Logic and Discrete Structures - LDS

Course 11 – First order logic

s.l. dr. ing. Cătălin Iapă

catalin.iapa@cs.upt.ro

Predicate logic - syntax

Formalization of natural language

Resolution

Proofs in Predicate Logic

Semantics in Predicate Logic

Logic: review

We use logic to rigorously express (formalize) reasoning.

Logic allows us to make demonstrations (inferences)

- from axioms (always true)
- and hypotheses (considered true in the given problem)
- using rules of inference (deduction)

$$\frac{p \qquad p \rightarrow q}{q}$$

modus ponens

Propositional logic cannot express everything

A classic example: (1) All humans are mortal.
(2) Socrates is human.
(3) So, (3) Socrates is mortal.

This is a syllogism (pattern of inference rule)

classical logic: Aristotle, Stoics

It looks like modus ponens

- but the premise in (1) ("all men")
- is not the same as (2) (Socrates, a certain man)

We could rephrase (1): If X is human, then X is mortal.
more precisely: For any X, if X is human, then X is mortal.

Modern logic: predicate logic (first-order logic)

Gottlob Frege, Charles Peirce (19th century)

We need more expressive formulas

Formulas consist of predicates linked by logical connectors

$\forall x\, ((folder(x) \wedge x \neq root) \rightarrow contains(parent(x), x))$

Instead of propositions (a, p, q) we have predicates : *file*(*x*), *contains*(*x*, *y*)

A predicate = a statement relative to one or more variables, which, by giving values to variables, can take the value true or false.

Predicates have arguments terms: variables x / functions: parent(x)
intuitive: represent objects/notions and functions in the universe

New: quantifiers appear: ∀ (any), ∃ (exist)

Define first-order logic

Syntax of predicate logic: Terms

We define, structurally recursively, the notions of term and formula:

Terms

- variable v

$f\,(t_1, \cdot \cdot \cdot, t_n)$ with f n-ary function and $t_1, \cdot \cdot \cdot, t_n$ *terms*

Exemple: *parent*(*x*), *cmmdc*(*x*, *y*), max(min(*x*, *y*), *z*)

- constant c: special case, function of zero arguments

Syntax of predicate logic: formulas

Formulas (well-formed formulas):

- $P(t_1, \cdot\cdot\cdot, t_n)$ with P predicate of n arg. And $t_1, \cdot\cdot\cdot, t_n$ *terms*

  Exemple: *contains*(*empty*, *x*), *divide*(*cmmdc*(*x*, *y*), *x*)

  - proposition p: particular case, predicate of zero arguments

$\neg\alpha$ where α is a formula

$\alpha \rightarrow \beta$ with α, β formulas

$\forall v\, \alpha$ with variable v, α formula: universal quantification

Exemple: $\forall x \neg$*contains*(*empty*, *x*), $\forall x \forall y$ *divide*(*cmmdc*(*x*, *y*), *x*)

$t_1 = t_2$ with t1, t2 terms (in first-order logic with equality)

Exemple: min(*x*, min(*y*, *z*)) = min(min(*x*, *y*), *z*)

About quantifiers. Existential quantifier ∃

Denote: $\exists x\phi \overset{def}{=} \neg\forall x\,(\neg\phi)$ ϕ - a formula

There are x for which ϕ is true ↔ not for every x ϕ is false. The two quantifiers are dual. We can also write $\forall x\phi = \neg\exists x\,(\neg\phi)$

The quantifiers have higher precedence than the connectives ¬, ∧, →

⇒ if the quantified formula has ∧, ∨, → we use parentheses:

$\exists x\,(P(x) \rightarrow Q(x))$ $\forall y\,(Q(y) \wedge R(x, y))$

Other notation: dot . quantifier applies to all the rest of the formula, up to the end or closed parenthesis

$P(x) \vee \forall y.Q(y) \wedge R(x, y)$ $(R(y) \vee \exists x.P(x) \rightarrow Q(x)) \wedge S(x)$

Distributivity of quantifiers to $\wedge$ and $\vee$

The universal quantifier is distributive to the conjunction:

$$\forall x\, (P(x) \wedge Q(x)) \leftrightarrow \forall x\, P(x) \wedge \forall x\, Q(x)$$

but the existential quantifier is NOT distributive to the conjunction:

$$\exists x\, (P(x) \wedge Q(x)) \not\leftrightarrow (\exists x\, P(x) \wedge \exists x\, Q(x))$$

we have implication $\rightarrow$, but not the converse, it may not be the same x !

