Artificial Intelligence

First-Order Logic

Dr. Bilgin Avenoğlu

First-order Logic

- Propositional logic is limited in what it can say.
- Propositional logic, as a factored representation, lacks the expressive power to concisely describe an environment with many objects.
- For example, we were forced to write a separate rule about breezes and pits for each square, such as

$$\boxed{B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})\,.}$$

- In English, “Squares adjacent to pits are breezy.”

First-order Logic

- *Objects*: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries ...
- Relations: these can be unary relations or properties such as red, round, bogus, prime, multistoried . . ., or more general n-ary relations such as brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
- Functions: relations in which there is only one “value” for a given “input”.
  - father of, best friend, third inning of, one more than, beginning of . . .

- FOL can also express facts about some or all of the objects in the universe.

First-order Logic

- "One plus two equals three."
  Objects: one, two, three, one plus two; Relation: equals; Function: plus. ("One plus two" is a name for the object that is obtained by applying the function "plus" to the objects "one" and "two." "Three" is another name for this object.)
- "Squares neighboring the wumpus are smelly."
  Objects: wumpus, squares; Property: smelly; Relation: neighboring.
- "Evil King John ruled England in 1200."
  Objects: John, England, 1200; Relation: ruled during; Properties: evil, king.

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 $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Figure 8.1 Formal languages and their ontological and epistemological commitments.

- Propositional logic assumes that there are facts that either hold or do not hold in the world.
  - Each fact can be in one of two states—true or false—and each model assigns true or false to each proposition symbol.
- FOL assumes more; the world consists of objects with certain relations among them that do or do not hold.

Models for first-order logic

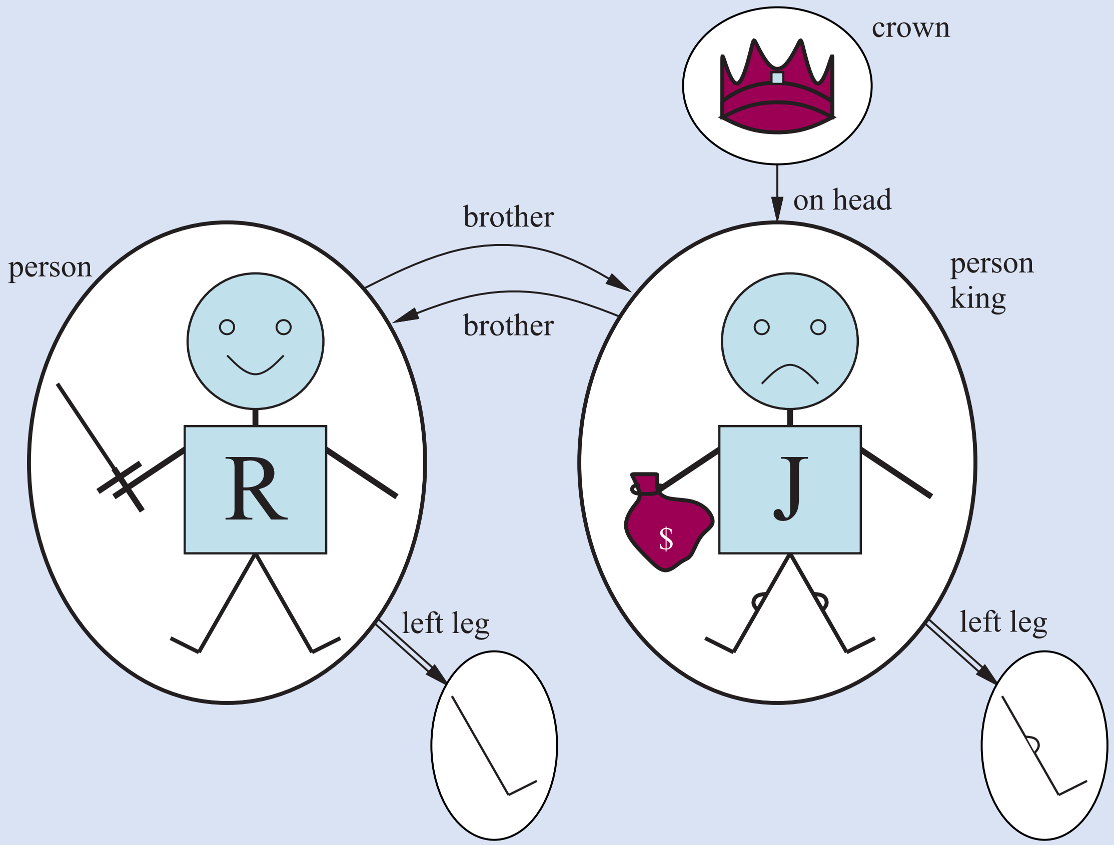


Figure 8.2 A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).

Symbols and interpretations

- The basic syntactic elements of first-order logic are:
  - constant symbols, which stand for objects;
  - predicate symbols, which stand for relations;
  - function symbols, which stand for functions.
- Every model must provide the information required to determine if any given sentence is true or false.

Symbols and interpretations

- *Richard* refers to Richard the Lionheart and John refers to the evil King John.
- *Brother* refers to the brotherhood relation
  - *OnHead* is a relation that holds between the crown and King John;
- *Person*, *King*, and *Crown* are unary relations that identify persons, kings, and crowns.
- *LeftLeg* refers to the “left leg” function

