Logic
(Métodos Formais em Engenharia de Software)

Maria João Frade

Departmento de Informática
Universidade do Minho

2011/2012

Roadmap

- Classical Propositional Logic
- **Classical First-Order Logic**
  - syntax; semantics; validity; satisfiability; modeling with FOL
  - normal forms; Herbrandization; Skolemization; Herbrand's theorem; semi-decidability; decidable fragments
  - FOL with equality; many-sorted FOL
- First-Order Theories
- Natural Deduction

(Classical) First-Order Logic

Introduction

First-order logic (FOL) is a richer language than propositional logic. Its lexicon contains not only the symbols $\wedge$, $\vee$, $\neg$, and $\rightarrow$ (and parentheses) from propositional logic, but also the symbols $\exists$ and $\forall$ for "there exists" and "for all", along with various symbols to represent variables, constants, functions, and relations.

There are two sorts of things involved in a first-order logic formula:

- *terms*, which denote the objects that we are talking about;
- *formulas*, which denote truth values.

Examples:

"Not all birds can fly."
"Every mother is older than her children."
"John and Peter have the same maternal grandmother."

Syntax

The alphabet of a first-order language is organised into the following categories.

- *Variables:* $x, y, z, \ldots \in \mathcal{X}$ (arbitrary elements of an underlying domain)
- *Constants:* $a, b, c, \ldots \in \mathcal{C}$ (specific elements of an underlying domain)
- *Functions:* $f, g, h, \ldots \in \mathcal{F}$ (every function f as a fixed arity, $\mathsf{ar}(f)$)
- *Predicates:* $P, Q, R, \ldots \in \mathcal{P}$ (every predicate P as a fixed arity, $\mathsf{ar}(P)$)
- *Logical connectives:* $\top$, $\bot$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\forall$ (*for all*), $\exists$ (*there exists*)
- *Auxiliary symbols*: ".", "(" and ")".

We assume that all these sets are disjoint. $\mathcal{C}$, $\mathcal{F}$ and $\mathcal{P}$ are the non-logical symbols of the language. These three sets constitute the *vocabulary* $\mathcal{V} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$.

Syntax

Terms

The set of *terms* of a first-order language over a vocabulary $\mathcal{V}$ is given by the following abstract syntax

$$\textbf{Term}_{\mathcal{V}} \ni t \;::=\; x \mid c \mid f(t_1, \ldots, t_{\mathsf{ar}(f)})$$

Formulas

The set $\textbf{Form}_{\mathcal{V}}$, of *formulas* of FOL, is given by the abstract syntax

$$\begin{aligned}\textbf{Form}_{\mathcal{V}} \ni \phi, \psi \;::=\;& P(t_1, \ldots, t_{\mathsf{ar}(P)}) \mid \bot \mid \top \mid (\neg\phi) \mid (\phi \wedge \psi) \mid (\phi \vee \psi) \\ & \mid (\phi \rightarrow \psi) \mid (\forall x.\, \phi) \mid (\exists x.\, \phi)\end{aligned}$$

An *atomic formula* has the form $\bot$, $\top$, or $P(t_1, \ldots, t_{\mathsf{ar}(P)})$. A *ground term* is a term without variables. *Ground formulas* are formulas without variables, i.e., quantifier-free formulas ϕ such that all terms occurring in ϕ are ground terms.

Syntax

Convention

We adopt some syntactical conventions to lighten the presentation of formulas:

- Outermost parenthesis are usually dropped.
- In absence of parentheses, we adopt the following convention about precedence. Ranging from the highest precedence to the lowest, we have respectively: $\neg$, $\wedge$, $\vee$ and $\rightarrow$. Finally we have that $\rightarrow$ binds more tightly than $\forall$ and $\exists$.
- All binary connectives are right-associative.
- Nested quantifications such as $\forall x. \forall y. \phi$ are abbreviated to $\forall x, y.\, \phi$.
- $\forall \overline{x}. \phi$ denotes the nested quantification $\forall x_1, \ldots, x_n.\, \phi$.

Modeling with FOL

"Not all birds can fly."