Dual, $\exists$ is distributive to disjunction:

$$\exists x\, P(x) \vee \exists x\, Q(x) \leftrightarrow \exists x.P(x) \vee Q(x)$$

$\forall$ is NOT distributive to disjunction. We just have:

$$\forall x\, P(x) \vee \forall x\, Q(x) \rightarrow \forall x.P(x) \vee Q(x)$$

Predicate logic - syntax

Formalization of natural language

Resolution

Proofs in Predicate Logic

Semantics in Predicate Logic

Formalising natural language

Formulas contain: variables, functions, predicates.

- Verbs become predicates (as in natural language):

  buys(X , Y), subtracts(X),

- Subject and (in)direct complements: predicate arguments
- Attributes (properties) become predicates about argument-values

  glad(X), golden(Y)

Variables in formulas can take values of any kind from the universe

- Categories also become predicates, with object argument of that kind

  child (*X*), *notebook*(*X*)

- Single entities become constants:

  mary, *emptyset*, *santaclaus*

Example of formalisation (1)

1. Each investor bought shares or bonds.

Quantifiers introduce variables with arbitrary values from the universe

⇒ impose categories by additional predicates

⇒ introduce a predicate inv (X) (X is investor)

For any X, if X is investor, it has done something

$$\forall X.inv(X) \rightarrow \boxed{\text{what } X \text{ does}}$$

What does it say about the investor? Is there anything he bought

$$\forall X.inv(X) \rightarrow \exists C.\ \text{bought}(X, C) \wedge \boxed{\text{what we know about } C}$$

$$\forall X.inv(X) \rightarrow \exists C.\ \text{bought}(X, C) \wedge (shares(C) \vee bonds(C))$$

Sursa: http://www.cs.utexas.edu/users/novak/reso.html

Example of formalisation (2)

2. If the Dow Jones falls, all shares except gold fall.

The Dow Jones index is a single notion ⇒ we use a constant dj
alternatively: we could also use a sentence falldj

$$fall(dj) \rightarrow \boxed{\text{what happens}}$$

$$fall(dj) \rightarrow \forall X. \boxed{\textit{conditions for } X} \rightarrow fall(X)$$

$$fall(dj) \rightarrow \forall X.\ shares(X) \wedge \neg gold(X) \rightarrow fall(X)$$

Example of formalisation (3)

3. *If the Treasury* increases *interest, all bonds fall.*

$$increasesinterest \rightarrow \forall X.bonds(X) \rightarrow fall(X)$$

Interest is the only thing in the problem that increases ⇒ alternative sentence: a constant interest + predicate increases

increases(interest)

Example of formalisation (4)

4. Any investor who bought something that decreases is not happy.

$$\forall X.inv(X) \rightarrow \boxed{what\ we\ know\ about\ X}$$

$$\forall X.inv(X) \rightarrow \boxed{condition\ for\ X} \rightarrow \neg happy(X))$$

$$\forall X.inv(X) \rightarrow (\exists C.bought(X, C) \wedge decreases(C)) \rightarrow \neg happy(X)$$

Example of formalisation (5)

5. If the Dow Jones index falls and the Treasury raises interest rates, all the happy investors have bought a few shares of gold.

$fall(dj) \wedge increasesinterest \rightarrow$ what happens

$fall(dj) \wedge increasesinterest \rightarrow$
$\forall X.inv(X) \wedge happy(X) \rightarrow$ what we know about X

$$fall(dj) \wedge increasesinterest \rightarrow \forall X.inv(X) \wedge happy(X) \rightarrow \exists C.bought(X, C) \wedge shares(C) \wedge gold(C)$$

Beware of quantifiers!

The universal quantifier ("all") quantifies an implication: All students are young people

student ⊆ *young*

$$\forall x. student(x) \rightarrow young(x)$$

Common error: ∧ instead of →: ~~$\forall x. student(x) \land young(x)$~~

Anyone/anyone in the universe is both a student and a young!

The existential quantifier ("some", "exists") quantifies a conjunction.

There are student prize winners.

