CS 4700:
Foundations of Artificial Intelligence

Bart Selman
selman@cs.cornell.edu

Module: Knowledge, Reasoning, and Planning
Part 2

First-Order Logic and Inference

R&N: Chapters 8 and 9

First-Order Logic

Richer language. Closer match to ontology / conceptual structure

objects / properties / relations / functions

Ex.:

objects: table, car, house, John...

relations: brother of, part of, has color...

properties: red, round, prime... ["unary relations"]

functions: father of, best fiend, one more than...

Q. Contrast relation with function.

Syntax and Semantics

Study section 8.2 of R&N carefully.

Note: Semantics can be defined more formally.

See e.g. "A course in mathematical logic" Bell & Machover.

R&N provide main ideas behind semantics.

Semantics give by **interpretations** (propositional analogue: truth assignment)

Sentence ("formula") evaluates to True or False under a given interpretation. If True, interpretation is called a **model** of the sentence.

We hope that models of KB are close to actual state of affairs (worl But often, we also have unexpected (non-standard) models.

Relation between mathematical notion of interpretation / model and actual physical world interesting philosophical issue. We'll ignore it.

Each interpretation is defined over a given **domain** U (set of individuals / objects).

Constant symbols: *A, B, C, John, chair-1, house-10...*

In interpretation these symbols correspond to elements of U.

(two constants can define the same element in U).

(morning-star / evening-star)

Predicate symbols: *Round, Brother, Part-of,...*

Each predicate symbol correspond to a relation on U.

E.g., a binary predicate, corresponds to a binary relation.

If U equals { car, tires, steering wheel, house }.

[tires, car], and [steering wheel, car].

could be the intended interpretation of "part-of"

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
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 0…1
Fuzzy logic	degree of truth	degree of belief 0…1

Function symbols: $Cosine$, $FatherOf$, $LeftLegOf \ldots$

Correspond to functions defined on U.

Relation to predicate symbols?

Again, "what's meant by embodying **knowledge** about the world?"

Example:

1) $On(A, Fl) \Rightarrow Clear(B)$
2) $(Clear(B) \wedge Clear(C)) \Rightarrow On(A, Fl)$
3) $Clear(B) \vee Clear(A)$
4) $Clear(B)$
5) $Clear(C)$

One interpretation:

U is the set { `A, B, C, Floor` }.

1) mapping constant symbols to elements of U.

e.g., A to `A`, B to `B`, C to `C`

and Fl to `Floor`

Could we have mapped Fl to `A`??

2) mapping of relation symbol On to relation on U.

e.g., On = { [`B, A`], [`A, Floor`], [`C, Floor`]}.

3) mapping of relation (property) $Clear$ to a unary rel. on U.

e.g., $Clear$ = { [`B`], [`C`] }.

Yet others . . .

```
  B                                   C
  A   C        A   B   C              A    B
-------        --------             ------
  floor          floor                floor
```

Including completely different interpretations!

E.g., use integers for domain. (Lowenheim 1915)

Try to add sufficient axioms (facts) to rule out unwanted models. E.g., add $clear(A)$.

Terms —- a logical expressions that refers to an object. Constant symbols are terms. Functions applied to constant symbols. $FatherOf(John)$. Also, **variables** are terms (later) and functions applied to variables or other terms.

The interpretation is given by whatever the Constant or Function maps to in U (vars later).

If no vars, called **atomic terms**.

Atomic sentences — A predicate symbol applied to atomic terms

E.g. $Married(FatherOf(Richard), MotherOf(John))$

Evaluated to **true** if predicate symbol holds between the objects referred to by the arguments.

Complex sentences — add **logical connectives**.

E.g. $Older(John, 30) \Rightarrow Older(Jane, 29)$

Quantifiers

Universal Quantification $\forall$ —

E.g., $\forall x \;\; Cat(x) \Rightarrow Mammal(x)$

Think of as:

$$(Cat(Spot) \Rightarrow Mammal(Spot)) \;\; \wedge$$
$$(Cat(Felix) \Rightarrow Mammal(Felix)) \;\; \wedge$$
$$(Cat(John) \Rightarrow Mammal(John)) \;\; \wedge$$
$$\ldots$$

Intuition: Expand over all object symbols.

