Previously . . . (what I expect you to know)

- propositional logic:
  - syntax: propositional symbols
    logical operators $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\top$, $\bot$
  - semantics: truth tables, truth values $\mathbf{1}$, $\mathbf{0}$
- conversion to clausal form
  - atoms, literals, clauses (= multi-sets)
- resolution calculus: resolution rule
  factoring rule
- tautology deletion, subsumption deletion
- multi-set extension ordering

First-order logic

Extension of propositional logic

with quantification over individuals

Most important 'unifying' formal logic system

- knowledge representation and reasoning in CS and AI
- specification and verification in software engineering
- semantics of programming languages
- SQL querying language of data bases
- rules in expert systems
- foundation for logic programming

Very expressive

Deficiencies of propositional logic

- In propositional logic we can make statements about truth of propositions

  "grass is green", "1 is an even number",

  "grass is green" $\wedge$ $\neg$("1 is an even number")

- But not about objects of structures and relations among several objects. E.g.

  "If b lies between a and c, then b also lies between c and a"

  is a true statement, but $F \rightarrow G$ is not an adequate representation

- Propositional logic cannot represent, e.g.,

  "*Every* student wants a job", "*Some* students don't like Logic"

What are the new concepts in first-order logic?

- First-order logic = extension of propositional logic
- Statements about objects of structures can be expressed

  $Between(b, a, c) \rightarrow Between(b, c, a)$ $Even(1)$

- Symbols denoting functions and constants are allowed

  b, a, c 1 ann $father\_of(ann)$ -1 $1 + 5$

- Abstract, schematic statements via variables

  $x + y$ $Even(x)$ $Student(x)$ $Between(0, -x, x)$

  ...and quantification ($\exists$ = 'there exists', $\forall$ = 'for all')

  $\exists x\, Even(x)$ $\forall x\big(Student(x) \rightarrow Want\_job(x)\big)$

Alphabet of a first-order language

- Logical symbols (domain-independent, fixed):
  logical connectives, quantifiers, variables, auxiliary symbols
- Non-logical symbols (domain-specific, flexible):
  function symbols, constants, predicate symbols

Logical symbols in the alphabet

- Logical connectives:

$\bot$	falsehood	$\top$	truth	
$\neg$	not	$\wedge$	and	$\vee$ or
$\rightarrow$	implication	$\leftrightarrow$	equivalence	
$\forall$	for all	$\exists$	there exists	(quantifiers)
$\approx$	equality symbol			

- Variables: A countably infinite set $\mathcal{X}$ of symbols $x_0, x_1, x_2, \ldots$
  We also use the symbols x, y, z to denote variables
- Auxiliary symbols: ( ) [ ] .

Note: $\approx$ is equality in FOL language; $=$ is syntactic equality

Non-logical symbols in the alphabet

- Function symbols: A finite or countably infinite set $\mathcal{F}$ of symbols f with arity $n \geq 0$, written f/n,
  - If $n = 0$ then f is also called a constant (symbol).

  Notation: f, g, h for function symbols a, b, c for constants
- Predicate symbols: A finite or countably infinite set $\mathcal{P}$ of symbols P with arity $m \geq 0$, written P/m.
  - If $m = 0$ then P is also called a propositional symbol.

  Notation: P, Q, R for pred. symbols; p, q, r for prop. symbols
- We refer to $\Sigma = (\mathcal{F}, \mathcal{P})$ as the signature.

Each function/predicate symbol has an arity which indicates the number of arguments it takes

Arity	0	1	2	3	n
Symbol	nullary	unary	binary	ternary	n-ary

Non-logical symbols for two sample domains

- Constants, functions, propositional symbols and predicate symbols are domain-specific symbols
  - Are flexibly chosen as appropriate for an application
  - Their interpretations are flexible
- Symbols for number theory:

  Constants: $0, 1, 5$
  Functions: s , $-$, $*$, $+$
  Predicates: $<$, *Even*, *Prime*
- Symbols for a student domain

  Constants: *ann*, 60332
  Functions: *grade*
  Predicates: *Student*, *Likes*, $<$

Terms

- Terms over Σ and $\mathcal{X}$ are formed according to this syntactic rule:

$$\begin{array}{lcll} s, t, u & \longrightarrow & x & \text{(first-order variable)} \\ & | & a & \text{(constant)} \\ & | & f(s_1, \ldots, s_n), \quad n > 0 & \text{(functional term)} \end{array}$$

  where $x \in \mathcal{X}$, $a/0 \in \mathcal{F}$ and $f/n \in \mathcal{F}$

- Examples, in the number theory domain:

$$0 \qquad 1 \qquad 5 \qquad -(1) \qquad *(1,5) \qquad x \qquad *(x,y)$$

- By $\mathrm{T}_\Sigma(\mathcal{X})$ we denote the set of Σ-terms (over $\mathcal{X}$).
- A term not containing any variable is called a ground term. By T_Σ we denote the set of ground Σ-terms.

Subterms

- If s is a term and s occurs as a part of another term t, then s is a subterm of t.
- Example:

$$+(+(x,y), s(0))$$

  Subterms:

$$0 \qquad x \qquad s(0) \qquad +(x,y) \qquad +(+(x,y), s(0))$$

  (Are there more?)

