Before we begin

About the midterm exam

- 2 hours
- you're allowed to bring one A4 sheet of handwritten notes
- French version available (but you should answer in English)
- Topics covered: all of propositional logic
- Typical exercices, one of them taken straight from the handout

Before we begin

About the midterm exam

- 2 hours
- you're allowed to bring one A4 sheet of handwritten notes
- French version available (but you should answer in English)
- Topics covered: all of propositional logic
- Typical exercices, one of them taken straight from the handout

Before the midterms

- Don't forget your project pre-report !

Before we begin

About the midterm exam

- 2 hours
- you're allowed to bring one A4 sheet of handwritten notes
- French version available (but you should answer in English)
- Topics covered: all of propositional logic
- Typical exercices, one of them taken straight from the handout

Before the midterms

- Don't forget your project pre-report !

Schedule reminder, archives on

`https://wackb.gricad-pages.univ-grenoble-alpes.fr/inf402/`

First-order logic
Part one:
Language and Semantics of Formulae

Frédéric Prost

Université Grenoble Alpes

March 2023

Overview of the course

- Propositional logic: $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
- Interpretation: boolean functions
- Deductive systems: resolution, natural deduction
- Algorithms: Complete Strategy, DPLL, DN tactics

Overview of the course

- Propositional logic: $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
- Interpretation: boolean functions
- Deductive systems: resolution, natural deduction
- Algorithms: Complete Strategy, DPLL, DN tactics

- First-order logic: $\forall, \exists$
- Interpretation
- "First-order resolution"
- First-order natural deduction

Overview

Structure of first-order logic

A non-empty domain (more than two elements)

Structure of first-order logic

A non-empty domain (more than two elements)

Three categories:

- Terms representing the elements of the domain
- Relations
- Formulae describing the interactions between relations

Structure of first-order logic

A non-empty domain (more than two elements)

Three categories:

- Terms representing the elements of the domain
- Relations
- Formulae describing the interactions between relations

Two new symbols (quantifiers) in the formulae :

$\forall$ (universal quantification) and $\exists$ (existential quantification)

Structure of first-order logic

A non-empty domain (more than two elements)

Three categories:

- Terms representing the elements of the domain
- Relations
- Formulae describing the interactions between relations

Two new symbols (quantifiers) in the formulae :

$\forall$ (universal quantification) and $\exists$ (existential quantification)

Examples:

- domain = members of a family
- the term $father(x)$ refers to a domain element (the father of x),
- the relation *brother* which applies to two elements,
- the formula $\forall x \exists y\ brother(y, x)$ means “everyone has a brother”.

Syllogism

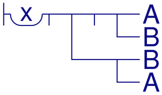
Every man is mortal.
Socrates is a man.
Hence Socrates is mortal.

Syllogism

Every man is mortal.
Socrates is a man.
Hence Socrates is mortal.

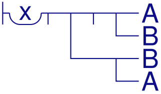
$$\forall x(man(x) \Rightarrow mortal(x))$$
$$man(Socrates)$$
$$mortal(Socrates)$$

Gottlob Frege's *Begriffsschrift* (ideography), 1879

- Like Leibniz, attempt at a formal "universal" language 

- First-order logical *system*
  (which contains rules such as Modus Ponens already known by Stoicists, but also new rules for the quantifiers)

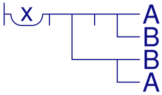
Gottlob Frege's *Begriffsschrift* (ideography), 1879

- Like Leibniz, attempt at a formal "universal" language 

- First-order logical *system*
  (which contains rules such as Modus Ponens already known by Stoicists, but also new rules for the quantifiers)
- Contains only *reasoning* rules but allows to express every *mathematical notion* (using sets)

Gottlob Frege's *Begriffsschrift* (ideography), 1879

- Like Leibniz, attempt at a formal "universal" language



- First-order logical *system*
  (which contains rules such as Modus Ponens already known by Stoicists, but also new rules for the quantifiers)
- Contains only *reasoning* rules but allows to express every *mathematical notion* (using sets)
- Also containts **second-order** logic:
  a variable may represent a property $\forall R \exists x R(x)$

Overview

Vocabulary

- Two propositional constants: $\bot$ and $\top$
- Connectives: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Vocabulary

- Two propositional constants: $\bot$ and $\top$
- Connectives: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- Quantifiers: the **universal** $\forall$ and the **existential** $\exists$
- Variables: $u, v, w, x, y, z, x1, x2\ldots$ ($\neq$ propositional vars.)
- Symbols: $a, b, c, p, brother, 12\ldots$
- Punctuation: the comma and the parentheses

Example 4.1.1

- $x, x1, x2, y$ are variables,
- *man*, *brother*, *succ*, 12, 24, $f1$, *itRains* are symbols:
  - **functions** with one, several or no arguments (constants)
  - **relations** with one, several or no arguments (propositional variables)
- For some (*special*) symbols
  we may use the infix notation $x = y$ or $z > 3$.

