Lectures 12 and 13 (Week 7)
Introducing Reasoning with First-Order Logic

COMP24412: Symbolic AI

Giles Reger

March 2020

Welcome to Part 2

The first part introduced

- Different ways to capture (from natural language) and represent knowledge
- The notions of querying a knowledge base and checking entailments – related to reasoning
- The Prolog language, used for both knowledge representation, knowledge capture (parsing), and reasoning

The second part looks at reasoning and focusses on doing this in first-order logic (we'll explore why later)

These Slides

I have colour-coded the slides in an attempt to be helpful:

- For admin (e.g. this slide)
- For introductory/motivational text
- For definitions
- For examples
- For important points (used sparingly)

I will be covering four major topics over the next four weeks. Slides will be prepared for each topic e.g. each slide deck will cover two lectures.

Aim and Learning Outcomes

The aim of these two lectures is to:

Revise the general language of first-order logic, including its syntax and semantics, and introduce the main reasoning tasks we want to carry out.

Learning Outcomes

By the end you will be able to:

1. Write formulas in first-order logic modelling real-world situations and describe their meaning
2. Recall the notion of an *interpretation* in first-order logic and apply it to determine the truth of a first-order logic
3. Explain the main reasoning tasks at a high level of abstraction and using first-order logic

Automated Reasoning

Knowledge representation is about representing knowledge (about the world) in an unambiguous and precise manner.

Automated reasoning is about automatically answering questions about that knowledge e.g. to find statements that follow from the knowledge

Example

I know

1. It rains in Manchester
2. If it rains somewhere I need to take an umbrella when I go there

I can reason that

3. When I go to Manchester I need to take an umbrella

This is something that is not in my original knowledge explicitly but is entailed by it.

Automated Reasoning

How does reasoning work?

Theoretically, the semantics of first-order logic is defined in terms of models (which we'll define soon) which capture the set of all possible statements that are consistent with our knowledge. Reasoning is defined in terms of querying these models, but they are abstract!

Practically, we define a set of inference rules that allow us to derive new information that is consistent with the current information (the rules are sound). Ideally we can derive all consistent things (the rules are complete).

For example, a famous rule is Modus Ponens

$$\frac{A \rightarrow B \qquad A}{B}$$

Don't worry if this doesn't make sense yet - we'll cover it properly later!

Automated Reasoning

How does reasoning work?

Theoretically, the semantics of first-order logic is defined in terms of models (which we'll define soon) which capture the set of all possible statements that are consistent with our knowledge. Reasoning is defined in terms of querying these models, but they are abstract!

Practically, we define a set of inference rules that allow us to derive new information that is consistent with the current information (the rules are sound). Ideally we can derive all consistent things (the rules are complete).

For example, a famous rule is Modus Ponens

$$\frac{\text{It rains in Manchester} \rightarrow \text{Take an umbrella to Manchester} \quad \text{It rains in Manchester}}{\text{Take an umbrella to Manchester}}$$

Don't worry if this doesn't make sense yet - we'll cover it properly later!

Automated Reasoning

How does reasoning work?

Theoretically, the semantics of first-order logic is defined in terms of models (which we'll define soon) which capture the set of all possible statements that are consistent with our knowledge. Reasoning is defined in terms of querying these models, but they are abstract!

Practically, we define a set of inference rules that allow us to derive new information that is consistent with the current information (the rules are sound). Ideally we can derive all consistent things (the rules are complete).

For example, a famous rule is Modus Ponens

$$\frac{\text{It rains in X} \rightarrow \text{Take an umbrella to X} \qquad \text{It rains in Manchester}}{\text{Take an umbrella to Manchester}}$$

Don't worry if this doesn't make sense yet - we'll cover it properly later!

Automated Reasoning

How does reasoning work?

Theoretically, the semantics of first-order logic is defined in terms of models (which we'll define soon) which capture the set of all possible statements that are consistent with our knowledge. Reasoning is defined in terms of querying these models, but they are abstract!

Practically, we define a set of inference rules that allow us to derive new information that is consistent with the current information (the rules are sound). Ideally we can derive all consistent things (the rules are complete).

For example, a famous rule is Modus Ponens

$$\frac{A \rightarrow B \qquad C}{B\theta} \quad \theta = \text{mgu}(A, C)$$

Don't worry if this doesn't make sense yet - we'll cover it properly later!

Knowledge Bases

Reasoning is dependent on the knowledge we have and how it is represented. For now we will consider first-order knowledge bases but we will relate these to alternatives later.

Reminder: a knowledge base is a symbolic abstraction of the real world.

We pick the thing we want to describe (the domain) and the symbols we will use to define it (the signature).

During modelling the mapping between domain and signature is in our heads, or at least is not captured logically.

When interpreting a knowledge base we do so over an abstract domain where abstract symbols represent the things we are modelling. This is because reasoning needs to work with any interpretation.

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

In the Signature we use

- Constants `red`, `blue`, `green`, `yellow`, and `orange` for the blocks
- A unary predicate symbol `block` to capture what the blocks are
- A function symbol `on` capturing what a block is on
- A binary predicate symbol `above` for one block being above another