- Alternative infix notation for the term: $(x + y) + s(0)$

Atomic formulae

- Atomic formulae over Σ are formed according to:

$$\begin{array}{lcll} A, B & \longrightarrow & P(s_1, \ldots, s_n), \quad n \geq 0 & \text{(non-equational atom)} \\ & | & s \approx t & \text{(equational atom)} \end{array}$$

  where $P/n \in \mathcal{P}$

- Atomic formulae are also called atoms
- Examples of atoms are:

$$Even(x), \qquad P(a, x), \qquad < (0, s(0)), \qquad s(x) \approx y$$

Aside: Whenever we admit equations as atomic formulae we are in the realm of first-order logic with equality. Admitting equality does not really increase the expressiveness of first-order logic. But deductive systems where equality is treated specifically can be much more efficient.

Formulae of a first-order language

- Formulae over Σ are formed according to:

$$\begin{array}{lcll} F, G, H & \longrightarrow & \bot \mid \top & \text{(logical constants)} \\ & | & A & \text{(atomic formula)} \\ & | & \neg F & \text{(not)} \\ & | & (F \star G) & \star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \text{ (binary conn.)} \\ & | & \forall x F & \text{(universal quantification)} \\ & | & \exists x F & \text{(existentially quantification)} \end{array}$$

- $\mathrm{F}_\Sigma(\mathcal{X})$ denotes the set of all first-order formulae over Σ and $\mathcal{X}$
- Formulae without variables are ground.

Formulae in number theory, informal meaning in $\mathbb{N}$

- $Even(1)$
- $<(1, 5)$ in infix form: $1 < 5$
- $Even(x)$
- $0 < x$
- $\exists x.Even(x)$
  There is an even number
- $\forall x.\exists y.x < y$
  For every number x there is a number y greater than x
- $\forall x.Even(*(x, x))$
  Every square number is even
- $\exists x.(Even(x) \wedge Prime(x) \wedge 0 < x)$
  There is an even prime number greater than 0
- $\forall x.\forall y.(x < y \rightarrow \neg(y < x))$
  For any x, y, if x is less than y then y is not less than x

From English to first-order logic

Assume the student domain and these symbols are at our disposal

Constants:	ann	for person Ann
	60332	for this course
Functions:	$grade(x, y)$	for the grade of student x in course y
Predicates:	$Student(x)$	for x is a student
	$Likes(x, y)$	for x likes y
	$x < y$	for $x < y$

- Ann is a student
- Some students don't like 60332
- Ann is the student with the best grade in 60332

Subformulae

- If F is a formula and F occurs as a part of another formula G, then F is a subformula of G.
- Example:

$$\forall x \forall y \big(\leq (x, y) \rightarrow \exists z (+(x, z) \approx y) \big)$$

Subformulae

$\leq (x, y)$ $\quad +(x, z) \approx y$ $\quad \exists z (+(x, z) \approx y)$

$\leq (x, y) \rightarrow \exists z (+(x, z) \approx y)$ $\quad \forall y \big(\leq (x, y) \rightarrow \exists z (+(x, z) \approx y) \big)$

(Are there more?)

Using brackets and notation

- We omit brackets according to the following criteria:
  - Binding precedences (from highest to lowest):

$$\neg \quad \forall x \quad \exists x$$

$$\vee \quad \wedge$$

$$\rightarrow \quad \leftrightarrow$$

  - $\vee$ and $\wedge$ are associative

$$P \wedge Q \rightarrow R \quad \text{for}$$

$$\exists x\, P(x) \wedge \forall y \neg Q(y) \quad \text{for}$$

$$P \vee Q \vee R \quad \text{for}$$

- Useful tip: When in doubt use brackets

Free & bound variables

- A quantified formula has the form $\mathcal{Q}xF$, where $\mathcal{Q} \in \{\exists, \forall\}$. x is the quantified variable and F is the scope of the quantifier $\mathcal{Q}x$.
- An *occurrence* of a variable x is bound, if it is inside the scope of a quantifier $\mathcal{Q}x$. Otherwise, it is free.

Examples of free and bound variables

$$\overbrace{\forall x \ \big(P(x) \wedge \big(\exists y \, \overbrace{P(y)}^{\text{scope of } \exists y} \rightarrow \ Q(y,x)\big)\big)}^{\text{scope of } \forall x}$$

- The two occurrences of x are both bound, as is the first occurrence of y. The second occurrence of y is free.
- Thus a variable may occur both free and bound in a formula.

$$\overbrace{\forall x \ \big(P(x) \wedge \overbrace{\exists y \, \big(\exists y \, \overbrace{P(y)}^{\text{scope of } \exists y} \rightarrow \ Q(y,x)\big)}^{\text{scope of } \exists y}\big)}^{\text{scope of } \forall x}$$

Open and closed formulae, universal closure

- Formulae without free variables are closed formulae.
  Formulae with at least one free occurrence of a variable are open formulae.

- If $x_1, \ldots, x_n$ are the free variables in F then $\forall F$ denotes the univeral closure of F. That is,

$$\forall F = \forall x_1 \ldots \forall x_n \, F.$$

  E.g., if $F = P(a, x)$ then

$$\forall F = \forall x \, P(a, x).$$

Summary

- syntax of first-order logic
  - logical symbols (fixed): logical connectives, variables
  - non-logical symbols (flexible, depends on application): signature = function symbols + predicate symbols
  - terms
  - atoms
  - formulae
- their informal meaning
- subterms, subformulae
- convention for omitting brackets
- scope of a quantifier, free and bound occurrences of variables

Substitution of terms for variables via an example

- Substitution is an operation on terms and formulae
- Substituting terms for variables means simultaneously and independently replacing the variables
- Example:

$$F = P(g(x), y, x)$$

Substituting a for x gives:

$$F\{x/a\} = P(g(a), y, a)$$
$$F\{x/a,\ y/b\} =$$
$$F\{x/a,\ y/f(z)\} =$$

Substitution of terms for variables

- Formally, a substitution is a function $\sigma : \mathcal{X} \to \mathrm{T}_\Sigma(\mathcal{X})$ such that the set

$$\mathrm{Dom}(\sigma) =^{\mathrm{def}} \{x \in \mathcal{X} \mid \sigma(x) \neq x\} \quad \text{is finite.}$$

- $\mathrm{Dom}(\sigma)$ is called the domain of σ.
- $\mathrm{Cod}(\sigma) =^{\mathrm{def}} \{\sigma(x) \mid x \in \mathrm{Dom}(\sigma)\}$ is the codomain of σ.
- The identity substitution, denoted by ϵ, is the (unique) substitution such that $\mathrm{Dom}(\epsilon) = \emptyset$. I.e., for every $x \in \mathcal{X}$, $\epsilon(x) = x$.