Term

Definition 4.1.2

A term is either :

- a symbol s alone
- or a variable
- or a symbol applied to terms $s(t_1, \ldots, t_n)$

Term

Definition 4.1.2

A term is either :

- a symbol s alone
- or a variable
- or a symbol applied to terms $s(t_1, \ldots, t_n)$

Example 4.1.3

x ; a ; $f(x1, x2, g(y))$; $sum(5, product(x, 42))$ are terms.

But $f(\bot, 2, y)$ is not a term.

Term

Definition 4.1.2

A term is either :

- a symbol s alone
- or a variable
- or a symbol applied to terms $s(t_1, \ldots, t_n)$

Example 4.1.3

x ; a ; $f(x1, x2, g(y))$; $sum(5, product(x, 42))$ are terms.

But $f(\perp, 2, y)$ is not a term.

Note that $42(1, y, 3)$ is also a term, but usually 42 is not used to denote a function or a relation.

Atomic formula

Definition 4.1.4 atomic formulae

An atomic formula is either:

- $\top$ or $\bot$
- or a symbol alone
- or a symbol applied to terms $s(t_1, \ldots, t_n)$

Atomic formula

Definition 4.1.4 atomic formulae

An atomic formula is either:

- $\top$ or $\bot$
- or a symbol alone
- or a symbol applied to terms $s(t_1, \ldots, t_n)$

Example 4.1.5:

- $P(x)$, a and $R(1, +(5, 42), g(z))$ are atomic formulae
- x and $A \vee f(4, 2, 6)$ are not atomic formulae

Beware : two-level interpretation

The set of **terms** and the set of **atomic formulae** are not disjoint.

For example $p(x)$ is **both** a term **and** an atomic formula.

Beware : two-level interpretation

The set of **terms** and the set of **atomic formulae** are not disjoint.

For example $p(x)$ is **both** a term **and** an atomic formula.

- $[\![t]\!]$ will be the value of t seen as a term
- $[t]$ will be the value of t seen as a formula.

(Strict) formula

Definition 4.1.6

A (strict) formula is either:

- an atomic formula
- $\neg A$
  where A is a formula
- $(A \circ B)$
  where A and B are formulae and $\circ$ a connective $\vee, \wedge, \Rightarrow, \Leftrightarrow$
- $\forall x\ A$ or $\exists x\ A$
  where A is a formula and x is any variable

Example 4.1.7

- $man(x), \quad brother(son(y), mother(Alice)), \quad =(x, +(f(x), g(y)))$

  are atomic formulae, hence formulae.

- On the opposite

  $\forall x\ (man(x) \Rightarrow man(Socrate))$

  is a non-atomic formula.

(Strict) formula: Examples

Among these expressions, which ones are strict formulae:

- x

(Strict) formula: Examples

Among these expressions, which ones are strict formulae:

- x
  not a formula
- a

(Strict) formula: Examples

Among these expressions, which ones are strict formulae:

- x
  not a formula
- a
  yes
- $(a(x) \Rightarrow b) \wedge a(x) \Rightarrow b$

(Strict) formula: Examples

Among these expressions, which ones are strict formulae:

- x
  not a formula
- a
  yes
- $(a(x) \Rightarrow b) \wedge a(x) \Rightarrow b$
  no, missing parentheses
- $\exists x((\bot \Rightarrow a(x)) \wedge b(x))$

(Strict) formula: Examples

Among these expressions, which ones are strict formulae:

- x
  not a formula
- a
  yes
- $(a(x) \Rightarrow b) \wedge a(x) \Rightarrow b$
  no, missing parentheses
- $\exists x((\perp \Rightarrow a(x)) \wedge b(x))$
  yes
- $\exists x \exists y < (-(x, y), +(a, y))$

(Strict) formula: Examples

Among these expressions, which ones are strict formulae:

- x
  not a formula
- a
  yes
- $(a(x) \Rightarrow b) \wedge a(x) \Rightarrow b$
  no, missing parentheses
- $\exists x((\bot \Rightarrow a(x)) \wedge b(x))$
  yes
- $\exists x \exists y < (-(x, y), +(a, y))$
  yes

Infix notations

Prioritized formulae: the symbols of the functions $+, -, *, /$ and the symbols of the relations $=, \neq, <, >, \leq, \geq$ are written in the usual manner.