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block}(\mathrm{red}) \wedge \mathrm{block}(\mathrm{blue}) \wedge \mathrm{block}(\mathrm{green}) \wedge \mathrm{block}(\mathrm{yellow}) \wedge \mathrm{block}(\mathrm{orange})$$
$$\mathrm{on}(\mathrm{orange}) = \mathrm{green} \wedge \mathrm{above}(\mathrm{yellow}, \mathrm{green}) \wedge \neg\mathrm{block}(\mathrm{on}(\mathrm{red}))$$
$$\forall b_1, b_2.(\mathrm{on}(\mathrm{b}_1) = \mathrm{on}(\mathrm{b}_2) \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above}(\mathrm{b}, \mathrm{on}(\mathrm{b})) \wedge \mathrm{on}(\mathrm{b}) \neq \mathrm{b} \wedge \neg\mathrm{above}(\mathrm{b}, \mathrm{b}))$$
$$\forall b_1, b_2, b_3.((\mathrm{above}(\mathrm{b}_1, \mathrm{b}_2) \wedge \mathrm{above}(\mathrm{b}_2, \mathrm{b}_3)) \rightarrow \mathrm{above}(\mathrm{b}_1, \mathrm{b}_3))$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block}(\mathrm{red}) \wedge \mathrm{block}(\mathrm{blue}) \wedge \mathrm{block}(\mathrm{green}) \wedge \mathrm{block}(\mathrm{yellow}) \wedge \mathrm{block}(\mathrm{orange})$$
$$\mathrm{on}(\mathrm{orange}) = \mathrm{green} \wedge \mathrm{above}(\mathrm{yellow}, \mathrm{green}) \wedge \neg\mathrm{block}(\mathrm{on}(\mathrm{red}))$$
$$\forall b_1, b_2.(\mathrm{on}(\mathrm{b}_1) = \mathrm{on}(\mathrm{b}_2) \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above}(\mathrm{b}, \mathrm{on}(\mathrm{b})) \wedge \mathrm{on}(\mathrm{b}) \neq \mathrm{b} \wedge \neg\mathrm{above}(\mathrm{b}, \mathrm{b}))$$
$$\forall b_1, b_2, b_3.((\mathrm{above}(\mathrm{b}_1, \mathrm{b}_2) \wedge \mathrm{above}(\mathrm{b}_2, \mathrm{b}_3)) \rightarrow \mathrm{above}(\mathrm{b}_1, \mathrm{b}_3))$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block(red)} \wedge \mathrm{block(blue)} \wedge \mathrm{block(green)} \wedge \mathrm{block(yellow)} \wedge \mathrm{block(orange)}$$
$$\mathrm{on(orange)} = \mathrm{green} \wedge \mathrm{above(yellow, green)} \wedge \neg\mathrm{block(on(red))}$$
$$\forall b_1, b_2.(\mathrm{on(b_1)} = \mathrm{on(b_2)} \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above(b, on(b))} \wedge \mathrm{on(b)} \neq \mathrm{b} \wedge \neg\mathrm{above(b, b)})$$
$$\forall b_1, b_2, b_3.((\mathrm{above(b_1, b_2)} \wedge \mathrm{above(b_2, b_3)}) \rightarrow \mathrm{above(b_1, b_3)})$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block}(\mathrm{red}) \wedge \mathrm{block}(\mathrm{blue}) \wedge \mathrm{block}(\mathrm{green}) \wedge \mathrm{block}(\mathrm{yellow}) \wedge \mathrm{block}(\mathrm{orange})$$
$$\mathrm{on}(\mathrm{orange}) = \mathrm{green} \wedge \mathrm{above}(\mathrm{yellow}, \mathrm{green}) \wedge \neg\mathrm{block}(\mathrm{on}(\mathrm{red}))$$
$$\forall b_1, b_2.(\mathrm{on}(\mathrm{b}_1) = \mathrm{on}(\mathrm{b}_2) \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above}(\mathrm{b}, \mathrm{on}(\mathrm{b})) \wedge \mathrm{on}(\mathrm{b}) \neq \mathrm{b} \wedge \neg\mathrm{above}(\mathrm{b}, \mathrm{b}))$$
$$\forall b_1, b_2, b_3.((\mathrm{above}(\mathrm{b}_1, \mathrm{b}_2) \wedge \mathrm{above}(\mathrm{b}_2, \mathrm{b}_3)) \rightarrow \mathrm{above}(\mathrm{b}_1, \mathrm{b}_3))$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block}(\mathrm{red}) \wedge \mathrm{block}(\mathrm{blue}) \wedge \mathrm{block}(\mathrm{green}) \wedge \mathrm{block}(\mathrm{yellow}) \wedge \mathrm{block}(\mathrm{orange})$$
$$\mathrm{on}(\mathrm{orange}) = \mathrm{green} \wedge \mathrm{above}(\mathrm{yellow}, \mathrm{green}) \wedge \neg\mathrm{block}(\mathrm{on}(\mathrm{red}))$$
$$\forall b_1, b_2.(\mathrm{on}(\mathrm{b}_1) = \mathrm{on}(\mathrm{b}_2) \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above}(\mathrm{b}, \mathrm{on}(\mathrm{b})) \wedge \mathrm{on}(\mathrm{b}) \neq \mathrm{b} \wedge \neg\mathrm{above}(\mathrm{b}, \mathrm{b}))$$
$$\forall b_1, b_2, b_3.((\mathrm{above}(\mathrm{b}_1, \mathrm{b}_2) \wedge \mathrm{above}(\mathrm{b}_2, \mathrm{b}_3)) \rightarrow \mathrm{above}(\mathrm{b}_1, \mathrm{b}_3))$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block(red)} \wedge \mathrm{block(blue)} \wedge \mathrm{block(green)} \wedge \mathrm{block(yellow)} \wedge \mathrm{block(orange)}$$
$$\mathrm{on(orange)} = \mathrm{green} \wedge \mathrm{above(yellow, green)} \wedge \neg\mathrm{block(on(red))}$$
$$\forall b_1, b_2.(\mathrm{on(b_1)} = \mathrm{on(b_2)} \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above(b, on(b))} \wedge \mathrm{on(b)} \neq \mathrm{b} \wedge \neg\mathrm{above(b, b)})$$
$$\forall b_1, b_2, b_3.((\mathrm{above(b_1, b_2)} \wedge \mathrm{above(b_2, b_3)}) \rightarrow \mathrm{above(b_1, b_3)})$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block}(\mathrm{red}) \wedge \mathrm{block}(\mathrm{blue}) \wedge \mathrm{block}(\mathrm{green}) \wedge \mathrm{block}(\mathrm{yellow}) \wedge \mathrm{block}(\mathrm{orange})$$
$$\mathrm{on}(\mathrm{orange}) = \mathrm{green} \wedge \mathrm{above}(\mathrm{yellow}, \mathrm{green}) \wedge \neg \mathrm{block}(\mathrm{on}(\mathrm{red}))$$
$$\forall b_1, b_2.(\mathrm{on}(\mathrm{b}_1) = \mathrm{on}(\mathrm{b}_2) \rightarrow b_1 = b_2)$$
$$\forall b.(\mathrm{above}(\mathrm{b}, \mathrm{on}(\mathrm{b})) \wedge \mathrm{on}(\mathrm{b}) \neq \mathrm{b} \wedge \neg \mathrm{above}(\mathrm{b}, \mathrm{b}))$$
$$\forall b_1, b_2, b_3.((\mathrm{above}(\mathrm{b}_1, \mathrm{b}_2) \wedge \mathrm{above}(\mathrm{b}_2, \mathrm{b}_3)) \rightarrow \mathrm{above}(\mathrm{b}_1, \mathrm{b}_3))$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\text{block}(\text{red}) \land \text{block}(\text{blue}) \land \text{block}(\text{green}) \land \text{block}(\text{yellow}) \land \text{block}(\text{orange})$$
$$\text{on}(\text{orange}) = \text{green} \land \text{above}(\text{yellow}, \text{green}) \land \neg\text{block}(\text{on}(\text{red}))$$
$$\forall b_1, b_2.(\text{on}(\text{b}_1) = \text{on}(\text{b}_2) \rightarrow b_1 = b_2)$$
$$\forall b.(\text{above}(\text{b}, \text{on}(\text{b})) \land \text{on}(\text{b}) \neq \text{b} \land \neg\text{above}(\text{b}, \text{b}))$$
$$\forall b_1, b_2, b_3.((\text{above}(\text{b}_1, \text{b}_2) \land \text{above}(\text{b}_2, \text{b}_3)) \rightarrow \text{above}(\text{b}_1, \text{b}_3))$$

A Knowledge Base Example

Reality - representing certain configurations of blocks:

- The five distinct blocks are red, blue, green, yellow, and orange
- The orange block must be on top of the green block
- The yellow block must be above the blue block
- The red block must not be on a block
- A block is on exactly one thing and has at most one thing on it
- If a block is on another block it is above it
- Blocks cannot be on or above themselves
- The notion of being above something is transitive

Knowledge base:

$$\mathrm{block}(\mathrm{red}) \land \mathrm{block}(\mathrm{blue}) \land \mathrm{block}(\mathrm{green}) \land \mathrm{block}(\mathrm{yellow}) \land \mathrm{block}(\mathrm{orange})$$
$$\mathrm{on}(\mathrm{orange}) = \mathrm{green} \land \mathrm{above}(\mathrm{yellow}, \mathrm{green}) \land \neg \mathrm{block}(\mathrm{on}(\mathrm{red}))$$
$$\forall b_1, b_2.(\mathrm{on}(\mathrm{b}_1) = \mathrm{on}(\mathrm{b}_2) \to b_1 = b_2)$$
$$\forall b.(\mathrm{above}(\mathrm{b}, \mathrm{on}(\mathrm{b})) \land \mathrm{on}(\mathrm{b}) \neq \mathrm{b} \land \neg \mathrm{above}(\mathrm{b}, \mathrm{b}))$$
$$\forall b_1, b_2, b_3.((\mathrm{above}(\mathrm{b}_1, \mathrm{b}_2) \land \mathrm{above}(\mathrm{b}_2, \mathrm{b}_3)) \to \mathrm{above}(\mathrm{b}_1, \mathrm{b}_3))$$

A Knowledge Base Example

Knowledge base:

$$\begin{gathered}\text{block(red)} \land \text{block(blue)} \land \text{block(green)} \land \text{block(yellow)} \land \text{block(orange)} \\ \text{on(orange)} = \text{green} \land \text{above(yellow, green)} \land \neg\text{block(on(red))} \\ \forall b_1, b_2.(\text{on}(\text{b}_1) = \text{on}(\text{b}_2) \rightarrow b_1 = b_2) \\ \forall b.(\text{above(b, on(b))} \land \text{on(b)} \neq \text{b} \land \neg\text{above(b, b)}) \\ \forall b_1, b_2, b_3.((\text{above}(\text{b}_1, \text{b}_2) \land \text{above}(\text{b}_2, \text{b}_3)) \rightarrow \text{above}(\text{b}_1, \text{b}_3))\end{gathered}$$

What is the red block on?