We can code this sentence assuming the two unary predicates B and F expressing

$B(x)$ – x is a bird
$F(x)$ – x can fly

The declarative sentence "Not all birds can fly" can now be coded as

$$\neg\forall x.\, B(x) \rightarrow F(x)$$

or, alternatively, as

$$\exists x. B(x) \wedge \neg F(x)$$

Modeling with FOL

"Every mother is older than her children."
"John and Peter have the same maternal grandmother."

Using constants symbols j and p for John and Peter, and predicates $=$, *mother* and *older* expressing that

$mother(x, y)$ – x is a mother of y
$older(x, y)$ – x is older than y

these sentences could be expressed by

$$\forall x.\forall y.\, mother(x, y) \rightarrow older(x, y)$$

$$\forall x, y, u, v.\, mother(x, y) \wedge mother(y, j) \wedge mother(u, v) \wedge mother(v, p) \rightarrow x = u$$

A different and more elegant encoding is to represent y'mother in a more direct way, by using a function instead of a relation. We write $m(y)$ to mean y'mother. This way the two sentences above have simpler encondings.

$$\forall x.\, older(m(x), x) \qquad \text{and} \qquad m(m(j)) = m(m(p))$$

Modeling with FOL

Assume further the following predicates and constant symbols

$flower(x)$ – x is a flower	$likes(x, y)$ – x likes y
$sport(x)$ – x is a sport	$brother(x, y)$ – x is brother of y
a – Anne	

- "Anne likes John's brother." $\exists x.\, brother(x, j) \wedge likes(a, x)$
- "John likes all sports." $\forall x.\, sports(x) \rightarrow likes(j, x)$
- "John's mother likes flowers." $\forall x.\, flower(x) \rightarrow likes(m(j), x)$
- "John's mother does not like some sports." $\exists y.\, sport(y) \wedge \neg likes(m(j), y)$
- "Peter only likes sports." $\forall x.\, likes(p, x) \rightarrow sports(x)$
- "Anne has two children."

$$\exists x_1, x_2.\, mother(a, x_1) \wedge mother(a, x_2) \wedge x_1 \neq x_2 \wedge \\ \forall z.\, mother(a, z) \rightarrow z = x_1 \vee z = x_2$$

Free and bound variables

- The *free variables* of a formula ϕ are those variables occurring in ϕ that are not quantified. FV(ϕ) denotes the set of free variables occurring in ϕ.
- The *bound variables* of a formula ϕ are those variables occurring in ϕ that do have quantifiers. BV(ϕ) denote the set of bound variables occurring in ϕ.

Note that variables can have both free and bound occurrences within the same formula. Let ϕ be $\exists x.\, R(x, y) \wedge \forall y.\, P(y, x)$, then

$$\text{FV}(\phi) = \{y\} \quad \text{and} \quad \text{BV}(\phi) = \{x, y\}.$$

- A formula ϕ is *closed* (or *a sentence*) if it does not contain any free variables.
- If FV(ϕ) $= \{x_1, \ldots, x_n\}$, then
  - its *universal closure* is $\forall x_1. \ldots \forall x_n.\, \phi$
  - its *existential closure* is $\exists x_1. \ldots \exists x_n.\, \phi$

Substitution

Substitution

- We define $u[t/x]$ to be the term obtained by replacing each occurrence of variable x in u with t.
- We define $\phi[t/x]$ to be the formula obtained by replacing each free occurrence of variable x in ϕ with t.

Care must be taken, because substitutions can give rise to undesired effects.

Given a term t, a variable x and a formula ϕ, we say that *t is free for x in ϕ* if no free x in ϕ occurs in the scope of $\forall z$ or $\exists z$ for any variable z occurring in t.

From now on we will assume that all substitutions satisfy this condition. That is when performing the $\phi[t/x]$ we are always assuming that t is free for x in ϕ.

Substitution

Convention

We write $\phi(x_1, \ldots, x_n)$ to denote a formula having free variables $x_1, \ldots, x_n$. We write $\phi(t_1, \ldots, t_n)$ to denote the formula obtained by replacing each free occurrence of x_i in ϕ with the term t_i. When using this notation, it should always be assumed that each t_i is free for x_i in ϕ. Also note that when writhing $\phi(x_1, ..., x_n)$ we do not mean that $x_1, ..., x_n$ are the only free variables of ϕ.