Infix notations

Prioritized formulae: the symbols of the functions $+, -, *, /$ and the symbols of the relations $=, \neq, <, >, \leq, \geq$ are written in the usual manner.

Example 4.1.9

- $\leq (*(3, x), +(y, 5))$ is abbreviated as

Infix notations

Prioritized formulae: the symbols of the functions $+, -, *, /$ and the symbols of the relations $=, \neq, <, >, \leq, \geq$ are written in the usual manner.

Example 4.1.9

- $\leq(*(3, x), +(y, 5))$ is abbreviated as $3 * x \leq y + 5$
- $+(x, *(y, z))$ is abbreviated as

Infix notations

Prioritized formulae: the symbols of the functions $+, -, *, /$ and the symbols of the relations $=, \neq, <, >, \leq, \geq$ are written in the usual manner.

Example 4.1.9

- $\leq(*(3, x), +(y, 5))$ is abbreviated as $3 * x \leq y + 5$
- $+(x, *(y, z))$ is abbreviated as $x + y * z$

Prioritized formulae

Definition 4.1.10

A prioritized formula is either:

- an atomic formula
- $\neg A$
- $A \circ B$ with a binary connective $\circ$
- $\forall x\ A$ or $\exists x\ A$
- (A)

Inverse transformation

Precedence

- Quantifiers have the same precedence as negation.
- Connectives have a lower precedence than relations.
- $=, \neq, <, \leq, >, \geq$ have a lower precedence than $+, -, *, /$

Table 4.1 summary of priorities

Decreasing precedence from top to bottom.

OPERATIONS	
$-, +$ unary	
$*, /$ binary	left associative
$+, -$ binary	left associative
RELATIONS	
$=, \neq, <, \leq, >, \geq$	
NEGATION, QUANTIFIERS	
$\neg, \forall, \exists$	
BINARY CONNECTIVES	
$\wedge$	left associative
$\vee$	left associative
$\Rightarrow$	**right** associative
$\Leftrightarrow$	left associative

Tree representation

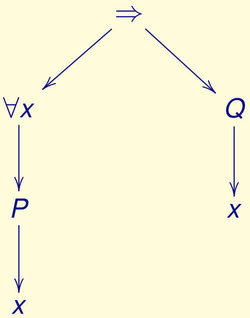
Example 4.1.12 $\forall x P(x) \Rightarrow Q(x)$

$\forall$ has higher priority: the left-hand side operand of $\Rightarrow$ is $\forall x P(x)$.

Tree representation

Example 4.1.12 $\forall x P(x) \Rightarrow Q(x)$

$\forall$ has higher priority: the left-hand side operand of $\Rightarrow$ is $\forall x P(x)$.



Overview

Idea

- The meaning of the formula $x + 2 = 4$ depends on x.
  The formula is not true (in arithmetics) unless $x = 2$.
  x is free in the previous formula.

Idea

- The meaning of the formula $x + 2 = 4$ depends on x.
  The formula is not true (in arithmetics) unless $x = 2$.
  x is free in the previous formula.
- $\forall x(x + 2 = 4)$ is unsatisfiable (in arithmetics)
  $\forall x(x + 0 = x)$ is valid
  x does not need to be assigned a value.
  There is no free variable in these two formulae.

Idea

- The meaning of the formula $x+2=4$ depends on x.
  The formula is not true (in arithmetics) unless $x=2$.
  x is free in the previous formula.
- $\forall x(x+2=4)$ is unsatisfiable (in arithmetics)
  $\forall x(x+0=x)$ is valid
  x does not need to be assigned a value.
  There is no free variable in these two formulae.
- Then the name of the variable doesn't matter.
  Frequent situation in mathematics $\int_0^1 f(x)dx$
  ... and in computer science

```
int Toto(int x) {
  return x + 1;
}
```

Free and bound occurrences

Definition 4.2.1

A quantifier binds a variable locally.

- In $\forall x\ A$ or $\exists x\ A$, the scope of the binding of x is A.

Free and bound occurrences

Definition 4.2.1

A quantifier binds a variable locally.

