Knowledge Representation

Lecture 2: Classical Logics

Patrick Koopmann

October 31, 2023

What is a Knowledge Representation?

after [Davis, Shrobe and Szolovits, 1993]:

1. Surrogate
2. Expression of ontological commitment
3. Theory of intelligent reasoning
4. Medium of efficient computation
5. Medium of human expression

Today we look at two examples: propositional logic and first-order logic

Propositional Logic

Propositional logic is an example of a simple KR

- Propositional variables abstract atoms of information
  - a: Tom comes to the VU.
  - c: Tom takes the bike.
  - b: Tom takes the tram.
  - d: It is sunny.
  - e: It is raining.
  - f: The bike is broken.
  - g: The tram company is on strike.
- Operators allow to build complex formulas
  1. $d \rightarrow \neg e$
  2. $a \leftrightarrow (b \vee c)$
  3. $(e \vee f) \rightarrow \neg b$
  4. $(d \wedge \neg f) \rightarrow b$
  5. $g \rightarrow \neg c$
  6. $e \wedge \neg g$
- A clear semantics defines what these formulas mean
- Automated reasoning can be used to infer implicit information
  - *Does Tom come to the VU?*

Propositional Logics: Assumptions

This KR formalism makes the following assumptions:

- Atomic sentences as building blocks
  - Represent facts we want to reason about
  - No inner structure

Propositional Logics: Assumptions

This KR formalism makes the following assumptions:

- Atomic sentences as building blocks
  - Represent facts we want to reason about
  - No inner structure
- We can build more complex sentences using operators

Propositional Logics: Assumptions

This KR formalism makes the following assumptions:

- Atomic sentences as building blocks
  - Represent facts we want to reason about
  - No inner structure
- We can build more complex sentences using operators
- Every sentence is either true or false

Propositional Logics: Reasoning Problems

What do we want to do with such sentences?

- Entailment
  - What does logically follow from my knowledge?
  - Example: *Does it follow from my knowledge that Tom comes to the VU?*

Propositional Logics: Reasoning Problems

What do we want to do with such sentences?

- Entailment
  - What does logically follow from my knowledge?
  - Example: *Does it follow from my knowledge that Tom comes to the VU?*
- Consistency
  - Is my knowledge consistent?
  - Does it describe a possible situation?
  - Example: "It rains", "It is sunny", "If it rains, it is not sunny" $\Rightarrow$ not consistent
  - In context of propositional logic, usually called satisfiability

Propositional Logic: Vocabulary

Let's define propositional logic formally!

Propositional Logic: Vocabulary

Let's define propositional logic formally!

Our vocabulary V consists of an infinite set of propositional variables:

$$V = \{a, b, c, \ldots\}$$

These are our basic building blocks:

- Represent sentences we reason about
- Using letters makes it easier to write complex formulas
- But we could also use strings: "It rains", "It is sunny".

Propositional Logic: Interpretations

Adding meaning:

Propositional Logic: Interpretations

Adding meaning:

- We don't know the value of propositional variables unless specified.

Propositional Logic: Interpretations

Adding meaning:

- We don't know the value of propositional variables unless specified.
- This incomplete knowledge is typical for most KR formalisms.
  - incomplete knowledge of the world
  - use reasoning to find out more

Propositional Logic: Interpretations

Adding meaning:

- We don't know the value of propositional variables unless specified.
- This incomplete knowledge is typical for most KR formalisms.
  - incomplete knowledge of the world
  - use reasoning to find out more

Propositional Logic: Interpretations

Adding meaning:

- We don't know the value of propositional variables unless specified.
- This incomplete knowledge is typical for most KR formalisms.
  - incomplete knowledge of the world
  - use reasoning to find out more
- These interpret the value of the propositional variables

Propositional Logic: Interpretations

Adding meaning:

- We don't know the value of propositional variables unless specified.
- This incomplete knowledge is typical for most KR formalisms.
  - incomplete knowledge of the world
  - use reasoning to find out more
- These interpret the value of the propositional variables
- and represent different possibilities.

Interpretations in Propositional Logic

An interpretation in propositional logic is a function $I : V \to \{\mathbf{true}, \mathbf{false}\}$

Examples:

- I_1("Es regnet") = **true**, I_1("Die Sonne scheint") = **false**

Interpretations in Propositional Logic

An interpretation in propositional logic is a function $I : V \rightarrow \{\textbf{true}, \textbf{false}\}$

Examples:

- $I_1(\text{“Es regnet”}) = \textbf{true}$, $I_1(\text{“Die Sonne scheint”}) = \textbf{false}$
- $I_2(\text{“Es regnet”}) = \textbf{false}$, $I_2(\text{“Die Sonne scheint”}) = \textbf{true}$

Interpretations in Propositional Logic

An interpretation in propositional logic is a function $I : V \to \{\textbf{true}, \textbf{false}\}$

Examples:

- I_1(“Es regnet”) $=$ **true**, I_1(“Die Sonne scheint”) $=$ **false**
- I_2(“Es regnet”) $=$ **false**, I_2(“Die Sonne scheint”) $=$ **true**
- I_3(“Es regnet”) $=$ **true**, I_3(“Die Sonne scheint”) $=$ **true**

Interpretations in Propositional Logic

An interpretation in propositional logic is a function $I : V \to \{\mathbf{true}, \mathbf{false}\}$

Examples:

- I_1(“Es regnet”) $=$ **true**, I_1(“Die Sonne scheint”) $=$ **false**
- I_2(“Es regnet”) $=$ **false**, I_2(“Die Sonne scheint”) $=$ **true**
- I_3(“Es regnet”) $=$ **true**, I_3(“Die Sonne scheint”) $=$ **true**
- I_4(“Es regnet”) $=$ **false**, I_4(“Die Sonne scheint”) $=$ **false**

Interpretations in Propositional Logic

An interpretation in propositional logic is a function $I : V \rightarrow \{\textbf{true}, \textbf{false}\}$

Examples:

- $I_1(\text{“Es regnet”}) = \textbf{true}$, $I_1(\text{“Die Sonne scheint”}) = \textbf{false}$
- $I_2(\text{“Es regnet”}) = \textbf{false}$, $I_2(\text{“Die Sonne scheint”}) = \textbf{true}$
- $I_3(\text{“Es regnet”}) = \textbf{true}$, $I_3(\text{“Die Sonne scheint”}) = \textbf{true}$
- $I_4(\text{“Es regnet”}) = \textbf{false}$, $I_4(\text{“Die Sonne scheint”}) = \textbf{false}$

We do not necessarily know which interpretation is the right one.