Existential Quantification $\exists$ —

E.g., $\exists x \;\; Sister(x, Spot) \wedge Cat(x)$

Think of as:

$$(Sister(Spot, Spot) \wedge Cat(Spot)) \quad \vee$$
$$(Sister(Rebecca, Spot) \wedge Cat(Rebecca)) \quad \vee$$
$$(Sister(Felix, Spot) \wedge Cat(Felix)) \quad \vee$$
$$\ldots$$

Intuition: Expand over all object symbols.

Equality $=$ —

E.g. $father(John) = Henry$

True iff refer to same object of U in interpretation.

(identity relation)

See Chapter 8 of R&N for more discussion and fine details.

E.g. can't just switch quantifiers around.

Compare $\forall x \exists y Loves(x, y)$ vs.

Compare $\exists x \forall y Loves(x, y)$

Reflex Agent

Directly connects percepts to actions:

$$\forall s, b, u, c, t \quad Percept([s, b, Glitter, u, c], t) \Rightarrow Action(Grab, t)$$

Or, more indirectly:

$$\forall s, b, u, c, t \quad Percept([s, b, Glitter, u, c], t) \Rightarrow AtGold(t)$$

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

Why more flexible? Limitations of reflex approach?

Done with propositional logic. Just check at home for syntax and FOL form.

Example: N-Queens, first-order

- **1) no row with two queens**

  $$\forall i, j, k \ \ (1 \leq i, j, k \leq N) \ \ (Queen(i, j) \wedge Queen(i, k)) \Rightarrow (j = k)$$

- **2) no column with two queens**

  $$\forall i, j, k \ \ (1 \leq i, j, k \leq N) \ \ (Queen(j, i) \wedge Queen(k, i)) \Rightarrow (j = k)$$

- **3) no diagonal with two queens**

$$\forall i,j,k,l \;\; (1 \le i,j,k,l \le N) \;\; [(Queen(i,j) \wedge Queen(k,l)$$
$$\wedge leftdiagonal(i,j,k,l)] \Rightarrow ((i=j) \wedge (k=l))$$

similarly for right diagonal.

$$\forall i, j, k, l \;\; (1 \leq i, j, k, l \leq N)$$
$$leftdiagonal(i, j, k, l) \Leftrightarrow [(i - j) = (k - l)]$$

Similarly, for $rightdiagonal$

Complete?

- **4) at least one queen per row**

  $\forall i \, \exists j \ \ (1 \leq i, j \leq N) \ \ Queen(i, j)$

Also discussed earlier. Here some additional axiom details. R&N Section 10.4.2.