A *sentence* of first-order logic is a formula having no free variables.

- The presence of free variables distinguishes formulas from sentences.
- This distinction did not exist in propositional logic.

Semantics

$\mathcal{V}$-structure

Let $\mathcal{V}$ be a vocabulary. A *$\mathcal{V}$-structure* $\mathcal{M}$ is a pair $\mathcal{M} = (D, I)$ where D is a nonempty set called the *interpretation domain*, and I is an *interpretation function* that assigns constants, functions and predicates over D to the symbols of $\mathcal{V}$ as follows:

- for each constant symbol $c \in \mathcal{C}$, the interpretation of c is a constant $I(c) \in D$;
- for each $f \in \mathcal{F}$, the interpretation of f is a function $I(f) : D^{\mathsf{ar}(f)} \to D$;
- for each $P \in \mathcal{P}$, the interpretation of P is a function $I(P) : D^{\mathsf{ar}(P)} \to \{0, 1\}$. In particular, 0-ary predicate symbols are interpreted as truth values.

$\mathcal{V}$-structures are also called *models* for $\mathcal{V}$.

Semantics

Assignment

An *assignment* for a domain D is a function $\alpha : \mathcal{X} \to D$.

We denote by $\alpha[x \mapsto a]$ the assignment which maps x to a and any other variable y to $\alpha(y)$.

Given a $\mathcal{V}$-structure $\mathcal{M} = (D, I)$ and given an assignment $\alpha : \mathcal{X} \to D$, we define an *interpretation function for terms*, $\alpha_{\mathcal{M}} : \textbf{Term}_{\mathcal{V}} \to D$, as follows:

$$\begin{array}{lcl} \alpha_{\mathcal{M}}(x) & = & \alpha(x) \\ \alpha_{\mathcal{M}}(c) & = & I(c) \\ \alpha_{\mathcal{M}}(f(t_1, \ldots, t_n)) & = & I(f)(\alpha_{\mathcal{M}}(t_1), \ldots, \alpha_{\mathcal{M}}(t_n)) \end{array}$$

Semantics

Satisfaction relation

Given a $\mathcal{V}$-structure $\mathcal{M} = (D, I)$ and given an assignment $\alpha : \mathcal{X} \to D$, we define the *satisfaction relation* $\mathcal{M}, \alpha \models \phi$ for each $\phi \in \textbf{Form}_{\mathcal{V}}$ as follows:

$$\begin{array}{lll} \mathcal{M}, \alpha \models \top & & \\ \mathcal{M}, \alpha \not\models \bot & & \\ \mathcal{M}, \alpha \models P(t_1, \ldots, t_n) & \text{iff} & I(P)(\alpha_{\mathcal{M}}(t_1), \ldots, \alpha_{\mathcal{M}}(t_n)) = 1 \\ \mathcal{M}, \alpha \models \neg\phi & \text{iff} & \mathcal{M}, \alpha \not\models \phi \\ \mathcal{M}, \alpha \models \phi \wedge \psi & \text{iff} & \mathcal{M}, \alpha \models \phi \text{ and } \mathcal{M}, \alpha \models \psi \\ \mathcal{M}, \alpha \models \phi \vee \psi & \text{iff} & \mathcal{M}, \alpha \models \phi \text{ or } \mathcal{M} \models \psi \\ \mathcal{M}, \alpha \models \phi \to \psi & \text{iff} & \mathcal{M}, \alpha \not\models \phi \text{ or } \mathcal{M}, \alpha \models \psi \\ \mathcal{M}, \alpha \models \forall x. \phi & \text{iff} & \mathcal{M}, \alpha[x \mapsto a] \models \phi \text{ for all } a \in D \\ \mathcal{M}, \alpha \models \exists x. \phi & \text{iff} & \mathcal{M}, \alpha[x \mapsto a] \models \phi \text{ for some } a \in D \end{array}$$

Validity and satisfiability

When $\mathcal{M}, \alpha \models \phi$, we say that $\mathcal{M}$ *satisfies* ϕ *with* α.

We write $\mathcal{M} \models \phi$ iff $\mathcal{M}, \alpha \models \phi$ holds for every assignment α.