Interpretations in Propositional Logic

An interpretation in propositional logic is a function $I : V \to \{\textbf{true}, \textbf{false}\}$

Examples:

- I_1(“Es regnet”) = **true**, I_1(“Die Sonne scheint”) = **false**
- I_2(“Es regnet”) = **false**, I_2(“Die Sonne scheint”) = **true**
- I_3(“Es regnet”) = **true**, I_3(“Die Sonne scheint”) = **true**
- I_4(“Es regnet”) = **false**, I_4(“Die Sonne scheint”) = **false**

We do not necessarily know which interpretation is the right one.

Propositional formulas restrict the space of interpretations to those that are models (abstract representations) of possible alternatives of the described situation.

Propositional Logic: Formulas

Propositional formulas are defined as follows:

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ ("not" / negation)

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ ("not" / negation)
   - $F \vee G$ ("or" / disjunction)

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ (“not” / negation)
   - $F \vee G$ (“or” / disjunction)
   - $F \wedge G$ (“and” / conjunction)

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ ("not" / negation)
   - $F \vee G$ ("or" / disjunction)
   - $F \wedge G$ ("and" / conjunction)
   - $F \rightarrow G$ ("implies" / implication)

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ (“not” / negation)
   - $F \vee G$ (“or” / disjunction)
   - $F \wedge G$ (“and” / conjunction)
   - $F \rightarrow G$ (“implies” / implication)
   - $F \leftrightarrow G$ (biimplication)

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ ("not" / negation)
   - $F \vee G$ ("or" / disjunction)
   - $F \wedge G$ ("and" / conjunction)
   - $F \rightarrow G$ ("implies" / implication)
   - $F \leftrightarrow G$ (biimplication)
3. Nothing else is a formula.

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ ("not" / negation)
   - $F \vee G$ ("or" / disjunction)
   - $F \wedge G$ ("and" / conjunction)
   - $F \rightarrow G$ ("implies" / implication)
   - $F \leftrightarrow G$ (biimplication)
3. Nothing else is a formula.

Examples:

- *"Wenn es regnet, scheint die Sonne nicht."*

  "Es regnet" $\rightarrow \neg$"Die Sonne scheint"

Propositional Logic: Formulas

Propositional formulas are defined as follows:

1. Every propositional variable is a formula (stating that this sentence is true)
2. If F and G are formulas, then the following are also formulas:
   - $\neg F$ ("not" / negation)
   - $F \vee G$ ("or" / disjunction)
   - $F \wedge G$ ("and" / conjunction)
   - $F \rightarrow G$ ("implies" / implication)
   - $F \leftrightarrow G$ (biimplication)
3. Nothing else is a formula.

Examples:

- *"Wenn es regnet, scheint die Sonne nicht."*
  "Es regnet" $\rightarrow \neg$"Die Sonne scheint"
- $(c \wedge d) \leftrightarrow (a \vee \neg b)$

Exercise: Formalization in Propositional Logic

Assume we have the following propositional variables:

- a: "The post brings a parcel."
- b: "I am at home."
- c: "My neighbour is at home."
- d: "I take the parcel."
- e: "My neighbour takes the parcel."
- f: "The parcel goes back."

Formalize the following facts into propositional logic:

"I am not at home, and the post brings a parcel."	
"My neighbour is at home."	
"If I am at home and the post brings a parcel, I take the parcel."	
"If I am not at home, I don't take the parcel."	
"If the post brings a parcel, and neither me nor the neighbour take the parcel, it goes back."	
"If the post brings a parcel and the neighbour is at home, the neighour takes the parcel if and only if I am not at home."	

Propositional Logic: Semantics

- What do the complex formulas mean formally?

Propositional Logic: Semantics

- What do the complex formulas mean formally?
- Again, interpretations capture the meaning

Propositional Logic: Semantics

- What do the complex formulas mean formally?
- Again, interpretations capture the meaning
- Each interpretation $I : V \to \{\textbf{true}, \textbf{false}\}$ is extended to formulas using truth tables:

F	$\neg F$
false	**true**
true	**false**

F	G	$F \lor G$	$F \land G$	$F \to G$	$F \leftrightarrow G$
false	**false**	**false**	**false**	**true**	**true**
false	**true**	**true**	**false**	**true**	**false**
true	**false**	**true**	**false**	**false**	**false**
true	**true**	**true**	**true**	**true**	**true**

Propositional Logic: Semantics

- What do the complex formulas mean formally?
- Again, interpretations capture the meaning
- Each interpretation $I : V \to \{\mathbf{true}, \mathbf{false}\}$ is extended to formulas using truth tables:

F	$\neg F$
false	**true**
true	**false**

F	G	$F \lor G$	$F \land G$	$F \to G$	$F \leftrightarrow G$
false	**false**	**false**	**false**	**true**	**true**
false	**true**	**true**	**false**	**true**	**false**
true	**false**	**true**	**false**	**false**	**false**
true	**true**	**true**	**true**	**true**	**true**

Example: $I(a) = \mathbf{true}$, $I(b) = \mathbf{false}$ and $I(c) = \mathbf{true}$:

- $I(\neg a) = \mathbf{false}$

Propositional Logic: Semantics

- What do the complex formulas mean formally?
- Again, interpretations capture the meaning
- Each interpretation $I : V \to \{\textbf{true}, \textbf{false}\}$ is extended to formulas using truth tables:

F	$\neg F$
false	**true**
true	**false**

F	G	$F \lor G$	$F \land G$	$F \to G$	$F \leftrightarrow G$
false	**false**	**false**	**false**	**true**	**true**
false	**true**	**true**	**false**	**true**	**false**
true	**false**	**true**	**false**	**false**	**false**
true	**true**	**true**	**true**	**true**	**true**

Example: $I(a) = \textbf{true}$, $I(b) = \textbf{false}$ and $I(c) = \textbf{true}$:

- $I(\neg a) = \textbf{false}$
- $I(a \lor b) = \textbf{true}$

Propositional Logic: Semantics

- What do the complex formulas mean formally?
- Again, interpretations capture the meaning
- Each interpretation $I : V \to \{\textbf{true}, \textbf{false}\}$ is extended to formulas using truth tables:

F	$\neg F$
false	**true**
true	**false**

F	G	$F \vee G$	$F \wedge G$	$F \to G$	$F \leftrightarrow G$
false	**false**	**false**	**false**	**true**	**true**
false	**true**	**true**	**false**	**true**	**false**
true	**false**	**true**	**false**	**false**	**false**
true	**true**	**true**	**true**	**true**	**true**

Example: $I(a) = \textbf{true}$, $I(b) = \textbf{false}$ and $I(c) = \textbf{true}$:

- $I(\neg a) = \textbf{false}$
- $I(a \vee b) = \textbf{true}$
- $I(a \wedge b) = \textbf{false}$

Propositional Logic: Semantics

- What do the complex formulas mean formally?
- Again, interpretations capture the meaning
- Each interpretation $I : V \to \{\textbf{true}, \textbf{false}\}$ is extended to formulas using truth tables:

F	$\neg F$
false	**true**
true	**false**

F	G	$F \vee G$	$F \wedge G$	$F \to G$	$F \leftrightarrow G$
false	**false**	**false**	**false**	**true**	**true**
false	**true**	**true**	**false**	**true**	**false**
true	**false**	**true**	**false**	**false**	**false**
true	**true**	**true**	**true**	**true**	**true**

Example: $I(a) = \textbf{true}$, $I(b) = \textbf{false}$ and $I(c) = \textbf{true}$:

- $I(\neg a) = \textbf{false}$
- $I(a \vee b) = \textbf{true}$
- $I(a \wedge b) = \textbf{false}$
- $I((a \wedge b) \to c) = \textbf{true}$

Exercise: Semantics

Exercise:

- Try to find an interpretation I that satisfies some of the formulas you have written down in the previous exercise.
- Of course, you only need to specify the variables that are relevant
- Example: $I(a) =$ **true** and $I(b) =$ **false** satisfies the first formula

Exercise: Semantics

Exercise:

- Try to find an interpretation I that satisfies some of the formulas you have written down in the previous exercise.
- Of course, you only need to specify the variables that are relevant
- Example: $I(a) =$ **true** and $I(b) =$ **false** satisfies the first formula

Propositional Logic: Semantics

F	G	$F \vee G$	$F \wedge G$
false	**false**	**false**	**false**
false	**true**	**true**	**false**
true	**false**	**true**	**false**
true	**true**	**true**	**true**

We note that conjunction ($\wedge$) and disjunction ($\vee$) are commutative and associative—the order and how we put brackets does not matter.

Propositional Logic: Semantics

F	G	$F \vee G$	$F \wedge G$
false	**false**	**false**	**false**
false	**true**	**true**	**false**
true	**false**	**true**	**false**
true	**true**	**true**	**true**

We note that conjunction ($\wedge$) and disjunction ($\vee$) are commutative and associative—the order and how we put brackets does not matter.

We can therefore leave out brackets for nested conjunctions/disjunctions:

- $(F \vee G) \vee H = F \vee (G \vee H) = F \vee G \vee H$
- $(F \wedge G) \wedge H = F \wedge (G \wedge H) = F \wedge G \wedge H$

Propositional Logic: Reasoning Problems

We can now define different reasoning problems:

- A formula F is satisfiable if there is an interpretation I s.t. $I(F) = \textbf{true}$
  - I is then called a model of F

Propositional Logic: Reasoning Problems

We can now define different reasoning problems:

- A formula F is satisfiable if there is an interpretation I s.t. $I(F) = \textbf{true}$
  - I is then called a model of F
- A formula G is entailed by a formula F is every model of F is also a model of G
  - We write this as $F \models G$

Propositional Logic: Reasoning Problems

We can now define different reasoning problems:

- A formula F is satisfiable if there is an interpretation I s.t. $I(F) = \mathbf{true}$
  - I is then called a model of F
- A formula G is entailed by a formula F is every model of F is also a model of G
  - We write this as $F \models G$

Having a method for satisfiability is sufficient:

- $F \models G$ if and only if $F \land \neg G$ is not satisfiable

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?
  - Example 2: Given a program P, does P eventually stop?

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?
  - Example 2: Given a program P, does P eventually stop?
- If there is a *decision procedure*, we can use it to decide decision problems.

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?
  - Example 2: Given a program P, does P eventually stop?
- If there is a *decision procedure*, we can use it to decide decision problems.

> Given a decision problem P, a decision procedure for P is an algorithm with the following properties:

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?
  - Example 2: Given a program P, does P eventually stop?
- If there is a *decision procedure*, we can use it to decide decision problems.

> Given a decision problem P, a decision procedure for P is an algorithm with the following properties:
>
> Soundness For each instance of P for which the method returns **yes**, the answer is also yes.

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?
  - Example 2: Given a program P, does P eventually stop?
- If there is a *decision procedure*, we can use it to decide decision problems.

> Given a decision problem P, a decision procedure for P is an algorithm with the following properties:
>
> **Soundness** For each instance of P for which the method returns **yes**, the answer is also yes.
>
> **Completeness** For each instance of P for which the method returns **no**, the answer is also no.

Decision Problems and Decision Procedures

A little bit of theory:

- Satisfiability and entailment are decision problems
  - Decision problems contain questions that have a yes or a no answer.
  - Example 1: Given a number X, is X a prime number?
  - Example 2: Given a program P, does P eventually stop?
- If there is a *decision procedure*, we can use it to decide decision problems.

Given a decision problem P, a decision procedure for P is an algorithm with the following properties:

Soundness For each instance of P for which the method returns **yes**, the answer is also yes.

Completeness For each instance of P for which the method returns **no**, the answer is also no.

Termination For each instance of P, the algorithm stops after a finite number of steps.

Decidability

- Many problems have corresponding decision problems:
  - Querying answers to some problem $\Rightarrow$ Deciding whether a given answer is true

Decidability

- Many problems have corresponding decision problems:
  - Querying answers to some problem $\Rightarrow$ Deciding whether a given answer is true
- A decision problem is decidable if there is a decision procedure for it.