Substitutions

- Substitutions are often written $\{x_1/s_1, \ldots, x_n/s_n\}$, where the x_i are pairwise distinct, and defined by:

$$\{x_1/s_1, \ldots, x_n/s_n\}(y) =^{\text{def}} \begin{cases} s_i, & \text{if } y = x_i \\ y, & \text{otherwise} \end{cases}$$

- The modification $\sigma[x \mapsto t]$ of a substitution σ at x is defined like σ but it substitutes t for x.

$$\text{Formally:} \qquad \sigma[x \mapsto t](y) =^{\text{def}} \begin{cases} t, & \text{if } y = x \\ \sigma(y), & \text{otherwise} \end{cases}$$

- Alternative notation: $y\sigma$ for $\sigma(y)$, and
  $y\sigma[x \mapsto t]$ for $\sigma[x \mapsto t](y)$.

Application of a substitution to a term

- Let σ be a substitution. For each term t the substitution instance $t\sigma$ is inductively defined by:

$$x\sigma = \sigma(x)$$

$$a\sigma = a$$

$$f(s_1, \ldots, s_n)\sigma = f(s_1\sigma, \ldots, s_n\sigma)$$

- Note: $x\sigma = x$, if $x \notin \text{Dom}(\sigma)$

Important restriction for formulae

- In formulae we want to substitute a term only for a free variable.
  Example: $F = \exists x(x > y)$

$$F\{y/2\} = \exists x(x > 2)$$
$$F\{y/z\} = \exists x(x > z)$$
$$F\{y/x\} = \exists x(x > x) \qquad \text{*trouble*}$$

  y with x substituted for it becomes bound $\rightsquigarrow$ not useful

- More precisely, we don't want any variables in the substituting terms to be captured by a quantifier in the formula.
- Hence the captured variable must be renamed into a "fresh", that is, previously unused, variable z.

Application of a substitution to a formula

- For each formula F the substitution instance $F\sigma$ is inductively defined by:

$$\begin{aligned}
\bot\sigma &= \bot \qquad & \top\sigma &= \top \\
P(s_1, \ldots, s_n)\sigma &= P(s_1\sigma, \ldots, s_n\sigma) \\
(u \approx v)\sigma &= (u\sigma \approx v\sigma) \\
(\neg F)\sigma &= \neg(F\sigma) \\
(F \star G)\sigma &= (F\sigma \star G\sigma) && \text{for each binary connective } \star \\
(\mathcal{Q}x\, F)\sigma &= \mathcal{Q}z\,(F\sigma[x \mapsto z]) && \text{with } z \text{ a fresh variable}
\end{aligned}$$

- We say $E\sigma$ is formed by applying σ to E, where E is an expression (a term or formula).

Important notes

- Applying a substitution does not change constants, function symbols, predicate symbols or logical connectives.
- Components in terms/formulae are changed simultaneously and independently by σ.
- Every substitution is completely determined by its effect on variables.

Formal semantics of first-order logic

- Aim: Give formal definition of the semantics of terms and formulae of FOL.
  Goes back to Tarski.
- FOL is two-valued: As in propositional logic, the truth values are

  1 ("true") **0** ("false")
- Semantics of $\wedge$, $\vee$, $\neg$, ... are the same as in propositional logic
- We also need to:
  - interpret function symbols as functions,
  - interpret predicate symbols as predicates/relations,
  - fix some domain of elements over which these are defined,
  - constants are interpreted as elements of the domain

Interpretation of constant, function & predicate symbols

Idea: We always assume there is a particular non-empty set of objects: the domain of interpretation. The constants, function symbols and predicate symbols are interpreted over this domain.

- Let F be a formula expressed over the signature $\Sigma = (\mathcal{F}, \mathcal{P})$.
- An interpretation for F (over Σ) is a pair

$$\mathcal{I} = (U, \cdot^{\mathcal{I}}), \quad \text{where}$$

  - U is a non-empty set, called the domain of $\mathcal{I}$, and
  - $\cdot^{\mathcal{I}}$ is a function that maps
    1. each constant to an element of U,
    2. each function symbol to a function on U, and
    3. each predicate symbol to a relation on U.

Examples

- Interpretations of the formula

$$\forall x\, P(a, x)$$

  can be:

$U = \mathbb{N}$	$U = \mathbb{N}$	$U = \mathbb{N}$
$a^{\mathcal{I}} = 0$	$a^{\mathcal{I}} = 3$	$a^{\mathcal{I}} = 0$
$P^{\mathcal{I}} = \leq$	$P^{\mathcal{I}} = \leq$	$P^{\mathcal{I}} = >$

  where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers.