What configurations are possible?

What about the following reasoning questions?

- Is yellow $=$ green consistent with the KB?
- Is on(yellow) $=$ green consistent with the KB?
- Does the KB entail $\forall b_1, b_2.(\text{above}(\text{b}_1, \text{b}_2) \rightarrow \text{on}(\text{b}_2) \neq b_1)$?
- Does the KB entail above(yellow, orange)?

Some KB Concepts

Some concepts we just saw:

Open or Closed World?

- If closed then only things that are true are the things that are stated/derived.
- If open an interpretation can add consistent true things

First-order logic has an open-world semantics, whereas Prolog has a closed world. The difference is subtle but can be important.

Some possible assumptions

- Domain Closure - All elements of the domain are explicitly mentioned.
- Unique Names - Things with different names are necessarily unique.

We return to these ideas next week.

First-Order Logic

We have seen first-order logic used informally.

We will now review precisely its syntax, semantics, and reasoning tasks.

You may have already seen first-order logic, e.g. in

Next week we will look at rules for reasoning in first-order logic and how to apply them to solve reasoning tasks.

First-order logic: Syntax

Definition (Signature)

The signature $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathsf{arr} \rangle$ of a first-order formula consists of a disjoint sets of function and predicate symbols $\mathcal{F}$ and $\mathcal{P}$, and a function $\mathsf{arr} : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}_0$ giving the arity (number of parameters) for each symbol.

First-order logic: Syntax

Definition (Signature)

The signature $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathsf{arr} \rangle$ of a first-order formula consists of a disjoint sets of function and predicate symbols $\mathcal{F}$ and $\mathcal{P}$, and a function $\mathsf{arr} : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}_0$ giving the arity (number of parameters) for each symbol.

Definition (Term - similar to Prolog)

A variable is a term. A constant symbol ($c \in \mathcal{F}$ s.t. $\mathsf{arr}(c) = 0$) is a term. If $f \in \mathcal{F}$ and $\mathsf{arr}(f) = n > 0$ then $f(t_1, \ldots t_n)$ is a term if each t_i is a term.

First-order logic: Syntax

Definition (Signature)

The signature $\Sigma = \langle \mathcal{F}, \mathcal{P}, \text{arr} \rangle$ of a first-order formula consists of a disjoint sets of function and predicate symbols $\mathcal{F}$ and $\mathcal{P}$, and a function $\text{arr} : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}_0$ giving the arity (number of parameters) for each symbol.

Definition (Term - similar to Prolog)

A variable is a term. A constant symbol ($c \in \mathcal{F}$ s.t. $\text{arr}(c) = 0$) is a term. If $f \in \mathcal{F}$ and $\text{arr}(f) = n > 0$ then $f(t_1, \ldots t_n)$ is a term if each t_i is a term.

Terms are similar to Prolog in syntax but not semantics. In first-order logic terms do not automatically have an interpretation as algebraic datatypes.

E.g. it is possible that $f(a, b) = a$, or $add(one, zero) = one$.

First-order logic: Syntax

Definition (Atoms)

A proposition ($p \in \mathcal{P}$ s.t. $\mathrm{arr}(p) = 0$) is an atom. If t_1 and t_2 are terms then $t_1 \neq t_2$ is an atom. If $p \in \mathcal{P}$ and $\mathrm{arr}(p) = n > 0$ then $p(t_1, \ldots t_n)$ is an atom if each t_i is a term.

First-order logic: Syntax

Definition (Atoms)

A proposition ($p \in \mathcal{P}$ s.t. $\text{arr}(p) = 0$) is an atom. If t_1 and t_2 are terms then $t_1 \neq t_2$ is an atom. If $p \in \mathcal{P}$ and $\text{arr}(p) = n > 0$ then $p(t_1, \ldots t_n)$ is an atom if each t_i is a term.

Definition (Formula)

All atoms are formulas. The boolean value true is a formula. If ϕ_1 and ϕ_2 are formulae then so are $\neg\phi_1$, $\phi_1 \wedge \phi_2$, and $\forall x.\phi_1$.

First-order logic: Syntax

Definition (Atoms)

A proposition ($p \in \mathcal{P}$ s.t. $\mathrm{arr}(p) = 0$) is an atom. If t_1 and t_2 are terms then $t_1 \neq t_2$ is an atom. If $p \in \mathcal{P}$ and $\mathrm{arr}(p) = n > 0$ then $p(t_1, \ldots t_n)$ is an atom if each t_i is a term.

Definition (Formula)

All atoms are formulas. The boolean value true is a formula. If ϕ_1 and ϕ_2 are formulae then so are $\neg\phi_1$, $\phi_1 \wedge \phi_2$, and $\forall x.\phi_1$.

We define false $= \neg$true, $\phi_1 \vee \phi_2 = \neg(\phi_1 \wedge \phi_2)$, $\phi_1 \rightarrow \phi_2 = \neg\phi_1 \vee \phi_2$, $\phi_1 \leftrightarrow \phi_2 = ((\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1))$ and $\exists x.\phi = \neg(\forall x.\neg\phi)$.

First-order logic: Syntax

Definition (Atoms)

A proposition ($p \in \mathcal{P}$ s.t. $\text{arr}(p) = 0$) is an atom. If t_1 and t_2 are terms then $t_1 \neq t_2$ is an atom. If $p \in \mathcal{P}$ and $\text{arr}(p) = n > 0$ then $p(t_1, \ldots t_n)$ is an atom if each t_i is a term.

Definition (Formula)

All atoms are formulas. The boolean value true is a formula. If ϕ_1 and ϕ_2 are formulae then so are $\neg\phi_1$, $\phi_1 \wedge \phi_2$, and $\forall x.\phi_1$.

We define false $= \neg$true, $\phi_1 \vee \phi_2 = \neg(\phi_1 \wedge \phi_2)$, $\phi_1 \rightarrow \phi_2 = \neg\phi_1 \vee \phi_2$, $\phi_1 \leftrightarrow \phi_2 = ((\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1))$ and $\exists x.\phi = \neg(\forall x.\neg\phi)$.

In terms of precedence, unary symbols bind tighter than binary symbols. Otherwise, I use parenthesis to disambiguate.

How to Read Formulas

$=$	equals
$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	implies, or sometimes if *left* then *right*
$\leftrightarrow$	equivalent, bi-implication, if-and-only-if
$\forall$	(for) all, (for) every
$\exists$	exists, some

I will write $\neq$ (read 'not equal') instead of $\neg(t_1 = t_2)$ as it is nicer.

