Introduction to Artificial Intelligence

Lecture 12: First-order Logic

Nina Gierasimczuk

Outline

Everybody Loves my Baby

From *Everybody loves my baby* by Boswell Sisters, 1932, *youtube link*

Everybody loves my baby
but my baby doesn't love anybody but me.

Everybody Loves my Baby

From *Everybody loves my baby* by Boswell Sisters, 1932, *youtube link*

Everybody loves my baby
but my baby doesn't love anybody but me.
Therefore, I am my baby.

Everybody Loves my Baby

From *Everybody loves my baby* by Boswell Sisters, 1932, youtube link

Everybody loves my baby
but my baby doesn't love anybody but me.
Therefore, I am my baby.

Is this a valid inference?

- We cannot answer this question with propositional logic.
- We should be able to use formal logic to analyze such inferences.

From Propositional Logic to First-order Logic

Propositional Logic:

Reasoning about situations using combinations of simple facts (with "and", "or", "not" etc). The *structure* of these facts was not analyzed further.

Example

p : *John cooks dinner for Ann.*

q : *Ann visits John.*

r : *John is happy.*

From Propositional Logic to First-order Logic

Propositional Logic:
Reasoning about situations using combinations of simple facts (with "and", "or", "not" etc). The *structure* of these facts was not analyzed further.

First-order Logic: First-order Logic looks at the internal structure of basic facts, especially, the objects that occur, the properties of these objects, and their relations to each other.

Example

j : John

a : Ann

Cxy : *x cooks for y.*	Cja : *John cooks for Ann.*
Vxy : *x visits y.*	Vaj : *Ann visits John.*
Hx : *x is happy.*	Hj : *John is happy.*

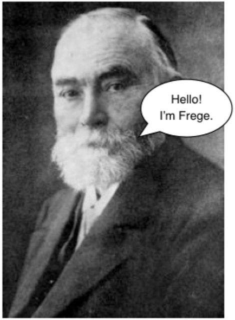
First-order Logic

- Reasoning about objects, predicates, and in principle, arbitrary forms of quantification.
- the most important system in logic today: it is a universal language for talking about **structure** (situation with objects, properties and relations).
  - Structures: friends on facebook, road systems, family trees, number systems
- First-order Logic has been used to increase precision in describing and studying all these structures, from linguistics and philosophy to computer science and mathematics.

Brief History

First-order Logic is a streamlined version of a "language of thought", proposed in 1878 by the German philosopher and mathematician

Gottlob Frege (1848 – 1925)



The Language of First-order Logic: The basic vocabulary

- Names for **objects**:
  - variables $x, y, z, \ldots$ when the object is indefinite.
  - function symbols, e.g., constants ('proper names') $a, b, c, \ldots$,for special objects,

The Language of First-order Logic: The basic vocabulary

- Names for **objects**:
  - variables $x, y, z, \ldots$ when the object is indefinite.
  - function symbols, e.g., constants ('proper names') a, b, c, ...,for special objects,
- **Properties and predicates** of objects:
  - Capital letters are predicate letters, with different numbers of arguments (arity):
  - 1-place predicates (unary predicates) are intransitive verbs ("walk") and common nouns ("boy" = "being a boy"),
  - 2-place predicates are transitive verbs ("see")
  - 3-place predicates are so-called ditransitive verbs ("give")
  - ...

The Language of First-order Logic: The basic vocabulary

- Names for **objects**:
  - variables $x, y, z, \ldots$ when the object is indefinite.
  - function symbols, e.g., constants ('proper names') a, b, c, ...,for special objects,
- **Properties and predicates** of objects:
  - Capital letters are predicate letters, with different numbers of arguments (arity):
  - 1-place predicates (unary predicates) are intransitive verbs ("walk") and common nouns ("boy"="being a boy"),
  - 2-place predicates are transitive verbs ("see")
  - 3-place predicates are so-called ditransitive verbs ("give")
  - ...
- **Sentence combination**:
  - The usual operators from propositional logic: $\neg, \vee, \wedge, \rightarrow, \iff$.