Intended interpretation

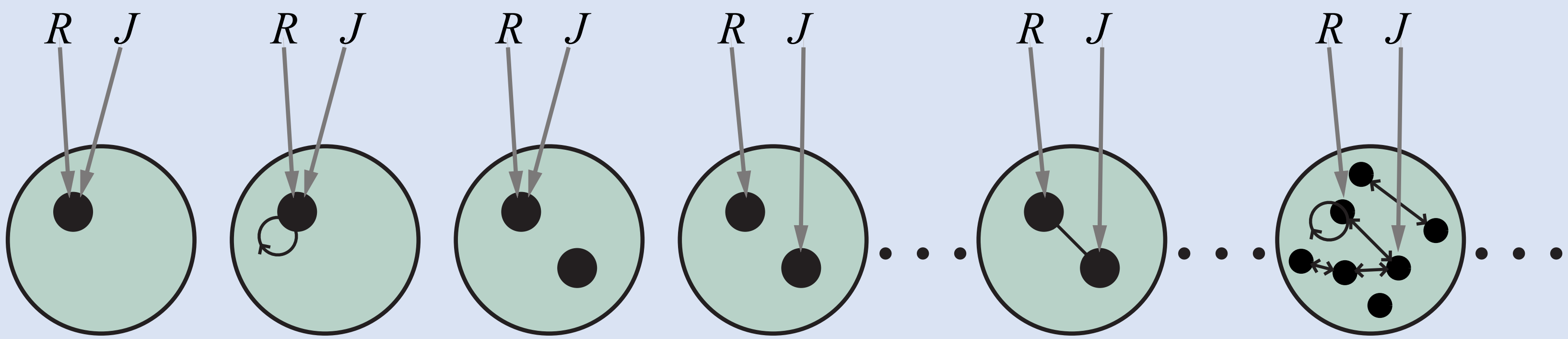


Figure 8.4 Some members of the set of all models for a language with two constant symbols, *R* and *J*, and one binary relation symbol. The interpretation of each constant symbol is shown by a gray arrow. Within each model, the related objects are connected by arrows.

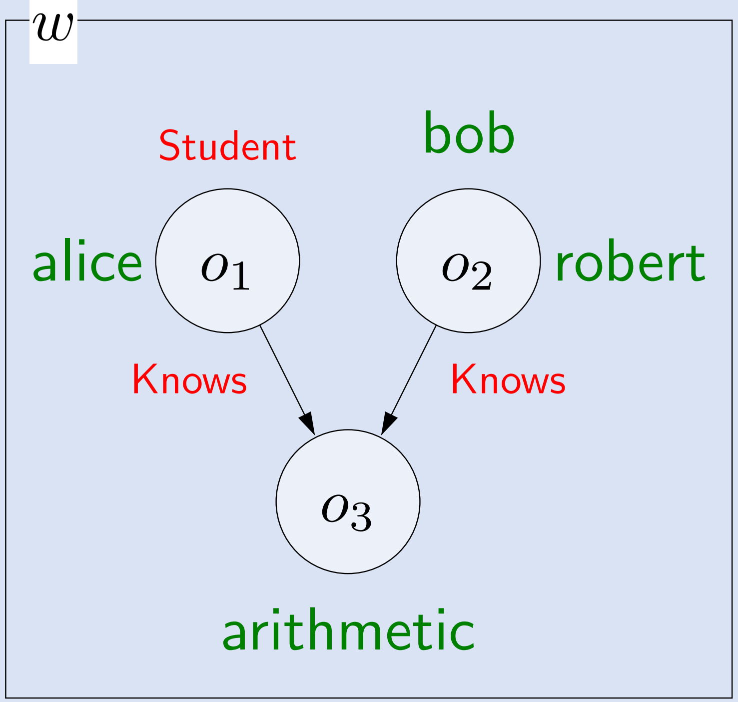
- One interpretation maps Richard to the crown and John to King John's left leg.
- There are five objects in the model, so there are 25 possible interpretations just for the constant symbols Richard and John.

Models in first-order logic

- Recall a model represents a possible situation in the world.
- Propositional logic: Model *w* maps propositional symbols to truth values.
  - w = {AliceKnowsArithmetic : 1, BobKnowsArithmetic : 0}
- First-order logic: ?

Models in first-order logic

- If only have unary and binary predicates, a model *w* can be represented as a directed graph:



- Nodes are objects, labeled with constant symbols
- Directed edges are binary predicates, labeled with predicate symbols;
- Unary predicates are additional node labels

Models in first-order logic

Definition: model in first-order logic

A model w in first-order logic maps:

- constant symbols to objects

$w(\text{alice}) = o_1, w(\text{bob}) = o_2, w(\text{arithmetic}) = o_3$

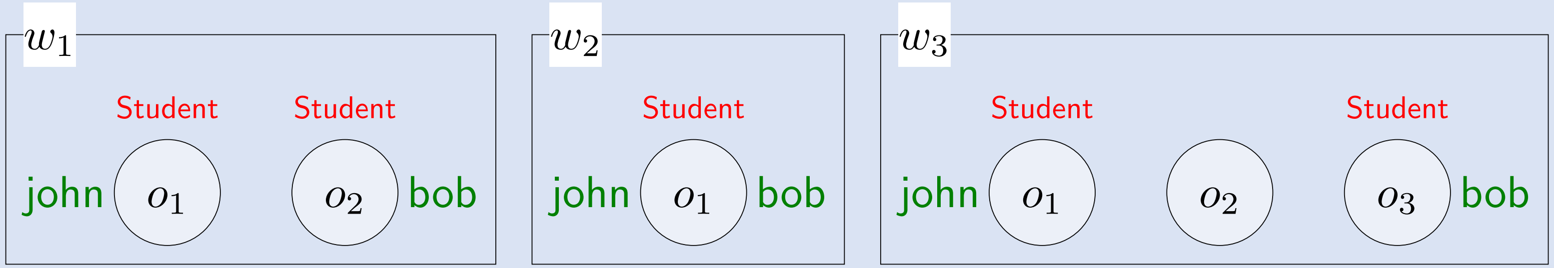
- predicate symbols to tuples of objects

$w(\text{Knows}) = \{(o_1, o_3), (o_2, o_3), \dots\}$

A restriction on models

John and Bob are students.

$$\text{Student}(\text{john}) \wedge \text{Student}(\text{bob})$$



- Unique names assumption: Each object has **at most one** constant symbol. This rules out w_2.
- Domain closure: Each object has **at least one** constant symbol. This rules out w_3.

Point:

constant symbol ⟷ object

Syntax of first-order logic

Terms (refer to objects):

- Constant symbol (e.g., arithmetic)
- Variable (e.g., x)
- Function of terms (e.g., Sum$(3, x)$)

Formulas (refer to truth values):

- Atomic formulas (atoms): predicate applied to terms (e.g., Knows$(x,$arithmetic$)$)
- Connectives applied to formulas (e.g., Student$(x) \rightarrow$ Knows$(x,$arithmetic$)$)
- Quantifiers applied to formulas (e.g., $\forall x\,$Student$(x) \rightarrow$ Knows$(x,$arithmetic$)$)

Terms

- A term is a logical expression that refers to an object.
- Constant symbols are terms, but it is not always convenient to have a distinct symbol to name every object.
  - In English we might use the expression “King John’s left leg” rather than giving a name to his leg.
  - This is what function symbols are for: instead of using a constant symbol, we use *LeftLeg(John)*.