Winner ∩ *Student* ≠ ∅

$$\exists x. winner(x) \land student(x)$$

Common error: → instead of ∧: ~~∃x.winner(x) → student(x)~~

It's true if there is a non-premium! (false implies anything)

Predicate logic - syntax

Formalization of natural language

Resolution

Proofs in Predicate Logic

Semantics in Predicate Logic

After translation into logic, we can prove it!

Having an infinity of interpretations (values in the universe, functions, values for relations/predicates), we cannot write truth tables.

But we can make demonstrations (inferences) according to rules of inference (purely syntactic), as in propositional logic.

Demonstration by resolution method

A formula is valid if and only if its negation is a contradiction.

We can prove a theorem by indirect proof (reduction to the absurd) by showing that its negation is an (unfeasible) contradiction.

Let the hypotheses A1, A2, . . . , An and conclusion C .

Let the theorem $A_1 \wedge A_2 \ldots \wedge A_n \rightarrow C$

i.e.: assumptions $A_1, A_2, \ldots$ together imply conclusion C

Negation of implication: $\neg(H \rightarrow C) = \neg(\neg H \vee C) = H \wedge \neg C$

So we show that $A_1 \wedge A_2 \ldots \wedge A_n \wedge \neg C$ is a contradiction (indirect proof: true hypothesis + false conclusion is impossible)

We show that a formula is a contradiction by the resolution method.

Resolution in propositional calculus

Resolution is a rule of inference that produces a new clause from two clauses with complementary literals (p and ¬p).

$$\frac{p \lor A \qquad \neg p \lor B}{A \lor B} \quad \textit{resolution}$$

"From clauses p ∨ A and ¬p ∨ B we deduce/derive clause A ∨ B"

Clause obtained = resolvent of the two clauses with regard to p

Exemple: $rez_p\,(p \lor q \lor \neg r, \neg p \lor s) = q \lor \neg r \lor s$

Modus ponens can be seen as a particular case of resolution:

$$\frac{p \lor \textit{false} \qquad \neg p \lor q}{\textit{false} \lor q}$$

Resolution example (1)

$(a \vee \neg b \vee \neg d)$ — *b* negated
$\wedge (\neg a \vee \neg b)$ — *b* negated
$\wedge (\neg a \vee c \vee \neg d)$
$\wedge (\neg a \vee b \vee c)$ — *b* positive

We take a sentence with both polarities (b) and construct solvers

$rez_b(a \vee \neg b \vee \neg d,\ \neg a \vee b \vee c) = a \vee \neg d \vee \neg a \vee c = T$
$rez_b(\neg a \vee \neg b,\ \neg a \vee b \vee c) = \neg a \vee \neg a \vee c = \neg a \vee c$

We add the new solvers (ignore T); we remove the old clauses with b

$(\neg a \vee c \vee \neg d)$
$\wedge (\neg a \vee c)$

We can no longer create solvers. We have no empty clause.⇒ the formula is satisfiable, e.g. with a = F. Or with c = T.

Resolution example (2)

$$\begin{aligned} & a \\ \wedge\ & (\neg a \vee b) \\ \wedge\ & (\neg b \vee c) \\ \wedge\ & (\neg a \vee \neg b \vee \neg c) \end{aligned}$$

c positive
c negated

We apply the resolution after c, we have one pair of clauses:
$rez_c(\neg b \vee c,\ \neg a \vee \neg b \vee \neg c) = \neg b \vee \neg a \vee \neg b = \neg a \vee \neg b$

We remove the two clauses with c and add the new clause:

$$\begin{aligned} & a \\ \wedge\ & (\neg a \vee b) \\ \wedge\ & (\neg a \vee \neg b) \end{aligned}$$

We apply the resolution to b:
$rez_b(\neg a \vee b, \neg a \vee \neg b) = \neg a \vee \neg a = \neg a$

Remove the two b clauses, add the new clause:

$$\begin{aligned} & a \\ \wedge\ & \neg a \end{aligned}$$

We apply the resolution after a: reza(a, ¬a) = F (empty clause)

So the original formula is a contradiction (it's infeasible).

Applying resolution in propositional calculus

Starting from a formula in conjunctive normal form (CNF), we add resolvents, trying to get the empty clause:

We choose a sentence p and add all resolvents relative to p: from m clauses with p and n clauses with ¬p, we create m - n resolvents we have removed p ⇒ we delete the original m+n clauses

If any solver is empty clause, the formula is infeasible
If we can't create any more resolvers (literals have single polarity), the formula is satisfiable (make T all remaining literals)

The number of clauses can increase exponentially (problematic!)