Situation Calculus

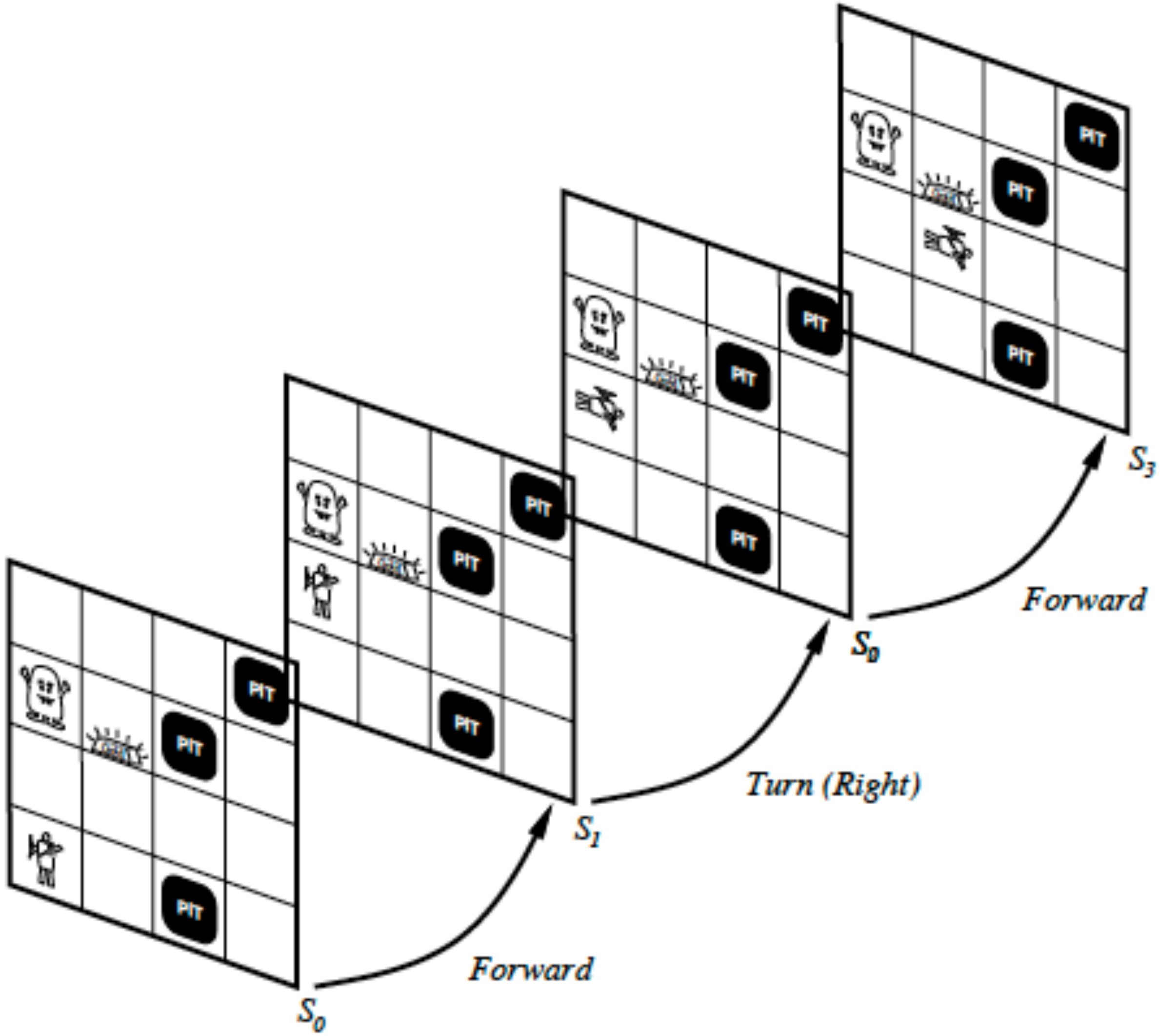
One approach: have time argument

We can be more concise, since we're only interested in when and how things **change**.

Focus on **"situations"**. (**"snapshots"**)

(John McCarthy 1963).

(Changing) World is represented by a series of *situations*.



PIT
PIT
PIT
S_0
Forward
PIT
PIT
PIT
S_1
Turn (Right)
PIT
PIT
PIT
S_0
Forward
PIT
PIT
PIT
S_3

E.g.:

$$At(Agent, [1,1], S_0) \wedge At(Agent, [1,2], S_1)$$

"talking" about change / actions:

$$Result(Forward, S_0) = S1$$

$$Result(Turn(Right), S_1) = S2$$

$$Result(Forward, S_2) = S3$$

Is $Result$ a relation or a function?

What about $Forward$?

Result(action, situation) → situation
(unique outcome)

Effect Axioms

$Portable(Gold)$

$\forall s \quad AtGold(s) \Rightarrow Present(Gold, s)$

$\forall x, s \quad (Present(x, s) \wedge Portable(x)) \Rightarrow Holding(x, Result(Grab, s))$

$\forall x, s \quad \neg Holding(x, Result(Release, s))$

Does this work?

Again, as discussed in propositional case.

