}\} \]
\[ S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill}) \]

• \( \text{Ask}(KB, S) \) returns some/all \( \sigma \) such that \( KB \models S\sigma \)
Knowledge base for the wumpus world

• “Perception”
  \[ \forall b, g, t \ Percept([\text{Smell}, b, g], t) \Rightarrow \text{Smell}(t) \]
  \[ \forall s, b, t \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{AtGold}(t) \]

• Reflex
  \[ \forall t \ \text{AtGold}(t) \Rightarrow \text{Action}(%27\text{Grab}, t) %27 \]

• Reflex with internal state: do we have the gold already?
  \[ \forall t \ \text{AtGold}(t) \land \neg \text{Holding}(\text{Gold}, t) \Rightarrow \text{Action}(\text{Grab}, t) \]

• \text{Holding}(\text{Gold}, t) cannot be observed \Rightarrow keeping track of change is essential
function \textit{KB-AGENT}(\textit{percept}) \textbf{returns} an \textit{action}

\textbf{static}: \textit{KB}, a knowledge base
\textit{t}, a counter, initially 0, indicating time

\textbf{Tell}(\textit{KB, Make-Percept-Sentence}(\textit{percept, t}))

\textit{action} ← \textbf{Ask}(\textit{KB, Make-Action-Query}(\textit{t}))

\textbf{Tell}(\textit{KB, Make-Action-Sentence}(\textit{action, t}))

\textit{t} ← \textit{t} + 1

\textbf{return} \textit{action}
Deducing hidden properties

- Properties of locations:
  \[ \forall x, t \ \text{At}(\text{Agent}, x, t) \land \text{Smelt}(t) \Rightarrow \text{Smelly}(x) \]
  \[ \forall x, t \ \text{At}(\text{Agent}, x, t) \land \text{Breeze}(t) \Rightarrow \text{Breezy}(x) \]

- Squares are breezy near a pit.

- **Diagnostic** rule—infer cause from effect
  \[ \forall y \ \text{Breezy}(y) \Rightarrow \exists x \ \text{Pit}(x) \land \text{Adjacent}(x, y) \]

- **Causal** rule—infer effect from cause
  \[ \forall x, y \ \text{Pit}(x) \land \text{Adjacent}(x, y) \Rightarrow \text{Breezy}(y) \]

- 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:
  \[ \forall y \ \text{Breezy}(y) \Leftrightarrow \left[ \exists x \ \text{Pit}(x) \land \text{Adjacent}(x, y) \right] \]
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 \textit{inherits} all the rules of PL.

• To understand FOL proof rules, need to understand \textit{substitution}.

• The most obvious rule, for $\forall$\text{-E}.
  
  Tells us that if everything in the domain has some property, then we can infer that any \textit{particular} individual has the property.

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

  Going from \textit{general} to \textit{specific}.

• If all Brooklyn College students are smart, then anyone in the class is smart.
Example 1.
Let’s use ∀-E to get the Socrates example out of the way.

\[ \text{Person}(s); \forall x \cdot \text{Person}(x) \Rightarrow \text{Mortal}(x) \vdash \text{Mortal}(s) \]

1. \( \text{Person}(s) \) Given
2. \( \forall x \cdot \text{Person}(x) \Rightarrow \text{Mortal}(x) \) Given
3. \( \text{Person}(s) \Rightarrow \text{Mortal}(s) \) 2, ∀-E
4. \( \text{Mortal}(s) \) 1, 3, \( \Rightarrow \)-E
• We can also go from the general to the slightly less specific!

\[
\begin{align*}
\vdash \forall x \cdot P(x); & \quad \exists\text{-I(1)} \\
\vdash \exists x \cdot P(x) & \quad \text{if domain not empty}
\end{align*}
\]

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.
• This is Fred.
• We can also go from the very specific to less specific.

\[ \vdash P(a); \quad \exists \text{-I(2)} \]

\[ \vdash \exists x \cdot P(x) \]

• 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”.
• There’s a \( \exists \) elimination rule also.
• 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!

$$\vdash \exists x \cdot P(x); \quad \exists \text{-E} \quad \vdash P(a)$$

$a$ 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:

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

• Here’s an invalid use of this rule:

1. \(Boring(AI)\) Given
2. \(\forall x \cdot Boring(x)\) 1, \(\forall\text{-I}\)
• With this we have a full set of rules for handling quantifiers.
• Adding them to the rules we had before, we have enough to do natural deduction with predicate logic.
• Let’s look at an example.
• 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:

$$\forall x. H(x) \lor R(x); \neg R(Simon) \vdash H(Simon)$$
1. \( \forall x. H(x) \lor R(x) \)  
   Given
2. \( \neg R(Simon) \)  
   Given
3. \( H(Simon) \lor R(Simon) \)  
   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. \( R(Simon) \land \neg R(Simon) \)  
   2, 6, \( \land \)-I
8. \( \neg \neg H(Simon) \)  
   5, 7, \( \neg \)-I
9. \( H(Simon) \Leftrightarrow \neg \neg H(Simon) \)  
   PL axiom
10. \( (H(Simon) \Rightarrow \neg \neg H(Simon)) \)  
   \( \Leftrightarrow (\neg \neg H(Simon) \Rightarrow H(Simon)) \)  
    9, defn \( \Leftrightarrow \)
11. \( \neg \neg H(Simon) \Rightarrow H(Simon) \)  
    10, \( \land \)-E
12. \( H(Simon) \)  
    8, 11, \( \Rightarrow \)-E
To summarise where we stand with logics and proof systems:

<table>
<thead>
<tr>
<th></th>
<th>Propositional Logic</th>
<th>Predicate Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural Deduction</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Forward Chaining</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Backward Chaining</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

We could, quite easily, extend FC, BC and resolution for predicate logic.

– We already know how to handle quantifiers, and that is the hardest bit.
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 textbook) but we will look at other things next week.