The Language of First-order Logic: The basic vocabulary

- Names for **objects**:
  - variables $x, y, z, \ldots$ when the object is indefinite.
  - function symbols, e.g., constants ('proper names') $a, b, c, \ldots$,for special objects,
- **Properties and predicates** of objects:
  - Capital letters are predicate letters, with different numbers of arguments (arity):
  - 1-place predicates (unary predicates) are intransitive verbs ("walk") and common nouns ("boy"="being a boy"),
  - 2-place predicates are transitive verbs ("see")
  - 3-place predicates are so-called ditransitive verbs ("give")
  - ...
- **Sentence combination**:
  - The usual operators from propositional logic: $\neg, \vee, \wedge, \rightarrow, \iff$.
- **Quantification**:
  - Quantifiers: $\forall x$ ("for all x") and $\exists x$ ("there exists an x")

The grammar of First-order Logic

$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
& \mid & \neg\ \textit{Sentence} \\
& \mid & \textit{Sentence} \wedge \textit{Sentence} \\
& \mid & \textit{Sentence} \vee \textit{Sentence} \\
& \mid & \textit{Sentence} \ \Rightarrow\ \textit{Sentence} \\
& \mid & \textit{Sentence} \ \Leftrightarrow\ \textit{Sentence} \\
& \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& \mid & \textit{Constant} \\
& \mid & \textit{Variable} \\
\\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots
\end{array}
$$

Simple Statements about Objects

natural language	First-order Logic
Alex sleeps.	Sa
Alex is a dragon.	Da
He walks.	Wx
Alex eats Tweety.	Eat
John gives Mary the book.	$Gjmb$

Predicates in Mathematics

informally	**logical/mathematical formula**
42 is smaller than 303	$42 < 303$
x is smaller than 42	$x < 42$
y is even	$y \mid 2$
Point p lies between q and r	$Bpqr$

Adding Propositional Logic

Propositional operators can be added in the obvious way to the preceding statements, and they function as before:

John does not see Mary.	$\neg Sjm$
Three is not less than two.	$\neg(3 < 2)$
Alex eats Tweety or Harry.	$Eat \vee Eah$
3 is less than 3 or 3 is less than 4.	$(3 < 3) \vee (3 < 4)$
x is odd	$\neg(2 \mid x)$
If John sees Mary, he is happy	$Sjm \rightarrow Hj$

Quantifiers

Existential: for saying that objects exist without naming them explicitly:

- Something happens: $\exists x Hx$

Quantifiers

Existential: for saying that objects exist without naming them explicitly:

- Something happens: $\exists x Hx$

Universal: for saying that all objects satisfy a property:

- Everything is fun: $\forall x Fx$

Quantifiers

Existential: for saying that objects exist without naming them explicitly:

- Something happens: $\exists x Hx$

Universal: for saying that all objects satisfy a property:

- Everything is fun: $\forall x Fx$

Let us think of more complex formulas:

go to: www.menti.com enter code: 7419 9461

More complex formulas

- Some dragon walks: $\exists x(Dx \wedge Wx)$
- Alex loves a girl: $\exists x(Gx \wedge Sax)$
- A girl loves Alex: $\exists x(Gx \wedge Lxa)$
- A dragon bites itself: $\exists x(Dx \wedge Bxx)$

- Every boy walks: $\forall x(Bx \rightarrow Wx)$
- Every girl loves Alex: $\forall x(Gx \rightarrow Lxa)$

Nested Quantifiers

- Brothers are siblings: $\forall x \forall y\ (B(x,y) \rightarrow S(x,y))$
- Being a sibling is symmetric: $\forall x, y\ (S(x,y) \leftrightarrow S(y,x))$
- Everybody loves somebody: $\forall x \exists y\ L(x,y)$
- There is someone who is loved by everybody: $\exists y \forall x\ L(x,y)$
- Consider: $\forall x(M(x) \vee \exists x\ L(John, x))$: **confusing, use a fresh variable**

Relationship between $\forall$ and $\exists$

$$\forall x\ \neg\varphi \equiv \neg\exists x\ \varphi$$

$$\neg\forall x\ \varphi \equiv \exists x\ \neg\varphi$$

$$\forall x\ \varphi \equiv \neg\exists x\ \neg\varphi$$

$$\exists x\ \varphi \equiv \neg\forall x\ \neg\varphi$$

Formulas and Models

B: property of being a motorbike
C: property of being a cow
M: property of being a man
R: relation of riding

$$\exists x \exists y \exists z(((Mx \wedge Cy) \wedge Bz\ ) \wedge (Rxz \wedge Ryz))$$

Models of First-order Logic

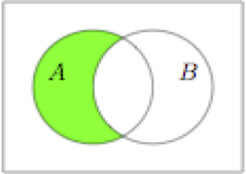
A model in first-order logic consists of a set of objects and an interpretation that maps constant symbols to objects, predicate symbols to relations on those objects, and function symbols to functions on those objects.