A formula ϕ is

valid	iff	$\mathcal{M}, \alpha \models \phi$ holds for all structure $\mathcal{M}$ and assignments α. A valid formula is called a *tautology*. We write $\models \phi$.
satisfiable	iff	there is some structure $\mathcal{M}$ and some assigment α such that $\mathcal{M}, \alpha \models \phi$ holds.
unsatisfiable	iff	it is not satisfiable. An unsatisfiable formula is called a *contradiction*.
refutable	iff	it is not valid.

Consequence and equivalence

Given a set of formulas Γ, a model $\mathcal{M}$ and an assignment α, $\mathcal{M}$ is said to *satisfy* Γ *with* α, denoted by $\mathcal{M}, \alpha \models \Gamma$, if $\mathcal{M}, \alpha \models \phi$ for every $\phi \in \Gamma$.

Γ *entails* ϕ (or that ϕ is a *logical consequence* of Γ), denoted by $\Gamma \models \phi$, iff for all structures $\mathcal{M}$ and assignments α, whenever $\mathcal{M}, \alpha \models \Gamma$ holds, then $\mathcal{M}, \alpha \models \phi$ holds as well.

ϕ is *logically equivalent* to ψ, denoted by $\phi \equiv \psi$, iff $\{\phi\} \models \psi$ and $\{\psi\} \models \phi$.

Deduction theorem

$\Gamma, \phi \models \psi \quad \text{iff} \quad \Gamma \models \phi \to \psi$

Consistency

The set Γ is *consistent* or *satisfiable* iff there is a model $\mathcal{M}$ and an assigment α such that $\mathcal{M}, \alpha \models \phi$ holds for all $\phi \in \Gamma$.

We say that Γ is *inconsistent* iff it is not consistent and denote this by $\Gamma \models \bot$.

Proposition

- $\{\phi, \neg\phi\} \models \bot$
- If $\Gamma \models \bot$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \models \bot$.
- $\Gamma \models \phi \quad \text{iff} \quad \Gamma, \neg\phi \models \bot$

Substitution

- Formula ψ is a *subformula* of formula ϕ if it occurs syntactically within ϕ.
- Formula ψ is a *strict subformula* of ϕ if ψ is a subformula of ϕ and $\psi \neq \phi$

Substitution theorem

Suppose $\phi \equiv \psi$. Let θ be a formula that contains ϕ as a subformula. Let θ' be the formula obtained by safe replacing (i.e., avoiding the capture of free variables of ϕ) some occurrence of ϕ in θ with ψ. Then $\theta \equiv \theta'$.

Adquate sets of connectives for FOL

Renaming of bound variables

If y is free for x in ϕ and $y \notin \mathsf{FV}(\phi)$, then the following equivalences hold.

- $\forall x.\phi \equiv \forall y.\phi[y/x]$
- $\exists x.\phi \equiv \exists y.\phi[y/x]$

Lemma

The following equivalences hold in first-order logic.

$$\begin{array}{rcl@{\qquad}rcl}
\forall x.\phi \wedge \psi &\equiv& (\forall x.\phi) \wedge (\forall x.\psi) & \exists x.\phi \vee \psi &\equiv& (\exists x.\phi) \vee (\exists x.\psi)\\
\forall x.\phi &\equiv& (\forall x.\phi) \wedge \phi[t/x] & \exists x.\phi &\equiv& (\exists x.\phi) \vee \phi[t/x]\\
\neg\forall x.\phi &\equiv& \exists x.\neg\phi & \neg\exists x.\phi &\equiv& \forall x.\neg\phi
\end{array}$$

As in propositional logic, there is some redundancy among the connectives and quantifiers since $\forall x.\,\phi \equiv \neg\exists x.\,\neg\phi$ and $\exists x.\,\phi \equiv \neg\forall x.\,\neg\phi$.

Decidability

Given formulas ϕ and ψ as input, we may ask:

Decision problems

Validity problem:	"Is ϕ valid ?"
Satisfiability problem:	"Is ϕ satisfiable ?"
Consequence problem:	"Is ψ a consequence of ϕ ?"
Equivalence problem:	"Are ϕ and ψ equivalent ?"

These are, in some sense, variations of the same problem.