Resolution: from sentences to predicates

In predicate logic, a literal is not a sentence, but a predicate
not just p and ¬p, but P(arg 1) and ¬P(arg 2) (different arguments)

To derive a new clause from A ∨ P(arg 1) and B ∨ ¬P(arg 2)
we must try to bring the arguments to a common expression.

We will have clauses with universal quantified default variables can take any value ⇒ we can substitute them with terms

Substitutions and mergers of terms

A substitution is a function that associates terms with variables:

{x1 '→ t1, . . . , xn '→ tn}

Two terms can be unified if there is a substitution that makes them equal

f (x, g (y, z), t){x '→ h(z), y '→ h(b), t '→ u} = f (h(z), g (h(b), z), u)

Unification rules

A variable x can be unified with any term t (substitution) if x does not appear in t (otherwise, substituting gives an infinite term)

so no: x with f (h(y), g (x, z))

Two terms f (...) can be unified only if they have the same function, and the arguments (terms) can be unified (position by position)

Two constants (functions with 0 arg.) ⇒ unified if they are identical

Resolution in predicate calculus

Either clauses: A with positive P(...) and B with ¬P(...) (negated)
Example:

A: $P(x, g(y)) \vee P(h(a), z) \vee Q(z)$

B: $\neg P(h(z), t) \vee R(t, z)$

We choose some (≥ 1) P(...) from A and some ¬P(...) from B.

Rename common variables (not related between A and B)

A: $P(x, g(y)) \vee P(h(a), z) \vee Q(z)$ B: $\neg P(h(z2), t) \vee R(t, z2)$

We unify (all at once) only those P(...) in A and ¬P(...) in B chosen

$\{P(x, g(y)), P(h(a), z), P(h(z2), t)\}$ x '→ h(a); z2 '→ a; z, t '→ g (y)

We eliminate P(...) and ¬P(...) chosen from A ∨ B. We apply the substitution resulting from unification and add the new clause to the list of clauses.

$$Q(g(y)) \vee R(g(y), a)$$

We keep the original clauses, they can be used with other predicate choices.

Resolution: in conclusion

Repeatedly generate new clauses (resolvers) by resolution with merging

If repeating yields the empty clause, the original formula is infeasible.

If we find no new resolvers, the original formula is satisfiable.

Recall: we started by trying to prove

$$A1 \land A2 \land \ldots \land An \rightarrow C$$

by reduction to the absurd, denying the conclusion and showing that

$A1 \land A2 \land \ldots \land An \land \neg C$ is a contradiction

The method of resolution is complete relative to the refutation for any non-realizable formula, will arrive at the empty clause but cannot determine the realizability of any formula

(there are formulas for which it runs to infinity)

Example of application of the resolution

We resume the exercise formalised above.
We use () and not . to avoid mistakes when applying quantification.

A_1: $\forall X\,(inv\,(X) \rightarrow \exists\, C\,(cump(X, C) \wedge (act(C) \vee oblig\,(C))))$

A_2: $scadedj \rightarrow \forall\ X\,(act(X) \wedge \neg aur\,(X) \rightarrow scade(X))$

A_3: $creștedob \rightarrow \forall\ X\,(oblig\,(X) \rightarrow scade(X))$

A_4: $\forall X\,(inv\,(X) \rightarrow (\exists\, C\,(cump(X, C) \wedge scade(C)) \rightarrow \neg bucur\,(X)))$

C : $scadedj \wedge creștedob \rightarrow$
$\forall\ X\,(inv\,(X) \wedge bucur\,(X) \rightarrow \exists\, C\,(cump(X, C) \wedge act(C) \wedge aur\,(C)))$

We negate the conclusion at the beginning, before turning quantifiers!

$\neg C$: $\neg(scadedj \wedge creștedob \rightarrow$
$\forall\ X\,(inv\,(X) \wedge bucur\,(X) \rightarrow \exists\, C\,(cump(X, C) \wedge act(C) \wedge aur\,(C))))$

We remove the implication, take the negation down to the predicate

1. Remove the implication : $A \rightarrow B = \neg A \vee B$, $\neg(A \rightarrow B) = A \wedge \neg B$

Any transformation in a formula does NOT affect what is outside of it!