- In the first case $\forall x\, P(a, x)$ is interpreted as

$$\text{for each } n \in \mathbb{N},\ (0 \leq n)$$

Interpretation of variables

- Intuitively, the interpretation of variables ranges over the elements of the domain U.
  - If the domain is $\mathbb{N}$:
    $x < y$ is interpreted as $\mathbf{1}$, if $x \mapsto 1$ and $y \mapsto 2$
    $x < y$ is interpreted as $\mathbf{0}$, if $x \mapsto 3$ and $y \mapsto 2$
  - We want to interpret $\forall x$ as 'for all elements x in U' and $\exists x$ as 'there is an element x in U'.
- A variable assignment relative to $\mathcal{I} = (U, \cdot^{\mathcal{I}})$ is a function

$$\beta : \mathcal{X} \to U$$

  that assigns the variables in $\mathcal{X}$ to elements of the domain.
- Define $\beta[x \mapsto a]$ to be the assignment that is the same as β except that x is mapped to a.

Interpretation of terms

- Let $\mathcal{I}$ be an interpretation and β a variable assignment.
  A term assignment is a function $\mathcal{I}_\beta : \mathrm{T}_\Sigma(\mathcal{X}) \to U$ defined by

$$\mathcal{I}_\beta(x) = \beta(x) \quad \text{for } x \in \mathcal{X}$$
$$\mathcal{I}_\beta(a) = a^{\mathcal{I}}$$
$$\mathcal{I}_\beta(f(s_1, \ldots, s_n)) = f^{\mathcal{I}}(\mathcal{I}_\beta(s_1), \ldots, \mathcal{I}_\beta(s_n)), \quad n > 0$$

- That is, for each term s its meaning under the interpretation $\mathcal{I}$ and variable assignment β is given by $\mathcal{I}_\beta(s)$.
- $\mathcal{I}_\beta[x \mapsto a]$ is defined to be the assignment $\mathcal{I}_{\beta[x \mapsto a]}$.
- Next we extend $\mathcal{I}_\beta$ to formulae.

Interpretation of formulae

As a formula assignment $\mathcal{I}_\beta : \mathrm{F}_\Sigma(\mathcal{X}) \to \{\mathbf{1}, \mathbf{0}\}$ is defined by:

$$\begin{aligned}
\mathcal{I}_\beta(\bot) &= \mathbf{0} && \mathcal{I}_\beta(\top) = \mathbf{1} \\
\mathcal{I}_\beta(P(s_1, \ldots, s_n)) &= \mathbf{1} && \text{iff} \quad P^{\mathcal{I}}(\mathcal{I}_\beta(s_1), \ldots, \mathcal{I}_\beta(s_n)) = \mathbf{1} \\
\mathcal{I}_\beta(s \approx t) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta(s) = \mathcal{I}_\beta(t) \\
\mathcal{I}_\beta(\neg F) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{0} \\
\mathcal{I}_\beta(F \wedge G) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ and } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(F \vee G) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ or } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(F \to G) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ implies } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(F \leftrightarrow G) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta(F) = \mathbf{1} \text{ iff } \mathcal{I}_\beta(G) = \mathbf{1} \\
\mathcal{I}_\beta(\forall x F) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta[x \mapsto u](F) = \mathbf{1}, \text{ for all } u \in U \\
\mathcal{I}_\beta(\exists x F) &= \mathbf{1} && \text{iff} \quad \mathcal{I}_\beta[x \mapsto u](F) = \mathbf{1}, \text{ for some } u \in U
\end{aligned}$$

Examples

- Determine the truth or falsity of $\forall x\, P(a, x)$ in these two interpretations:

$U = \mathbb{N}$	$U = \mathbb{N}$
$a^{\mathcal{I}} = 0$	$a^{\mathcal{I}'} = 1$
$P^{\mathcal{I}} = \leq$	$P^{\mathcal{I}'} = \leq$

- In the first case,

$$\mathcal{I}_\beta(\forall x\, P(a, x)) = \mathbf{1}, \quad \text{because for all } n \in \mathbb{N},\ 0 \leq n.$$

(We have $0 \leq x$ is true under $\beta[x \mapsto 0]$, $\beta[x \mapsto 1]$, $\beta[x \mapsto 2]$, etc.)

- In the second case,

$$\mathcal{I}'_\beta(\forall x\, P(a, x)) =$$

Satisfiability, models and validity

- F is true in $\mathcal{I}$ under assignment β, if $\mathcal{I}_\beta(F) = \mathbf{1}$.
- F is satisfiable, if for some interpretation $\mathcal{I}$ and some assignment β, $\mathcal{I}_\beta(F) = \mathbf{1}$.
  Otherwise, F is unsatisfiable. Notation: $F \models \bot$
- F is true in $\mathcal{I}$, if for *every* assignment β, $\mathcal{I}_\beta(F) = \mathbf{1}$.
  We also say $\mathcal{I}$ is a model of F. Notation: $\mathcal{I} \models F$.
- F is valid, if for every interpretation $\mathcal{I}$, $\mathcal{I} \models F$. Notation: $\models F$.
- $\mathcal{I}$ is a model for a set N of formulae, if every formula in N is true in $\mathcal{I}$. Notation: $\mathcal{I} \models N$.

Semantic entailment and equivalence

Let N be a set of formulae, and F and G formulae.

- N entails F, or F semantically follows from N, if every model of N is also a model of F. Notation: $N \models F$.
- We write $G \models F$ instead of $\{G\} \models F$ and
  $\models F$ instead of $\{\} \models F$.
- F and G are semantically equivalent, if $F \models G$ and $G \models F$.
  Notation: $F \equiv G$.