Decidability

- Many problems have corresponding decision problems:
  - Querying answers to some problem $\Rightarrow$ Deciding whether a given answer is true
- A decision problem is decidable if there is a decision procedure for it.
- Many decision problems are decidable:
  - Example: *Is X a prime number?*

Decidability

- Many problems have corresponding decision problems:
  - Querying answers to some problem $\Rightarrow$ Deciding whether a given answer is true
- A decision problem is decidable if there is a decision procedure for it.
- Many decision problems are decidable:
  - Example: *Is X a prime number?*
- But there are also problems that are undecidable:
  - Example *Does program P eventually stop?*
  - This corresponds to the famous halting problem
  - It is impossible to devise an algorithm that always answers this correctly.

Reasoning in Propositional Logic in Practice

How do we reason in propositional logic?

- Deciding satisfiability is usually sufficient

Reasoning in Propositional Logic in Practice

How do we reason in propositional logic?

- Deciding satisfiability is usually sufficient
- Checking whether an interpretation is a model is easy

Reasoning in Propositional Logic in Practice

How do we reason in propositional logic?

- Deciding satisfiability is usually sufficient
- Checking whether an interpretation is a model is easy
- But finding a model can be difficult:
  - with n variables, 2^n interpretations to consider!

Reasoning in Propositional Logic in Practice

How do we reason in propositional logic?

- Deciding satisfiability is usually sufficient
- Checking whether an interpretation is a model is easy
- But finding a model can be difficult:
  - with n variables, 2^n interpretations to consider!
- Whether there is a tractable method is one of the big open problems in computer science
  - The famous $P = NP$ problem

Reasoning in Propositional Logic in Practice

How do we reason in propositional logic?

- Deciding satisfiability is usually sufficient
- Checking whether an interpretation is a model is easy
- But finding a model can be difficult:
  - with n variables, 2^n interpretations to consider!
- Whether there is a tractable method is one of the big open problems in computer science
  - The famous $P = NP$ problem

In practice:

- SAT-solvers are tools to determine satisfiability of propositional formulas
- Modern SAT-solvers are highly optimized, and can often deal with very large formulas in short time
- Examples of SAT solvers: MiniSAT, PicoSAT, CaDiCaL, ...

Applications of Propositional Logic

- Symbolic AI applications with limited/finite set of variables

Applications of Propositional Logic

- Symbolic AI applications with limited/finite set of variables
- Software and hardware verification

Applications of Propositional Logic

- Symbolic AI applications with limited/finite set of variables
- Software and hardware verification
- NP-complete problems: find "easy" to verify solution of fixed size
  - "Easy": we can describe it using propositional logic
  - Many puzzles and games like Sudoku have this property
  - Describing the problem with propositional logic and using a SAT solver can be more efficient than implementing a search procedure from scratch

Applications of Propositional Logic

- Symbolic AI applications with limited/finite set of variables
- Software and hardware verification
- NP-complete problems: find "easy" to verify solution of fixed size
  - "Easy": we can describe it using propositional logic
  - Many puzzles and games like Sudoku have this property
  - Describing the problem with propositional logic and using a SAT solver can be more efficient than implementing a search procedure from scratch
- Explaining classifiers obtained through machine learning

Applications of Propositional Logic

- Symbolic AI applications with limited/finite set of variables
- Software and hardware verification
- NP-complete problems: find "easy" to verify solution of fixed size
  - "Easy": we can describe it using propositional logic
  - Many puzzles and games like Sudoku have this property
  - Describing the problem with propositional logic and using a SAT solver can be more efficient than implementing a search procedure from scratch
- Explaining classifiers obtained through machine learning

What we cannot do well is reason using the inner structure of sentences:

- "Socrates is a human."
- "All humans are mortal."
- Entails: "Socrates is mortal."

First Order Logic

First Order Logic

First Order Logic (FOL) introduces structure:

- Reason about objects, functions and relations
- Atomic formulas now have structure
- Additional logical operators

First Order Logic

First Order Logic (FOL) introduces structure:

- Reason about objects, functions and relations
- Atomic formulas now have structure
- Additional logical operators

Examples:

- Petra is the neighbour of Tom: $\mathit{neighbours}(\mathit{Petra}, \mathit{Tom})$

First Order Logic

First Order Logic (FOL) introduces structure:

- Reason about objects, functions and relations
- Atomic formulas now have structure
- Additional logical operators

Examples:

- Petra is the neighbour of Tom: $neighbours(Petra, Tom)$
- If x is a neighbour of y, then y is a neighbour of x:
  $\forall x, y : (neighbours(x, y) \rightarrow neighbours(y, x))$

First Order Logic

First Order Logic (FOL) introduces structure:

- Reason about objects, functions and relations
- Atomic formulas now have structure
- Additional logical operators

Examples:

- Petra is the neighbour of Tom: $\mathit{neighbours}(\mathit{Petra}, \mathit{Tom})$
- If x is a neighbour of y, then y is a neighbour of x:
  $\forall x, y : (\mathit{neighbours}(x, y) \rightarrow \mathit{neighbours}(y, x))$
- Every parent has a child: $\forall x : (\mathit{Parent}(x) \rightarrow \exists y : \mathit{hasChild}(x, y))$

First Order Logic: Vocabulary

The vocabulary is now more involved:

- constants a, b, c, . . . denote specified objects
- variables x, y, z, . . . denote unspecified objects

First Order Logic: Vocabulary

The vocabulary is now more involved:

- constants a, b, c, ... denote specified objects
- variables x, y, z, ... denote unspecified objects
- function names f, g, h, ... denote functions
  - Every function name f has an arity $ar(f) \in \mathbb{N}^+$
  - The arity determines how many arguments the function takes
  - *Examples:* successor, sum $\quad ar(\text{successor}) = 1, \quad ar(\text{sum}) = 2$

First Order Logic: Vocabulary

The vocabulary is now more involved:

- constants a, b, c, ... denote specified objects
- variables x, y, z, ... denote unspecified objects
- function names f, g, h, ... denote functions
  - Every function name f has an arity $ar(f) \in \mathbb{N}^+$
  - The arity determines how many arguments the function takes
  - *Examples:* successor, sum $\quad ar(\text{successor}) = 1, \quad ar(\text{sum}) = 2$
- predicate names P, Q, R, ... denote relations
  - Again, every predicate name P has an arity $ar(P)$
  - which determines how many arguments P takes
  - *Examples:* *neighbours*, *Parent* $\quad ar(\mathit{neighbours}) = 2, \quad ar(\mathit{Parent}) = 1$