Atomic sentences

- An atomic sentence is formed from a predicate symbol optionally followed by a parenthesized list of terms
  - *Brother(Richard, John).* : Richard the Lionheart is the brother of King John
- Atomic sentences can have complex terms as arguments
  - *Married(Father(Richard), Mother(John))* : Richard the Lionheart's father is married to King John's mother

An atomic sentence is ***true*** *in a given model if the relation referred to by the predicate symbol holds among the objects referred to by the arguments.*

Complex sentences

- We can use logical connectives to construct more complex sentences, as in propositional calculus.

$$\neg Brother(LeftLeg(Richard), John)$$
$$Brother(Richard, John) \land Brother(John, Richard)$$
$$King(Richard) \lor King(John)$$
$$\neg King(Richard) \Rightarrow King(John).$$

Quantifiers - Universal quantification (∀)

- The universal quantifier ∀ is usually pronounced "For all . . .".
- The sentence says, "For all x, if x is a king, then x is a person." The symbol *x* is called a variable.

$$\boxed{\forall x \;\; King(x) \;\Rightarrow\; Person(x)\,.}$$

Quantifiers - Universal quantification (∀)

- The sentence ∀x P, where *P* is any logical sentence, says that *P* is true for every object *x*.
- Precisely, ∀x P is true in a given model if P is true in all possible extended interpretations constructed from the interpretation given in the model, where each extended interpretation specifies a domain element to which x refers.

$$
\begin{aligned}
&x \rightarrow \text{Richard the Lionheart},\\
&x \rightarrow \text{King John},\\
&x \rightarrow \text{Richard's left leg},\\
&x \rightarrow \text{John's left leg},\\
&x \rightarrow \text{the crown}.
\end{aligned}
$$

Quantifiers - Universal quantification (∀)

- The universally quantified sentence $\forall x\ King(x) \Rightarrow Person(x)$ is true in the original model if the sentence King(x) ⇒ Person(x) is true under each of the five extended interpretations.

Richard the Lionheart is a king ⇒ Richard the Lionheart is a person.
King John is a king ⇒ King John is a person.
Richard's left leg is a king ⇒ Richard's left leg is a person.
John's left leg is a king ⇒ John's left leg is a person.
The crown is a king ⇒ the crown is a person.

Looking at the truth table for ⇒ we see that the implication is true whenever its premise is false —*regardless* of the truth of the conclusion.

Quantifiers - Universal quantification (∀)

- A common mistake, is to use conjunction instead of implication.

$$\boxed{\forall x \;\; King(x) \land Person(x)}$$

$$\boxed{\begin{array}{l}\text{Richard the Lionheart is a king} \land \text{Richard the Lionheart is a person,}\\ \text{King John is a king} \land \text{King John is a person,}\\ \text{Richard's left leg is a king} \land \text{Richard's left leg is a person,}\end{array}}$$

Quantifiers - Existential quantification (∃)

- We can make a statement about some object without naming it, by using an existential quantifier.
- $\exists x$ is pronounced "There exists an x such that ..." or "For some x...".
- $\exists x\ P$ says that P is true for at least one object x.
- $\exists x\ P$ is true in a given model if P is true in at least one extended interpretation that assigns x to a domain element.

Richard the Lionheart is a crown $\wedge$ Richard the Lionheart is on John's head;
King John is a crown $\wedge$ King John is on John's head;
Richard's left leg is a crown $\wedge$ Richard's left leg is on John's head;
John's left leg is a crown $\wedge$ John's left leg is on John's head;
The crown is a crown $\wedge$ the crown is on John's head.

Quantifiers - Existential quantification (∃)

- Just as ⇒ appears to be the natural connective to use with ∀, ∧ is the natural connective to use with ∃.
- Using ⇒ with ∃ usually leads to a very weak statement.

$$\boxed{\exists x \;\; Crown(x) \;\Rightarrow\; OnHead(x, John)\,.}$$

Richard the Lionheart is a crown $\Rightarrow$ Richard the Lionheart is on John's head;
King John is a crown $\Rightarrow$ King John is on John's head;
Richard's left leg is a crown $\Rightarrow$ Richard's left leg is on John's head;

Nested quantifiers

- "Brothers are siblings"

$$\forall x \ \forall y \ \ Brother(x,y) \ \Rightarrow \ Sibling(x,y)\,.$$

- Consecutive quantifiers of the same type can be written as one quantifier with several variables.

$$\forall x,y \ \ Sibling(x,y) \ \Leftrightarrow \ Sibling(y,x)\,.$$

- "Everybody loves somebody"

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

- "There is someone who is loved by everyone"

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

Connections between ∀ and ∃

- The two quantifiers are actually intimately connected with each other, through negation.

$$\forall x\ \neg Likes(x, Parsnips) \quad \text{is equivalent to} \quad \neg\exists x\ Likes(x, Parsnips)\,.$$

- "Everyone likes ice cream" means that there is no one who does not like ice cream.

$$\forall x\ Likes(x, IceCream) \quad \text{is equivalent to} \quad \neg\exists x\ \neg Likes(x, IceCream)\,.$$

- Because ∀ is really a conjunction over the universe of objects and ∃ is a disjunction, it should not be surprising that they obey De Morgan's rules.

$$\begin{array}{lll} \neg\exists x\ P & \equiv & \forall x\ \neg P \\ \neg\forall x\ P & \equiv & \exists x\ \neg P \\ \forall x\ P & \equiv & \neg\exists x\ \neg P \\ \exists x\ P & \equiv & \neg\forall x\ \neg P \end{array} \qquad\qquad \begin{array}{lll} \neg(P \vee Q) & \equiv & \neg P \wedge \neg Q \\ \neg(P \wedge Q) & \equiv & \neg P \vee \neg Q \\ P \wedge Q & \equiv & \neg(\neg P \vee \neg Q) \\ P \vee Q & \equiv & \neg(\neg P \wedge \neg Q)\,. \end{array}$$

Equality

- We can use the equality symbol to signify that two terms refer to the same object.

$$\boxed{Father(John) = Henry}$$

- The equality symbol can be used to state facts about a given function, as we just did for the Father symbol.
- It can also be used with negation to insist that two terms are not the same object.