In ∀x A, transforming however on A (→, ¬, ...) does NOT change ∀x

We take ¬ inside : $\neg \forall x P(x) = \exists x \neg P(x)$ $\neg \exists x P(x) = \forall x \neg P(x)$

A_1: $\forall X\,(inv(X) \rightarrow \exists C\,(cump(X, C) \wedge (act(C) \vee oblig(C))))$
$\forall X\,(\neg inv(X) \vee \exists C\,(cump(X, C) \wedge (act(C) \vee oblig(C))))$

A_2: $scadedj \rightarrow \forall X\,(act(X) \wedge \neg aur(X) \rightarrow scade(X))$
$\neg scadedj \vee \forall X\,(\neg act(X) \vee aur(X) \vee scade(X))$

A_3: $creștedob \rightarrow \forall X\,(oblig(X) \rightarrow scade(X))$
$\neg creștedob \vee \forall X\,(\neg oblig(X) \vee scade(X))$

A_4: $\forall X\,(inv(X) \rightarrow (\exists C\,(cump(X, C) \wedge scade(C)) \rightarrow \neg bucur(X)))$
$\forall X\,(\neg inv(X) \vee \neg \exists C\,(cump(X, C) \wedge scade(C)) \vee \neg bucur(X))$
$\forall X\,(\neg inv(X) \vee \forall C\,(\neg cump(X, C) \vee \neg scade(C)) \vee \neg bucur(X))$

Remove the implication, take the negation in (cont.)

$\neg C : \neg(scadedj \wedge creștedob \rightarrow$
$\forall X\, (inv(X) \wedge bucur(X) \rightarrow \exists C\, (cump(X, C) \wedge act(C) \wedge aur(C))))$

$\neg C : scadedj \wedge creștedob \wedge$
$\neg\forall\, X\, (inv(X) \wedge bucur(X) \rightarrow \exists C\, (cump(X, C) \wedge act(C) \wedge aur(C)))$

$scadedj \wedge creștedob \wedge$
$\exists X\, (inv(X) \wedge bucur(X) \wedge \neg\exists C\, (cump(X, C) \wedge act(C) \wedge aur(C)))$

$scadedj \wedge creștedob \wedge$
$\exists X\, (inv(X) \wedge bucur(X) \wedge \forall C\, (\neg cump(X, C) \vee \neg act(C) \vee \neg aur(C)))$

Rename: unique names to quantified variables

3. We give unique names to the quantified variables in each formula so that we can later remove the quantifiers. For example:

$\forall x\, P(x) \vee \forall x\, \exists y\, Q(x, y)$ devine $\forall x\, P(x) \vee \forall z\, \exists y\, Q(z, y)$

No need in our example:

A_1: $\forall X\, (\neg inv\, (X) \vee \exists\, C\, (cump(X, C) \wedge (act(C) \vee oblig\, (C))))$

A_2: $\neg scadedj \vee \forall X\, (\neg act(X) \vee aur\, (X) \vee scade(X))$

A_3: $\neg\, creștedob \vee \forall X\, (\neg oblig\, (X) \vee scade(X))$

A_4: $\forall X\, (\neg inv\, (X) \vee \forall C\, (\neg cump(X, C) \vee \neg scade(C)) \vee \neg bucur\, (X))$

$\neg C$: $scadedj \wedge creștedob \wedge$
$\exists X\, (inv(X) \wedge bucur(X) \wedge \forall C\, (\neg cump(X, C) \vee \neg act(C) \vee \neg aur(C)))$

Skolemization: eliminating existential quantifiers

4. Skolemization: in ∀x1...∀xn∃y , the choice of y depends on x1, . . . xn; we introduce a new Skolem function y = g (x1, . . . , xn), ∃y disappears

A_1: $\forall X\,(\neg inv\,(X) \vee \exists C\,(cump(X, C) \wedge (act(C) \vee oblig\,(C))))$

C of ∃ depends on X ⇒ C becomes a new function f (X), ∃C disappears

$\forall X\,(\neg inv\,(X) \vee (cump(X, f\,(X)) \wedge (act(f\,(X)) \vee oblig\,(f\,(X)))))$

Attention! each ∃ quantifier gets a new Skolem function!