First Order Logic: Vocabulary

The vocabulary is now more involved:

- constants a, b, c, ... denote specified objects
- variables x, y, z, ... denote unspecified objects
- function names f, g, h, ... denote functions
  - Every function name f has an arity $ar(f) \in \mathbb{N}^+$
  - The arity determines how many arguments the function takes
  - *Examples:* successor, sum $\quad ar(\mathrm{successor}) = 1, \quad ar(\mathrm{sum}) = 2$
- predicate names P, Q, R, ... denote relations
  - Again, every predicate name P has an arity $ar(P)$
  - which determines how many arguments P takes
  - *Examples:* *neighbours*, *Parent* $\quad ar(neighbours) = 2, \quad ar(Parent) = 1$

Again, interpretations determine what these mean

First Order Logic: Syntax

The vocabulary is used as follows to build formulas:

- First, we define terms:
  - terms refer to objects

First Order Logic: Syntax

The vocabulary is used as follows to build formulas:

- First, we define terms:
    - terms refer to objects
    - every constant and every variable is a term

First Order Logic: Syntax

The vocabulary is used as follows to build formulas:

- First, we define terms:
  - terms refer to objects
  - every constant and every variable is a term
  - if f is a function of arity n, and t_1, ..., t_n are terms, then $f(t_1, \ldots, t_n)$ is also a term

First Order Logic: Syntax

The vocabulary is used as follows to build formulas:

- First, we define terms:
  - terms refer to objects
  - every constant and every variable is a term
  - if f is a function of arity n, and t_1, ..., t_n are terms, then $f(t_1, \ldots, t_n)$ is also a term
  - Examples: x, tom, $successor(x)$, $sum(x, y)$

First Order Logic: Syntax

The vocabulary is used as follows to build formulas:

- First, we define terms:
  - terms refer to objects
  - every constant and every variable is a term
  - if f is a function of arity n, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is also a term
  - Examples: x, tom, $successor(x)$, $sum(x, y)$
- Then, we define atoms:
  - if P is a predicate name of arity n, and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is an atom
  - Example: $EvenNumber(x)$, $Neighbours(sister(anna), peter)$
  - atoms are like the propositional variables in propositional logic

First Order Logic: Syntax

First order formulas are now defined as follows:

- every atom is a formula

First Order Logic: Syntax

First order formulas are now defined as follows:

- every atom is a formula
- if F and G are formulas, and x is a variable, then the following are also formulas:

First Order Logic: Syntax

First order formulas are now defined as follows:

- every atom is a formula
- if F and G are formulas, and x is a variable, then the following are also formulas:
  - $\neg F$, $F \wedge G$, $F \vee G$, $F \rightarrow G$, $F \leftrightarrow G$

First Order Logic: Syntax

First order formulas are now defined as follows:

- every atom is a formula
- if F and G are formulas, and x is a variable, then the following are also formulas:
  - $\neg F$, $F \wedge G$, $F \vee G$, $F \rightarrow G$, $F \leftrightarrow G$
  - $\exists x : F$ ("there exists", existential quantification)

First Order Logic: Syntax

First order formulas are now defined as follows:

- every atom is a formula
- if F and G are formulas, and x is a variable, then the following are also formulas:
  - $\neg F$, $F \wedge G$, $F \vee G$, $F \rightarrow G$, $F \leftrightarrow G$
  - $\exists x : F$ ("there exists", existential quantification)
  - $\forall x : F$ ("for all", universal quantification)

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : \mathit{hasStudentJob}(\mathit{peter}, x)$

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : \mathit{hasStudentJob}(\mathit{peter}, x)$
  - $\forall x : F$: for every x, F holds
    - Example: $\forall x : (\mathit{Student}(x) \rightarrow \mathit{Person}(x))$

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : \mathit{hasStudentJob}(\mathit{peter}, x)$
  - $\forall x : F$: for every x, F holds
    - Example: $\forall x : (\mathit{Student}(x) \rightarrow \mathit{Person}(x))$
- this of course only makes sense if F contains x

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : \mathit{hasStudentJob}(\mathit{peter}, x)$
  - $\forall x : F$: for every x, F holds
    - Example: $\forall x : (\mathit{Student}(x) \rightarrow \mathit{Person}(x))$
- this of course only makes sense if F contains x
- we then say that x is bound in $\exists x : F$ / $\forall x : F$

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : \mathit{hasStudentJob}(\mathit{peter}, x)$
  - $\forall x : F$: for every x, F holds
    - Example: $\forall x : (\mathit{Student}(x) \rightarrow \mathit{Person}(x))$
- this of course only makes sense if F contains x
- we then say that x is bound in $\exists x : F$ / $\forall x : F$
- a variable that is not bound is free

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : \mathit{hasStudentJob}(\mathit{peter}, x)$
  - $\forall x : F$: for every x, F holds
    - Example: $\forall x : (\mathit{Student}(x) \rightarrow \mathit{Person}(x))$
- this of course only makes sense if F contains x
- we then say that x is bound in $\exists x : F$ / $\forall x : F$
- a variable that is not bound is free
- a formula without free variables is called sentence

First Order Logic: Quantifiers and Sentences

- The intuitive idea of the quantifiers is as follows:
  - $\exists x : F$: for some x, F holds
    - Example: $\exists x : hasStudentJob(peter, x)$
  - $\forall x : F$: for every x, F holds
    - Example: $\forall x : (Student(x) \rightarrow Person(x))$
- this of course only makes sense if F contains x
- we then say that x is bound in $\exists x : F$ / $\forall x : F$
- a variable that is not bound is free
- a formula without free variables is called sentence

Some examples:

- $\forall x : (EvenNumber(x) \leftrightarrow OddNumber(sum(x, 1)))$
- $\forall x : (Parent(x) \rightarrow \exists y : HasChild(x, y))$
- $\forall x : \forall y : (\exists z : DeliversParcelTo(z, y, x) \wedge AtHome(x)) \rightarrow ReceivedParcel(x, y))$

First-Order Logic: Example

We can use first-order logic to model the parcel example a bit better:

$$\neg AtHome(patrick)$$
$$DeliversParcelTo(mailMan, patrick, parcel)$$
$$Neighbour(patrick, lucia)$$
$$\forall x : \forall y : \forall z : ((BringsParcelTo(x, y, z) \wedge AtHome(y)) \rightarrow Receives(x, z))$$
$$\vdots$$