$$\boxed{\exists x, y \;\; Brother(x, Richard) \wedge Brother(y, Richard) \wedge \neg(x = y)\,.}$$

Database semantics

- We believe that Richard has two brothers, John and Geoffrey.

$$\boxed{Brother(John, Richard) \land Brother(Geoffrey, Richard)\,,}$$

- First, this assertion is true in a model where Richard has only one brother—we need to add John ≠ Geoffrey.
- Second, the sentence doesn't rule out models in which Richard has many more brothers besides John and Geoffrey.
- "Richard's brothers are John and Geoffrey"

$$\boxed{\begin{array}{l} Brother(John, Richard) \land Brother(Geoffrey, Richard) \land John \neq Geoffrey \\ \quad \land \forall x \;\; Brother(x, Richard) \;\Rightarrow\; (x = John \lor x = Geoffrey)\,. \end{array}}$$

Database semantics

- First, we insist that every constant symbol refer to a distinct object—the unique-names assumption.
- Second, we assume that atomic sentences not known to be true are in fact false - the closed-world assumption.
- Third, we invoke domain closure, meaning that each model contains no more domain elements than those named by the constant symbols.

Using First-Order Logic

- Assertions and queries in first-order logic
  - Sentences are added to a knowledge base using *TELL*, exactly as in propositional logic. Such sentences are called assertions.
- John is a king, Richard is a person, and all kings are persons:

$$\text{TELL}(KB, King(John)).$$
$$\text{TELL}(KB, Person(Richard)).$$
$$\text{TELL}(KB, \forall x\ King(x) \Rightarrow Person(x)).$$

- We can ask questions of the knowledge base using *ASK*.

$$\text{ASK}(KB, King(John))$$ true

- Questions asked with *ASK* are called queries or goals.

$$\text{ASK}(KB, Person(John))$$ true $$\text{ASK}(KB, \exists x\ Person(x)).$$ true

Assertions and queries in first-order logic

- If we want to know what value of x makes the sentence true, we will need a different function, which we call *ASKVARS*,

$$\text{ASKVARS}(KB, Person(x))$$

- There are two answers: {x/John} and {x/Richard}

$$\text{TELL}(KB, King(John))\,.$$
$$\text{TELL}(KB, Person(Richard))\,.$$
$$\text{TELL}(KB, \forall x\ \ King(x) \ \Rightarrow\ Person(x))\,.$$

Some examples of first-order logic

There is some course that every student has taken.

$$\exists y\, \mathsf{Course}(y) \wedge [\forall x\, \mathsf{Student}(x) \rightarrow \mathsf{Takes}(x, y)]$$

Every even integer greater than 2 is the sum of two primes.

$$\forall x\, \mathsf{EvenInt}(x) \wedge \mathsf{Greater}(x, 2) \rightarrow \exists y\, \exists z\, \mathsf{Equals}(x, \mathsf{Sum}(y, z)) \wedge \mathsf{Prime}(y) \wedge \mathsf{Prime}(z)$$

If a student takes a course and the course covers a concept, then the student knows that concept.

$$\forall x\, \forall y\, \forall z\, (\mathsf{Student}(x) \wedge \mathsf{Takes}(x, y) \wedge \mathsf{Course}(y) \wedge \mathsf{Covers}(y, z)) \rightarrow \mathsf{Knows}(x, z)$$

The wumpus world

- A typical percept : Percept is a binary predicate, and Stench and so on are constants placed in a list.

$$\boxed{Percept([Stench, Breeze, Glitter, None, None], 5)\,.}$$

- The actions in the wumpus world can be represented by logical terms:

$$\boxed{Turn(Right),\;\; Turn(Left),\;\; Forward,\;\; Shoot,\;\; Grab,\;\; Climb\,.}$$

- To determine which action is best, the agent program executes the query which returns a binding list such as {a/Grab}

$$\boxed{\textsc{AskVars}(KB, BestAction(a, 5))\,,}$$

The wumpus world

- The raw percept data implies certain facts about the current state:

$$
\begin{array}{l}
\forall t,s,g,w,c \;\; Percept([s,Breeze,g,w,c],t) \;\Rightarrow\; Breeze(t) \\
\forall t,s,g,w,c \;\; Percept([s,None,g,w,c],t) \;\Rightarrow\; \neg Breeze(t) \\
\forall t,s,b,w,c \;\; Percept([s,b,Glitter,w,c],t) \;\Rightarrow\; Glitter(t) \\
\forall t,s,b,w,c \;\; Percept([s,b,None,w,c],t) \;\Rightarrow\; \neg Glitter(t)
\end{array}
$$

- Simple “reflex” behavior can also be implemented by quantified implication sentences.

$$\forall t \;\; Glitter(t) \;\Rightarrow\; BestAction(Grab,t)\,.$$

- Adjacency of any two squares can be defined as

$$
\begin{array}{l}
\forall x,y,a,b \;\; Adjacent([x,y],[a,b]) \;\Leftrightarrow \\
\qquad (x=a \wedge (y=b-1 \vee y=b+1)) \vee (y=b \wedge (x=a-1 \vee x=a+1))\,.
\end{array}
$$

The wumpus world

- The agent's location changes over time, so we write *At(Agent,s,t)* to mean that the agent is at square s at time t.

$$\boxed{\forall t\, At(Wumpus, [1,3], t).}$$

- We can then say that objects can be at only one location at a time:

$$\boxed{\forall x, s_1, s_2, t \;\; At(x, s_1, t) \wedge At(x, s_2, t) \;\Rightarrow\; s_1 = s_2\,.}$$

- Given its current location, the agent can infer properties of the square from properties of its current percept.

$$\boxed{\forall s, t \;\; At(Agent, s, t) \wedge Breeze(t) \;\Rightarrow\; Breezy(s)\,.}$$

The wumpus world

- Whereas propositional logic necessitates a separate axiom for each square and would need a different set of axioms for each geographical layout of the world, first-order logic just needs one axiom:

$$\boxed{\begin{array}{llll} R_2: & B_{1,1} & \Leftrightarrow & (P_{1,2} \vee P_{2,1}). \\ R_3: & B_{2,1} & \Leftrightarrow & (P_{1,1} \vee P_{2,2} \vee P_{3,1}). \end{array}} \longrightarrow \boxed{\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r,s) \wedge Pit(r).}$$