Ontological Committments

Language	Ontological Commitment	Epistemological Commitment
Propositional	facts	true/false/unknown
First-order	facts, objects, relations	true/false/unknown
Temporal	facts, objects, relations, times	true/false/unknown
Probability	facts	degree of belief $\in [0, 1]$
Fuzzy	facts with degree of truth $\in [0, 1]$	known interval value

Venn Diagrams

- Venn diagrams depict objects having certain properties (unary predicates).

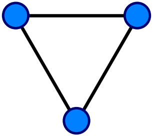


The green area is the set of individuals that make the formula $Ax \wedge \neg Bx$ true.

The variable x is not bound by quantifier. It is a **free variable**.

- With Venn diagrams, we can reason about sets of objects with certain *properties*.
- But, if we want to talk about *relations* between objects, we need another kind of pictures: Graphs!

Example



R : relation of being linked by an edge
True in this situation: $\forall x \exists y Rxy$, $\forall x \neg Rxx$
False: $\exists x \forall y Rxy$ $\exists x \exists y (Rxy \wedge \neg Ryx)$

Equality

- First-order Logic is more powerful if we can talk about things being equal.
- Equality can be expressed with the relation $=$.

Example

Olivia loves Peter but Peter loves another girl.

$$Lop \wedge \exists x((Gx \wedge \neg(x = o)) \wedge Lpx)$$

$$Lop \wedge \exists x((Gx \wedge x \neq o) \wedge Lpx)$$

Expressing Uniqueness

We can use equality in order to express uniqueness.

Example

Let P be the property of being a pope.

There is exactly one individual that has the property of being a pope.

$$\exists x (Px \wedge \forall y (Py \rightarrow (y = x)))$$

Counting in Prediate Logic

Similar to uniqueness, we can also express that there is a certain number of individuals that have some property.

Example

John has two sisters.

$$\exists x \exists y (((Sxj \wedge Syj) \wedge x \neq y) \wedge \forall z (Szj \rightarrow ((z = x) \vee (z = y))))$$

where S : relation of being a sister of someone.

This formula says that there are two individuals which are different and both are sisters of John, and every individual that is a sister of John has to be one of those two.

An Alternative Semantics

John has two sisters, Ann and Barbara.

$$Sister(John, Ann) \wedge Sister(John, Barbara)$$

$$Sister(John, Ann) \wedge Sister(John, Barbara) \wedge \ Ann \neq Barbara$$
$$\wedge \ \forall x \ Sister(John, x) \rightarrow (x = Ann \ \vee \ x = Barbara)$$

Too cumbersome, so in **database semantics**, the following are assumed:

1. unique names assumption;
2. closed-world assumption;
3. domain closure.

Knowledge Engineering in FOL

1. Identify the task.
2. Assemble the relevant knowledge.
3. Decide on a vocabulary (ontology).
4. Encode general knowledge about the domain.
5. Encode a description of the specific problem instance.
6. Pose queries to the inference procedure and get answers.
7. Debug the knowledge base.

See the textbook (Chapter 9 of R&N) for the **electronic circuit** domain.

Outline

Semantic Example: Venn Diagrams

If we are only concerned with unary predicates, we can use Venn diagrams to determine if an *inference* of First-order Logic is *valid*.

Example

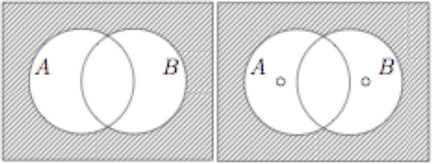
Is it the case that $\forall x(Ax \vee Bx) \models \forall x\ Ax \vee \forall x\ Bx$?

Semantic Example: Venn Diagrams

If we are only concerned with unary predicates, we can use Venn diagrams to determine if an *inference* of First-order Logic is *valid*.

Example

Is it the case that $\forall x(Ax \vee Bx) \models \forall x\ Ax \vee \forall x\ Bx$?



No! $\forall x(Ax \vee Bx) \not\models \forall x\ Ax \vee \forall x\ Bx$

Finding a model for which the premiss is true and the conclusion false!

Inference Rules for Quantifiers: Universal Instantiation

Let us now look at ways to derive new formulas from old formulas.