- In $\forall x\ A$ or $\exists x\ A$, the scope of the binding of x is A.
- An occurrence of x is bound if it is in the scope of a binding for x.
- Otherwise it is said to be free.

Free and bound occurrences

Definition 4.2.1

A quantifier binds a variable locally.

- In $\forall x\ A$ or $\exists x\ A$, the scope of the binding of x is A.
- An occurrence of x is bound if it is in the scope of a binding for x.
- Otherwise it is said to be free.

If we represent a formula by a tree:

- An occurrence of x is bound if it is

Free and bound occurrences

Definition 4.2.1

A quantifier binds a variable locally.

- In $\forall x\ A$ or $\exists x\ A$, the scope of the binding of x is A.
- An occurrence of x is bound if it is in the scope of a binding for x.
- Otherwise it is said to be free.

If we represent a formula by a tree:

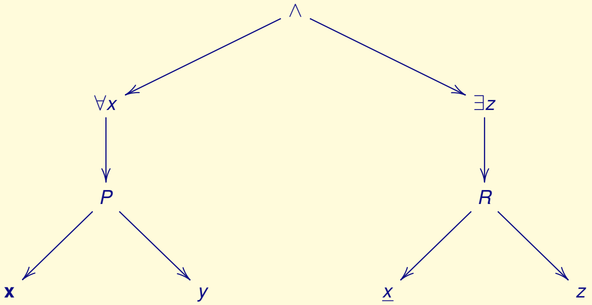
- An occurrence of x is bound if it is below a node $\exists x$ or $\forall x$.
- Any other occurrence of x is free.

Example 4.2.2

$\forall x P(\mathbf{x}, y) \land \exists z R(\underline{x}, z)$

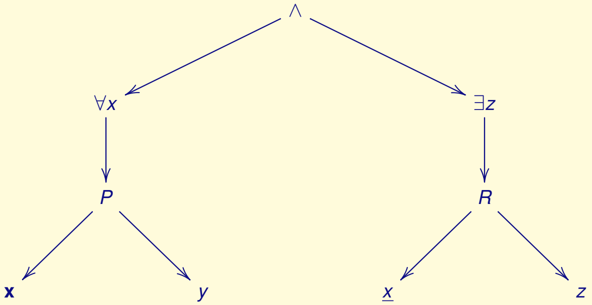
Example 4.2.2

$\forall x P(\mathbf{x}, y) \land \exists z R(\underline{x}, z)$



Example 4.2.2

$\forall x P(\mathbf{x}, y) \wedge \exists z R(\underline{x}, z)$



- The occurrence of z is bound, the occurrence of y is free.
- The bold occurrence of x is bound.
- The underlined occurrence of x is free.

Free, bound variables

Definition 4.2.3

- A formula without free variables is also called a closed formula.

Free, bound variables

Definition 4.2.3

- A formula without free variables is also called a closed formula.

Remark

- In $\forall x P(x) \vee Q(x)$, the variable x is both free and bound (thus the formula is not closed).

Free, bound variables

Definition 4.2.3

- A formula without free variables is also called a closed formula.

Remark

- In $\forall x P(x) \vee Q(x)$, the variable x is both free and bound (thus the formula is not closed).

Example 4.2.6

The free variables of $\forall x P(x, y) \wedge \exists z R(x, z)$ are x and y.

Overview

Declaring a symbol

Definition 4.3.1

A symbol declaration is a triple denoted by s^{gn} where:

- s is a symbol
- g is one of the letters f (for a function) or r (for a relation)
- n is a natural number.

Declaring a symbol

Definition 4.3.1

A symbol declaration is a triple denoted by s^{gn} where:

- s is a symbol
- g is one of the letters f (for a function) or r (for a relation)
- n is a natural number.

Remark 4.3.3

If the context is clear, we omit g and n.

Example: **equal** is always a 2 arguments relation.
Thus, we abbreviate the declaration $=^{r2}$ as $=$.

Symbol declaration: Example

Example 4.3.2

- $brother^{r2}$ is a **(r)elation** with **2** arguments
- $*^{f2}$ is a **(f)unction** with **2** arguments
- man^{r1} is a unary **relation**

Signature

Definition 4.3.4

A signature Σ is a set of symbol declarations.