- In FOL, we can quantify over time, so we need just one successor-state axiom for each predicate, rather than a different copy for each time step.

$$\boxed{HaveArrow^{t+1} \Leftrightarrow (HaveArrow^t \wedge \neg Shoot^t).} \longrightarrow \boxed{\forall t\ HaveArrow(t+1) \Leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot,t)).}$$

INFERENCE IN FIRST-ORDER LOGIC

Propositionalization

If one-to-one mapping between constant symbols and objects (**unique names** and **domain closure**),

first-order logic is syntactic sugar for propositional logic:

Knowledge base in first-order logic

$$\text{Student}(\text{alice}) \wedge \text{Student}(\text{bob})$$
$$\forall x\, \text{Student}(x) \to \text{Person}(x)$$
$$\exists x\, \text{Student}(x) \wedge \text{Creative}(x)$$

Knowledge base in propositional logic

$$\text{Studentalice} \wedge \text{Studentbob}$$
$$(\text{Studentalice} \to \text{Personalice}) \wedge (\text{Studentbob} \to \text{Personbob})$$
$$(\text{Studentalice} \wedge \text{Creativealice}) \vee (\text{Studentbob} \wedge \text{Creativebob})$$

Point: use any inference algorithm for propositional logic!

Substitution

- Suppose our KB contains the axiom that all greedy kings are evil:

$$\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)\,.$$

From that we can infer any of the following sentences:

$$\begin{aligned}
&King(John) \wedge Greedy(John) \;\Rightarrow\; Evil(John)\\
&King(Richard) \wedge Greedy(Richard) \;\Rightarrow\; Evil(Richard)\\
&King(Father(John)) \wedge Greedy(Father(John)) \;\Rightarrow\; Evil(Father(John))\,.\\
&\quad\vdots
\end{aligned}$$

The rule of Universal Instantiation (UI for short) says that we can infer any sentence obtained by substituting a ground term (a term without variables) for a universally quantified variable.

$$\frac{\forall v \;\; \alpha}{\textsc{Subst}(\{v/g\}, \alpha)} \longrightarrow \{x/John\},\ \{x/Richard\},\ \text{and}\ \{x/Father(John)\}.$$

Substitution

The rule of Existential Instantiation replaces an existentially quantified variable with a single *new constant symbol*.

$$\frac{\exists v \;\; \alpha}{\textsc{Subst}(\{v/k\},\alpha)} \, .$$

For example, from the sentence

$$\exists x \;\; Crown(x) \land OnHead(x, John)$$

we can infer the sentence

$$Crown(C_1) \land OnHead(C_1, John)$$

Skolem constant.

Substitution

- Whereas Universal Instantiation can be applied many times to the same axiom to produce many different consequences, Existential Instantiation need only be applied once, and then the existentially quantified sentence can be discarded.
  - We no longer need $\exists x\ Kill(x,Victim)$ once we have added the sentence Kill(Murderer,Victim).

Unification

- Generalized Modus Ponens: For atomic sentences p_i, p_i', and q, where there is a substitution θ such that

$$\boxed{\begin{array}{l}\textsc{Subst}(\theta, p_i') = \textsc{Subst}(\theta, p_i), \text{ for all } i, \\ \dfrac{p_1',\ p_2',\ \ldots,\ p_n',\ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{\textsc{Subst}(\theta, q)}\ .\end{array}}$$

$$\boxed{\forall x\ King(x) \land Greedy(x)\ \Rightarrow\ Evil(x)} \qquad \boxed{\forall y\ Greedy(y)\,.}$$

p_1' is $King(John)$	p_1 is $King(x)$
p_2' is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	q is $Evil(x)$
$\textsc{Subst}(\theta, q)$ is $Evil(John)$.	

Example: modus ponens in first-order logic

Premises:

$\text{Takes}(\text{alice}, \text{cs221})$

$\text{Covers}(\text{cs221}, \text{mdp})$

$\forall x\, \forall y\, \forall z\, \text{Takes}(x, y) \wedge \text{Covers}(y, z) \rightarrow \text{Knows}(x, z)$

Conclusion:

$\theta = \{x/\text{alice}, y/\text{cs221}, z/\text{mdp}\}$

Derive $\text{Knows}(\text{alice}, \text{mdp})$

Unification

- The *UNIFY* algorithm takes two sentences and returns a unifier for them (a substitution) if one exists:

$$\boxed{\text{UNIFY}(p,q) = \theta \text{ where } \text{SUBST}(\theta,p) = \text{SUBST}(\theta,q)\,.}$$

$$\boxed{AskVars(Knows(John,x)):\ \text{whom does John know?}}$$

$$\begin{array}{|l|}\hline
\text{UNIFY}(Knows(John,x),\ Knows(John,Jane)) = \{x/Jane\} \\
\text{UNIFY}(Knows(John,x),\ Knows(y,Bill)) = \{x/Bill, y/John\} \\
\text{UNIFY}(Knows(John,x),\ Knows(y,Mother(y))) = \{y/John, x/Mother(John)\} \\
\text{UNIFY}(Knows(John,x),\ Knows(x,Elizabeth)) = failure\,. \\ \hline
\end{array}$$

$$\text{UNIFY}(Knows(John,x),\ Knows(x_{17},Elizabeth)) = \{x/Elizabeth, x_{17}/John\}\,.$$

Forward Chaining

- First-order definite clauses are disjunctions of literals of which exactly one is positive.
- That means a definite clause is either atomic, or is an implication whose antecedent is a conjunction of positive literals and whose consequent is a single positive literal.
- Existential quantifiers are not allowed, and universal quantifiers are left implicit:
  - If you see an x in a definite clause, that means there is an implicit ∀x quantifier.
  - A typical first-order definite clause looks like this:

$$\boxed{King(x) \wedge Greedy(x) \Rightarrow Evil(x)\,,}$$

  - The literals *King(John)* and Greedy(y) also count as definite clauses.

Example

- “The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.”

- “The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.”

First, we will represent these facts as first-order definite clauses:

“... it is a crime for an American to sell weapons to hostile nations”:

$$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x) . \tag{9.3}$$

“Nono ... has some missiles.” The sentence $\exists x\ Owns(Nono, x) \land Missile(x)$ is transformed into two definite clauses by Existential Instantiation, introducing a new constant M_1:

$$Owns(Nono, M_1) \tag{9.4}$$

$$Missile(M_1) \tag{9.5}$$

“All of its missiles were sold to it by Colonel West”:

$$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono) . \tag{9.6}$$

- “The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.”