ϕ is valid	iff	$\neg\phi$ is unsatisfiable
$\phi \models \psi$	iff	$\neg(\phi \to \psi)$ is unsatisfiable
$\phi \equiv \psi$	iff	$\phi \models \psi$ and $\psi \models \phi$
ϕ is satisfiable	iff	$\neg\phi$ is not valid

Decidability

A *solution* to a decision problem is a program that takes problem instances as input and always terminates, producing a correct "yes" or "no" output.

- A decision problem is *decidable* if it has a solution.
- A decision problem is *undecidable* if it is not decidable.

In PL we could, in theory, compute a truth table to determine whether or not a formula is satisfiable. In FOL, we would have to check every model to do this.

Theorem (Church & Turing)

- The decision problem of validity in first-order logic is undecidable: no program exists which, given any ϕ, decides whether $\models \phi$.
- The decision problem of satisfiability in first-order logic is undecidable: no program exists which, given any ϕ, decides whether ϕ is satisfiable.

Semi-decidability

However, there is a procedure that halts and says "yes" if ϕ is valid.

A decision problem is *semi-decidable* if exists a procedure that, given an input,

- halts and answers "yes" iff "yes" is the correct answer,
- halts and answers "no" if "no" is the correct answer, or
- does not halt if "no" is the correct answer

Unlike a decidable problem, the procedure is only guaranteed to halt if the correct answer is "yes".

The decision problem of validity in first-order logic is semi-decidable.

Normal forms

A first-order formula is in *negation normal form (NNF)* if the implication connective is not used in it, and negation is only applied to atomic formulas.

If x does not occur free in ψ, then the following equivalences hold.

$$\begin{array}{ll}
(\forall x.\phi) \land \psi \equiv \forall x.\phi \land \psi & \psi \land (\forall x.\phi) \equiv \forall x.\psi \land \phi \\
(\forall x.\phi) \lor \psi \equiv \forall x.\phi \lor \psi & \psi \lor (\forall x.\phi) \equiv \forall x.\psi \lor \phi \\
(\exists x.\phi) \land \psi \equiv \exists x.\phi \land \psi & \psi \land (\exists x.\phi) \equiv \exists x.\psi \land \phi \\
(\exists x.\phi) \lor \psi \equiv \exists x.\phi \lor \psi & \psi \lor (\exists x.\phi) \equiv \exists x.\psi \lor \phi
\end{array}$$

The applicability of these equivalences can always be assured by appropriate renaming of bound variables.

Normal forms

A formula is in *prenex form* if it is of the form $Q_1x_1.Q_2x_2.\dots.Q_nx_n.\psi$ where each Q_i is a quantifier (either $\forall$ or $\exists$) and ψ is a quantifier-free formula.

Prenex form of $\forall x.(\forall y.P(x,y) \lor Q(x)) \rightarrow \exists z.P(x,z)$

First we compute the NNF and then we go for the prenex form.

$$\begin{array}{lll}
 & \forall x.(\forall y.P(x,y) \lor Q(x)) \rightarrow \exists z.P(x,z) & \equiv \\
 & \forall x.\neg(\forall y.P(x,y) \lor Q(x)) \lor \exists z.P(x,z) & \equiv \\
\text{(NNF)} & \forall x.\exists y.(\neg P(x,y) \land \neg Q(x)) \lor \exists z.P(x,z) & \equiv \\
\text{(prenex)} & \forall x.\exists y.\exists z.(\neg P(x,y) \land \neg Q(x)) \lor P(x,z) &
\end{array}$$

Herbrand/Skolem normal forms

Let ϕ be a first-order formula in prenex normal form.

- The *Herbrandization* of ϕ (written ϕ^H) is an existential formula obtained from ϕ by repeatedly and exhaustively applying the following transformation:

$$\exists x_1,\dots,x_n.\forall y.\psi \ \leadsto\ \exists x_1,\dots,x_n.\psi[f(x_1,\dots,x_n)/y]$$

  with f a fresh function symbol with arity n (i.e. f does not occur in ψ).
- The *Skolemization* of ϕ (written ϕ^S) is a universal formula obtained from ϕ by repeatedly applying the transformation:

$$\forall x_1,\dots,x_n\exists y.\psi \ \leadsto\ \forall x_1,\dots,x_n.\psi[f(x_1,\dots,x_n)/y]$$

  with f a fresh function symbol with arity n.
- *Herbrand normal form* (resp. *Skolem normal form*) formulas are those obtained by the process of Herbrandization (resp. Skolemization).