From:

$$\forall x \; (King(x) \wedge Greedy(x) \rightarrow Evil(x))$$

Inference Rules for Quantifiers: Universal Instantiation

Let us now look at ways to derive new formulas from old formulas.

From:

$$\forall x \; (King(x) \land Greedy(x) \to Evil(x))$$

it follows that:

$$King(John) \land Greedy(John) \to Evil(John)$$
$$King(Richard) \land Greedy(Richard) \to Evil(Richard)$$
$$King(Father(John)) \land Greedy(Father(John)) \to Evil(Father(John))$$

Inference Rules for Quantifiers: Universal Instantiation

Universal Instantiation (UI)
infer any sentence obtained by substituting a ground term (a term without variables) for the variable.

Inference Rules for Quantifiers: Universal Instantiation

Universal Instantiation (UI)
infer any sentence obtained by substituting a ground term (a term without variables) for the variable.

Formally, let $SUBST(\theta, \alpha)$ be the result of applying θ to α.

$$\frac{\forall v \;\alpha}{SUBST(\{v/g\}, \alpha)}$$

for any variable v and ground term g.

Inference Rules for Quantifiers: Universal Instantiation

Universal Instantiation (UI)
infer any sentence obtained by substituting a ground term (a term without variables) for the variable.

Formally, let $SUBST(\theta, \alpha)$ be the result of applying θ to α.

$$\frac{\forall v \ \alpha}{SUBST(\{v/g\}, \alpha)}$$

for any variable v and ground term g.

For example, we got the three sentences above via $\{x/John\}$, $\{x/Richard\}$, and $\{x/Father(John)\}$.

Inference Rules for Quantifiers: Existential Instantiation

Existential Instantiation (EI)
replace the variable by a single new constant symbol.

Existential Instantiation (EI)
replace the variable by a single new constant symbol.

Formally, for any sentence α, variable v, and constant symbol k that does not appear elsewhere in the knowledge base:

$$\frac{\exists v \;\alpha}{SUBST(\{v/k\}, \alpha)}$$

Existential Instantiation (EI)
replace the variable by a single new constant symbol.

Formally, for any sentence α, variable v, and constant symbol k that does not appear elsewhere in the knowledge base:

$$\frac{\exists v \ \alpha}{SUBST(\{v/k\}, \alpha)}$$

For example, from the sentence $\exists x \ Crown(x) \wedge OnHead(x, John)$ we can infer the sentence $Crown(C1) \wedge OnHead(C1, John)$, as long as C1 does not appear elsewhere in the knowledge base.

Existential Instantiation (EI)
replace the variable by a single new constant symbol.

Formally, for any sentence α, variable v, and constant symbol k that does not appear elsewhere in the knowledge base:

$$\frac{\exists v \ \alpha}{SUBST(\{v/k\}, \alpha)}$$

For example, from the sentence $\exists x \ Crown(x) \wedge OnHead(x, John)$ we can infer the sentence $Crown(C1) \wedge OnHead(C1, John)$, as long as C1 does not appear elsewhere in the knowledge base.

In logic, the new name is called a **Skolem constant**.

Inference Rules for Quantifiers: Discussion

- UI can be applied many times to produce different consequences
- EI can be applied once, then the existential sentence is discarded
- the new knowledge base is not logically equivalent to the old
- but it can be shown to be inferentially equivalent:
- it is satisfiable exactly when the original knowledge base is satisfiable

Semi-decidability of FOL

Propositionalisation gives completeness: any entailed sentence can be proved.

But we do not know until the proof is done that the sentence is entailed!

What happens when the sentence is not entailed? Can we tell?

In FOL we cannot: proof can go on and on and we'll never know if it'll stop.
Just like the **halting problem for Turing machines**. See here: movie

A first-order inference rule: Generalized Modus Ponens

Propositionalization approach is rather inefficient.

For example, given the query $\mathit{Evil}(\mathit{John})$ and the knowledge base:

$$\forall x \; \mathit{King}(x) \wedge \mathit{Greedy}(x) \rightarrow \mathit{Evil}(x)$$

$$\mathit{King}(\mathit{John})$$

$$\mathit{Greedy}(\mathit{John})$$

$$\mathit{Brother}(\mathit{Richard}, \mathit{John})$$

why would we generate all the instantiations of the first sentence?