We will also need to know that missiles are weapons:

$$Missile(x) \Rightarrow Weapon(x) \tag{9.7}$$

and we must know that an enemy of America counts as “hostile”:

$$Enemy(x, America) \Rightarrow Hostile(x)\,. \tag{9.8}$$

“West, who is American …”:

$$American(West)\,. \tag{9.9}$$

“The country Nono, an enemy of America …”:

$$Enemy(Nono, America)\,. \tag{9.10}$$

Forward chaining

- Starting from the known facts, it triggers all the rules whose premises are satisfied, adding their conclusions to the known facts.
- The process repeats until the query is answered (assuming that just one answer is required) or no new facts are added.

Forward chaining

$$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)\,. \tag{9.3}$$

$$Owns(Nono, M_1) \tag{9.4}$$

$$Missile(M_1) \tag{9.5}$$

$$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)\,. \tag{9.6}$$

$$Missile(x) \Rightarrow Weapon(x) \tag{9.7}$$

$$Enemy(x, America) \Rightarrow Hostile(x)\,. \tag{9.8}$$

$$American(West)\,. \tag{9.9}$$

$$Enemy(Nono, America)\,. \tag{9.10}$$

- On the first iteration, rule (9.3) has unsatisfied premises.
  Rule (9.6) is satisfied with $\{x/M_1\}$, and $Sells(West, M_1, Nono)$ is added.
  Rule (9.7) is satisfied with $\{x/M_1\}$, and $Weapon(M_1)$ is added.
  Rule (9.8) is satisfied with $\{x/Nono\}$, and $Hostile(Nono)$ is added.
- On the second iteration, rule (9.3) is satisfied with $\{x/West, y/M_1, z/Nono\}$, and the inference $Criminal(West)$ is added.

Forward chaining proof

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$.

$Owns(Nono, M_1)$

$Missile(M_1)$

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$.

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$.

$Enemy(x,America) \Rightarrow Hostile(x)$.

$American(West)$.

$Enemy(Nono,America)$.

American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

Forward chaining proof

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$.

$Owns(Nono, M_1)$

$Missile(M_1)$

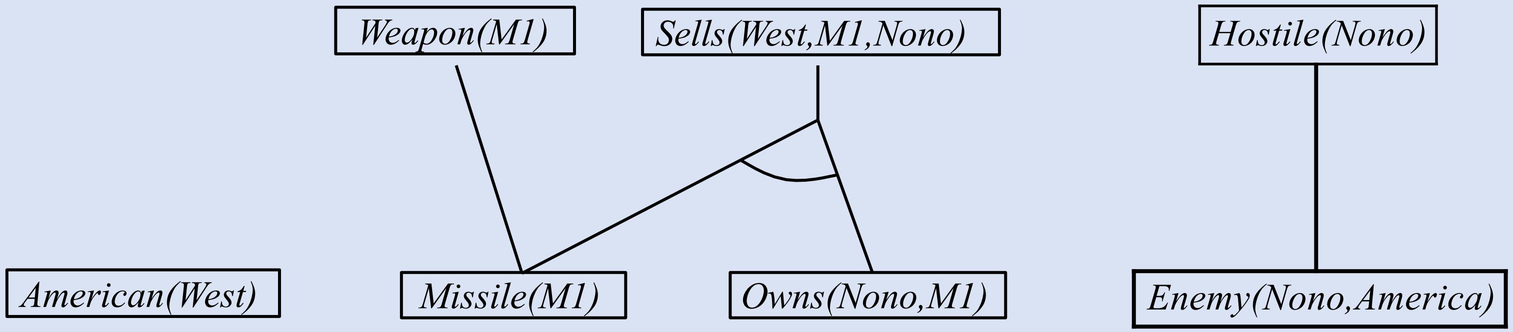
$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$.

$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$.

$Enemy(x,America) \Rightarrow Hostile(x)$.

$American(West)$.

$Enemy(Nono,America)$.



Forward chaining proof

$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$.

$Owns(Nono, M_1)$

$Missile(M_1)$

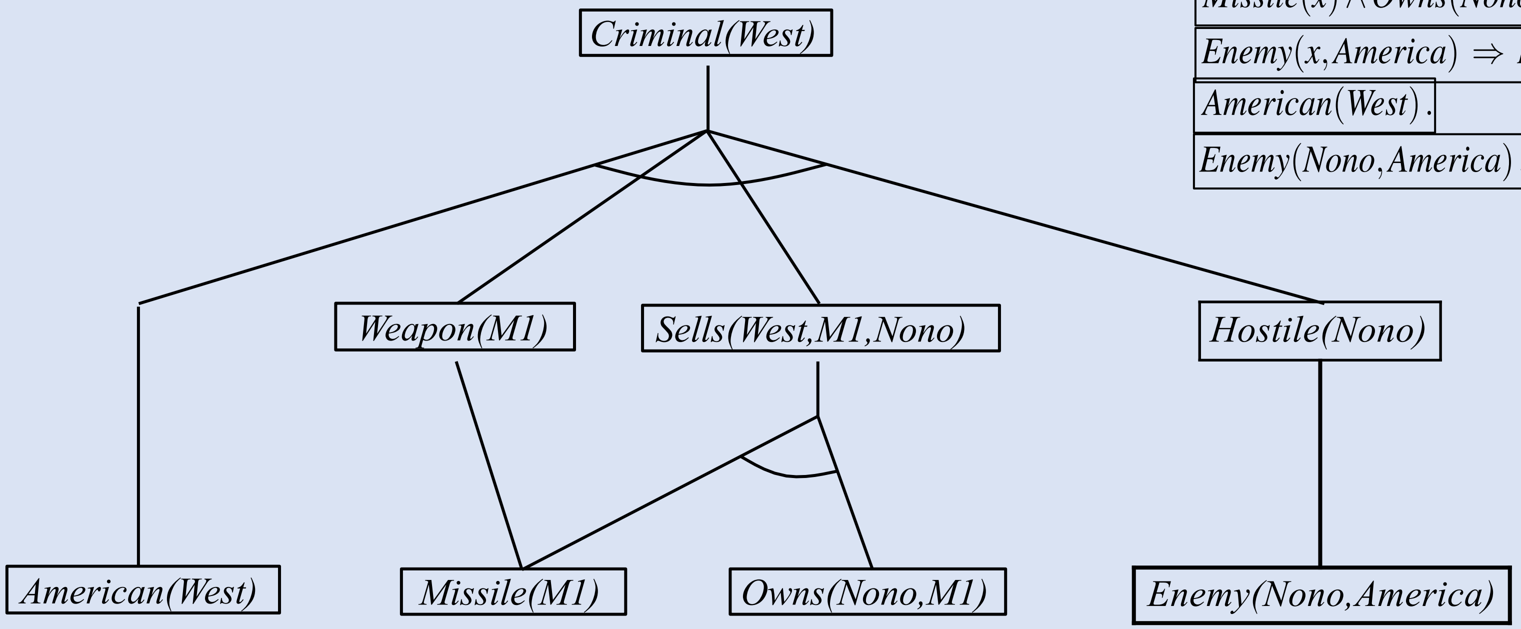
$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$.