Herbrandization/Skolemization

A formula ϕ and its Herbrandization/Skolemization are not logically equivalent.

Proposition

Let ϕ be a first-order formula in prenex normal form.

- ϕ is valid iff its Herbrandization ϕ^H is valid.
- ϕ is unsatisfiable iff its Skolemization ϕ^S is unsatisfiable.

- It is convenient to write Herbrand and Skolem formulas using vector notation $\exists\overline{x}.\psi$ and $\forall\overline{x}.\psi$ (with ψ quantifier free), respectively.
- The quantifier-free sub-formula can be furthered normalised:
  - *Universal CNF:* $\forall\overline{x}.\bigwedge_i \bigvee_j l_{ij}$
  - *Existencial DNF:* $\exists\overline{x}.\bigvee_i \bigwedge_j l_{ij}$

  where *literals* are either atomic predicates or negation of atomic predicates.
- Herbrandization/Skolemization change the underlying vocabulary. These additional symbols are called *Herbrand/Skolem functions*.

Herbrandization/Skolemization

- The requirement that the original formula be in prenex normal form is too strong: Skolemization/Herbrandization can be applied to formulas in negative normal form. In fact, this relaxation results in potentially simpler Skolem normal forms (with fewer Skolem functions, or functions with lower arity).
- Herbrand (resp. Skolem) normal form is important since it allows to reduce validity (resp. unsatisfiability) of a first-order formula to the existence of suitable instances of a quantifier-free formula, considering models constructed directly from the syntax of terms.

Herbrand models

Herbrand interpretation

Let $\mathcal{V}$ be a first-order vocabulary and assume $\mathcal{V}$ has at least one constant symbol (otherwise, we explicitly expand the vocabulary with such a symbol). A *Herbrand Interpretation* $\mathcal{H} = (D_{\mathcal{H}}, I_{\mathcal{H}})$ is a $\mathcal{V}$-structure specified by a set of closed atomic predicates (i.e. atomic predicates applied to ground terms), also denoted by $\mathcal{H}$. The interpretation structure is given as follows:

- Interpretation domain: $D_{\mathcal{H}}$ is the set of ground terms for the vocabulary $\mathcal{V}$. It is called the *Herbrand universe* for $\mathcal{V}$.
- Interpretation of constants: for every $c \in \mathcal{V}$, $I_{\mathcal{H}}(c) = c$;
- Interpretation of functions: for every $f \in \mathcal{V}$ with ar$(f) = n$, $I_{\mathcal{H}}(f)$ consists of the n-ary function that, given ground terms $t_1, \ldots, t_n$, returns the ground term $f(t_1, \ldots, t_n)$;
- Interpretation of predicates: for every $P \in \mathcal{V}$ with ar$(P) = n$, $I_{\mathcal{H}}(P)$ is the n-ary relation $\{(t_1, \ldots, t_n) \mid P(t_1, \ldots, t_n) \in \mathcal{H}\}$.

Herbrand's theorem

Lemma

- An existential formula ϕ is valid iff for every Herbrand model $\mathcal{H}$, $\mathcal{H} \models \phi$.
- A universal formula ϕ is unsatisfiable iff there exists no Herbrand model $\mathcal{H}$ such that $\mathcal{H} \models \phi$.

Herbrand's Theorem

- An existential first-order formula $\exists \overline{x}.\psi$ (with ψ quantifier-free) is valid iff there exists an integer k and ground instances $\psi\sigma_1, \ldots, \psi\sigma_k$ such that $\psi\sigma_1 \vee \ldots \vee \psi\sigma_k$ is propositionally valid.
- A universal formula $\forall \overline{x}.\psi$ (with ψ quantifier-free) is unsatisfiable iff there exists an integer k and closed instances $\psi\sigma_1, \ldots, \psi\sigma_k$ such that $\psi\sigma_1 \wedge \cdots \wedge \psi\sigma_k$ is propositionally unsatisfiable.