Entailment, equivalence and validity

- Let $N = \{F_1, \ldots, F_n\}$. For *closed* formulae:

  Property 1

  1. $F \equiv G$ iff $\models F \leftrightarrow G$ (F and G are equiv. iff $F \leftrightarrow G$ is valid)
  2. $F \models G$ iff $\models F \rightarrow G$ (F entails G iff $F \rightarrow G$ is valid)

  Property 2 (Deduction theorem)

  $N \models F$ iff $\models (F_1 \wedge \ldots \wedge F_n) \rightarrow F$

- Aside: In general Property 3

  1. $F \equiv G$ iff $\models \forall F \leftrightarrow \forall G$
  2. $F \models G$ iff $\models \forall F \rightarrow \forall G$

  Property 4 (Deduction theorem)

  $N \models F$ iff $\models (\forall F_1 \wedge \ldots \wedge \forall F_n) \rightarrow \forall F$

  $\forall F$ is the univeral closure of F; e.g., if $F = P(x)$ then $\forall F = \forall x\, P(x)$

Duality between validity and satisfiability

Recall:

- F is satisfiable, if *for some interpr.* $\mathcal{I}$ *and some* β, $\mathcal{I}_\beta(F) = \mathbf{1}$.
- F is unsatisfiable, if *for all interpr.* $\mathcal{I}$ *and all* β, $\mathcal{I}_\beta(F) = \mathbf{0}$.
- F is valid, if *for every interpretation* $\mathcal{I}$, $\mathcal{I} \models F$.

Property 5

1. F is valid iff $\neg F$ is unsatisfiable.
2. $N \models F$ iff $N \cup \{\neg F\}$ is unsatisfiable.

- Validity and unsatisfiability can be interreduced.
- Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for (un)satisfiability.

Previously ...

- syntax of first-order logic
- substitutions
  - $\{x_1/s_1, \ldots, x_n/s_n\}$
- semantics of first-order logic (standard interpretations)
- semantic entailment, semantic equivalence

- reasoning for propositional logic using resolution
- resolution calculi operate on clauses
- conversion to clausal form for propositional formulae

Normal forms

Recall: $\rightarrow$ is 'superfluous' since it can be expressed using $\neg$ and $\vee$

$$\begin{aligned} A \rightarrow B &\equiv \neg A \vee B, \\ &\equiv \neg(A \wedge \neg B), \\ &\equiv \neg\neg(A \rightarrow B), \\ &\text{etc} \end{aligned}$$

Problem: Formulas have *very many* equivalent forms

Solution: Transform formulas to normal form

- Simplifies formulas: formulas limited to fixed simple patterns
- Easier to define and determine truth
- Easier development of efficient automated reasoning tools, without penalty

Main problem in first-order logic

- . . . is the treatment of quantifiers.
- Solution: Use additional normal form transformations to eliminate the quantifiers
  - Prenxe normal form transformation
  - Skolemisation

Prenex Normal Form

- A formula is in prenex normal form (PNF) if it is in the form

$$\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n \; F,$$

  where F is a quantifier-free formula and $\mathcal{Q}_i \in \{\forall, \exists\}$
- $\mathcal{Q}_1 x_1 \ldots \mathcal{Q}_n x_n$ is called the quantifier prefix and F the matrix of the formula.

Useful semantic equivalences involving $\forall$, $\exists$

Let F and G be any closed formulae. Then

1. $\models \neg\forall x F \leftrightarrow \exists x \neg F$

   $\models \neg\exists x F \leftrightarrow \forall x \neg F$

2. $\models \forall x \forall y F \leftrightarrow \forall y \forall x F$

   $\models \exists x \exists y F \leftrightarrow \exists y \exists x F$

   But only: $\models \exists x \forall y F \rightarrow \forall y \exists x F$

3. $\models \forall x(F \wedge G) \leftrightarrow (\forall x F) \wedge (\forall x G)$

   $\models \exists x(F \vee G) \leftrightarrow (\exists x F) \vee (\exists x G)$

   But: $\not\models \forall x(F \vee G) \leftrightarrow (\forall x F) \vee (\forall x G)$

   $\not\models \exists x(F \wedge G) \leftrightarrow (\exists x F) \wedge (\exists x G)$

Useful semantic equivalences involving $\forall$, $\exists$ (cont'd)

4. If x does not occur freely in G then

   $\models \forall x(F \star G) \leftrightarrow (\forall x F \star G)$ for $\star \in \{\wedge, \vee\}$

   $\models \exists x(F \star G) \leftrightarrow (\exists x F \star G)$ for $\star \in \{\wedge, \vee\}$

5. If x does not occur freely in G then

   $\models \forall x(F \rightarrow G) \leftrightarrow (\exists x F \rightarrow G)$

   $\models \exists x(F \rightarrow G) \leftrightarrow (\forall x F \rightarrow G)$

   $\models \forall x(G \rightarrow F) \leftrightarrow (G \rightarrow \forall x F)$

   $\models \exists x(G \rightarrow F) \leftrightarrow (G \rightarrow \exists x F)$

Useful semantic equivalences involving $\forall$, $\exists$ (cont'd)

6. $\models \forall x F \leftrightarrow \forall y F\{x/y\}$ where y is a fresh variable
   $\models \exists x F \leftrightarrow \exists y F\{x/y\}$ where y is a fresh variable
7. $\models (\forall x F) \vee (\forall x G) \leftrightarrow \forall x \forall y (F \vee G\{x/y\})$
   where y is a fresh variable
   $\models (\exists x F) \wedge (\exists x G) \leftrightarrow \exists x \exists y (F \wedge G\{x/y\})$
   where y is a fresh variable