For ∃y outside any ∀, we choose a new Skolem constant

$\neg C$: $scadedj \wedge creștedob \wedge \exists\, X\,(inv\,(X) \wedge bucur\,(X)$
$\wedge \forall\; C\,(\neg cump(X, C) \vee \neg act(C) \vee \neg aur\,(C)))$

X becomes a new constant b (depends on nothing), ∃X disappears

$scadedj \wedge creștedob \wedge inv\,(b) \wedge bucur\,(b)$
$\wedge \forall C\,(\neg cump(b, C) \vee \neg act(C) \vee \neg aur\,(C))$

Normal prenex shape. Eliminate universal quantifiers

5.Bringing universal quantifiers to the front: prenex normal form

6.A_4: $\forall X\,(\neg inv(X) \vee \forall C\,(\neg cump(X, C) \vee \neg scade(C)) \vee \neg bucur(X))$

$\forall X\,\forall C\,(\neg inv(X) \vee \neg cump(X, C) \vee \neg scade(C) \vee \neg bucur(X))$

6. Eliminate universal quantifiers

(become default, a variable can be replaced by any term).

A_1: $(\neg inv(X) \vee (cump(X, f(X)) \wedge (act(f(X)) \vee oblig(f(X)))))$

A_2: $\neg scadedj \vee \neg act(X) \vee aur(X) \vee scade(X)$

A_3: $\neg creștedob \vee \neg oblig(X) \vee scade(X)$

A_4: $\neg inv(X) \vee \neg cump(X, C) \vee \neg scade(C) \vee \neg bucur(X)$

$\neg C$: $scadedj \wedge creștedob \wedge inv(b) \wedge bucur(b)$
$\wedge(\neg cump(b, C) \vee \neg act(C) \vee \neg aur(C))$

Clausal form

7. We take the conjunction outside the disjunction (distributivity) and write each clause separately (clause form, CNF)

$\neg inv(X) \vee cump(X, f(X))$

(1) $\neg inv(X) \vee act(f(X)) \vee oblig(f(X))$

(2) $\neg scadedj \vee \neg act(X) \vee aur(X) \vee scade(X)$

(3) $\neg creștedob \vee \neg oblig(X) \vee scade(X)$

(4) $\neg inv(X) \vee \neg cump(X, C) \vee \neg scade(C) \vee \neg bucur(X)$

(5) $scadedj$

(6) $creștedob$

(7) $inv(b)$

(8) $bucur(b)$

(9) $\neg cump(b, C) \vee \neg act(C) \vee \neg aur(C)$

We generate resolvers down to the empty clause

We search for predicates P(...) and ¬P(...) and unify, obtaining solvers:

(11) $\neg act(X) \vee aur(X) \vee scade(X)$ (3, 6)

(12) $\neg cump(b, C) \vee \neg act(C) \vee scade(C)$ (10, 11, $X = C$)

(13) $\neg oblig(X) \vee scade(X)$ (4, 7)

Când unificăm, redenumim clauzele să nu aibă variabile comune:

(13) $\neg oblig(Y) \vee scade(Y)$ vom unifica cu (2), redenumim X

(14) $\neg inv(X) \vee act(f(X)) \vee scade(f(X))$ (2, 13, $Y = X$)

(15) $\neg cump(b, f(X)) \vee \neg inv(X) \vee scade(f(X))$ (12, 14, $C = f(X)$)

(16) $\neg cump(b, C) \vee \neg scade(C) \vee \neg bucur(b)$ (5, 8, $X = b$)

(17) $\neg cump(b, C) \vee \neg scade(C)$ (9, 16)

(18) $\neg cump(b, f(X)) \vee \neg inv(X)$ (15, 17, $C = f(X)$)

(19) $\neg inv(b)$ (1, 18, $X = b$)

(20) $\varnothing$ (contradiction = success in indirect proof) (8, 19)

Predicate logic - syntax
Formalization of natural language
Resolution
Proofs in Predicate Logic
Semantics in Predicate Logic

Axioms of predicate calculus

A1: $\alpha \rightarrow (\beta \rightarrow \alpha)$ (A1-A3 from propositional logic)
A2: $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
A3: $(\neg\beta \rightarrow \neg\alpha) \rightarrow (\alpha \rightarrow \beta)$
A4: $\forall x\,(\alpha \rightarrow \beta) \rightarrow (\forall x\alpha \rightarrow \forall x\beta)$
A5: $\forall x\alpha \rightarrow \alpha[x \leftarrow t]$ if x can be substituted* by t in α
A6: $\alpha \rightarrow \forall x\alpha$ if x does not occur freely in α

*Define: we can substitute the variable x with the term t in ∀yϕ if:
x does not occur freely in ϕ (the substitution has no effect) or
x can substitute with t in ϕ and y does not appear in t
(we cannot substitute related variables)

In the logic with equality, we also add
A7: $x = x$
A8: $x = y \rightarrow \alpha = \beta$

where β is obtained from α by replacing any occurrences of x by y .

