TDT4136 Introduction to Artificial Intelligence

Assignment Lecture: First-Order Logic

Håkon Måløy

Norwegian University of Science and Technology

First-Order Logic

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

- Propositional Logic assumes the world contains facts.

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

- Propositional Logic assumes the world contains facts.
- First-Order Logic assumes that the world consists of:

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

- Propositional Logic assumes the world contains facts.
- First-Order Logic assumes that the world consists of:
  - Objects: Cars, People, Colors, Animals, ...

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

- Propositional Logic assumes the world contains facts.
- First-Order Logic assumes that the world consists of:
  - Objects: Cars, People, Colors, Animals, ...
  - Relations: Green, Big, prime, ...

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

- Propositional Logic assumes the world contains facts.
- First-Order Logic assumes that the world consists of:
  - Objects: Cars, People, Colors, Animals, ...
  - Relations: Green, Big, prime, ...
  - Functions: FatherOf, Plus, OwnerOf, ...

First-Order Logic

First-Order Logic (or predicate logic) is more expressive than Propositional Logic.

- Propositional Logic assumes the world contains facts.
- First-Order Logic assumes that the world consists of:
  - Objects: Cars, People, Colors, Animals, ...
  - Relations: Green, Big, prime, ...
  - Functions: FatherOf, Plus, OwnerOf, ...
  - Quantifiers: Existential and Universal

Syntax of First-Order Logic

Syntax of First-Order Logic

The basic syntactic elements in First-Order Logic are the symbols that stand for objects, relations and functions. There are three types of symbols:

Syntax of First-Order Logic

The basic syntactic elements in First-Order Logic are the symbols that stand for objects, relations and functions. There are three types of symbols:

- Constant symbols: Objects (Roar, Eirik)

Syntax of First-Order Logic

The basic syntactic elements in First-Order Logic are the symbols that stand for objects, relations and functions. There are three types of symbols:
- Constant symbols: Objects (Roar, Eirik)
- Predicate symbols: Relations (Brother, Person, Male)

Syntax of First-Order Logic

The basic syntactic elements in First-Order Logic are the symbols that stand for objects, relations and functions. There are three types of symbols:

- Constant symbols: Objects (Roar, Eirik)
- Predicate symbols: Relations (Brother, Person, Male)
- Function symbols: Functions (BrotherOf, LeftLeg)

Overview of FOL Syntax

Overview of FOL Syntax

The full overview of the syntax of FOL is listed in the table below:

Overview of FOL Syntax

The full overview of the syntax of FOL is listed in the table below:

Sentence	$\rightarrow$	AtomicSentence, ComplexSentence
AtomicSentence	$\rightarrow$	Predicate, Predicate(Term,...), Term = Term
ComplexSentence	$\rightarrow$	(Sentence), [Sentence]
		$\neg$ Sentence
		Sentence $\wedge$ Sentence
		Sentence $\vee$ Sentence
Sentence	$\rightarrow$	Sentence
		Sentence $\iff$ Sentence
		Quantifier Variable, ... Sentence
Terms	$\rightarrow$	Function(Term, ...)
		Constant
		Variable
Quantifier	$\rightarrow$	$\forall$, $\exists$
Constant	$\rightarrow$	A, X_1, $John$, ...
Variable	$\rightarrow$	a, x, s, ...
Predicate	$\rightarrow$	True, False, After, Loves, Raining
Function	$\rightarrow$	Mother, LeftLeg, BelongsTo

Functions vs. Predicates

Functions vs. Predicates

- A function is a 'box' that takes an argument and returns a value.

Functions vs. Predicates

- A function is a 'box' that takes an argument and returns a value.
- Predicates takes an argument and returns a boolean.

Logical Quantifiers

Logical Quantifiers

Quantifiers let us express properties of entire collections of objects instead of enumerating them by name. There are two standard quantifiers in FOL:

Logical Quantifiers

Quantifiers let us express properties of entire collections of objects instead of enumerating them by name. There are two standard quantifiers in FOL:

- Universal quantification ($\forall$):

Logical Quantifiers

Quantifiers let us express properties of entire collections of objects instead of enumerating them by name. There are two standard quantifiers in FOL:

- Universal quantification ($\forall$):
  Lets us express a sentence such as "all kings are persons":
  $\forall x \; King(x) \Rightarrow Person(x)$

Logical Quantifiers

Quantifiers let us express properties of entire collections of objects instead of enumerating them by name. There are two standard quantifiers in FOL:

- Universal quantification ($\forall$):
  Lets us express a sentence such as "all kings are persons":
  $\forall x \; King(x) \Rightarrow Person(x)$
- Existential quantification ($\exists$):

Logical Quantifiers

Quantifiers let us express properties of entire collections of objects instead of enumerating them by name. There are two standard quantifiers in FOL:

- Universal quantification ($\forall$):
  Lets us express a sentence such as “all kings are persons”:
  $\forall x\ King(x) \Rightarrow Person(x)$
- Existential quantification ($\exists$):
  Makes a statement about <u>some</u> object in the universe without naming the exact object. We can say that “king John has a crown on his head”:
  $\exists x\ Crown(x) \land OnHead(x, John)$

Universal Instantiation

Universial Instantiation

Universial Instantiation is a inference rule that says that we can infer any sentence just by substituting a variable with a ground term (terms without variables).

Universial Instantiation

Universial Instantiation is a inference rule that says that we can infer any sentence just by substituting a variable with a ground term (terms without variables).

We use it to get rid of universal quantifications in sentences.

Universial Instantiation

Universial Instantiation is a inference rule that says that we can infer any sentence just by substituting a variable with a ground term (terms without variables).

We use it to get rid of universal quantifications in sentences.

Universal Instantiation can be applied many times to produce many different consequences.

Existential Instantiation

Existential Instantiation

Existential instantiation says that we can infer any sentence that is found by replacing a variable with a new constant symbol.

Existential Instantiation

Existential instantiation says that we can infer any sentence that is found by replacing a variable with a new constant symbol.

We use it to get rid of existential quantifications in sentences

Existential Instantiation

Existential instantiation says that we can infer any sentence that is found by replacing a variable with a new constant symbol.

We use it to get rid of existential quantifications in sentences

Existential Instantiation can only be applied once and then we can discard the existentially quantified sentence.

Skolemization

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

- When the existential quantifier has precedent over all universal quantifiers we can eliminate the existential quantifier and substitute the variable in the existential quantifier with a constant (Skolem Constant) using Existential Instantiation.

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

- When the existential quantifier has precedent over all universal quantifiers we can eliminate the existential quantifier and substitute the variable in the existential quantifier with a constant (Skolem Constant) using Existential Instantiation.
  $\exists x \, \forall y \, \forall z \, (P(x, y)) \Rightarrow Q(x, z)$, can be skolemized into:

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

- When the existential quantifier has precedent over all universal quantifiers we can eliminate the existential quantifier and substitute the variable in the existential quantifier with a constant (Skolem Constant) using Existential Instantiation.
  $\exists x \, \forall y \, \forall z \, (P(x, y)) \Rightarrow Q(x, z)$, can be skolemized into:
  $\forall y \, \forall z \, (P(C, y)) \Rightarrow Q(C, z)$

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

- When the existential quantifier has precedent over all universal quantifiers we can eliminate the existential quantifier and substitute the variable in the existential quantifier with a constant (Skolem Constant) using Existential Instantiation.
  $\exists x \, \forall y \, \forall z \, (P(x, y)) \Rightarrow Q(x, z)$, can be skolemized into:
  $\forall y \, \forall z \, (P(C, y)) \Rightarrow Q(C, z)$
- When the existential quantifier is nested within universal quantifiers we can eliminate the existential quantifier and substitute its variable with a function of all the universal quantifiers that precedes the existential one (Skolem Functions).

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

- When the existential quantifier has precedent over all universal quantifiers we can eliminate the existential quantifier and substitute the variable in the existential quantifier with a constant (Skolem Constant) using Existential Instantiation.
  $\exists x\ \forall y\ \forall z\ (P(x, y)) \Rightarrow Q(x, z)$, can be skolemized into:
  $\forall y\ \forall z\ (P(C, y)) \Rightarrow Q(C, z)$
- When the existential quantifier is nested within universal quantifiers we can eliminate the existential quantifier and substitute its variable with a function of all the universal quantifiers that precedes the existential one (Skolem Functions).
  $\forall x\ \exists y\ \forall z\ \exists u\ A(x, y, z, u)$ can be skolemized into:

Skolemization

Skolemization is the process of removing Existential Quantifiers by elimination. This can be done in two ways:

- When the existential quantifier has precedent over all universal quantifiers we can eliminate the existential quantifier and substitute the variable in the existential quantifier with a constant (Skolem Constant) using Existential Instantiation.
  $\exists x \, \forall y \, \forall z \, (P(x, y)) \Rightarrow Q(x, z)$, can be skolemized into:
  $\forall y \, \forall z \, (P(C, y)) \Rightarrow Q(C, z)$
- When the existential quantifier is nested within universal quantifiers we can eliminate the existential quantifier and substitute its variable with a function of all the universal quantifiers that precedes the existential one (Skolem Functions).
  $\forall x \, \exists y \, \forall z \, \exists u \, A(x, y, z, u)$ can be skolemized into:
  $\forall x \, \forall z \, A(f(x)/y, g(x, z)/u)$

CNF in FOL

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

- Eliminate Implications

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

- Eliminate Implications
- Move $\neg$ inward

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

- Eliminate Implications
- Move $\neg$ inward
- Standardize variables (replace variable names to avoid confusion)

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

- Eliminate Implications
- Move $\neg$ inward
- Standardize variables (replace variable names to avoid confusion)
- Skolemize

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

- Eliminate Implications
- Move $\neg$ inward
- Standardize variables (replace variable names to avoid confusion)
- Skolemize
- Drop Universal Quantifiers

CNF in FOL

The differences when converting sentences to CNF in FOL as compared to Propositional Logic are the need to eliminate Existential Quantifiers.

The steps are:

- Eliminate Implications
- Move $\neg$ inward
- Standardize variables (replace variable names to avoid confusion)
- Skolemize
- Drop Universal Quantifiers
- Distribute $\vee$ over $\wedge$

CFN in FOL Example

ex: Translate the sentence "Everyone who loves all animals is loved by someone":

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

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x \; [\neg\forall y \; \neg Animal(y) \vee Loves(x,y)] \vee [\exists y \; Loves(y,x)]$

- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x \; p$ becomes $\exists x \; \neg p$
  $\neg\exists x \; p$ becomes $\forall x \; \neg p$
  thus we have:
  $\forall x \; [\exists y \; \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y \; Loves(y,x)]$.

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$
- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:
  $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y\ Loves(y, x)]$.
  $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$.

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:
  $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$.
  $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$.
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y\ Loves(y,x)]$.

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$
- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:
  $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y\ Loves(y, x)]$.
  $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$.
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y))] \vee [\exists y\ Loves(y, x)]$.
- Standardize variables (replace variable names to avoid confusion):
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y))] \vee [\exists z\ Loves(z, x)]$.

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:
  $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y\ Loves(y, x)]$.
  $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$.
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y))] \vee [\exists y\ Loves(y, x)]$.

- Standardize variables (replace variable names to avoid confusion):
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y))] \vee [\exists z\ Loves(z, x)]$.

- Skolemize (Remove Existential Quantifiers):
  $\forall x\ [Animal(F(x)) \wedge \neg Loves(x, F(x)))] \vee Loves(G(z), x)$.

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:
  $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$.
  $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$.
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y\ Loves(y,x)]$.
- Standardize variables (replace variable names to avoid confusion):
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z\ Loves(z,x)]$.
- Skolemize (Remove Existential Quantifiers):
  $\forall x\ [Animal(F(x)) \wedge \neg Loves(x, F(x)))] \vee Loves(G(z), x)$.
- Drop Universal Quantifiers:
  Since all variables are universally quantified we can drop the universal quantifiers.
  $[Animal(F(x)) \wedge \neg Loves(x, F(x)))] \vee Loves(G(z), x)$.

CFN in FOL Example

ex: Translate the sentence "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 $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
- Move $\neg$ inward: We also need the rules for negated quantifiers here:
  $\neg\forall x\ p$ becomes $\exists x\ \neg p$
  $\neg\exists x\ p$ becomes $\forall x\ \neg p$
  thus we have:
  $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$.
  $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$.
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists y\ Loves(y,x)]$.
- Standardize variables (replace variable names to avoid confusion):
  $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y))] \vee [\exists z\ Loves(z,x)]$.
- Skolemize (Remove Existential Quantifiers):
  $\forall x\ [Animal(F(x)) \wedge \neg Loves(x, F(x)))] \vee Loves(G(z), x)$.
- Drop Universal Quantifiers:
  Since all variables are universally quantified we can drop the universal quantifiers.
  $[Animal(F(x)) \wedge \neg Loves(x, F(x)))] \vee Loves(G(z), x)$.
- Distribute $\vee$ over $\wedge$
  $[Animal(F(x)) \vee Loves(G(z), x)] \wedge [Loves(x, F(x)) \vee Loves(G(z), x)]$.
  This step can also require flattening out nested conjunctions and disjunctions.