Computing prenex normal form

- The prenex normal form $\mathrm{PNF}(F)$ of a formula F can be computed by applying these rewrite rules:

$$
\begin{array}{rcll}
(F \leftrightarrow G) & \Rightarrow_{\mathrm{PNF}} & (F \rightarrow G) \wedge (G \rightarrow F) & \\
\neg \mathcal{Q} x F & \Rightarrow_{\mathrm{PNF}} & \overline{\mathcal{Q}} x \neg F & (\neg \mathcal{Q}) \\
(\mathcal{Q} x F \star G) & \Rightarrow_{\mathrm{PNF}} & \mathcal{Q} y (F\{x/y\} \star G), & \star \in \{\wedge, \vee\} \\
(F \star \mathcal{Q} x G) & \Rightarrow_{\mathrm{PNF}} & \mathcal{Q} y (F \star G\{x/y\}), & \star \in \{\wedge, \vee, \rightarrow\} \\
(\mathcal{Q} x F \rightarrow G) & \Rightarrow_{\mathrm{PNF}} & \overline{\mathcal{Q}} y (F\{x/y\} \rightarrow G), &
\end{array}
$$

  In the last three lines y must be a fresh variable in each case.
- $\overline{\mathcal{Q}}$ denotes the quantifier dual to $\mathcal{Q}$, i.e., $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$.
- The rules can be applied in any order

Exercise

- Obtain the prenex normal form for the formula

$$\exists x \forall y \big(\exists z(P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)\big).$$

$$\exists x \forall y \big(\exists z(P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)\big)$$

$$\Rightarrow_{\mathsf{PNF}}$$

Eliminating $\exists$ quantifiers using Skolemisation

- Transformation $\Rightarrow_{\mathsf{Sk}}$

$$\forall x_1 \ldots \forall x_n \exists y F \quad \Rightarrow_{\mathsf{Sk}} \quad \forall x_1 \ldots \forall x_n F\{y/f(x_1,\ldots,x_n)\}$$

  where f/n is a fresh function symbol.

- f is called a Skolem function.
  $f(x_1,\ldots,x_n)$ is called a Skolem term, or Skolem constant when $n = 0$.
- Example:

$$\forall x \exists y P(x,y) \;\Rightarrow_{\mathsf{Sk}}\; \forall x P(x, f(x))$$

- Intuition: f is choice function computing y from all the arguments that y depends on.
- **Note:** Always apply outermost first, not in subformulae

Exercise

- Apply Skolemisation to this formula

$$\exists x \forall y \forall z \exists u \forall v \, P(x, y, z, u, v)$$

Applying transformation to PNF and Skolemising

Together: $F \Rightarrow^*_{\text{PNF}} \underbrace{G}_{\text{PNF}} \Rightarrow^*_{\text{Sk}} \underbrace{H}_{\text{PNF , no } \exists}$

Property 6

Let F, G, and H be as above and closed. Then

(i) F and G are equivalent.

(ii) $H \models G$, but the converse is not true in general.

(iii) G is satisfiable iff H is satisfiable

In the last case: In fact, G is satisfiable in an interpretation $\mathcal{I}$ iff H is satisfiable in an extenstion $\mathcal{I}'$ of $\mathcal{I}$.
$\mathcal{I}$ is an interpr. over $\Sigma = (\mathcal{F}, \mathcal{P})$, while
$\mathcal{I}'$ is an interpr. over $\Sigma' = (\mathcal{F} \cup SKF, \mathcal{P})$.

Recall from week 1–2: Literals, clauses

- Literals

$$\begin{array}{rcll} L & \longrightarrow & A & \text{(atom, positive literal)} \\ & | & \neg A & \text{(negative literal)} \end{array}$$

- Clauses

$$\begin{array}{rcll} C, D & \longrightarrow & \bot & \text{(empty clause)} \\ & | & L_1 \vee \ldots \vee L_k, \ k \geq 1 & \text{(non-empty clause)} \end{array}$$

- **Important assumptions:**
  - $\vee$ is associative and commutative, repetitions matter.
    I.e. we regard clauses as multi-sets of literals
  - Thus, $C = P \vee P \vee \neg Q$ is identical to $C' = P \vee \neg Q \vee P$.
    But neither C nor C' are the same as $D = P \vee \neg Q$.

Transformation to conjunctive normal form

- The conjunctive normal form $\mathrm{CNF}(F)$ of a formula F can be computed by persistently applying these rewrite rules:

$$\begin{array}{rcl} F \leftrightarrow G & \Rightarrow_{\mathrm{CNF}} & (F \rightarrow G) \wedge (G \rightarrow F) \\ F \rightarrow G & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee G) \\ \neg(F \vee G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \wedge \neg G) \\ \neg(F \wedge G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee \neg G) \\ \neg\neg F & \Rightarrow_{\mathrm{CNF}} & F \\ (F \wedge G) \vee H & \Rightarrow_{\mathrm{CNF}} & (F \vee H) \wedge (G \vee H) \end{array}$$

$$\begin{array}{rcl@{\qquad}rcl} F \wedge \top & \Rightarrow_{\mathrm{CNF}} & F & F \wedge \bot & \Rightarrow_{\mathrm{CNF}} & \bot \\ F \vee \top & \Rightarrow_{\mathrm{CNF}} & \top & F \vee \bot & \Rightarrow_{\mathrm{CNF}} & F \\ \neg\top & \Rightarrow_{\mathrm{CNF}} & \bot & \neg\bot & \Rightarrow_{\mathrm{CNF}} & \top \end{array}$$

- These rules are to be applied modulo associativity and commutativity of $\wedge$ and $\vee$.