Examples of Signature, Terms, Atoms, and Formulas

Let $\langle\{a, f, g\}, \{p, q\}, \{(a, 0), (f, 1), (g, 2), (p, 1), (q, 2)\}\rangle$ be a signature *(note the set-based syntax for representing functions with finite-domain)*

Examples of terms where x and y are variables:

$$x \quad a \quad f(a) \quad f(f(a)) \quad g(a, x) \quad g(f(g(x, x)), f(y))$$

Examples of atoms

$$p(a) \quad p(x) \quad q(x, y) \quad a = f(a) \quad x = y \quad g(x, y) = g(y, x)$$

Examples of formulas (including defined symbols):

$$\neg p(a) \quad p(x) \rightarrow p(a) \quad \exists x.\neg p(x) \quad \forall x.(p(x) \rightarrow p(a)) \quad (\forall x.p(x)) \rightarrow p(a)$$

$$x = y \leftrightarrow y = x \quad \exists x.\forall x.\exists x.(x = x)$$

Non-Examples

Let $\langle \{a, f, g\}, \{p, q\}, \{(a, 0), (f, 1), (g, 2), (p, 1), (q, 2)\}\rangle$ be a signature

These are **not** terms

- b as it's not in the signature (but signature normally implicit)
- $f(x, x)$ or f as they have the wrong arity for f
- $f(x = a)$ of $f(p(a))$ as $x = a$ and $p(a)$ are not terms

These are **not** formulas

- $x = (a = b)$ as $=$ can only be applied to terms
- $\forall f.f(x) = a$ as we cannot quantify over function symbols
- $1 + 2 = 3$ as 1, 2, and $+$ are not in the signature and functions are prefix not infix (e.g. it would be $+(1,2) = 3$)

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human
Every red car is cool
Every bird that is not a penguin or an ostrich can fly
If something is either fruit or cheese then it is food
Every car is either petrol or diesel
No car is both petrol and diesel

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	
Every bird that is not a penguin or an ostrich can fly	
If something is either fruit or cheese then it is food	
Every car is either petrol or diesel	
No car is both petrol and diesel	

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \wedge car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	
If something is either fruit or cheese then it is food	
Every car is either petrol or diesel	
No car is both petrol and diesel	

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \wedge car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	$\forall x.((bird(x) \wedge x \neq penguin \wedge x \neq ostrich) \rightarrow fly(x))$
If something is either fruit or cheese then it is food	
Every car is either petrol or diesel	
No car is both petrol and diesel	

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \wedge car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	$\forall x.((bird(x) \wedge x \neq penguin \wedge x \neq ostrich) \rightarrow fly(x))$
If something is either fruit or cheese then it is food	$(\forall x.fruit(x) \rightarrow food(x)) \wedge (\forall x.cheese(x) \rightarrow food(x))$
Every car is either petrol or diesel	
No car is both petrol and diesel	

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \land car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	$\forall x.((bird(x) \land x \neq penguin \land x \neq ostrich) \rightarrow fly(x))$
If something is either fruit or cheese then it is food	$\forall x. \left(\begin{array}{c} fruit(x) \lor \\ cheese(x) \end{array} \right) \rightarrow fruit(x)$
Every car is either petrol or diesel	
No car is both petrol and diesel	

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \wedge car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	$\forall x.((bird(x) \wedge x \neq penguin \wedge x \neq ostrich) \rightarrow fly(x))$
If something is either fruit or cheese then it is food	$\forall x.\left(\begin{array}{l} fruit(x) \vee \\ cheese(x) \end{array}\right) \rightarrow fruit(x)$
Every car is either petrol or diesel	$\forall x.car(x) \rightarrow \left(\begin{array}{l} petrol(x) \vee \\ diesel(x) \end{array}\right)$
No car is both petrol and diesel	

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \wedge car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	$\forall x.((bird(x) \wedge x \neq penguin \wedge x \neq ostrich) \rightarrow fly(x))$
If something is either fruit or cheese then it is food	$\forall x.\left(\begin{array}{l} fruit(x) \vee \\ cheese(x) \end{array}\right) \rightarrow fruit(x)$
Every car is either petrol or diesel	$\forall x.car(x) \rightarrow \left(\begin{array}{l} petrol(x) \vee \\ diesel(x) \end{array}\right)$
No car is both petrol and diesel	$\forall x.\neg(petrol(x) \wedge diesel(x))$

Examples of Universal Quantification

Some Examples of how to use $\forall$

Every man is human	$\forall x.(man(x) \rightarrow human(x))$
Every red car is cool	$\forall x.((red(x) \wedge car(x)) \rightarrow cool(x))$
Every bird that is not a penguin or an ostrich can fly	$\forall x.((bird(x) \wedge x \neq penguin \wedge x \neq ostrich) \rightarrow fly(x))$
If something is either fruit or cheese then it is food	$\forall x. \left(\begin{array}{l} fruit(x) \vee \\ cheese(x) \end{array} \right) \rightarrow fruit(x)$
Every car is either petrol or diesel	$\forall x.car(x) \rightarrow \left(\begin{array}{l} petrol(x) \vee \\ diesel(x) \end{array} \right)$
No car is both petrol and diesel	$\neg\exists x.(petrol(x) \wedge diesel(x))$

Examples of Existential Quantification

Some examples of how to use $\exists$

There is a human man
There are at least two men
Everybody has a mother

If two different people have parents who are siblings then they are cousins

Examples of Existential Quantification

Some examples of how to use $\exists$

There is a human man	$\exists x.(man(x) \wedge human(x))$
There are at least two men	
Everybody has a mother	

If two different people have parents who are siblings then they are cousins

Examples of Existential Quantification

Some examples of how to use $\exists$

There is a human man	$\exists x.(man(x) \wedge human(x))$
There are at least two men	$\exists x.\exists y.(man(x) \wedge man(y) \wedge x \neq y)$
Everybody has a mother	

If two different people have parents who are siblings then they are cousins

Examples of Existential Quantification

Some examples of how to use $\exists$

There is a human man	$\exists x.(man(x) \wedge human(x))$
There are at least two men	$\exists x.\exists y.(man(x) \wedge man(y) \wedge x \neq y)$
Everybody has a mother	$\forall x.\exists y.(person(x) \rightarrow mother\_of(y, x))$

If two different people have parents who are siblings then they are cousins

Examples of Existential Quantification

Some examples of how to use $\exists$

There is a human man	$\exists x.(man(x) \wedge human(x))$
There are at least two men	$\exists x.\exists y.(man(x) \wedge man(y) \wedge x \neq y)$
Everybody has a mother	$\forall x.\exists y.(person(x) \rightarrow mother\_of(y, x))$

If two different people have parents who are siblings then they are cousins

$$\forall x, y.(x \neq y \wedge (\exists u, v. \left(\begin{array}{l} parent(u, x) \wedge \\ parent(v, y) \wedge \\ sibling(u, v) \end{array} \right))) \rightarrow cousins(x, y)$$

Examples of Existential Quantification

Some examples of how to use $\exists$

There is a human man	$\exists x.(man(x) \land human(x))$
There are at least two men	$\exists x.\exists y.(man(x) \land man(y) \land x \neq y)$
Everybody has a mother	$\forall x.\exists y.(person(x) \rightarrow mother\_of(y, x))$

If two different people have parents who are siblings then they are cousins

$$\forall x, y, u, v.(x \neq y \land \begin{pmatrix} parent(u, x) \land \\ parent(v, y) \land \\ sibling(u, v) \end{pmatrix}) \rightarrow cousins(x, y)$$

Examples of Existential Quantification

Existential quantification over an empty domain is false.

Existential quantification over a finite domain is finite disjunction

$$(\forall x.(x = a \vee x = b)) \rightarrow ((\exists x.p(x)) \leftrightarrow (p(a) \vee p(b)))$$

Which Formula?

Everybody is either a Manchester United Supporter or a Manchester City Supporter and cannot be both.

1. $\forall p.(person(p) \rightarrow (manU(p) \vee manC(p)))$
2. $\forall p.(person(p) \rightarrow ((manU(p) \vee manC(p)) \wedge \neg(manU(p) \wedge manC(p))))$
3. $\exists p_1, p_2.(person(p_1) \wedge person(p_2) \wedge p_1 \neq p_2 \wedge manU(p_1) \wedge manC(p_2))$
4. $\forall p.(person(p) \rightarrow (\exists s.(supports(p, s) \wedge (s = manU \vee s = manC))))$

Which Formula?