Depending on its declaration, a symbol s will be called:

1. for s^{fn} : a function symbol with n arguments
2. for s^{f0} : a constant
3. for s^{rn} : a relation symbol with n arguments
4. for s^{r0} : a propositional variable

Example in mathematics

Let us define a signature for arithmetic:

Example in mathematics

Let us define a signature for arithmetic:

- Constants

Example in mathematics

Let us define a signature for arithmetic:

- Constants $0^{f0}, 1^{f0}$
- Functions

Example in mathematics

Let us define a signature for arithmetic:

- Constants $0^{f0}, 1^{f0}$
- Functions $+^{f2}, -^{f2}, *^{f2}$
- Relations

Example in mathematics

Let us define a signature for arithmetic:

- Constants $0^{f0}, 1^{f0}$
- Functions $+^{f2}, -^{f2}, *^{f2}$
- Relations $=^{r2}$

Remarques :

- The context being well-known, we write 0, 1, $+$, $-$, $*$ and $=$.
- But note that $-$ requires two arguments
  (the symbol will not be used with only one argument).

Example in mathematics

Let us define a signature for arithmetic:

- Constants $0^{f0}, 1^{f0}$
- Functions $+^{f2}, -^{f2}, *^{f2}$
- Relations $=^{r2}$

Remarques :

- The context being well-known, we write 0, 1, $+$, $-$, $*$ and $=$.
- But note that $-$ requires two arguments
  (the symbol will not be used with only one argument).

Unary relation : a relation with only 1 argument denotes a **property** of a term (for instance here $prime^{r1}$).

Term over a signature

Definition 4.3.8

A term over Σ is either:

- a variable,
- or a constant s^{f0},
- or a term $s(t_1, \ldots, t_n)$ where
  - s^{fn}
  - $n \geq 1$
  - $t_1, \ldots, t_n$ are terms over Σ.

Atomic formula over a signature

Definition 4.3.9

An atomic formula over Σ is either:

- a constant $\top$ or $\bot$
- or a propositional variable s^{r0}
- or an expression $s(t_1, \ldots, t_n)$ where
  - s^{rn}
  - $n \geq 1$
  - $t_1, \ldots, t_n$ are **terms** over Σ

Formula over a signature

Definition 4.3.10

A formula over a signature Σ is a formula whose atomic sub-formulae are atomic formulae over Σ.

Formula over a signature

Definition 4.3.10

A formula over a signature Σ is a formula whose atomic sub-formulae are atomic formulae over Σ.

Example 4.3.11

$\forall x\ (p(x) \Rightarrow \exists y\ q(x,y))$ is a formula over $\Sigma = \{p^{r1}, q^{r2}, h^{f1}, c^{f0}\}$.

Formula over a signature

Definition 4.3.10

A formula over a signature Σ is a formula whose atomic sub-formulae are atomic formulae over Σ.

Example 4.3.11

$\forall x\ (p(x) \Rightarrow \exists y\ q(x, y))$ is a formula over $\Sigma = \{p^{r1}, q^{r2}, h^{f1}, c^{f0}\}$.

But it is also a formula over the signature $\Sigma' = \{p^{r1}, q^{r2}\}$, since the symbols h and c are not in the formula.

The signature associated to a formula is the smallest signature Σ such that the formula is correctly built.

Overview

Interpretation

Definition 4.3.16

An interpretation I over a signature Σ is defined by:

- a non-empty domain D

Interpretation

Definition 4.3.16

An interpretation I over a signature Σ is defined by:

- a non-empty domain D
- every symbol s^{gn} is mapped to its value s_I^{gn} as follows:

Interpretation

Definition 4.3.16

An interpretation I over a signature Σ is defined by:

- a non-empty domain D
- every symbol s^{gn} is mapped to its value s_I^{gn} as follows:

(constant) s_I^{f0} is an element of D

Interpretation

Definition 4.3.16

An interpretation I over a signature Σ is defined by:

- a non-empty domain D
- every symbol s^{gn} is mapped to its value s_I^{gn} as follows:

(constant)	s_I^{f0} is an element of D
(function)	s_I^{fn} is a function from $D^n \to D$

Interpretation

Definition 4.3.16

An interpretation I over a signature Σ is defined by:

- a non-empty domain D
- every symbol s^{gn} is mapped to its value s_I^{gn} as follows:

(constant)	s_I^{f0} is an element of D
(function)	s_I^{fn} is a function from $D^n \to D$
(propositional variable)	s_I^{r0} is either 0 or 1

Interpretation

Definition 4.3.16

An interpretation I over a signature Σ is defined by:

- a non-empty domain D
- every symbol s^{gn} is mapped to its value s_I^{gn} as follows:

(constant)	s_I^{f0} is an element of D
(function)	s_I^{fn} is a function from $D^n \to D$
(propositional variable)	s_I^{r0} is either 0 or 1
(relation)	s_I^{rn} is a set of n-uples in D
	(the ones that satisfy this relation)

Example 4.3.17

Let *friend* be a binary relation and the domain $D = \{1, 2, 3\}$.
We consider the interpretation I where
$friend_I^{r2} = \{(1,2),(1,3),(2,3)\}$.