Computing the clausal form of a first-order formula

$$
\begin{array}{lll}
F & \Rightarrow^*_{\text{PNF}} & \mathcal{Q}_1 y_1 \ldots \mathcal{Q}_n y_n \; G \qquad (G \text{ quantifier-free}) \\
& \Rightarrow^*_{\text{Sk}} & \forall x_1 \ldots \forall x_m \; H \qquad (m \leq n,\ H \text{ quantifier-free}) \\
& \Rightarrow^*_{\text{CNF}} & \underbrace{\underbrace{\forall x_1 \ldots \forall x_m}_{\text{leave out}} \; \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clause } C_i}}_{F'} \\
& \Rightarrow & \{C_1, \ldots, C_k\}
\end{array}
$$

- $N = \{C_1, \ldots, C_k\}$ is called the clausal (normal) form of F.
- **Note:** the variables in the clauses are implicitly universally quantified.

Sample transformation to clausal form

- Given formula:
  $$\exists x \big[\forall y \, (R(x,y) \rightarrow (\neg P(y) \vee \exists z \, (R(y,z) \wedge P(z))))\big]$$
- Prenex normal form:
  $$\exists x \forall y \exists z \, \big[R(x,y) \rightarrow (\neg P(y) \vee (R(y,z) \wedge P(z)))\big]$$
- Skolemisation:
  $$\forall y \, \big[R(a,y) \rightarrow (\neg P(y) \vee (R(y,f(y)) \wedge P(f(y))))\big]$$
  a: Sk. const. for $\exists x$; $f(y)$: Sk. term for $\exists z$
- CNF:
  $$\forall y \, \big[(\neg R(a,y) \vee \neg P(y) \vee R(y,f(y))) \wedge (\neg R(a,y) \vee \neg P(y) \vee P(f(y)))\big]$$
- Clausal form: drop $\forall$, $\wedge$ and outer brackets
  $$\neg R(a,y) \vee \neg P(y) \vee R(y,f(y))$$
  $$\neg R(a,y) \vee \neg P(y) \vee P(f(y))$$

Properties of CNFs and clausal forms

Property 7

For every formula F:

$$\text{If } \; F \Rightarrow^*_{\mathsf{CNF}} F' \; \text{ then } \; F \equiv F'.$$

Property 8

Let F be closed. Suppose $F \Rightarrow^*_{\mathsf{PNF}} \circ \Rightarrow^*_{\mathsf{Sk}} \circ \Rightarrow^*_{\mathsf{CNF}} F'$ and N is clausification of F'.

$$\text{Then } \; F' \models F \; \text{ and } \; N \models F.$$

The converses are not true in general. But:

Property 9

Let F be closed. Then

F is satisfiable iff F' is satisfiable iff N is satisfiable

Optimising the transformation to clausal form

- Issues:
  - Size of the CNF can be exponential;
  - Want to preserve the original formula structure;
  - Want Skolem functions with small arity.
- These can all be addessed since we can/need to preserve only satisfiability anyway $\rightsquigarrow$ lots of room for optimisation
- The last point can be addressed with miniscoping of quantifiers (essentially moving quantifiers inwards) and a better form of $\Rightarrow_{\mathsf{Sk}}$ (not discussed)
- The first two points can be addressed with structural transformation (idea similar as for propositional logic; not discussed)

Previously ...

- Every FO formula F can be transformed into a set N of clauses so that

$$F \text{ is satisfiable} \quad \text{iff} \quad N \text{ is satisfiable.}$$

- We will extend resolution to first-order clauses

$$\neg R(a, y) \vee \neg P(y) \vee R(y, f(y))$$
$$\neg R(a, y) \vee \neg P(y) \vee P(f(y))$$

Herbrand semantics for first-order clauses

- Problem with classical semantics: There are soooo many ways to define interpretations
- Herbrand interpretations are a special interpretations that allow for a very simple definition and analysis of the truth of clauses.

- Key idea of Herbrand's theorem:
  It suffices that terms are interpreted as themselves.
  For establishing the truth of clauses it suffices to consider only Herbrand interpretations.

Ground expressions, ground instances

- Ground terms are terms with no occurrences of variables.
- Ground atoms are atoms with no occurrences of variables.
- Ground literals, ground clauses, ground formulae are defined similarly.
- A ground instance of an expression (term, atom, literal, clause, formula) is obtained by uniformly substituting the variables in it with ground terms.

Herbrand universe

- Herbrand semantics allows us to fix a special domain s.t. if F is unsatisfiable, then every truth assignment over this special domain is false.
- The Herbrand universe is T_Σ, i.e., the set of all ground terms over the signature $\Sigma = (\mathcal{F}, \mathcal{P})$.
- Example: Suppose $\mathcal{F}$ has one binary function symbol f and two constants a and b. Herbrand universe over Σ:

$$a,\ b,\ f(a,a),\ f(a,b),\ f(b,a),\ f(b,b),\ f(a,f(a,a)),\ \ldots$$

- If Σ contains non-constant function symbols then T_Σ is infinite.
- Important assumption: There is at least one constant in the signature Σ.

Exercise

- Suppose Σ is a signature with one unary function symbol f and one constant a. I.e. $\mathcal{F} = \{f/1, a/0\}$.
  Write down the elements of the Herbrand universe T_Σ.