Frame Axioms

Need also to state what **doesn't** change!

$$\forall a, x, s \quad Holding(x, s) \wedge (a \neq Release)) \\ \Rightarrow Holding(x, Result(a, x))$$

$$\forall a, x, s \quad (\neg Holding(x, s) \wedge (a \neq grab)) \\ \Rightarrow \neg Holding(x, Result(a, x))$$

More compactly:

$$\forall a, x, s \;\; Holding(x, Results(a, s)) \Leftrightarrow$$
$$[(a = Grab \wedge Present(x, s) \wedge Portable(x))$$
$$\vee (Holding(x, s) \wedge a \neq Release)]$$

successor-state axioms: need to list all the ways in wich any predicate can become true / false.

Frankly, representing and dealing with dynamic / changing
is not a "strong point" of first-order logic.

- work on different logics:
  e.g. dynamic logic / nonmonotonic logic.

nonmon: long "struggle". Yale shooting problem (warn.: R):

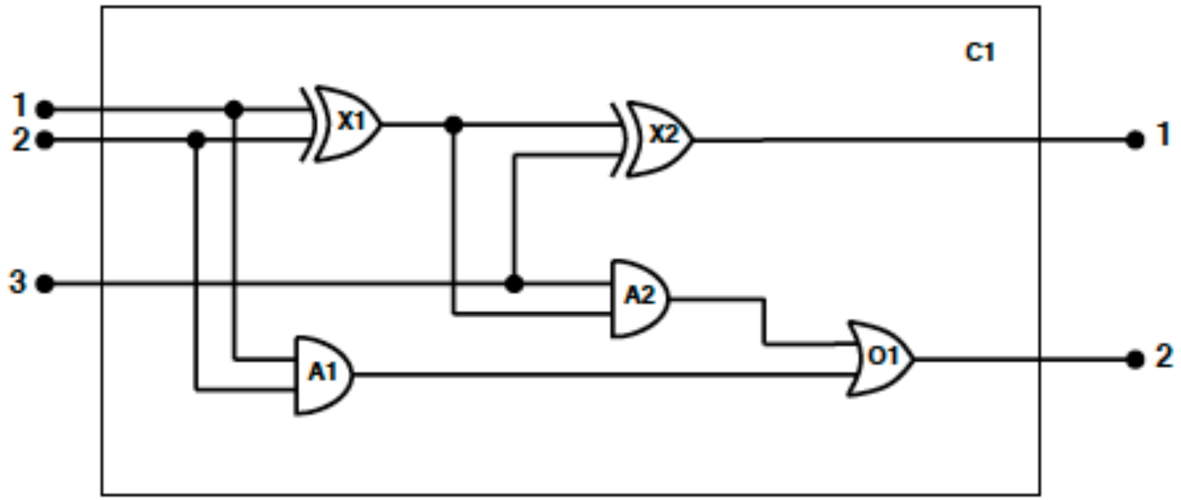
`load gun / point gun / wait 5 seconds / fire gun`

question: target dead? about 100+ research papers ...,
since 1986; still not fully resolved.

first-order logic better at "static" information.

R&N 8.4.2.

Another example: Electronic Circuits



Further illustration of FOL formulation.
Expands on earlier diagnosis example.

Formalization

one-bit full adder: two inputs and a carry / one output and carry

four gates: AND, OR, XOR and NOT.

Goal: analyze design to see if it matches specification.

Consider: circuits (gates and gate types), terminals, and signals.

formalization — keep task in mind.

e.g., for fault diagnosis: might want to specify "wires"

could be broken... (e.g., $Wire(x, y)$)

Always define first and carefully.

Vocabulary

pick: functions / predicates / constants.

constant symbols: X_1, X_2, etc.

type gate: $Type(X_1) = XOR$, note XOR new constant.

Alt.: $Type(X_1, XOR)$. Q. Advantage function?

terminals: $Out(1, X_1), In(1, X_1), In(2, X_1)$.

connectivity: $Connected(Out(1, X_1), In(1, X_2))$.

Note: we don't have to **name** the terminals explicity.

the semantics of the function will assign some unique "object" to it.

(Skolemization can bring back name.)

signal values: function $Signal(x)$, e.g., $Signal(In(1, X_1))$.

signal values: On and Off.