In this interpretation, $friend(2,3)$ is true.
On the other hand, $friend(2,1)$ is false.

Example 4.3.17

Let *friend* be a binary relation and the domain $D = \{1, 2, 3\}$.
We consider the interpretation I where
$friend_I^{r2} = \{(1, 2), (1, 3), (2, 3)\}$.

In this interpretation, $friend(2, 3)$ is true.
On the other hand, $friend(2, 1)$ is false.

Remark 4.3.18

In all interpretations, the symbol $=$ maps to the set $\{(d, d) \mid d \in D\}$.
In other words, the equality is always interpreted as the identity over D.

State, assignment

An interpretation defines only the meaning of the signature (the symbols), never the variables nor the formulae.

State, assignment

An interpretation defines only the meaning of the signature (the symbols), never the variables nor the formulae.

Definition 4.3.21

A state e of an interpretation maps each variable to an element in the domain D.

State, assignment

An interpretation defines only the meaning of the signature (the symbols), never the variables nor the formulae.

Definition 4.3.21

A state e of an interpretation maps each variable to an element in the domain D.

Definition 4.3.22

An assignment is a pair (I, e) composed of an interpretation I and a state e.

Example 4.3.23

Let the domain $D = \{1,2,3\}$ and the interpretation I where $friend_I^{r2} = \{(1,2),(1,3),(2,3)\}$

The interpretation I alone does not give us the truth value of $friend(x,y)$.

Example 4.3.23

Let the domain $D = \{1,2,3\}$ and the interpretation I where $friend_I^{r2} = \{(1,2),(1,3),(2,3)\}$

The interpretation I alone does not give us the truth value of $friend(x,y)$.

Let e be the state which maps x to 2 and y to 1.

The assignment (I,e) makes the formula $friend(x,y)$ false.

Example 4.3.31

Let I be the interpretation of domain $D = \{1,2,3\}$ where $friend_I^{r2} = \{(1,2),(1,3),(2,3)\}$.

How to interpret the formula $friend(1,2) \wedge friend(2,3) \Rightarrow friend(1,3)$ in I ?

Example 4.3.31

Let I be the interpretation of domain $D = \{1,2,3\}$ where $friend_I^{r2} = \{(1,2),(1,3),(2,3)\}$.

How to interpret the formula $friend(1,2) \wedge friend(2,3) \Rightarrow friend(1,3)$ in I ?

We know how to interpret the atomic formulae:

- $[friend(1,2)]_I = true$
- $[friend(2,3)]_I = true$
- $[friend(1,3)]_I = true$

Example 4.3.31

Let I be the interpretation of domain $D = \{1,2,3\}$ where $friend_I^{r2} = \{(1,2),(1,3),(2,3)\}$.

How to interpret the formula $friend(1,2) \wedge friend(2,3) \Rightarrow friend(1,3)$ in I ?
We know how to interpret the atomic formulae:

- $[friend(1,2)]_I = true$
- $[friend(2,3)]_I = true$
- $[friend(1,3)]_I = true$

Then we proceed as usual with the connectives, hence $[friend(1,2) \wedge friend(2,3) \Rightarrow friend(1,3)]_I = true$.
This formula is true in the interpretation I.

Overview

Finite model

Definition

A finite model of a closed formula is an interpretation of the formula in a *finite domain*, which makes the formula true.

Finite model

Definition

A finite model of a closed formula is an interpretation of the formula in a *finite domain*, which makes the formula true.

Remark

- The name of the elements of the domain is not important.
- Hence for a model with n elements, we'll use the domain of integers less than n.

Building a finite model

Naive idea: In order to know whether a closed formula has a model of domain $\{0, \ldots, n-1\}$, just

- **enumerate** all the possible interpretations of the associated signature of the formula
- **evaluate** the formula for these interpretations.

Building a finite model

Naive idea: In order to know whether a closed formula has a model of domain $\{0, \ldots, n-1\}$, just

- **enumerate** all the possible interpretations of the associated signature of the formula
- **evaluate** the formula for these interpretations.