Instead, we'd prefer to conclude $\mathit{Evil}(\mathit{John})$ directly.

A first-order inference rule: Generalized Modus Ponens

For atomic sentences p_i, p_i', and q,
if there is a substitution θ s.t. $SUBST(\theta, p_i') = SUBST(\theta, p_i)$, for all i:

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow q)}{SUBST(\theta, q)}$$

Generalized Modus Ponens is a **lifted** version of Modus Ponens:
raised from ground (variable-free) propositional logic to first-order logic.

Analogously, lifted versions of the forward chaining, backward chaining, and resolution algorithms can be provided (Chapter 9 of R&N).

Unification

Lifted inference requires substitutions that make different expressions identical.

Unification and is a key component of all first-order inference algorithms.
The UNIFY algorithm takes two sentences and returns a unifier (if exists):

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

Unification: Example

Query: $AskVars(Knows(John, x))$: whom does John know?

To answer we need to find all sentences in KB that unify with $Knows(John, x)$.

For example:

$$UNIFY(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$$
$$UNIFY(Knows(John, x), Knows(y, Bill)) = \{x/Bill, y/John\}$$
$$UNIFY(Knows(John, x), Knows(y, Mother(y))) = \{y/John, x/Mother(John)\}$$
$$UNIFY(Knows(John, x), Knows(x, Elizabeth)) = fail$$

Last unification can be solved by introducing a new variable:
standarizing apart one of the sentences.

Most General Unifier

$UNIFY(Knows(John, x), Knows(y, z))$ could return:
$\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$

Which is better?

Most General Unifier

$UNIFY(Knows(John, x), Knows(y, z))$ could return:
$\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$

Which is better? First gives $Knows(John, z)$, second $Knows(John, John)$.

Most General Unifier

$UNIFY(Knows(John, x), Knows(y, z))$ could return:
$\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$

Which is better? First gives $Knows(John, z)$, second $Knows(John, John)$.
The first one is more general!

Most General Unifier

$UNIFY(Knows(John, x), Knows(y, z))$ could return:
$\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$

Which is better? First gives $Knows(John, z)$, second $Knows(John, John)$.
The first one is more general!

Theorem:
For every unifiable pair of expressions, there is a unique **most general unifier** (MGU).

Towards Resolution: CNF for FOL

As before, resolution requires conjunctive normal form (CNF).

Towards Resolution: CNF for FOL

As before, resolution requires conjunctive normal form (CNF).

In FOL literals can contain variables, assumed to be universally quantified:

$$\forall x, y, z \; American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$$

becomes, in CNF:

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$

Towards Resolution: CNF for FOL

As before, resolution requires conjunctive normal form (CNF).

In FOL literals can contain variables, assumed to be universally quantified:

$$\forall x, y, z \; American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$$

becomes, in CNF:

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$

Every FOL sentence can be converted into an inferentially equivalent CNF.

Towards Resolution: CNF for FOL

As before, resolution requires conjunctive normal form (CNF).

In FOL literals can contain variables, assumed to be universally quantified:

$$\forall x, y, z \; American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$$

becomes, in CNF:

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$

Every FOL sentence can be converted into an inferentially equivalent CNF.

The CNF sentence will be unsatisfiable just when the original is too,
so we can do proofs by contradiction on the CNF sentences.

Converting to CNF in FOL

Everyone who loves all animals is loved by someone.

Converting to CNF in FOL

Everyone who loves all animals is loved by someone.

$$\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$$

Converting to CNF in FOL

- $\forall x \; (\forall y \; (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y \; Loves(y, x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \to Loves(x, y)) \to \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \lor Loves(x, y)) \lor \exists y\ Loves(y, x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y\ Loves(y, x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x,y)) \rightarrow \exists y\ Loves(y,x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x,y)) \vee \exists y\ Loves(y,x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x,y))) \vee \exists y\ Loves(y,x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x,y)) \vee \exists y\ Loves(y,x))$
  - $\forall x\ (\exists y\ (Animal(y) \wedge \neg Loves(x,y)) \vee \exists y\ Loves(y,x))$
- **Standarize variables**: $\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x,y)) \vee \exists z\ Loves(z,x))$