First-Order Logic: Semantics

While the intuition is maybe easy to understand, the semantics is a bit more involved now.
This time, interpretations are structures $\langle \Delta, \cdot^I \rangle$:

First-Order Logic: Semantics

While the intuition is maybe easy to understand, the semantics is a bit more involved now.
This time, interpretations are structures $\langle \Delta, \cdot^I \rangle$:

- Δ is a possibly infinite set called the domain
  - The elements occurring in the interpretation
  - these are called domain elements

First-Order Logic: Semantics

While the intuition is maybe easy to understand, the semantics is a bit more involved now.
This time, interpretations are structures $\langle \Delta, \cdot^I \rangle$:

- Δ is a possibly infinite set called the domain
  - The elements occurring in the interpretation
  - these are called domain elements
- $\cdot^I$ is a function that interprets constants, functions and variables

First-Order Logic: Semantics

While the intuition is maybe easy to understand, the semantics is a bit more involved now.
This time, interpretations are structures $\langle \Delta, \cdot^I \rangle$:

- Δ is a possibly infinite set called the domain
  - The elements occurring in the interpretation
  - these are called domain elements
- $\cdot^I$ is a function that interprets constants, functions and variables
- We then write their interpretations as c^I, f^I, P^I, etc.

Interpreting Terms

The interpretation function $\cdot^I$ works as follows:

- Every constant c is assigned some element $c^I \in \Delta$ in the domain

Interpreting Terms

The interpretation function $\cdot^I$ works as follows:

- Every constant c is assigned some element $c^I \in \Delta$ in the domain
- Every function name f with arity n is assigned a function $f^I : \Delta^n \to \Delta$
  - It maps tuples of elements to elements

Interpreting Terms

The interpretation function $\cdot^I$ works as follows:

- Every constant c is assigned some element $c^I \in \Delta$ in the domain
- Every function name f with arity n is assigned a function $f^I : \Delta^n \to \Delta$
  - It maps tuples of elements to elements
- Every predicate name P with arity n is assigned a relation $P \in \Delta^n$
  - This means, P is a set of tuples
  - For example, if P has arity 1, it is a subset of Δ
  - If P has arity 3, it contains tuples $\langle a, b, c \rangle$, where a, b and c are from Δ

Example: First-Order Interpretation

$$\Delta^I = \{a, b, c, d\}$$

$$patrick^I = a \qquad mailMan^I = b \qquad parcel^I = c$$

$$Neighbour^I = \{\langle a, d\rangle, \langle d, a\rangle\} \qquad AtHome^I = \{a, d\}$$

$$DeliversParcelTo^I = \{\langle b, c, a\rangle\} \qquad ReceivesParcel^I = \{\langle a, d\rangle\}$$

Example: First-Order Interpretation

$$\Delta^I = \{a, b, c, d\}$$

$$patrick^I = a \qquad mailMan^I = b \qquad parcel^I = c$$

$$Neighbour^I = \{\langle a, d\rangle, \langle d, a\rangle\} \qquad AtHome^I = \{a, d\}$$

$$DeliversParcelTo^I = \{\langle b, c, a\rangle\} \qquad ReceivesParcel^I = \{\langle a, d\rangle\}$$

Which formulas hold in this interpretation?

Interpreting Variables

Since formulas contain variables, we need to also take care of them.

Interpreting Variables

Since formulas contain variables, we need to also take care of them.

A variable assignment σ_I for an interpretation I is a function that assigns to every variable x an element $\sigma_I(x)$ of the domain Δ.

Interpreting Variables

Since formulas contain variables, we need to also take care of them.

A variable assignment σ_I for an interpretation I is a function that assigns to every variable x an element $\sigma_I(x)$ of the domain Δ.

We can modify variable assignments:

- $\sigma_I[x \mapsto d]$ is a new variable assignment which is like σ_I, but assigns x to d

Interpreting Variables

Since formulas contain variables, we need to also take care of them.

A variable assignment σ_I for an interpretation I is a function that assigns to every variable x an element $\sigma_I(x)$ of the domain Δ.

We can modify variable assignments:

- $\sigma_I[x \mapsto d]$ is a new variable assignment which is like σ_I, but assigns x to d

We extend σ_I inductively to work on arbitrary terms:

- For every constant c, $\sigma_I(c) = c^I$.

Interpreting Variables

Since formulas contain variables, we need to also take care of them.

A variable assignment σ_I for an interpretation I is a function that assigns to every variable x an element $\sigma_I(x)$ of the domain Δ.

We can modify variable assignments:

- $\sigma_I[x \mapsto d]$ is a new variable assignment which is like σ_I, but assigns x to d

We extend σ_I inductively to work on arbitrary terms:

- For every constant c, $\sigma_I(c) = c^I$.
- For every function f of arity n and terms $t_1, \ldots, t_n$,
  $\sigma_I(f(t_1, \ldots, t_n)) = f^I(\sigma_I(t_1), \ldots, \sigma_I(t_n))$

Interpreting Variables

Since formulas contain variables, we need to also take care of them.

A variable assignment σ_I for an interpretation I is a function that assigns to every variable x an element $\sigma_I(x)$ of the domain Δ.

We can modify variable assignments:

- $\sigma_I[x \mapsto d]$ is a new variable assignment which is like σ_I, but assigns x to d

We extend σ_I inductively to work on arbitrary terms:

- For every constant c, $\sigma_I(c) = c^I$.
- For every function f of arity n and terms $t_1, \ldots, t_n$, $\sigma_I(f(t_1, \ldots, t_n)) = f^I(\sigma_I(t_1), \ldots, \sigma_I(t_n))$

Using variable assigments, we can now define what it means for a first-order formula to be satisfied in an interpretation.