Example

Let $\Sigma = \{a^{f0}, f^{f1}, P^{r2}\}$

Building a finite model

Naive idea: In order to know whether a closed formula has a model of domain $\{0,\ldots,n-1\}$, just

- **enumerate** all the possible interpretations of the associated signature of the formula
- **evaluate** the formula for these interpretations.

Example

Let $\Sigma = \{a^{f0}, f^{f1}, P^{r2}\}$

Over a domain of 5 elements, Σ has $5 \times 5^5 \times 2^{25}$ interpretations!

Building a finite model

Naive idea: In order to know whether a closed formula has a model of domain $\{0,\ldots,n-1\}$, just

- **enumerate** all the possible interpretations of the associated signature of the formula
- **evaluate** the formula for these interpretations.

Example

Let $\Sigma = \{a^{f0}, f^{f1}, P^{r2}\}$

Over a domain of 5 elements, Σ has $5 \times 5^5 \times 2^{25}$ interpretations!

This method is unusable in practice.

Method for finding a finite model

We look for models with n elements **by reduction to the propositional case**
Base case: a formula with no function symbol and no constant.

Method for finding a finite model

We look for models with n elements **by reduction to the propositional case**
Base case: a formula with no function symbol and no constant.

Building the n-elements model

1. Quantifiers removal: replace A by its n-expansion B.

Method for finding a finite model

We look for models with n elements **by reduction to the propositional case**

Base case: a formula with no function symbol and no constant.

Building the n-elements model

1. Quantifiers removal: replace A by its n-expansion B.
2. In B,
   - replace equalities by their truth value ($i = j$ is true iff i and j are identical)
   - Apply the usual simplifications

Method for finding a finite model

We look for models with n elements **by reduction to the propositional case**
Base case: a formula with no function symbol and no constant.

Building the n-elements model

1. Quantifiers removal: replace A by its n-expansion B.
2. In B,
   - replace equalities by their truth value ($i = j$ is true iff i and j are identical)
   - Apply the usual simplifications

   Let C be the obtained formula.
3. Look for a model of C by building a propositional assignment of the atomic formulae in C.

Expansion of a formula

Definition 4.3.39

The *n-expansion of A* consists in replacing:

- every sub-formula of A of the form $\forall xB$ with the conjunction $\bigwedge_{i<n} B < x := i >$
- every sub-formula of A of the form $\exists xB$ with the disjunction $\bigvee_{i<n} B < x := i >$

Expansion of a formula

Definition 4.3.39

The *n-expansion of A* consists in replacing:

- every sub-formula of A of the form $\forall x B$ with the conjunction $\bigwedge_{i<n} B < x := i >$
- every sub-formula of A of the form $\exists x B$ with the disjunction $\bigvee_{i<n} B < x := i >$

Example 4.3.40

The 2-expansion of the formula $\exists x P(x) \Rightarrow \forall x P(x)$ is

Expansion of a formula

Definition 4.3.39

The *n-expansion of A* consists in replacing:

- every sub-formula of A of the form $\forall xB$ with the conjunction $\bigwedge_{i<n} B < x := i >$
- every sub-formula of A of the form $\exists xB$ with the disjunction $\bigvee_{i<n} B < x := i >$

Example 4.3.40

The 2-expansion of the formula $\exists xP(x) \Rightarrow \forall xP(x)$ is

$$P(0) \vee P(1) \Rightarrow P(0) \wedge P(1)$$

Example 4.3.45

$A = \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

Example 4.3.45

$A = \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

A has no model with one element.
$(P(0) \wedge \neg P(0) \wedge (P(0) \wedge P(0) \Rightarrow 0 = 0)$ is unsatisfiable.)

Example 4.3.45

$A = \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

A has no model with one element.

$(P(0) \wedge \neg P(0) \wedge (P(0) \wedge P(0) \Rightarrow 0 = 0)$ is unsatisfiable.)

2-expansion of A

Example 4.3.45

$A \ = \ \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

A has no model with one element.
$(P(0) \wedge \neg P(0) \wedge (P(0) \wedge P(0) \Rightarrow 0 = 0)$ is unsatisfiable.)