Everybody is either a Manchester United Supporter or a Manchester City Supporter and cannot be both.

1. $\forall p.(person(p) \rightarrow (manU(p) \vee manC(p)))$
2. $\forall p.(person(p) \rightarrow ((manU(p) \vee manC(p)) \wedge \neg(manU(p) \wedge manC(p))))$
3. $\exists p_1, p_2.(person(p_1) \wedge person(p_2) \wedge p_1 \neq p_2 \wedge manU(p_1) \wedge manC(p_2))$
4. $\forall p.(person(p) \rightarrow (\exists s.(supports(p, s) \wedge (s = manU \vee s = manC))))$

Which Formula?

Everybody is either a Manchester United Supporter or a Manchester City Supporter and cannot be both.

1. $\forall p.(person(p) \rightarrow (manU(p) \vee manC(p))) \wedge (manU(p) \leftrightarrow \neg manC(p))$
2. $\forall p.(person(p) \rightarrow ((manU(p) \vee manC(p)) \wedge \neg(manU(p) \wedge manC(p))))$
3. $\exists p_1, p_2.(person(p_1) \wedge person(p_2) \wedge p_1 \neq p_2 \wedge manU(p_1) \wedge manC(p_2))$
4. $\forall p.(person(p) \rightarrow (\exists s.(supports(p, s) \wedge (s = manU \vee s = manC))))$

Which Formula?

Everybody is either a Manchester United Supporter or a Manchester City Supporter and cannot be both.

1. $\forall p.(person(p) \rightarrow (manU(p) \vee manC(p))) \wedge (manU(p) \leftrightarrow \neg manC(p))$
2. $\forall p.(person(p) \rightarrow ((manU(p) \vee manC(p)) \wedge \neg(manU(p) \wedge manC(p))))$
3. $\exists p_1, p_2.(person(p_1) \wedge person(p_2) \wedge p_1 \neq p_2 \wedge manU(p_1) \wedge manC(p_2))$
4. $\forall p.(person(p) \rightarrow (\exists s.(supports(p, s) \wedge (s = manU \vee s = manC))))$
5. $\forall p.(person(p) \rightarrow \left(\begin{array}{l} supports(p) = manU \vee \\ supports(p) = manC \end{array}\right)) \wedge manU \neq manC$

Which Formula?

There is a book that if I read it I will score the best mark in all of my exams.

1. $\forall e.\exists b.(read(b) \wedge (take(e) \rightarrow best(e)))$
2. $\exists b.(read(b) \rightarrow \forall e.(take(e) \rightarrow best(e)))$
3. $(\exists b.read(b)) \wedge (\forall e.(take(e) \rightarrow best(e)))$
4. $\exists b.(read(b) \rightarrow \forall e.(take(e) \rightarrow \forall p.(takes(p,e) \rightarrow better(e,p))))$

Which Formula?

There is a book that if I read it I will score the best mark in all of my exams.

1. $\forall e.\exists b.(read(b) \wedge (take(e) \rightarrow best(e)))$
2. $\exists b.(read(b) \rightarrow \forall e.(take(e) \rightarrow best(e)))$
3. $(\exists b.read(b)) \wedge (\forall e.(take(e) \rightarrow best(e)))$
4. $\exists b.(read(b) \rightarrow \forall e.(take(e) \rightarrow \forall p.(takes(p,e) \rightarrow better(e,p))))$

Which Formula?

There is a book that if I read it I will score the best mark in all of my exams.

1. $\forall e.\exists b.(read(b) \land (take(e) \rightarrow best(e)))$
2. $\exists b.(read(b) \rightarrow \forall e.(take(e) \rightarrow best(e)))$
3. $(\exists b.read(b)) \land (\forall e.(take(e) \rightarrow best(e)))$
4. $\exists b.(read(b) \rightarrow \forall e.(take(e) \rightarrow \forall p.(takes(p, e) \rightarrow better(e, p))))$

Why First-Order Logic

The order of a logic is related to the kind of things one can quantify over.

Propositional logic is ‘zero’ ordered as one cannot quantify.

Extensions of propositional logic such as QBF or PLFD are useful for modelling and allow for efficient reasoning but do not increase expressive power.

First-order logic allows one to quantify over individuals

Second-order logic allows one to quantify over sets of individuals or equivalently predicts over individuals.

And so on.

First-order logic: Semantics

So far we have been working with an intuition of what first-order formulas mean but this isn't good enough.

We will define their semantics by defining

- Formulas that can have meaning e.g. those without free variables
- Interpretations to assign truth values to sentences
- Ways to handle equality
- The notion of a model

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $\forall x.p(x, y)$?

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $\forall x.(p(x) \rightarrow \exists x.q(x))$?

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $(\forall y.p(x,y)) \wedge (\exists x.p(x,y))$

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $(\forall y.p(x, y)) \wedge (\exists x.p(x, y))$

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $(\forall y.p(x,y)) \wedge (\exists x.p(x,y))$

This last one is a bit confusing as x and y occur bound and free. Later we rewrite formulas to avoid this (rectification).

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $(\forall y.p(x,y)) \wedge (\exists x.p(x,y))$

This last one is a bit confusing as x and y occur bound and free. Later we rewrite formulas to avoid this (rectification).

Some useful notation - If ϕ has free variables X we might write it $\phi[X]$. We write $\phi[V]$ for the formula $\phi[X]$ where X is replaced by V.

Free Variables, Sentences and Closure

The free variables of a formula f are those not captured by a quantifer. Otherwise they are bound variables.

What are the free and bound variables in $(\forall y.p(x,y)) \wedge (\exists x.p(x,y))$

This last one is a bit confusing as x and y occur bound and free. Later we rewrite formulas to avoid this (rectification).

Some useful notation - If ϕ has free variables X we might write it $\phi[X]$. We write $\phi[V]$ for the formula $\phi[X]$ where X is replaced by V.

A formula is a sentence if it does not contain any free variables

The universal closure of $\phi[X]$ is $\forall X.\phi[X]$

Interpretation

An Interpretation allows us to assign a truth value to every sentence.

Let $\langle \mathcal{D}, \mathcal{I} \rangle$ be a structure such that $\mathcal{I}$ is an interpretation over a non-empty (possibly infinite) domain $\mathcal{D}$. We often leave $\mathcal{D}$ implicit and refer directly to $\mathcal{I}$.

The map $\mathcal{I}$ maps

- Every constant symbol to an element of $\mathcal{D}$
- Every function symbol of arity n to a function in $\mathcal{D}^n \rightarrow \mathcal{D}$
- Every proposition symbol to a truth value in $\mathbb{B}$
- Every predicate symbol of arity n to a function in $\mathcal{D}^n \rightarrow \mathbb{B}$

Interpretation of Atoms

We can then lift interpretations to non-constant terms recursively as

$$\mathcal{I}(f(t_1,\ldots,t_n)) = \mathcal{I}(f)(\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n))$$

(variables are ignored as we only consider ground terms)

We can lift interpretations to non-propositional atoms similarly e.g.

$$\begin{array}{rcl} \mathcal{I}(t_1 = t_2) & = & \mathcal{I}(t_1) = \mathcal{I}(t_2) \\ \mathcal{I}(p(t_1,\ldots,t_n)) & = & \mathcal{I}(p)(\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n)) \end{array}$$

Every atom is interpreted as a truth value.

Interpretation of Atoms

We can then lift interpretations to non-constant terms recursively as

$$\mathcal{I}(f(t_1, \ldots, t_n)) = \mathcal{I}(f)(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$$

(variables are ignored as we only consider ground terms)

We can lift interpretations to non-propositional atoms similarly e.g.

$$\begin{array}{rcl} \mathcal{I}(t_1 = t_2) & = & \mathcal{I}(t_1) = \mathcal{I}(t_2) \\ \mathcal{I}(p(t_1, \ldots, t_n)) & = & \mathcal{I}(p)(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)) \end{array}$$

Every atom is interpreted as a truth value.