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\quad \sigma_I \models P(t_1, \ldots, t_n) \quad$ if $\langle t_1, \ldots, t_n \rangle \in P^I$

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\quad \sigma_I \models P(t_1, \ldots, t_n) \quad$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\quad \sigma_I \models \neg F \quad$ if $\sigma_I \not\models F$

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\sigma_I \models P(t_1, \ldots, t_n)$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\sigma_I \models \neg F$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G$ if $\sigma_I \models F$ or $\sigma_I \models G$

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\sigma_I \models P(t_1, \ldots, t_n)$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\sigma_I \models \neg F$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G$ if $\sigma_I \models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \wedge G$ if $\sigma_I \models F$ and $\sigma_I \models G$

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\sigma_I \models P(t_1, \ldots, t_n)$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\sigma_I \models \neg F$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G$ if $\sigma_I \models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \wedge G$ if $\sigma_I \models F$ and $\sigma_I \models G$
  - $\sigma_I \models F \rightarrow G$ if $\sigma_I \not\models F$ or $\sigma_I \models G$

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \dots, t_n)$: $\quad \sigma_I \models P(t_1, \dots, t_n) \quad$ if $\langle t_1, \dots, t_n \rangle \in P^I$
- For any formula F: $\quad \sigma_I \models \neg F \quad$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G \quad$ if $\sigma_I \models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \wedge G \quad$ if $\sigma_I \models F$ and $\sigma_I \models G$
  - $\sigma_I \models F \rightarrow G \quad$ if $\sigma_I \not\models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \leftrightarrow G \quad$ if $\sigma_I \models F \rightarrow G$ and $\sigma_I \models G \rightarrow F$

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\sigma_I \models P(t_1, \ldots, t_n)$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\sigma_I \models \neg F$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G$ if $\sigma_I \models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \wedge G$ if $\sigma_I \models F$ and $\sigma_I \models G$
  - $\sigma_I \models F \rightarrow G$ if $\sigma_I \not\models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \leftrightarrow G$ if $\sigma_I \models F \rightarrow G$ and $\sigma_I \models G \rightarrow F$
- For a variable x and a formula F:

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\sigma_I \models P(t_1, \ldots, t_n)$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\sigma_I \models \neg F$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G$ if $\sigma_I \models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \wedge G$ if $\sigma_I \models F$ and $\sigma_I \models G$
  - $\sigma_I \models F \rightarrow G$ if $\sigma_I \not\models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \leftrightarrow G$ if $\sigma_I \models F \rightarrow G$ and $\sigma_I \models G \rightarrow F$
- For a variable x and a formula F:
  - $\sigma_I \models \exists x : F$ if there is some domain element $d \in \Delta$ s.t. $\sigma_I[x \mapsto d] \models F$.

Satisfaction of First-Order Formulas

- Satisfaction of formulas F is first defined relative to a variable assigment σ_I for the interpretation in question.
- We write this as $\sigma_I \models F$ (F is satisfied under σ_I).

We define satisfaction under a variable assignment *inductively*:

- For any atom $P(t_1, \ldots, t_n)$: $\quad \sigma_I \models P(t_1, \ldots, t_n) \quad$ if $\langle t_1, \ldots, t_n \rangle \in P^I$
- For any formula F: $\quad \sigma_I \models \neg F \quad$ if $\sigma_I \not\models F$
- For any formulas F and G:
  - $\sigma_I \models F \vee G \quad$ if $\sigma_I \models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \wedge G \quad$ if $\sigma_I \models F$ and $\sigma_I \models G$
  - $\sigma_I \models F \rightarrow G \quad$ if $\sigma_I \not\models F$ or $\sigma_I \models G$
  - $\sigma_I \models F \leftrightarrow G \quad$ if $\sigma_I \models F \rightarrow G$ and $\sigma_I \models G \rightarrow F$
- For a variable x and a formula F:
  - $\sigma_I \models \exists x : F \quad$ if there is some domain element $d \in \Delta$ s.t. $\sigma_I[x \mapsto d] \models F$.
  - $\sigma_I \models \forall x : F \quad$ if for every domain element $d \in \Delta$, $\sigma_I[x \mapsto d] \models F$.

Satisfaction of First-Order Formulas

Now observe:

- If all variables are bound, the variable assignment is not relevant anymore
- We can then write $I \models F$ instead of $\sigma_I \models F$
- This is how *satisfaction of sentences* is defined
  - Recall: every variable in a sentence is bound

First-Order Logic: Reasoning

- Satisfiability and entailment are defined the same way as for propositional logic
  - Satisfiable: has some model
  - Entailed: $F \models G$ if every model of F is a model of G

First-Order Logic: Reasoning

- Satisfiability and entailment are defined the same way as for propositional logic
  - Satisfiable: has some model
  - Entailed: $F \models G$ if every model of F is a model of G
- Theorem provers are systems that can be used to determine wether a formula is satisfiable
  - Examples: Vampire, E, SPASS, Otter, etc.

First-Order Logic: Reasoning

- Satisfiability and entailment are defined the same way as for propositional logic
  - Satisfiable: has some model
  - Entailed: $F \models G$ if every model of F is a model of G
- Theorem provers are systems that can be used to determine wether a formula is satisfiable
  - Examples: Vampire, E, SPASS, Otter, etc.
- However, reasoning in first-order logic is much much harder than in propositional logic

First-Order Logic: Reasoning

- Satisfiability and entailment are defined the same way as for propositional logic
  - Satisfiable: has some model
  - Entailed: $F \models G$ if every model of F is a model of G
- Theorem provers are systems that can be used to determine wether a formula is satisfiable
  - Examples: Vampire, E, SPASS, Otter, etc.
- However, reasoning in first-order logic is much much harder than in propositional logic
- In particular, it is only semi-decidable

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.
- The bad news: there are satisfiable formulas for which no algorithm can ever show that they are satisfiable.

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.
- The bad news: there are satisfiable formulas for which no algorithm can ever show that they are satisfiable.
  - Recall: the domain of an interpretation can be infinite.

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.
- The bad news: there are satisfiable formulas for which no algorithm can ever show that they are satisfiable.
  - Recall: the domain of an interpretation can be infinite.
  - Don't ask for an example of such a formula.

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.
- The bad news: there are satisfiable formulas for which no algorithm can ever show that they are satisfiable.
  - Recall: the domain of an interpretation can be infinite.
  - Don't ask for an example of such a formula.
- This means: even if we want to know whether the formula is unsatisfiable, we might have to wait forever.

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.
- The bad news: there are satisfiable formulas for which no algorithm can ever show that they are satisfiable.
  - Recall: the domain of an interpretation can be infinite.
  - Don't ask for an example of such a formula.
- This means: even if we want to know whether the formula is unsatisfiable, we might have to wait forever.
- Future technological advancements will not help us here (this is a result from theoretical computer science).