General Rules

how signals behave:

1) $\forall t_1, t_2 \;\; Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$

2a) $\forall t \;\; Signal(t) = On \lor Signal(t) = Off$

2b) $On \neq Off$

3) $\forall t_1, t_2 \;\;\; Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$

how gates behave:

4) $\forall g \;\; Type(g) = OR \Rightarrow$

$$Signal(Out(1, g)) = On \Leftrightarrow \exists n \;\; Signal(In(n, g) = On.$$

5) how AND?

6) $\forall g \;\; Type(g) = XOR \Rightarrow$

$$(Signal(Out(1, g)) = On \Leftrightarrow (Signal(In(1, g) \neq Signal(In(2, g)$$

7) NOT, similarly.

few rules (7): good ontology

clear rules: good vocabulary

what remains?

atomic facts — describing actual circuit under consideration.

types of gates:

$Type(X_1) = XOR,\ \ Type(X_2) = XOR,\ \ Type(A_1) = AND, \ldots$

conectivity:

$$Connected(Out(1, X_1),\ \ In(1, X_2)),$$
$$Connected(In(1, C_1),\ \ In(1, X_1)),$$
$$Connected(Out(1, X_1),\ \ In(1, A_2))$$
$$Connected(In(1, C_1),\ \ In(1, A_1)),$$

etc.

Queries

Our theory captures full behavior.

Can now ask many different queries about behavior etc.

E.g.

$$\exists i_1, i_2, i_3 \;\; Signal(In(1, C_1)) = i_1 \wedge Signal(In(2, C_1)) = i_2 \wedge Signal(In_3,$$

$$\wedge Signal(Out(1, C_1)) = Off \wedge Signal(Out(2, C_1)) = On$$

A.: $(I_1 = On \wedge i_2 = On \wedge i_3 = Off) \vee$

$(I_1 = On \wedge i_2 = Off \wedge i_3 = On) \vee$

$(I_1 = Off \wedge i_2 = On \wedge i_3 = On)$

What is the advantage over direct simulation?

etc.

Aside: previously KB $\models \alpha$ The same here but we want a bit more "detailed answer". Inference will also give us variable bindings if existential query is entailed.

Consistency based diagnosis/abduction, requires addition of "OK" or "Functioning" predicates. See earlier.

Of course, formalization is somewhat facilitated by the "closeness" between logical formalisms and digital circuitr Starting with Shannon, allows for very powerful design methods (but did not prevent Pentium bug...).

Done with prop. logic. Just check for syntax and FOL form.

One more example: Graph Coloring

Graph: N nodes, K colors.

- 1) $\forall i \;\; (1 \leq i \leq N) \;\; \exists j \;\; (1 \leq j \leq K) \;\; Color(i, j)$

  $\forall i, j, l \;\; (1 \leq i \leq N) \;\; (1 \leq j, l \leq K)$

  $[(Color(i, j) \wedge Color(i, l)) \Rightarrow (j = l)]$

- 2) $\forall i, j \;\; (1 \leq i, j \leq N) \;\; [(i \neq j) \Rightarrow$

  $(Edge(i, j) \Rightarrow [\neg \exists \;\; k(1 \leq k \leq K)$

  $((Color(i, k) \wedge Color(j, k))])]$

alternative:

- 3) $\forall i, j \;\; (1 \leq i, j \leq N) \;\; [(i \neq j) \Rightarrow$
  $(Edge(i, j) \Rightarrow [\forall \;\; k(1 \leq k \leq K)$
  $(\neg Color(i, k) \vee \neg Color(j, k))])]$

Now actual graph given by, e.g.,:

- 4) $Edge(1, 3), Edge(2, 4), Edge(5, 6) \ldots$ etc.

reasoning: 3 & 4 gives e.g.:

$$\forall \; k(1 \leq k \leq K) \;\; (\neg Color(1,k) \vee \neg Color(3,k))$$

uses "unification" $\{i/1, j/3\}$with Modus Ponens.