Careful, this line mixes three kinds of equality, what are they?

Interpreting Equality

Usually FOL is introduced without equality and then extended. But I am introducing FOL with equality directly as it makes modelling things in FOL easier.

Usually in the extension the set of models is restricted to normal models that interpret equality as equality and such that all domain elements are distinct with respect to equality. We will enforce this straight away here.

Interpreting Equality

Usually FOL is introduced without equality and then extended. But I am introducing FOL with equality directly as it makes modelling things in FOL easier.

Usually in the extension the set of models is restricted to normal models that interpret equality as equality and such that all domain elements are distinct with respect to equality. We will enforce this straight away here.

We do not *need* to make equality directly part of the language. Adding

$$\forall x.x = x \qquad \forall x, y.(x = y \rightarrow y = x) \qquad \forall x, y, z.((x = y \wedge y = z) \rightarrow x = z)$$
$$\forall x_1, \ldots x_n, y_1, \ldots y_n.((x_1 = y_1 \wedge \ldots x_n = y_n) \rightarrow f(x_1, \ldots x_n) = f(y_1, \ldots, y_n))$$
$$\forall x_1, \ldots x_n, y_1, \ldots y_n.((x_1 = y_1 \wedge \ldots x_n = y_n) \rightarrow p(x_1, \ldots x_n) \leftrightarrow p(y_1, \ldots, y_n))$$

(for all functions f and predicates p) forces $=$ to behave as above.

Interpreting Equality

Usually FOL is introduced without equality and then extended. But I am introducing FOL with equality directly as it makes modelling things in FOL easier.

Usually in the extension the set of models is restricted to normal models that interpret equality as equality and such that all domain elements are distinct with respect to equality. We will enforce this straight away here.

We do not *need* to make equality directly part of the language. Adding

$$\forall x. x = x \qquad \forall x, y.(x = y \rightarrow y = x) \qquad \forall x, y, z.((x = y \wedge y = z) \rightarrow x = z)$$
$$\forall x_1, \ldots x_n, y_1, \ldots y_n.((x_1 = y_1 \wedge \ldots x_n = y_n) \rightarrow f(x_1, \ldots x_n) = f(y_1, \ldots, y_n))$$
$$\forall x_1, \ldots x_n, y_1, \ldots y_n.((x_1 = y_1 \wedge \ldots x_n = y_n) \rightarrow p(x_1, \ldots x_n) \leftrightarrow p(y_1, \ldots, y_n))$$

(for all functions f and predicates p) forces $=$ to behave as above.

We will do this next week as an initial approach to reasoning with equality (it is sound and complete to do so, but can be inefficient).

Interpretation of Formulas

Finally we interpret formulas

$\mathcal{I}(\mathit{true})$	is always true
$\mathcal{I}(\neg\phi)$	*iff* $\mathcal{I}(\phi)$ is not true
$\mathcal{I}(\phi_1 \wedge \phi_2)$	*iff* both $\mathcal{I}(\phi_1)$ and $\mathcal{I}(\phi_2)$ are true
$\mathcal{I}(\forall x.\phi[x])$	*iff* for every $d \in \mathcal{D}$ we have that $\mathcal{I}(\phi[d])$ is true

Hopefully the first 3 are straightforward. For $\forall$ we take each element d of the domain and see if the quantified formula holds if we replace x by d.

We could extend this for the derived operators but do not need to.

When is an Interpretation a Model?

We use the term model to refer to interpretations that make a sentence true - this doesn't apply to all interpretations.

For example, consider the sentence $a = f(a)$ and $\mathcal{I} = \langle \{1,2\}, \{a \mapsto 1, f \mapsto \{(1,2),(2,1)\}\} \rangle$ e.g. a domain with two elements where f returns the opposite element. We have

$$\begin{array}{rcl} \mathcal{I}(f(a)) & = & \mathcal{I}(f)(\mathcal{I}(a)) = (\{(1,2),(2,1)\}(1) = 2 \\ \mathcal{I}(a = f(a)) & = & (\mathcal{I}(a) = \mathcal{I}(f(a))) = (1 = 2) = \mathit{false} \end{array}$$

An interpretation $\mathcal{I}$ is a model for a sentence φ if $\mathcal{I}(\varphi)$ is true.

We might use the notation $\mathcal{I} \models \varphi$ and I might refer to models using $\mathcal{M}$.

Variable Renaming

Note that the semantics of a formula is independent of the names given to bound variables.

For example, the sentences

$$\forall banana.(\mathrm{like}(banana) \rightarrow \mathrm{eat}(banana))$$

has the same models as

$$\forall rock.(\mathrm{like}(rock) \rightarrow \mathrm{eat}(rock))$$

This means that it is always safe to consistently rename variables. In fact, we will do this very often when applying reasoning rules later.

It also means that we will usually assume that formulas have been rectified, which means renaming variables so that all bound variables are different.

Defining Functions (an aside)

In these slides I use two kinds of syntax to define functions.

Firstly, for finite functions I might use the set representation we saw before e.g. $\{(1, 2), (2, 1)\}$ gives the arguments and result as pairs/tuples and defines $f(1) = 2, f(2) = 1$.

Alternatively, I will use λ-abstractions e.g.

$$\lambda x.\ (\textit{expression using } x)$$

to represent an anonymous function that takes something (x) as input and returns the result of evaluating the expression on that input.

This notation is borrowed from the λ-calculus, which you have seen briefly.

A Formula Can Have Many Models

$$
\begin{array}{l}
\mathit{parent}(\mathit{giles}, \mathit{mark}) \\
\mathit{man}(\mathit{giles}) \\
\forall x.((\mathit{man}(x) \land \exists y.\mathit{parent}(x, y)) \rightarrow \mathit{father}(x))
\end{array}
$$

We need to interpret $\mathit{parent}, \mathit{giles}, \mathit{mark}, \mathit{man}, \mathit{father}$. Fix $\mathcal{D} = \{1, 2\}$.

A Formula Can Have Many Models

$$\begin{array}{l} parent(giles, mark) \\ man(giles) \\ \forall x.((man(x) \land \exists y.parent(x, y)) \rightarrow father(x)) \end{array}$$

We need to interpret $parent, giles, mark, man, father$. Fix $\mathcal{D} = \{1, 2\}$.

Giles is Mark. Everything is true.

$$\begin{array}{l} \mathcal{I}(giles) = 1 \\ \mathcal{I}(mark) = 1 \\ \mathcal{I}(parent) = (\lambda x, y.(true)) \\ \mathcal{I}(man) = (\lambda x.(true)) \\ \mathcal{I}(father) = (\lambda x.(true)) \end{array}$$

A Formula Can Have Many Models

$$\begin{array}{l} parent(giles, mark) \\ man(giles), mark \neq giles \\ \forall x.((man(x) \land \exists y.parent(x,y)) \rightarrow father(x)) \end{array}$$

We need to interpret *parent*, *giles*, *mark*, *man*, *father*. Fix $\mathcal{D} = \{1, 2\}$.

Giles and Mark different. Everything is true.

$$\begin{array}{l} \mathcal{I}(giles) = 1 \\ \mathcal{I}(mark) = 2 \\ \mathcal{I}(parent) = (\lambda x, y.(true)) \\ \mathcal{I}(man) = (\lambda x.(true)) \\ \mathcal{I}(father) = (\lambda x.(true)) \end{array}$$

A Formula Can Have Many Models

$$\begin{array}{l} parent(giles, mark) \\ man(giles), mark \neq giles \\ \forall x.((man(x) \land \exists y.parent(x, y)) \rightarrow father(x)) \end{array}$$

We need to interpret $parent, giles, mark, man, father$. Fix $\mathcal{D} = \{1, 2\}$.