$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$.

$Enemy(x, America) \Rightarrow Hostile(x)$.

$American(West)$.

$Enemy(Nono, America)$.



Resolution

Recall: First-order logic includes non-Horn clauses

$$\forall x\, \mathsf{Student}(x) \rightarrow \exists y\, \mathsf{Knows}(x, y)$$

High-level strategy (same as in propositional logic):

- Convert all formulas to CNF
- Repeatedly apply resolution rule

Resolution - Conversion to CNF

- Everyone who loves all animals is loved by someone

$$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)] .$$

Eliminate implications: Replace $P \Rightarrow Q$ with $\neg P \vee Q$. For our sample sentence, this needs to be done twice:

$$\forall x\ \neg[\forall y\ Animal(y) \Rightarrow Loves(x,y)] \vee [\exists y\ Loves(y,x)]$$
$$\forall x\ \neg[\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)] .$$

Resolution - Conversion to CNF

$$\boxed{\forall x \;\neg[\forall y \;\neg Animal(y) \vee Loves(x,y)] \vee [\exists y \;Loves(y,x)]\,.}$$

Move $\neg$ inwards: In addition to the usual rules for negated connectives, we need rules for negated quantifiers. Thus, we have

$$\begin{array}{lll} \neg\forall x \;p & \text{becomes} & \exists x \;\neg p \\ \neg\exists x \;p & \text{becomes} & \forall x \;\neg p\,. \end{array}$$

Our sentence goes through the following transformations:

$$\forall x \;[\exists y \;\neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y \;Loves(y,x)]\,.$$
$$\forall x \;[\exists y \;\neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y \;Loves(y,x)]\,.$$
$$\forall x \;[\exists y \;Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y \;Loves(y,x)]\,.$$

- The sentence now reads "Either there is some animal that x doesn't love, or (if this is not the case) someone loves x."

Resolution - Conversion to CNF

Standardize variables: For sentences like $(\exists x P(x)) \vee (\exists x Q(x))$ that use the same variable name twice, change the name of one of the variables. This avoids confusion later when we drop the quantifiers. Thus, we have

$$\forall x \ [\exists y \ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z \ Loves(z,x)] .$$

Resolution - Conversion to CNF

$$\boxed{\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y)] \lor [\exists z\ Loves(z,x)]\,.}$$

Skolemize: **Skolemization** is the process of removing existential quantifiers by elimination. In the simple case, it is just like the Existential Instantiation rule of Section 9.1: translate $\exists x\, P(x)$ into $P(A)$, where A is a new constant. However, we can't apply Existential Instantiation to our sentence above because it doesn't match the pattern $\exists v\ \alpha$; only parts of the sentence match the pattern. If we blindly apply the rule to the two matching parts we get

$$\forall x\ [Animal(A) \land \neg Loves(x,A)] \lor Loves(B,x)\,,$$

which has the wrong meaning entirely: it says that everyone either fails to love a particular animal A or is loved by some particular entity B. In fact, our original sentence allows each person to fail to love a different animal or to be loved by a different person. Thus, we want the Skolem entities to depend on x:

$$\forall x\ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)\,.$$

Resolution - Conversion to CNF

Drop universal quantifiers: At this point, all remaining variables must be universally quantified. Therefore, we don't lose any information if we drop the quantifier:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x).$$

Distribute $\lor$ **over** $\land$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)].$$

Resolution

Definition: resolution rule (first-order logic)

$$\frac{f_1 \vee \cdots \vee f_n \vee p, \qquad \neg q \vee g_1 \vee \cdots \vee g_m}{\text{Subst}[\theta, f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m]}$$

where $\theta = \text{Unify}[p, q]$.

Example: resolution

$$\frac{\text{Animal}(Y(x)) \vee \text{Loves}(Z(x), x), \qquad \neg\text{Loves}(u, v) \vee \text{Feeds}(u, v)}{\text{Animal}(Y(x)) \vee \text{Feeds}(Z(x), x)}$$

Substitution: $\theta = \{u/Z(x), v/x\}$.

Example

$KB \vDash \alpha$, we show that
($KB \wedge \neg\alpha$) is unsatisfiable

- If something is intelligent, it has common sense
- Deep Blue does not have common sense
- Prove that Deep Blue is not intelligent