For K = 5, we get:

$$(\neg Color(1,1) \vee \neg Color(3,1)), (\neg Color(1,2) \vee \neg Color(3,2))$$
$$\ldots (\neg Color(1,5) \vee \neg Color(3,5))$$

in propositional form.

uses Universal Elimination, e.g., substitute $\{k/1\}$, etc.

See R&N p. 443
FOL formalizations can be challenging for "everyday" concepts.

Defining **natural kinds** is much more difficult.

e.g. a *game*, or a *chair*.

difficulty with necessary and sufficient conditions.

problem with "strict definition" (Quine 1953)

"the Pope is a bachelor."

Approaches?

Probabilistic representations (extending prop. logic / FOL) can help!

Inference

Resolution / Unification

Chapter 9 R&N.

But for finite domains that are not too large, better to "ground to" propositional and use SAT solver.

Inference

We've considered various first-order formalizations.

But, how do we reason with them? Derive new info?

A. Use **resolution** as in propositional case

From $(\alpha \vee p) \wedge (\neg p \vee \beta)$
conclude $\alpha \vee \beta$ until you reach contradiction.

Need some extra "tricks" to deal with
quantifiers and **variables.**

Resolution

I **put in clausal form**

all variables universally quantified

main trick: "Skolemization" to remove existantials.

idea: invent names for unkown objects known to exist

II **use unification** to match atomic sentences

III **apply resolution rule** to the clausal set combined with negated goal. Attempt to generate empty clause.

Tricks

- **unification**: needed to match variables and terms
  between clauses that look similar
  See also R&N.

- **normalization**: put in **clausal form**
  move quantifiers / $\wedge$ / $\vee$ etc.
  and **Skolemization** — remove $\exists$ by giving
  an arbitrary, but unique name to the object in question.
    E.g. D for the dog owned by Jack.

Unification

UNIFY (P,Q) takes two atomic sentences P and Q and returns a substitution that makes P and Q **look the same.**

Rules for substitutions:

- Can replace a variable by a constant.
- Can replace a variable by a variable.
- Can replace a variable by a function expression, as long as the function expression does not contain the variable.

Unifier: a substitution that makes two clauses resolvable.

$v_1 \rightarrow C; v_2 \rightarrow v_3; v_4 \rightarrow f(...)$

Unification — Purpose

Given: $\forall$ **x (¬ Knows(John, x) $\vee$ Hates(John,x))**

$$Knows(John, x) \rightarrow Hates(John, x)$$
$$Knows(John, Jim)$$

Derive

$$Hates(John, Jim)$$

Need **unifier** $\{x/Jim\}$ before resolution.
(simplest case)

Find the "right" substitution for a universal quantified variable.

$\neg Knows(John, x) \vee Hates(John, x)$ and $Knows(John, Jim)$

How do we resolve? First, match them.

Solution:

$$\text{UNIFY}(Knows(John, x), Knows(John, Jim)) = \{x/Jim\}$$

Gives

$$\neg Knows(John, Jim) \vee Hates(John, Jim) \text{ and}$$
$$Knows(John, Jim)$$

Conclude by resolution

$$Hates(John, Jim)$$

Unification (example)

general rule:

$$Knows(John, x) \rightarrow Hates(John, x)$$

facts:

$$Knows(John, Jim)$$
$$Knows(y, Leo)$$
$$Knows(y, Mother(y))$$
$$Knows(x, Jane)$$

"matching facts to general rules"

Can substitute in because original clause universally quantified.

$$\text{UNIFY}(Knows(John, x), Knows(John, Jim)) = \{x/Jim\}$$

$$\text{UNIFY}(Knows(John, x), Knows(y, Leo)) = \{x/Leo, y/John\}$$

$$\text{UNIFY}(Knows(John, x), Knows(y, Mother(y))) = \{y/John, x/Mother(John)\};$$

$$\text{UNIFY}(Knows(John, x), Knows(x, Jane)) = fail$$

- Last one fails because x can't take on both the value John **and** the value Jane. But intuitively we know that everyone John knows he hates and everyone knows Jane so we should be able to infer that John hates Jane.
- This is why we require, if possible, that every variable has a separate name. Knows(John,x) and Knows(y,Jane) works.