Giles and Mark different. Least is true.

$$\begin{array}{l} \mathcal{I}(giles) = 1 \\ \mathcal{I}(mark) = 2 \\ \mathcal{I}(parent) = (\lambda x, y.(x = 1 \land y = 2))) \\ \mathcal{I}(man) = (\lambda x.(x = 2)) \\ \mathcal{I}(father) = (\lambda x.(x = 2)) \end{array}$$

A Formula Can Have Many Models

$$\begin{array}{l} parent(giles, mark) \\ man(giles), mark \neq giles \\ \forall x.((man(x) \land \exists y.parent(x, y)) \rightarrow father(x)) \end{array}$$

We need to interpret $parent, giles, mark, man, father$. Fix $\mathcal{D} = \{1, 2\}$.

Giles and Mark different. A bit more is true.

$$\begin{array}{l} \mathcal{I}(giles) = 1 \\ \mathcal{I}(mark) = 2 \\ \mathcal{I}(parent) = (\lambda x, y.(x = 1 \land y = 2))) \\ \mathcal{I}(man) = (\lambda x.(x = 2 \lor x = 1)) \\ \mathcal{I}(father) = (\lambda x.(x = 2)) \end{array}$$

Is this a Model?

$$p(a, b) \wedge p(b, a)$$

$$\begin{array}{rcl}
\mathcal{I}(a) & = & 1 \\
\mathcal{I}(b) & = & 2 \\
\mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\}
\end{array}$$

Is this a Model?

$$p(a, b) \wedge p(b, a)$$

$$\begin{array}{rcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

Yes

Is this a Model?

$$p(a,b) \wedge p(b,a) \wedge \neg p(a,a)$$

$$\begin{array}{lcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

Is this a Model?

$$p(a, b) \wedge p(b, a) \wedge \neg p(a, a)$$

$$\begin{array}{rcl}\mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\}\end{array}$$

No

Is this a Model?

$$\forall x.\forall y.p(x,y)$$

$$\begin{array}{rcl}\mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\}\end{array}$$

Is this a Model?

$$\forall x.\forall y.p(x,y)$$

$$\begin{array}{rcl}\mathcal{I}(a) & = & 1\\ \mathcal{I}(b) & = & 2\\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\}\end{array}$$

$\mathcal{I}(\forall x.\phi[x])$ *iff* for every $d \in \mathcal{D}$ we have that $\mathcal{I}(\phi[d])$ is true

Is this a Model?

$$\forall x.\forall y.p(x,y)$$

$$\begin{array}{lcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

$\mathcal{I}(\forall x.\forall y.p(x,y))$ *iff* for every $d \in \mathcal{D}$ we have that $\mathcal{I}(\forall y.p(d,y))$ is true

Is this a Model?

$$\forall x. \forall y. p(x, y)$$

$$\begin{array}{lcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

$\mathcal{I}(\forall y.p(1, y))$ is true and $\mathcal{I}(\forall y.p(2, y))$ is true

Is this a Model?

$$\forall x. \forall y. p(x, y)$$

$$\begin{array}{lcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

$\mathcal{I}(p(1,1))$ is true, $\mathcal{I}(p(1,2))$ is true, $\mathcal{I}(p(2,1))$ is true, $\mathcal{I}(p(2,2))$ is true

Is this a Model?

$$\forall x.\forall y.p(x, y)$$

$$\begin{aligned}\mathcal{I}(a) &= 1\\ \mathcal{I}(b) &= 2\\ \mathcal{I}(p) &= \{(1,2),(2,1),(1,1)\}\end{aligned}$$

$\mathcal{I}(p(1,1))$ is true, $\mathcal{I}(p(1,2))$ is true, $\mathcal{I}(p(2,1))$ is true, $\mathcal{I}(p(2,2))$ is true

Is this a Model?

$$\forall x.\forall y.p(x,y)$$

$$\begin{aligned}\mathcal{I}(a) &= 1\\ \mathcal{I}(b) &= 2\\ \mathcal{I}(p) &= \{(1,2),(2,1),(1,1)\}\end{aligned}$$

No

Is this a Model?

$$\exists x, y, z.(x \neq y \wedge x \neq z \wedge y \neq z)$$

$$\begin{array}{rcl}
\mathcal{I}(a) & = & 1 \\
\mathcal{I}(b) & = & 2 \\
\mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\}
\end{array}$$

Is this a Model?

$$\exists x, y, z.(x \neq y \wedge x \neq z \wedge y \neq z)$$

$$\begin{array}{rcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

No

Is this a Model?

$$\exists x, y, z.(x \neq y \wedge x \neq z \wedge y \neq z)$$

$$\begin{array}{lcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(c) & = & 3 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

Is this a Model?

$$\exists x, y, z.(x \neq y \wedge x \neq z \wedge y \neq z)$$

$$\begin{array}{rcl} \mathcal{I}(a) & = & 1 \\ \mathcal{I}(b) & = & 2 \\ \mathcal{I}(c) & = & 3 \\ \mathcal{I}(p) & = & \{(1,2),(2,1),(1,1)\} \end{array}$$

Yes

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \end{array}$$

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$

$$\begin{array}{rcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \end{array}$$

No

Is this a Model?

$$\forall x.(x \neq suc(x)) \wedge \forall x.\exists y.(suc(x) = y) \wedge \neg\exists x.(zero = suc(x))$$

$$\begin{array}{lcl}
\mathcal{I}(zero) & = & 1 \\
\mathcal{I}(suc(1)) & = & 2 \\
\mathcal{I}(suc(2)) & = & 3 \\
\mathcal{I}(suc(3)) & = & 4
\end{array}$$

Is this a Model?

$$\forall x.(x \neq suc(x)) \wedge \forall x.\exists y.(suc(x) = y) \wedge \neg\exists x.(zero = suc(x))$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \\ \mathcal{I}(suc(2)) & = & 3 \\ \mathcal{I}(suc(3)) & = & 4 \end{array}$$

No

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \\ \mathcal{I}(suc(2)) & = & 3 \\ \mathcal{I}(suc(3)) & = & 2 \end{array}$$

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \\ \mathcal{I}(suc(2)) & = & 3 \\ \mathcal{I}(suc(3)) & = & 2 \end{array}$$

Yes

Is this a Model?

$$\forall x.(x \neq suc(x)) \wedge \forall x.\exists y.(suc(x) = y) \wedge \neg\exists x.(zero = suc(x))$$

$$\forall x, y.(suc(x) = suc(y) \rightarrow x = y)$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \\ \mathcal{I}(suc(2)) & = & 3 \\ \mathcal{I}(suc(3)) & = & 2 \end{array}$$

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$
$$\forall x, y.(suc(x) = suc(y) \rightarrow x = y)$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \\ \mathcal{I}(suc(2)) & = & 3 \\ \mathcal{I}(suc(3)) & = & 2 \end{array}$$

No

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$
$$\forall x, y.(suc(x) = suc(y) \rightarrow x = y)$$

$$\begin{array}{lcl}
\mathcal{I}(zero) & = & 1 \\
\mathcal{I}(suc(1)) & = & 2 \\
\mathcal{I}(suc(2)) & = & 3 \\
\mathcal{I}(suc(3)) & = & 4 \\
\mathcal{I}(suc(4)) & = & 5 \\
\ldots & &
\end{array}$$

Is this a Model?

$$\forall x.(x \neq suc(x)) \land \forall x.\exists y.(suc(x) = y) \land \neg\exists x.(zero = suc(x))$$
$$\forall x, y.(suc(x) = suc(y) \rightarrow x = y)$$

$$\begin{array}{lcl} \mathcal{I}(zero) & = & 1 \\ \mathcal{I}(suc(1)) & = & 2 \\ \mathcal{I}(suc(2)) & = & 3 \\ \mathcal{I}(suc(3)) & = & 4 \\ \mathcal{I}(suc(4)) & = & 5 \\ \ldots & & \end{array}$$

Yes, it's infinite

Reasoning Tasks

A first-order knowledge base is a set of sentences.

