- The appeal of logic as knowledge representation

- Logic as "applied AI"

- Styles of working with logic

    Automated theorem proving
    Automated proof checking
    Logic programming

Propositional Logic

- Only "atomic" propositions (P, Q, etc.) with binary truth values

- Connectives AND, OR, NOT, IMPLIES

- Values of sentences are determined as functions of the truth values of individual propositions.

- Rules of inference: modus ponens, commutativity, etc.

- Standard issues in propositional logic: satisfiability, contradiction, tautology, implication, equivalence.

Solving Problems in Propositional Logic

- Satisfiability is a search problem (and is NP-complete)
- Brute force: truth table method
- Strategies: most constraining variable, heuristic repair

First-Order Predicate Calculus

- We introduce *objects* and *properties*:

  Beau [object]
  Furry [property]

- *Predicates* (or *relations*) are true/false statements about objects:

  Furry(Beau)
  Four-footed(Beau)

- *Functions* are applied to objects to produce other objects:

  Nose-of(Beau)
  Collar-of(Beau)

• Atoms (or atomic sentences) are just predicates applied to object arguments:

Four-footed(Beau)

• Sentences may be combined into more complex "compound forms" via connectives like AND, OR, NOT, IMPLIES:

Four-footed(Beau) AND Furry(Beau)

Furry(Beau) IMPLIES Mammal(Beau)

```
Furry (Beau)
   IMPLIES NOT(Reptile(Mother-of(Beau)))
```

• Quantifiers FOR-ALL and THERE-EXISTS can be used to make statements about large (possibly infinite) sets

FOR-ALL (x) [Man(x) IMPLIES Mortal (x)]

```
FOR-ALL (x)
  [THERE-EXISTS (y) [Greater-than (y x)]]
```

```
FOR-ALL(x)
 [Person(x) IMPLIES
  THERE-EXISTS (y)
      Time(y) AND Fool-at (x, y)]

THERE-EXISTS (x)
  [Person(x)
  AND
  FOR-ALL (y) Time(y) IMPLIES Fool-at(x, y)]

NOT FOR-ALL(x, y)
        [Person(x) AND Time(y)]
        IMPLIES Fool-at (x, y)
```

To determine the truth or falsehood of logical sentences we have to place an interpretation on the objects and relations of a given logical system. The sentence above, for instance is true when x, y are taken from the set of real numbers but not true if they are taken from the set of non-negative integers. A *model* for a given set of sentences is an interpretation for which the sentences are true statements. (Sometimes the model is formally identified with the set of sentences alone.)

• Issues in thinking about logical systems and rules of inference

Entailment
Soundness
Completeness
Decidability
"Difficulty"

S1. Beau is a retriever.
S2. Retrievers are dogs.
S3. Dogs have sensitive noses.
S4. Sensitive noses are useful for hunting.

P1. Retriever (Beau)

P2. FOR-ALL(x) [Retriever (x) --> Dog (x)]

P3. FOR-ALL (x, y) [Dog (x) AND Nose (y, x) --> Sensitive (y)]

```
P4. FOR-ALL (y)
      [(Sensitive (y) AND THERE-EXISTS (x) [Nose (y, x)])
                              ---> Useful-for-hunting (y)]
```

P5. FOR-ALL (x) [Nose(Nose-of(x), x)]

[1] Retriever (Beau)

[2] Retriever (Beau) ---> Dog (Beau)

[3] Nose (Nose-of (Beau), Beau)

[4] Nose (Nose-of (Beau), Beau) AND Dog (Beau) -->
Sensitive (Nose-of(Beau))

[5] Sensitive(Nose-of(Beau)) AND
Nose (Nose-of(Beau), Beau) -->
Useful-for-hunting(Nose-of(Beau))

Backward Chaining

• Propose the negation of what you want to prove:

[R1] Useful-for-hunting(Nose-of(Beau)) --> F

English: It is not the case that the object Nose-of(Beau) is useful for hunting.

• Resolve [R1] and [5]:

[R2] Sensitive(Nose-of(Beau)) AND Nose (Nose-of(Beau), Beau) --> F

English: It is not the case that the object Nose-of(Beau) is sensitive and is a nose that belongs to Beau.

• Resolve [R2] and [3]:
Note that [3] can be written T--> Nose(Nose-of(Beau), Beau)

[R3] Sensitive (Nose-of(Beau)) --> F

English: It is not the case that the object Nose-of(Beau) is sensitive.

• Resolve [R3] and [4]

[R4] Nose(Nose-of(Beau), Beau) AND Dog(Beau) --> F

English: It is not the case that the object Nose-of(Beau) is a nose belonging to Beau and that Beau is a dog.

• Resolve [R4] and [3]

[R5] Dog (Beau) --> F

English: It is not the case that Beau is a dog.

• Resolve [R5] and [2]

[R6] Retriever (Beau) --> F

English: It is not the case that Beau is a retriever.

- Resolve [R7] and [1]

T --> F [Contradiction]

Putting Propositional Logic Sentences into Normal Form:

Step 1. Change all instances of IMPLIES into OR form:

A IMPLIES B
becomes
(NOT A) OR B

Step 2. Bring negation inside parentheses by using DeMorgan's rules:

NOT (A AND B)
becomes
(NOT A) OR (NOT B)

NOT (A OR B)
becomes
(NOT A) AND (NOT B)

Step 3. Rewrite the sentence as a conjunction of disjunctions:

A OR (B AND C)
becomes
(A OR B) AND (A OR C)

Step 4. Rewrite the large conjunction as separate sentences

(A OR B) AND (A OR C)
becomes
A OR B
A OR C

Step 5. We now have a bunch of disjunctions of terms, some of which have a negation sign in front of them. Move the negation terms to the left:

(NOT C) OR (NOT D) OR E OR F

Then use DeMorgan's laws to get the sentence in the equivalent IMPLIES form:

NOT (C AND D) OR E OR F

Step 6. Rewrite using IMPLIES to put the sentence back into normal form:

NOT (C AND D) OR E OR F

becomes

(C AND D) IMPLIES (E OR F)

Redoing the Previous Algorithm for First-Order Predicate Logic:

1. Eliminate occurrences of IMPLIES

2. Bring negation inside parentheses.

Note that
NOT (FOR-ALL(x) P(x)) becomes
THERE-EXISTS (x) [NOT P(x)]

NOT (THERE-EXISTS(x) Q(x)) becomes
FOR-ALL (x) [NOT Q (x)]

3. For every occurrence of a quantifier, provide the bound variable with its own unique name. (This prevents name conflicts when quantifiers are eventually "moved to the front.")

4. Use Skolemization to replace existential quantifiers. The idea here is that an existential quantifier specifies a "choosing function" (that may depend on other variables):

FOR-ALL (x) [THERE-EXISTS [y] Successor (y x)]

becomes

FOR-ALL (x) [Successor (Successor-of(x), x)]

5. We now have a sentence in which only universal quantifiers occur, and in which all variables have unique names. We can now move all these quantifiers to the left of the overall sentence without changing the meaning of the sentence.

6. Rewrite the sentence as a conjunction of disjunctions.

7. Rewrite the conjunction as a series of separate sentences, each of which is now a disjunction.

8. Bring the negated terms to the left, and rewrite each sentence as an implication.

9. Within each sentence, rename variables to sentence-specific values. This is in preparation for...

10. (For brevity) Eliminate the universal quantifiers.

- Resolution is, at heart, a search problem
  Different sorts of strategies for searching:
  "set-of-support" strategy, "unit-preference" strategy

- Interactive systems

- More elaborate types of logic