Converting to CNF in FOL

- $\forall x \; (\forall y \; (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y \; Loves(y, x))$
- **Eliminate implication**:
  $\forall x \; (\neg \forall y \; (\neg Animal(y) \vee Loves(x, y)) \vee \exists y \; Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x \; (\exists y \; (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y \; Loves(y, x))$
  - $\forall x \; (\exists y \; (\neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee \exists y \; Loves(y, x))$
  - $\forall x \; (\exists y \; (Animal(y) \wedge \neg Loves(x, y)) \vee \exists y \; Loves(y, x))$
- **Standarize variables**: $\forall x \; (\exists y \; Animal(y) \wedge \neg Loves(x, y)) \vee \exists z \; Loves(z, x))$
- **Skolemize (EI)**: $\forall x \; ((Animal(A) \wedge \neg Loves(x, A)) \vee Loves(B, x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Standarize variables**: $\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x, y)) \vee \exists z\ Loves(z, x))$
- **Skolemize (EI)**: $\forall x\ ((Animal(A) \wedge \neg Loves(x, A)) \vee Loves(B, x))$ **NO!**

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Standarize variables**: $\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x, y)) \vee \exists z\ Loves(z, x))$
- **Skolemize (EI)**: $\forall x\ ((Animal(A) \wedge \neg Loves(x, A)) \vee Loves(B, x))$ **NO!**
  $\forall x\ ((Animal(F(x)) \wedge \neg Loves(x, F(x))) \vee Loves(G(x), x))$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x, y)) \rightarrow \exists y\ Loves(y, x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x, y))) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
  - $\forall x\ (\exists y\ (Animal(y) \wedge \neg Loves(x, y)) \vee \exists y\ Loves(y, x))$
- **Standarize variables**: $\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x, y)) \vee \exists z\ Loves(z, x))$
- **Skolemize (EI)**: $\forall x\ ((Animal(A) \wedge \neg Loves(x, A)) \vee Loves(B, x))$ **NO!**

  $\forall x\ ((Animal(F(x)) \wedge \neg Loves(x, F(x))) \vee Loves(G(x), x))$
- **Drop universal quantifiers**:
  $(Animal(F(x)) \wedge \neg Loves(x, F(x))) \vee Loves(G(x), x)$

Converting to CNF in FOL

- $\forall x\ (\forall y\ (Animal(y) \rightarrow Loves(x,y)) \rightarrow \exists y\ Loves(y,x))$
- **Eliminate implication**:
  $\forall x\ (\neg\forall y\ (\neg Animal(y) \vee Loves(x,y)) \vee \exists y\ Loves(y,x))$
- **Move $\neg$ inwards**:
  - $\forall x\ (\exists y\ (\neg(\neg Animal(y) \vee Loves(x,y))) \vee \exists y\ Loves(y,x))$
  - $\forall x\ (\exists y\ (\neg\neg Animal(y) \wedge \neg Loves(x,y)) \vee \exists y\ Loves(y,x))$
  - $\forall x\ (\exists y\ (Animal(y) \wedge \neg Loves(x,y)) \vee \exists y\ Loves(y,x))$
- **Standarize variables**: $\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x,y)) \vee \exists z\ Loves(z,x))$
- **Skolemize (EI)**: $\forall x\ ((Animal(A) \wedge \neg Loves(x,A)) \vee Loves(B,x))$ **NO!**

  $\forall x\ ((Animal(F(x)) \wedge \neg Loves(x,F(x))) \vee Loves(G(x),x))$
- **Drop universal quantifiers**:
  $(Animal(F(x)) \wedge \neg Loves(x,F(x))) \vee Loves(G(x),x)$
- **Distribute $\vee$ over $\wedge$**:
  $(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))!$

Resolution Rule for FOL

Two standardized apart clauses are resolved if they have complementary literals.

Resolution Rule for FOL

Two standardized apart clauses are resolved if they have complementary literals.

FOL literals are complementary if one unifies with the negation of the other.

$$\frac{\ell_1 \vee \ldots \vee \ell_k, m_1 \vee \ldots \vee m_n,}{SUBST(\theta, \ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)}$$

where $UNIFY(\ell_i, \neg m_j) = \theta$.

Resolution Rule for FOL

Two standardized apart clauses are resolved if they have complementary literals.

FOL literals are complementary if one unifies with the negation of the other.

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \; m_1 \vee \ldots \vee m_n,}{SUBST(\theta, \ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)}$$

where $UNIFY(\ell_i, \neg m_j) = \theta$.

Full resolution, which extends the above to unifying sets of literals is a complete inference procedure for FOL.

End of Lecture 12