Now we understand how the meaning of first-order knowledge bases are defined, we can consider what reasoning tasks we might specify.

Let Γ be a knowledge base. We might ask

- Is Γ consistent
  - e.g. does it have a model?
  - We might also ask if some new knowledge is consistent with existing knowledge e.g. is Γ consistent with ϕ
- Does Γ entail ϕ
  - e.g. is ϕ always true if Γ is true
  - Note that this talks about all possible models of Γ
- Which substitutions θ make Γ entail $\phi\theta$
  - e.g. if ϕ contains unknowns, what knowns can be substituted for them to make it follow from the KB

Satisfiability/Consistency and Validity

An interpretation satisfies a sentence if the sentence evaluates to true in it. Recall, we say that the interpretation is a model of that sentence

Satisfiability/Consistency and Validity

An interpretation satisfies a sentence if the sentence evaluates to true in it. Recall, we say that the interpretation is a model of that sentence

An interpretation satisfies a formula if it satisfies its universal closure.

A formula is satisfiable or consistent if it has a model.

Satisfiability/Consistency and Validity

An interpretation satisfies a sentence if the sentence evaluates to true in it. Recall, we say that the interpretation is a model of that sentence

An interpretation satisfies a formula if it satisfies its universal closure.

A formula is satisfiable or consistent if it has a model.

A formula is valid if *every* interpretation satisfies it.

Satisfiability/Consistency and Validity

An interpretation satisfies a sentence if the sentence evaluates to true in it. Recall, we say that the interpretation is a model of that sentence

An interpretation satisfies a formula if it satisfies its universal closure.

A formula is satisfiable or consistent if it has a model.

A formula is valid if *every* interpretation satisfies it.

A formula unsatisfiable or inconsistent if it has no models.

Satisfiability/Consistency and Validity

An interpretation satisfies a sentence if the sentence evaluates to true in it. Recall, we say that the interpretation is a model of that sentence

An interpretation satisfies a formula if it satisfies its universal closure.

A formula is satisfiable or consistent if it has a model.

A formula is valid if *every* interpretation satisfies it.

A formula unsatisfiable or inconsistent if it has no models.

Try writing down a consistent, a valid, and an inconsistent formula.

Entailment

We lift these notions to sets of formulas by interpreting them as conjunctions e.g. we can talk about a consistent knowledge-base.

Entailment

We lift these notions to sets of formulas by interpreting them as conjunctions e.g. we can talk about a consistent knowledge-base.

A formula ϕ is entailed by a set of formulas Γ if every model of Γ is also a model of ϕ, we write this

$$\Gamma \models \phi$$

also read as *f* is a consequence of Γ.

Entailment

We lift these notions to sets of formulas by interpreting them as conjunctions e.g. we can talk about a consistent knowledge-base.

A formula ϕ is entailed by a set of formulas Γ if every model of Γ is also a model of ϕ, we write this

$$\Gamma \models \phi$$

also read as f is a consequence of Γ.

This is equivalent to $\Gamma \rightarrow \phi$ being valid.

Entailment

We lift these notions to sets of formulas by interpreting them as conjunctions e.g. we can talk about a consistent knowledge-base.

A formula ϕ is entailed by a set of formulas Γ if every model of Γ is also a model of ϕ, we write this

$$\Gamma \models \phi$$

also read as f is a consequence of Γ.

This is equivalent to $\Gamma \rightarrow \phi$ being valid.

What if Γ is inconsistent?

Demonstrating Validity, Consistency, and Entailment

When there can be many models the notion of truth can change.

Let us take an example

$$man(aristotle) \qquad human(cleopatra) \qquad \forall x.(man(x) \rightarrow human(x))$$

There exists a model where cleopatra is a man.

In every model aristotle is human.

Demonstrating Validity, Consistency, and Entailment

When there can be many models the notion of truth can change.

Let us take an example

$$man(aristotle) \qquad human(cleopatra) \qquad \forall x.(man(x) \rightarrow human(x))$$

There exists a model where cleopatra is a man.

The set $\Gamma \cup man(cleopatra)$ is satisfiable or consistent

In every model aristotle is human.

Demonstrating Validity, Consistency, and Entailment

When there can be many models the notion of truth can change.

Let us take an example

$$man(aristotle) \qquad human(cleopatra) \qquad \forall x.(man(x) \rightarrow human(x))$$

There exists a model where cleopatra is a man.

The set $\Gamma \cup man(cleopatra)$ is satisfiable or consistent

In every model aristotle is human.

The statement $man(aristotle)$ is entailed by Γ

Demonstrating Validity, Consistency, and Entailment

When there can be many models the notion of truth can change.

Let us take an example

$$man(aristotle) \qquad human(cleopatra) \qquad \forall x.(man(x) \rightarrow human(x))$$

There exists a model where cleopatra is a man.

The set $\Gamma \cup man(cleopatra)$ is satisfiable or consistent

In every model aristotle is human.

The statement $man(aristotle)$ is entailed by Γ
The formula $\Gamma \rightarrow man(aristotle)$ is valid

Relation Between (In)Consistency and Validity

Let Γ be a set (conjunction) of formulas and ϕ be a sentence

The models of ϕ are exactly those that are not the models of $\neg\phi$

If Γ is inconsistent then $\Gamma \models \mathit{false}$.

$\Gamma \models \phi$ if and only if $\Gamma \cup \{\neg\phi\} \models \mathit{false}$

Relation Between (In)Consistency and Validity

Let Γ be a set (conjunction) of formulas and ϕ be a sentence

The models of ϕ are exactly those that are not the models of $\neg\phi$

If Γ is inconsistent then $\Gamma \models \mathit{false}$.

If Γ is inconsistent then it has no models, *false* has no models, all models of Γ are also models of *false*.

$\Gamma \models \phi$ if and only if $\Gamma \cup \{\neg\phi\} \models \mathit{false}$

Relation Between (In)Consistency and Validity

Let Γ be a set (conjunction) of formulas and ϕ be a sentence

The models of ϕ are exactly those that are not the models of $\neg\phi$

If Γ is inconsistent then $\Gamma \models$ *false*.

If Γ is inconsistent then it has no models, *false* has no models, all models of Γ are also models of *false*.

$\Gamma \models \phi$ if and only if $\Gamma \cup \{\neg\phi\} \models$ *false*

If ϕ is true in all models of Γ then $\neg\phi$ must be true in no models of Γ

Question Answering

A question or query can either be viewed as an existential conjecture or a formula with free variables needing instantiation. We take the former approach and assume any query formula with free variables has been existentially closed e.g. query $\phi[X]$ becomes $\exists X.\phi[X]$

Given knowledge base Γ and a query $\exists X.\phi[X]$ we search for a substitution σ such that

$$\Gamma \models \phi[X]\sigma$$

we do this be transforming the problem into

$$\Gamma \models \exists X.(\mathsf{ans}(X) \wedge \phi[X]) \wedge \exists X.\neg\mathsf{ans}(X)$$

for fresh predicate ans.

The result is that we have a single predicate capturing the result of the question, which we will then exploit during reasoning to extract the answer. We'll return to how we do this next week.

Summary

First-Order Logic has explicit universal and existential quantification and allows arbitrary boolean structure in formulas.

The semantics of a formula is defined in terms of interpretations.

A FOL formula can have many models. It can also have a single model (up to renaming of domain constants) or no models.

The main reasoning task in FOL is consistency-checking. This can be used to encode validity and entailment checking. We can encode question answering as a special kind of entailment check.

Next week:

- How are Prolog and FOL related?
- Efficient algorithms for deciding reasoning tasks in a subset of FOL
- Generalising this to a simple rule for all of FOL