Semi-Decidable?

First order logic is semi-decidable:

- The good news: for every unsatisfiable formula, an algorithm can in theory show in finite time that it is unsatisfiable.
- The bad news: there are satisfiable formulas for which no algorithm can ever show that they are satisfiable.
  - Recall: the domain of an interpretation can be infinite.
  - Don't ask for an example of such a formula.
- This means: even if we want to know whether the formula is unsatisfiable, we might have to wait forever.
- Future technological advancements will not help us here (this is a result from theoretical computer science).

This limits the usefulness of FOL as KR drastically!

And now?

This is very bad news! What now?

And now?

This is very bad news! What now?

There are restrictions that make FOL fully decidable.

And now?

This is very bad news! What now?

There are restrictions that make FOL fully decidable.

- Restrict the number of variables to 2

And now?

This is very bad news! What now?

There are restrictions that make FOL fully decidable.

- Restrict the number of variables to 2
- Restrict the use of quantifiers

And now?

This is very bad news! What now?

There are restrictions that make FOL fully decidable.

- Restrict the number of variables to 2
- Restrict the use of quantifiers
- Or more involved restrictions on formulas

From First-Order to Propositional Logic

- In some cases, we can use SAT solvers also in first-order contexts
- The basic idea is grounding

From First-Order to Propositional Logic

- In some cases, we can use SAT solvers also in first-order contexts
- The basic idea is grounding
- A formula is ground if it contains no variables
  - $\mathit{hasLocation}(\mathit{robot}, \mathit{livingRoom}) \vee \mathit{hasLocation}(\mathit{robot}, \mathit{kitchen})$

From First-Order to Propositional Logic

- In some cases, we can use SAT solvers also in first-order contexts
- The basic idea is grounding
- A formula is ground if it contains no variables
  - $\mathit{hasLocation}(\mathit{robot}, \mathit{livingRoom}) \lor \mathit{hasLocation}(\mathit{robot}, \mathit{kitchen})$
- Ground formulas are like propositional formulas:
  - every ground atom can be treated as a propositional variable

From First-Order to Propositional Logic

- In some cases, we can use SAT solvers also in first-order contexts
- The basic idea is grounding
- A formula is ground if it contains no variables
  - $\mathit{hasLocation}(\mathit{robot}, \mathit{livingRoom}) \vee \mathit{hasLocation}(\mathit{robot}, \mathit{kitchen})$
- Ground formulas are like propositional formulas:
  - every ground atom can be treated as a propositional variable
- We can thus use SAT-solvers to reason with them.

From First-Order to Proposition Logic: Grounding

Under certain circumstances, we can focus on ground formulas

- For instance, if we have the following restrictions:
  - No functions
  - All quantifiers are universal quantifiers
  - No quantifiers under a negation symbol

From First-Order to Proposition Logic: Grounding

Under certain circumstances, we can focus on ground formulas

- For instance, if we have the following restrictions:
  - No functions
  - All quantifiers are universal quantifiers
  - No quantifiers under a negation symbol
- Examples:
  - $\forall x : (\mathit{Student}(x) \rightarrow \mathit{Person}(x))$
  - $\forall x : \forall y : (\mathit{Neighbour}(x, y) \leftrightarrow \mathit{Neighbour}(y, x))$
  - $\forall x : \forall y : ((\mathit{Person}(x) \wedge \mathit{hasChild}(x, y)) \rightarrow \mathit{Parent}(x)$

From First-Order to Proposition Logic: Grounding

Under certain circumstances, we can focus on ground formulas

- For instance, if we have the following restrictions:
  - No functions
  - All quantifiers are universal quantifiers
  - No quantifiers under a negation symbol
- Examples:
  - $\forall x : (Student(x) \rightarrow Person(x))$
  - $\forall x : \forall y : (Neighbour(x, y) \leftrightarrow Neighbour(y, x))$
  - $\forall x : \forall y : ((Person(x) \land hasChild(x, y)) \rightarrow Parent(x)$
- Idea: use all groundings of quantified variables:
  - $\forall x : (Student(x) \rightarrow Person(x))$
    $\Longrightarrow Student(tom) \rightarrow Person(tom) \ \land \ Student(peter) \rightarrow Person(peter) \ \land \ \ldots$
  - We use only the constants that occur in our formulas.
  - If we have no constant, we use an arbitrary one (e.g. a)

From First-Order to Proposition Logic: Grounding

- This trick fails once we have functions or existential quantifiers
  - refer to new objects
  - there may be many: $\forall x : A(x) \rightarrow A(f(x)) \quad \Rightarrow \quad A(a), \quad A(f(a)), \quad A(f(f(a))), \quad \ldots$

From First-Order to Proposition Logic: Grounding

- This trick fails once we have functions or existential quantifiers
  - refer to new objects
  - there may be many: $\forall x : A(x) \rightarrow A(f(x)) \quad \Rightarrow \quad A(a), \quad A(f(a)), \quad A(f(f(a))), \quad \ldots$
- negating universal quantifiers has the same effect:
  $\neg\forall x : A(x) \equiv \exists x . \neg A(x)$

From First-Order to Proposition Logic: Grounding

- This trick fails once we have functions or existential quantifiers
  - refer to new objects
  - there may be many: $\forall x : A(x) \to A(f(x)) \quad \Rightarrow \quad A(a), \quad A(f(a)), \quad A(f(f(a))), \quad \ldots$
- negating universal quantifiers has the same effect:
  $\neg\forall x : A(x) \equiv \exists x.\neg A(x)$

Conclusion

Propositional Logic

- Limited expressivity
- Reasoning may take exponential time
- Efficient SAT-solvers

First-Order Logic

- High expressivity
- Semantics a bit involved
- Only semi-decidable
- But there are syntactical restrictions that makes it decidable

Friday

Next: A family of fragments of first-order logic that

- can deal with existential and universal quantification
- is decidable
- has a more readable syntax (optimized for its use case)
- has easier semantics
- allows for efficient reasoning in practical applications

Friday

Next: A family of fragments of first-order logic that

- can deal with existential and universal quantification
- is decidable
- has a more readable syntax (optimized for its use case)
- has easier semantics
- allows for efficient reasoning in practical applications

Meet the Description Logics!