2-expansion of A

$$(P(0)+P(1)).\ \ (\overline{P(0)}+\overline{P(1)}).\quad \begin{array}{l}(P(0).P(0) \Rightarrow 0=0).(P(0).P(1) \Rightarrow 0=1).\\ (P(1).P(0) \Rightarrow 1=0).(P(1).P(1) \Rightarrow 1=1)\end{array}$$

Example 4.3.45

$A = \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

A has no model with one element.
$(P(0) \wedge \neg P(0) \wedge (P(0) \wedge P(0) \Rightarrow 0 = 0)$ is unsatisfiable.)

2-expansion of A

$$(P(0)+P(1)).\ (\overline{P(0)}+\overline{P(1)}).\quad (P(0).P(0) \Rightarrow 0=0).(P(0).P(1) \Rightarrow 0=1).$$
$$(P(1).P(0) \Rightarrow 1=0).(P(1).P(1) \Rightarrow 1=1)$$

We replace equalities by their values

$$(P(0)+P(1)).\ (\overline{P(0)}+\overline{P(1)}).$$
$$(P(0).P(0) \Rightarrow \top).\ (P(0).P(1) \Rightarrow \bot).\ (P(1).P(0) \Rightarrow \bot).\ (P(1).P(1) \Rightarrow \top)$$

Example 4.3.45

$A = \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

A has no model with one element.
$(P(0) \wedge \neg P(0) \wedge (P(0) \wedge P(0) \Rightarrow 0 = 0)$ is unsatisfiable.)

2-expansion of A

$$(P(0)+P(1)).\ (\overline{P(0)}+\overline{P(1)}).\quad (P(0).P(0) \Rightarrow 0=0).(P(0).P(1) \Rightarrow 0=1).$$
$$(P(1).P(0) \Rightarrow 1=0).(P(1).P(1) \Rightarrow 1=1)$$

We replace equalities by their values

$$(P(0)+P(1)).\ (\overline{P(0)}+\overline{P(1)}).$$
$$(P(0).P(0) \Rightarrow \top).\ (P(0).P(1) \Rightarrow \bot).\ (P(1).P(0) \Rightarrow \bot).\ (P(1).P(1) \Rightarrow \top)$$

Which simplifies to $(P(0)+P(1)).(\overline{P(0)}+\overline{P(1)})$

Example 4.3.45

$A = \exists x P(x) \wedge \exists x \neg P(x) \wedge \forall x \forall y (P(x) \wedge P(y) \Rightarrow x = y))$

A has no model with one element.

$(P(0) \wedge \neg P(0) \wedge (P(0) \wedge P(0) \Rightarrow 0 = 0)$ is unsatisfiable.)

2-expansion of A

$$(P(0)+P(1)).\ (\overline{P(0)}+\overline{P(1)}).\quad (P(0).P(0) \Rightarrow 0=0).(P(0).P(1) \Rightarrow 0=1).$$
$$(P(1).P(0) \Rightarrow 1=0).(P(1).P(1) \Rightarrow 1=1)$$

We replace equalities by their values

$$(P(0)+P(1)).\ (\overline{P(0)}+\overline{P(1)}).$$
$$(P(0).P(0) \Rightarrow \top).\ (P(0).P(1) \Rightarrow \bot).\ (P(1).P(0) \Rightarrow \bot).\ (P(1).P(1) \Rightarrow \top)$$

Which simplifies to $(P(0)+P(1)).(\overline{P(0)}+\overline{P(1)})$

The assignment $P(0) = \mathit{true}$, $P(1) = \mathit{false}$ is a propositional model of that, hence the interpretation I of domain $\{0, 1\}$ where $P_I = \{0\}$ is a model of A.

Software for building a finite model

MACE

- **translation** of first-order formulae in propositional formulae
- **performant algorithms to find the satisfiability** of a propositional formula (e.g., different versions of the `DPLL` algorithm)

`http://www.cs.unm.edu/~mccune/mace4`

An actual example:

`http://www.cs.unm.edu/~mccune/mace4/examples/2009-11A/mace4-misc/`

Overview

Today

- **First-order logic** uses the quantifiers ∀ et ∃
- We quantify over variables representing the elements of a domain
- The **atomic formulae** are built using function symbols and relations between the elements in the domain
- To give a truth value to a formula:
  - The symbols need to be interpreted in a domain
  - The free variables need to be evaluated refering to a state
- Method for finding **(counter-)model** by finite interpretation and expansion

Next lecture

- Interpretation of a first order formula
- Notion of model
- Important equivalences

Next lecture

- Interpretation of a first order formula
- Notion of model
- Important equivalences

Homework: formalize in first-order logic

- Some people love each other.
- If two people are in love, then they're spouses.
- No one can love two distinct persons.