Herbrand interpretations

- A Herbrand interpretation, denoted I, is a set of ground atoms over Σ.
- Truth in I of ground formulae is defined inductively by:

$$I \models \top \qquad\qquad I \not\models \bot$$

$$(*) \qquad I \models A \text{ iff } A \in I, \text{ for any ground atom } A$$

$$I \models \neg F \text{ iff } I \not\models F$$

$$I \models F \wedge G \text{ iff } I \models F \text{ and } I \models G$$

$$I \models F \vee G \text{ iff } I \models F \text{ or } I \models G$$

- Note: $(*)$ is equivalent to $I \not\models A$ iff $A \notin I$
- This means: $A \notin I$ implies A is false in I, which implies $I \models \neg A$

Herbrand interpretations (cont'd)

- Truth in I of any quantifier-free formula F with free variables $x_1, \ldots, x_n$ is defined by:

$$I \models F(x_1, \ldots, x_n) \text{ iff } I \models F(t_1, \ldots, t_n), \text{ for every } t_i \in \mathrm{T}_\Sigma$$

- Truth in I of any set N of clauses/quantifier-free formulae is defined by:

$$I \models N \quad \text{iff} \quad I \models C, \text{ for each } C \in N$$

- A Herbrand interpretation I is called a Herbrand model of F, if $I \models F$.

Exercise

- Suppose $\mathcal{F} = \{f/1, a/0\}$ and $\mathcal{P} = \{P/1\}$.
  Which of the following are Herbrand interpretations over Σ?
  1. $I_1 = \{P(a)\}$
  2. $I_2 = \{P(a),\ P(f(a))\}$
  3. $I_3 = \{P(a),\ \neg P(f(a))\}$
- For I_2 determine whether the following is true?
  1. $I_2 \models P(a)$
  2. $I_2 \models \neg P(a)$
  3. $I_2 \models \neg P(f(a))$
  4. $I_2 \models P(a) \wedge P(f(a))$
  5. $I_2 \models P(x)$

Examples of truth in Herbrand interpretations

Suppose Σ is any signature. Let I be a Herbrand interpretation over Σ.

- $I \models P(x)$ iff $P(t) \in I$ for every ground term $t \in \mathrm{T}_\Sigma$.
- $I \models P(x) \vee Q(x)$ iff for every ground term $t \in \mathrm{T}_\Sigma$ $P(t) \in I$ or $Q(t) \in I$.
- $I \models \neg S(x, a)$ iff $S(t, a) \notin I$ for every ground term t in T_Σ.
- $I \models \neg R(x, y) \vee R(y, x)$ iff if $R(s, t) \in I$ then $R(t, s) \in I$, for any ground terms $s, t \in \mathrm{T}_\Sigma$.

Relation to standard interpretations (aside)

- In a Herbrand interpretation values are fixed to be ground terms and functions are fixed to be the (Skolem) functions in Σ.
- Let I be a Herbrand interpretation over Σ with domain T_Σ.
- Let $\mathcal{I} = (U, \cdot^{\mathcal{I}})$ be a standard interpretation where $U = \mathrm{T}_\Sigma$ and the interpretation function $\cdot^{\mathcal{I}}$ maps terms to themselves, i.e.:

  constant a: $a^{\mathcal{I}} = a$

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

  Only predicate symbols P may be freely interpreted as relations $P^{\mathcal{I}}$ over T_Σ.

Property 10

Every Herbrand interpretation I (set of ground atoms) uniquely determines a standard interpretation $\mathcal{I}$ via

$$P^{\mathcal{I}}(s_1, \ldots, s_n) = \mathbf{1} \text{ iff } P(s_1, \ldots, s_n) \in I$$

The set of all ground instances $G_\Sigma(N)$

- Let N be a set clauses over the signature Σ and suppose $\mathcal{X}$ denotes the set of variables in N. Define

  $$G_\Sigma(N) = \{C\sigma \mid C \in N, \sigma : \mathcal{X} \to \mathrm{T}_\Sigma \text{ a ground substitution}\}$$

  $G_\Sigma(N)$ is the set of all ground instances of the clauses in N.

- Example: $N = \{P(x),\ Q(f(y)) \vee R(y)\}$

  $T_\Sigma = \{a, f(a), f(f(a)), \ldots\}$

  - $P(a)$ and $P(f(a))$ are both ground instances of the first clause $P(x)$ in N.
  - **Exercise**: Give examples of ground instances of the second clause in N.

Existence of Herbrand models

Property 11 (Herbrand)

Let N be a set of Σ-clauses. Then

N is true in a standard interpretation

iff N has a Herbrand model (over Σ)

iff $G_\Sigma(N)$ has a Herbrand model (over Σ)

- Many theorem proving approaches exploit this property, e.g., approaches based on instantiation, e.g., tableau approaches, Inst-Gen, but also resolution.
- We use it in the completeness proof of the resolution calculus.

Using Herbrand's theorem to find a model

- Suppose the signature is based on $\mathcal{F} = \{1/0, +/2\}$ and $\mathcal{P} = \{P/1\}$. Is the following set satisfiable?

$$N = \{\ P(1),\ \neg P(x) \vee P(x+1)\ \}$$

- One obtains the following ground instances:

$$\begin{aligned} G_\Sigma(N) = \{\ & P(1), \\ & \neg P(1) \vee P(1+1), \\ & \neg P(1+1) \vee P(1+1+1), \\ & \neg P(1+1+1) \vee P(1+1+1+1), \\ & \cdots \\ & \} \end{aligned}$$

Exercise

- Write down a Herbrand model of $G_\Sigma(N)$.
  I.e. write down a Herbrand interpretation in which all clauses of $G_\Sigma(N)$ are true.

- Is N satisfiable?

Summary

- ground expressions, ground instances
- Herbrand universe
- Herbrand interpretation, Herbrand model
- truth in I: $\models$
- Herbrand's theorem