Semi-decidability

Theorem

The problem of validity of first-order formulas is semi-decidable, i.e. there exists a procedure that, given a first-order formula, answers “yes” iff the formula is valid (but might not terminate if the formula is not valid).

- An interesting refinement is to investigate fragments in which bounds can be established for searching the ground instance space.
- This immediately leads to a bound on the number of instances whose search is required by Herbrand's theorem, turning validity of formulas decidable.
- Clearly if the set of ground terms is finite, the set of ground instances of the formula under scrutiny will be finite as well.

Semi-decidability

Consider a vocabulary with two constants 1 and 2, and two unary predicates P and Q. To prove the validity of the formula $\exists x.\ (P(x) \rightarrow \forall y.\ P(y))$, we can

1. Calculate the prenex normal form equivalent of the formula.
$$\exists x. \forall y.\ (\neg P(x) \vee P(y))$$
2. Calculate the Herbrandization of the result. This process introduces a new function f in the vocabulary.
$$\exists x.\ \neg P(x) \vee P(f(x))$$
3. The set of ground terms for the new vocabulary is $\{1, 2, f(1), f(2), f(f(1)), f(f(2)), \ldots\}$
4. Using the Herbrand's Theorem we can say that $\exists x.\ \neg P(x) \vee P(f(x))$ is valid, because the formula
$$(\neg P(1) \vee P(f(1))) \vee (\neg P(2) \vee P(f(2))) \vee (\neg P(f(1)) \vee P(f(f(1))))$$
5. Hence, $\exists x.\ (P(x) \rightarrow \forall y.\ P(y))$ is valid.

Decidable fragments

- If the underlying vocabulary has a finite set of constants and no function symbol, the set of ground terms is finite. Note however that function symbols might be introduced during the Herbrandization/Skolemization.
- Restricting attention to formulas whose prenex normal form has the shape $\forall \overline{x}. \exists \overline{y}.\ \psi$ ensures that only constants are introduced by Herbrandization.
  - This fragment of formulas is normally known as the *AE fragment*.
  - The class of formulas can be further enlarged by observing that a formula not in AE may be equivalent to one in AE (e.g. *miniscope* – pushing existential quantifiers inside the formula, thus minimizing their scopes).

$$\exists x. \forall y.\ P(x) \vee Q(y) \ \equiv\ \exists x. P(x) \vee \forall y. Q(y) \ \equiv\ \forall y. \exists x.\ P(x) \vee Q(y)$$

Decidable fragments

Although first-order validity is undecidable, there are special simple fragments of FOL where it is decidable, e.g.

- *Monadic predicate logic* (i.e. only unary predicates and no function symbols) is decidable.
- *The Bernays-Schönfinkel class* of formulas (i.e. formulas that can be written with all quantiers appearing at the beginning of the formula with existentials before universals and that do not contain any function symbols) is decidable.

FOL with equality

There are different conventions for dealing with equality in first-order logic.

- We have follow the approach of considering equality predicate $(=)$ as a non-logical symbol, treated in the same way as any other predicate. We are working with what are usually known as *"first-order languages without equality"*.
- An alternative approach, usually called *"first-order logic with equality"*, considers equality as a logical symbol with a fixed interpretation.

  In this approach the equality symbol $(=)$ is interpreted as the equality relation in the domain of interpretation. So we have, for a structure $\mathcal{M} = (D, I)$ and an assignment $\alpha : \mathcal{X} \rightarrow D$, that

$$\mathcal{M}, \alpha \models t_1 = t_2 \quad \text{iff} \quad \alpha_{\mathcal{M}}(t_1) \text{ and } \alpha_{\mathcal{M}}(t_2) \text{ are the same element of } D$$

FOL with equality

To understand the significant difference between having equality with the status of any other predicate, or with a fixed interpretation as in first-order logic with equality, consider the formulas

- $\exists x_1, x_2. \forall y.\ y = x_1 \vee y = x_2$
  With a fixed interpretation of equality, the validity of this formula implies that the cardinality of the interpretation domain is at most two – the quantifiers can actually be used to fix the cardinality of the domain, which is not otherwise possible in first-order logic.
- $\exists x_1, x_2. \neg(x_1 = x_2)$
  The validity of this formula implies that there exist at least two distinct elements in the domain, thus its cardinality must be at least two.