Most General Unifier

In cases where there is more than one substitution choose the one that makes the least commitment (most general) about the bindings.

$$\text{UNIFY}(Knows(John, x), Knows(y, z))$$

$$= \{y/John, x/z\}$$

$$\text{or } \{y/John, x/z, z/Freda\}$$

$$\text{or } \{y/John, x/John, z/John\}$$

or ....

Normal form: Clausal

See also, R&N.

- **Eliminate implication**

  $p \Rightarrow q$ becomes $\neg p \vee q$

- **Move $\neg$ inwards**

  e.g., $\neg(p \vee q)$ becomes $(\neg p \wedge \neg q)$

  $\neg \exists x \,.\, p$ becomes $\forall x \; \neg p$

  $\neg \forall x \,.\, p$ becomes $\ldots$

- **Standarize variables**

  rename variables to avoid conflicts.

- **Move quantifiers left**

  e.g., $p \vee \forall x \; q$ becomes $\forall x \; (p \vee q)$

- **Skolemize** (remove existentials)

  e.g. $\forall x \, Person(x) \Rightarrow \exists y \, Heart(y) \wedge Has(x, y)$

  consider:

  $\forall x \, Person(x) \Rightarrow Heart(H) \wedge Has(x, H)$

  problem??

  $\forall x \, Person(x) \Rightarrow Heart(F(x)) \wedge Has(x, F(x))$

- **Distribute** $\wedge$ **over** $\vee$

  $(a \wedge b) \vee c$ becomes $(a \vee c) \wedge (b \vee c)$

- **Flatten nested conjunctions and disjunctions**

  e.g. $(a \vee b) \vee c$ becomes $(a \vee b \vee c)$

Example

Jack owns a dog.

Every dog owner is an animal lover.

No animal lover kills an animal.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

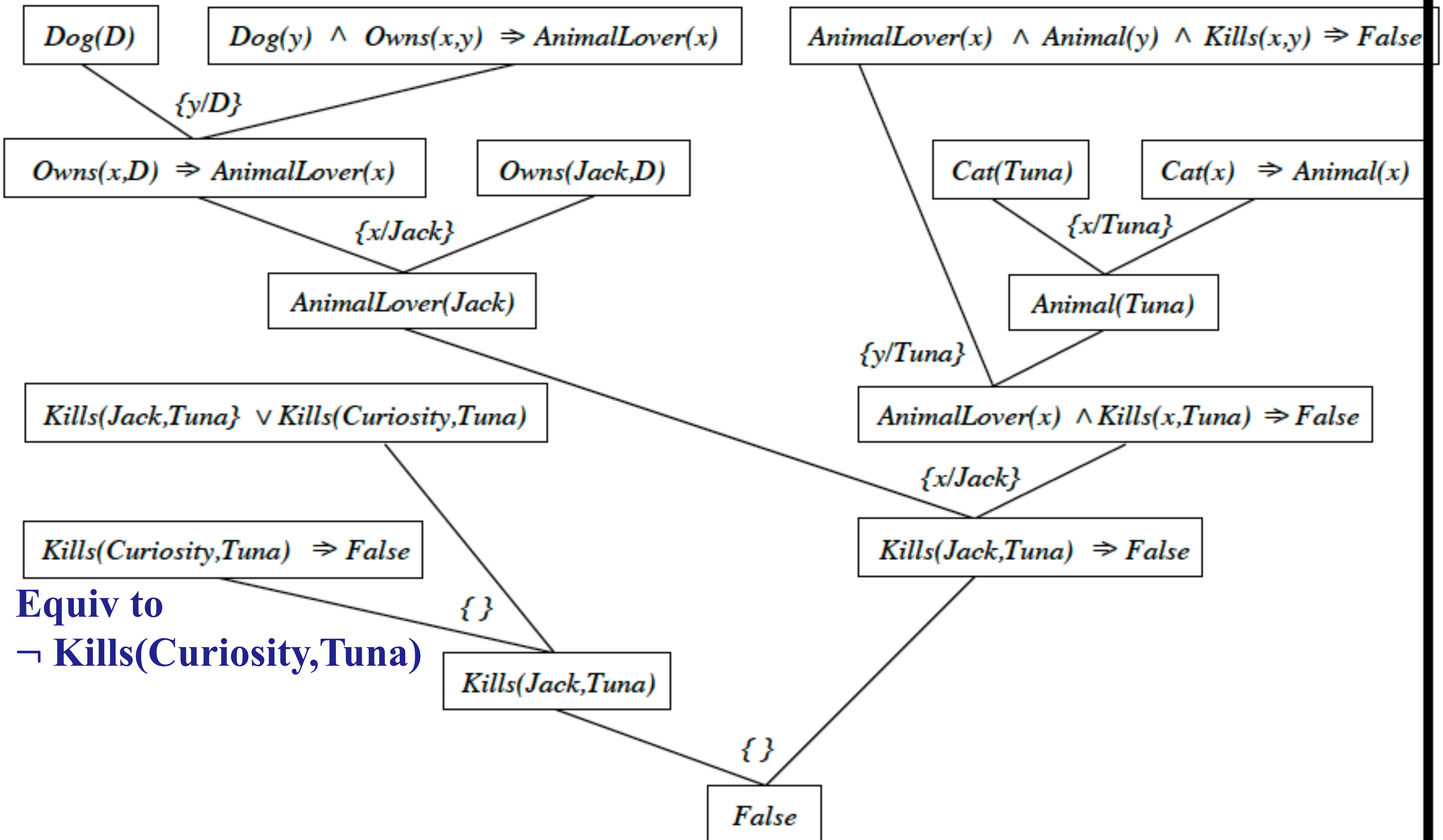
Original Sentences (Plus Background Knowledge)