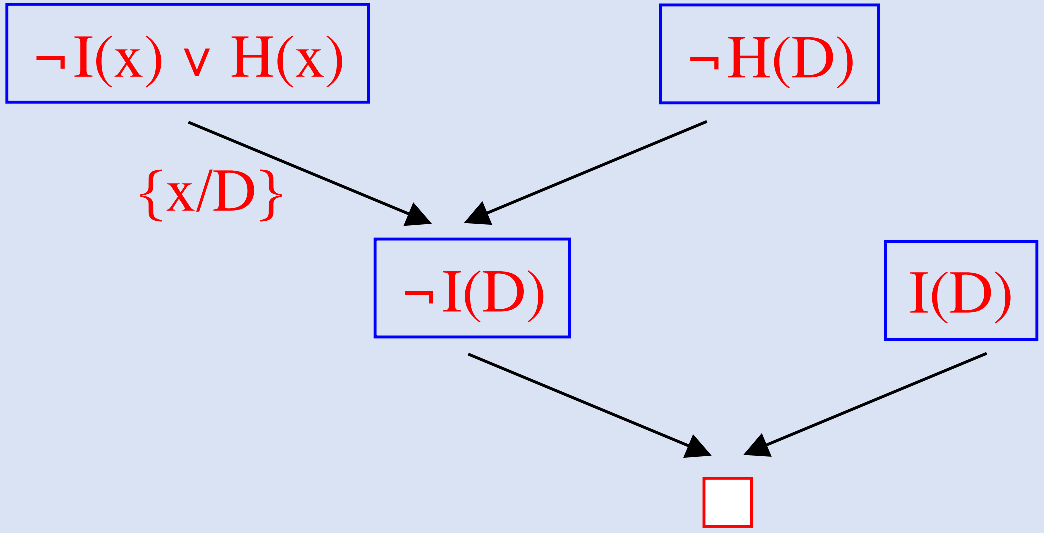
1. $\forall x. I(x) \Rightarrow H(x)$
2. $\neg H(D)$

Conclusion: $\neg I(D)$
Denial: C3: $I(D)$

C1: $\neg I(x) \vee H(x)$
C2: $\neg H(D)$

CNF

A resolution proof of $\neg I(D)$:



Example

1. All hounds howl at night.
2. Anyone who has any cats will not have any mice.
3. Light sleepers do not have anything which howls at night.
4. John has either a cat or a hound.
5. (Conclusion) If John is a light sleeper, then John does not have any mice.

1. $\forall x\,(HOUND(x) \rightarrow HOWL(x))$
2. $\forall x\, \forall y\,(HAVE\,(x,y) \wedge CAT\,(y) \rightarrow \neg \exists z\,(HAVE(x,z) \wedge MOUSE\,(z)))$
3. $\forall x\,(LS(x) \rightarrow \neg \exists y\,(HAVE\,(x,y) \wedge HOWL(y)))$
4. $\exists x\,(HAVE\,(John,x) \wedge (CAT(x) \vee HOUND(x)))$
5. $LS(John) \rightarrow \neg \exists z\,(HAVE(John,z) \wedge MOUSE(z))$

1. $\forall x\,(HOUND(x) \rightarrow HOWL(x))$

2. $\forall x\, \forall y\,(HAVE\,(x,y) \wedge CAT\,(y) \rightarrow \neg\, \exists z\,(HAVE(x,z) \wedge MOUSE\,(z)))$

3. $\forall x\,(LS(x) \rightarrow \neg\, \exists y\,(HAVE\,(x,y) \wedge HOWL(y)))$

4. $\exists x\,(HAVE\,(John,x) \wedge (CAT(x) \vee HOUND(x)))$

5. $LS(John) \rightarrow \neg\, \exists z\,(HAVE(John,z) \wedge MOUSE(z))$

1. $\forall x\,(HOUND(x) \rightarrow HOWL(x))$

   $\neg\, HOUND(x) \vee HOWL(x)$

2. $\forall x\, \forall y\,(HAVE\,(x,y) \wedge CAT\,(y) \rightarrow \neg\, \exists z\,(HAVE(x,z) \wedge MOUSE\,(z)))$

   $\forall x\, \forall y\,(HAVE\,(x,y) \wedge CAT\,(y) \rightarrow \forall z\, \neg\,(HAVE(x,z) \wedge MOUSE\,(z)))$

   $\forall x\, \forall y\, \forall z\,(\neg\,(HAVE\,(x,y) \wedge CAT\,(y)) \vee \neg\,(HAVE(x,z) \wedge MOUSE\,(z)))$

   $\neg\, HAVE(x,y) \vee \neg\, CAT(y) \vee \neg\, HAVE(x,z) \vee \neg\, MOUSE(z)$

3. $\forall x\,(LS(x) \rightarrow \neg\, \exists y\,(HAVE\,(x,y) \wedge HOWL(y)))$

   $\forall x\,(LS(x) \rightarrow \forall y\, \neg\,(HAVE\,(x,y) \wedge HOWL(y)))$

   $\forall x\, \forall y\,(LS(x) \rightarrow \neg\, HAVE(x,y) \vee \neg\, HOWL(y))$

   $\forall x\, \forall y\,(\neg\, LS(x) \vee \neg\, HAVE(x,y) \vee \neg\, HOWL(y))$

   $\neg\, LS(x) \vee \neg\, HAVE(x,y) \vee \neg\, HOWL(y)$

4. $\exists x\,(HAVE\,(John,x) \wedge (CAT(x) \vee HOUND(x)))$

   $HAVE(John,a) \wedge (CAT(a) \vee HOUND(a))$

5. $\neg\,[LS(John) \rightarrow \neg\, \exists z\,(HAVE(John,z) \wedge MOUSE(z))]$ (negated conclusion)

   $\neg\,[\neg\, LS\,(John) \vee \neg\, \exists z\,(HAVE\,(John, z) \wedge MOUSE(z))]$

   $LS(John) \wedge \exists z\,(HAVE(John, z) \wedge MOUSE(z)))$

   $LS(John) \wedge HAVE(John,b) \wedge MOUSE(b)$

The set of CNF clauses for this problem is thus as follows:

1. $\neg HOUND(x) \lor HOWL(x)$
2. $\neg HAVE(x,y) \lor \neg CAT(y) \lor \neg HAVE(x,z) \lor \neg MOUSE(z)$
3. $\neg LS(x) \lor \neg HAVE(x,y) \lor \neg HOWL(y)$
4.
    1. $HAVE(John,a)$
    2. $CAT(a) \lor HOUND(a)$
5.
    1. $LS(John)$
    2. $HAVE(John,b)$
    3. $MOUSE(b)$

[1.,4.(b):] 6. $CAT(a) \lor HOWL(a)$

[2,5.(c):] 7. $\neg HAVE(x,y) \lor \neg CAT(y) \lor \neg HAVE(x,b)$

[7,5.(b):] 8. $\neg HAVE(John,y) \lor \neg CAT(y)$

[6,8:] 9. $\neg HAVE(John,a) \lor HOWL(a)$

[4.(a),9:] 10. $HOWL(a)$

[3,10:] 11. $\neg LS(x) \lor \neg HAVE(x,a)$

[4.(a),11:] 12. $\neg LS(John)$

[5.(a),12:] 13. $\square$

The End!