Rule of inference: modus ponens is sufficient: $\dfrac{A \qquad A \rightarrow B}{B}$

Deduction

Let H be a lot of formulas. A deduction (proof) from H is a string of formulas A1, A2, - - - , An, such that $\forall i \in 1, n$

1. Ai is an axiom, or
2. Ai is a hypothesis (a formula from H), or
3. Ai follows by modus ponens from the previous Aj, Ak(j, k<i)

We say that An follows from H (it is deductible, it is a consequence).

We denote:

$$H \vdash An$$

Other inference rules

$$\frac{\forall x \ \phi(x)}{\phi(c)}$$

universal instantiation (see A5)

where c is an arbitrary constant (not previously shown in the proof)

If ϕ is valid for any x , then also for an arbitrary value c.

$$\frac{\phi(c)}{\forall x \ \phi(x)}$$

universal generalisation (see A6)

where c is an arbitrary value (does not appear in assumptions)

If ϕ is valid for an arbitrary value, it is valid for any x .

$$\frac{\exists x \ \phi(x)}{\phi(c)}$$

existential instantiation

If a value with property ϕ exists, we instantiate it (with a new name).

$$\frac{\phi(c)}{\exists x \ \phi(x)}$$

existential generalization

If ϕ is true for a value, there is a value that makes it true

Predicate logic - syntax
Formalization of natural language
Resolution
Proofs in Predicate Logic
Semantics in Predicate Logic

Semantics

We define the notions:

model

interpretation

universe

semantic consequence

How do we interpret a formula?

Intuitively, we find a meaning for each symbol in the formula:

An interpretation (structure) I in predicate logic consists of:

- a non-empty manifold U called the universe or domain of I

(the set of values that variables can take)

- for any constant symbol c, a value cI∈ U
- for any n-ary function symbol f , a function fI : Un → U
- for any n-ary predicate symbol P, a submultiplet PI ⊆ Un.
  (an n-ary relation on U)

So we give an interpretation to each symbol in the formula.

An interpretation does not give values to variables (see later: assignment).

Examples of interpretations

$\forall x \, \forall y \, \forall z . P(x, y) \wedge P(y, z) \rightarrow P(x, z)$ transitivity

For example:

universe U = real numbers;

predicate P: relation ≤

$\exists e \forall x \, \neg A(x, e)$

the existence of the empty set:

predicate A(x, y) e x ∈ y

Logical implication (semantic consequence)

Let H be a formula set and C a formula.

We denote $I \models H$ if I is a model for every formula in H.

We say that H implies C ($H \models C$) if for any interpretation I ,

$I \models H$ implies $I \models C$.

(C is true in any interpretation that satisfies all assumptions in H)

There are more expressive logics than first-order logic

The principle of mathematical induction is (despite the name) a rule of deduction in the arithmetic theory of natural numbers

$$\forall P[P(0) \wedge \forall k \in \mathbb{N}.P(k) \rightarrow P(k+1)] \rightarrow \forall n \in \mathbb{N}\, P(n)$$

formula in 2nd order logic (quantification over predicates)

Logic has its limitations

The theory of natural numbers with addition (Presburger arithmetic) is decidable (anything we can express about the addition of natural numbers is provable).
But: we cannot express divisibility, prime numbers, etc.

Peano's arithmetic (with addition and multiplication) is richer but it is undecidable: there are statements that cannot be decided whether they are true or not.

Summary

We can translate (formalize) from natural language into first order logic

We can prove theorems by indirect proof:

negate the conclusion,

transform to clause form (conjunction of disjunctions)

by resolution method

find a contradiction (empty clause).

Thank you!

Bibliography

The content of the course is based on the material from the LSD course taught by Prof. Dr. Eng. Marius Minea and S.l. Dr. Eng. Casandra Holotescu (http://staff.cs.upt.ro/~marius/curs/lsd/index.html)

The logic questions at the beginning of the course were taken from the Introduction to Logic course at Stanford University (https://www.coursera.org/learn/logic-introduction)