1. $\exists x : Dog(x) \land Owns(Jack, x)$
2. $\forall x \ \ (\exists y \ \ Dog(y) \land Owns(x, y)) \rightarrow AnimalLover(x)$
3. $\forall x \ \ AnimalLover(x) \rightarrow \forall y \ \ Animal(y) \rightarrow \neg Kills(x, y)$
4. $Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$
5. $Cat(Tuna)$
6. $\forall x \ \ Cat(x) \rightarrow Animal(x)$

Clausal Form

1. $Dog(D)$ (D is the function that finds Jack's dog)
2. $Owns(Jack, D)$
3. $\neg Dog(S(x)) \vee \neg Owns(x, S(x)) \vee AnimalLover(x)$
4. $\neg AnimalLover(w) \vee \neg Animal(y) \vee \neg Kills(w, y)$
5. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
6. $Cat(Tuna)$
7. $\neg Cat(z) \vee Animal(z)$

Proof by Resolution with Refutation



Equiv to
¬ Kills(Curiosity,Tuna)

Completeness

F.O. **Resolution** with **unification** applied to **clausal form**, is **refutation** complete.

Interesting proof! Based on building an "artificial" domain of interpretation, called the **Herbrand universe**.

Practice

Complete in principle, in practice still significant search problem!

Many different search strategies: **resolution strategies**

Schubert Steamroller

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them.

Also there are some grains, and grains are plants.

Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants.

Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are much smaller than wolves.

Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails.

Caterpillars and snails like to eat some plants.

Some logical forms:

$\forall x \; (Wolf(x) \Rightarrow animal(x))$

$\forall x \, \forall y \; ((Caterpillar(x) \wedge Bird(y)) \Rightarrow Smaller(x, y).$

$\exists x \;\; bird(x)$

To prove:

`There is an animal that likes to eat`
`            a grain-eating animal.`

Requires almost 150 resolution steps (minimal)

Significant challenge for early systems.

Open for about 15 years; solved in late 80s.

Relatively straightforward KB can quickly
overwhelm general resolution methods.

Resolution strategies reduce the problem somewhat,
but not completely.

As a consequence, many **practical** Knowledge Represenation
formalisms in AI use a **restricted form**
and **specialized inference.**

Can often understand them in terms of standard
first-order logic! (clear **syntax & semantics**)

Or, better yet, for finite domains, fall back to SAT solvers.