Many-sorted FOL

- A natural variant of first-order logic that can be considered is the one that results from allowing different domains of elements to coexist in the framework. This allows distinct "sorts" or types of objects to be distinguished at the syntactical level, constraining how operations and predicates interact with these different sorts.
- Having full support for different sorts of objects in the language allows for cleaner and more natural encodings of whatever we are interested in modeling and reasoning about.
- By adding to the formalism of FOL the notion of sort, we can obtain a flexible and convenient logic called *many-sorted first-order logic*, which has the same properties as FOL.

Many-sorted FOL

- A many-sorted vocabulary (signature) is composed of a set of sorts, a set of function symbols, and a set of predicate symbols.
  - Each function symbol f has associated with a type of the form $S_1 \times \ldots \times S_{\mathsf{ar}(f)} \to S$ where $S_1, \ldots, S_{\mathsf{ar}(f)}, S$ are sorts.
  - Each predicate symbol P has associated with it a type of the form $S_1 \times \ldots \times S_{\mathsf{ar}(P)}$.
  - Each variable is associated with a sort.
- The formation of terms and formulas is done only accordingly to the typing policy, i.e., respecting the "sorts".
- The domain of discourse of any structure of a many-sorted vocabulary is fragmented into different subsets, one for every sort.
- The notions of assignment and structure for a many-sorted vocabulary, and the interpretation of terms and formulas are defined in the expected way.

Proof system

- So far we have taken the *"semantic"* approach to logic, with the aim of characterising the semantic concept of model, from which validity, satisfiability and semantic entailment were derived.
- However, this is not the only possible point of view.
- Instead of adopting the view based on the notion of truth, we can think of logic as a codification of reasoning. This alternative approach to logic, called *"deductive"*, focuses directly on the deduction relation that is induced on formulas, i.e., on what formulas are logical consequences of other formulas.
- We will explore this perspective later in this course.

Exercises (from [Huth&Ryan 2004])

- Use the predicates

$A(x,y)$:	x admires y	$P(x)$:	x is a professor
$B(x,y)$:	x attended y	$S(x)$:	x is a student
		$L(x)$:	x is a lecture

  and the nullary function symbol (constant) 'Mary' to translate the following into predicate logic:

  1. Mary admires every professor.
  2. Some professor admires Mary.
  3. Mary admires herself.
  4. No student attended every lecture.
  5. No lecture was attended by every student.
  6. No lecture was attended by any student.

Exercises (from [Huth&Ryan 2004])

- Find appropriate predicates and their specification to translate the following into predicate logic:

  1. All red things are in the box.
  2. Only red things are in the box.
  3. No animal is both a cat and a dog.
  4. Every prize was won by a boy.
  5. A boy won every prize.

Exercises (from [Huth&Ryan 2004])

- Let $F(x,y)$ mean that x is the father of y; $M(x,y)$ denotes x is the mother of y. Similarly, $H(x,y)$, $S(x,y)$, and $B(x,y)$ say that x is the husband/sister/brother of y, respectively. You may also use constants to denote individuals, like 'Ed' and 'Patsy'. However, you are not allowed to use any predicate symbols other than the above to translate the following sentences into predicate logic:

  1. Everybody has a mother.
  2. Everybody has a father and a mother.
  3. Whoever has a mother has a father.
  4. Ed is a grandfather.
  5. All fathers are parents.
  6. All husbands are spouses.
  7. No uncle is an aunt.
  8. Nobody's grandmother is anybody's father.
  9. Ed and Patsy are husband and wife.
  10. Carl is Monique's brother-in-law.

Exercises (from [RSD 2011])

- Compute Herbrand and Skolem normal forms for each of the following formulas:

  1. $\exists x.(\forall y.\exists z.P(x,y,z)) \wedge \exists x.\forall y.\neg P(x,y,z)$
  2. $\forall x.(\exists y.Q(x,y)) \vee \forall y.\exists z.R(x,y,z)$
  3. $\forall x.(\exists y.P(x,y)) \rightarrow \exists y.Q(x,y)$
  4. $\forall x.P(x) \wedge (\forall y.Q(y)) \rightarrow \exists y.R(x,f(x,y))$