Intro. to 1st Order Logic (a.k.a. Predicate Calculus)

Moonzoo Kim
CS Dept. KAIST

moonzoo@cs.kaist.ac.kr

Introduction to predicate calculus (1/2)

- Propositional logic (sentence logic) dealt quite satisfactorily with sentences using conjunctive words (접속사) like not, and, or, and if … then. But it fails to reflect the *finer logical structure* of the sentence
- What can we reason about a sentence itself which deals with its target domain?
  - ex. Jane is taller than Alice (target domain : human being)
  - ex2. For natural numbers x and y, $x+y \geq -(x+y)$ (target domain: $\mathcal{N}$)
- What can we reason about a sentence itself which also deal with modifiers like there exists…, all …, among … and only …. ?
  - Note that these modifiers enable us to reason about an infinite domain because we do not have to enumerate all elements in the domain

KAIST

Introduction to predicate calculus (2/2)

- Ex. Every student is younger than some instructor
  - We could simply identify this assertion with a propositional atom p. However, this fails to reflect the finer logical structure of this sentence
- This statement is about being a student, being an instructor and being younger than somebody else for a set of university members as a target domain
  - We need to express them and use predicates for this purpose
  - S(yunho), I(moonzoo), Y(yunho,moonzoo)
- We need variables x, y to not to write down all instance of S(-), I(-), Y(-)
  - Every student x is young than some instructor y
- Finally, we need quantifiers to capture the actual elements by variables
  - For every x, if x is a student, then there is some y which is an instructor such that x is younger than y compare with
  - $\forall x\ (S(x) \rightarrow (\exists y\ (I(y) \wedge Y(x,y))))$
    - Compare with $\forall x\ (S(x) \rightarrow (\exists y\ (I(y) \rightarrow Y(x,y))))$

KAIST

Examples of 1st Order-logic Formula

predicate, element of domain, a variable for a domain, quantifier

- Bill is a student. $Student(Bill)$
- All students are smart. $\forall x\ (Student(x) \rightarrow Smart(x))$
- There exists a student. $\exists x\ Student(x)$.
- There exists a smart student. $\exists x\ (Student(x) \wedge Smart(x))$
- Every student loves some student. $\forall x\ (Student(x) \rightarrow \exists y\ (Student(y) \wedge Loves(x,y)))$
- Every student loves some other student. $\forall x(Student(x) \rightarrow \exists y\ (Student(y) \wedge \neg(x=y) \wedge Loves(x,y)))$
- There is a student who is loved by every other student.
  $\exists x\ (Student(x) \wedge \forall y\ (Student(y) \wedge \neg(x = y) \rightarrow Loves(y,x)))$
- No student loves Bill. $\neg \exists x\ (Student(x) \wedge Loves(x, Bill))$
- Bill does not take Analysis. $\neg\ Takes(Bill, Analysis)$.
- Bill takes Analysis or Geometry (or both). $Takes(Bill,Analysis) \vee Takes(Bill, Geometry)$
- Bill takes Analysis and Geometry. $Takes(Bill, Analysis) \wedge Takes(Bill, Geometry)$
- Bill takes either Analysis or Geometry (but not both)
  $Takes(Bill, Analysis) \leftrightarrow \neg\ Takes(Bill, Geometry)$

KAIST ***Excerpted from** www.uobabylon.edu.iq/eprints/publication_5_29514_1380.pdf*

Relations and predicates

- The axioms and theorems of mathematics are defined on arbitrary sets (domain) such as the set of integers $\mathcal{Z}$
  - ex. Fermat's last theorem
    - If an integer *n* is greater than 2, then the equation $a^n + b^n = c^n$ has no solutions in non-zero integers *a*, *b*, and *c*.
  - Can we express the Fermat's last theorem in propositional logic?
- The predicate calculus extends the propositional calculus with predicate letters that are interpreted as relations on a domain
  - i.e., predicates are interpreted upon domain
- Def 5.2. A relation can be represented by a boolean valued function $R:D^n \rightarrow \{T,F\}$, by mapping an n-tuple to T iff it is included in the relation
  - $R(d_1,\ldots d_n) = T$ iff $(d_1,\ldots d_n) \in \mathcal{R}$

KAIST

Predicate formulas

- Let $\mathcal{P}$, $\mathcal{A}$ and $\mathcal{V}$ be countable sets of symbols called predicate letters, constants, and variables, respectively.
  - $\mathcal{P}$={p,q,r} $\mathcal{A}$={a,b,c}, $\mathcal{V}$={x,y,z}
- Def 5.4 Atomic formulas and formulas
  - atomic formula
    - argument ::= x for any x $\in \mathcal{V}$
    - argument ::= a for any a $\in \mathcal{A}$
    - argument_list ::= argument$^+$
    - atomic_formula ::= p |
      p(argument_list) for any p $\in \mathcal{P}$
  - formula ::== atomic_formula
  - formula ::= $\neg$ formula
  - formula ::= formula $\vee$ formula
  - formula ::= $\forall$ x formula
  - formula ::= $\exists$ x formula

1. $\forall x \forall y (p(x, y) \rightarrow p(y, x))$.
2. $\forall x \exists y p(x, y)$.
3. $\exists x \exists y (p(x) \wedge \neg p(y))$.
4. $\forall x p(a, x)$.
5. $\forall x (p(x) \wedge q(x)) \leftrightarrow (\forall x p(x) \wedge \forall x q(x))$.
6. $\exists x (p(x) \vee q(x)) \leftrightarrow (\exists x p(x) \vee \exists x q(x))$.
7. $\forall x (p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))$.
8. $(\forall x p(x) \rightarrow \forall x q(x)) \rightarrow \forall x (p(x) \rightarrow q(x))$.

Free and bound variables

- Def 5.6
  - ∀ is the universal quantifier and is read ‘for all’.
  - ∃ is the existential quantifier and is read ‘there exists’.
  - In a quantified formula ∀ xA, x is the quantified variable and A is the scope of the quantified variable.
- Def 5.7 Let A be a formula. An occurrence of a variable x in A is a free variable of A iff x is not within the scope of a quantified variable x.
  - Notation: $A(x_1,\ldots x_n)$ indicates that the set of free variables of the formula A is a subset of $\{x_1,\ldots x_n\}$. A variable which is not free is bound.
  - If a formula has no free variable it is closed
  - If $\{x_1,\ldots,x_n\}$ are all the free variables of A, the universal closure of A is $\forall x_1 \ldots \forall x_n A$ and the existential closure is $\exists x_1 \ldots \exists x_n A$
- Ex 5.8 p(x,y), ∃ y p(x,y), ∀x ∃y p(x,y)
- Ex 5.9
  - In (∀x p(x)) ∧ q(x), the occurrence of x in p(x) is bound and the occurrence in q(x) is free. The universal closure is ∀x (∀xp(x) ∧ q(x)).
  - Obviously, it would have been better to write the formula as ∀xp(x)∧ q(y) where y is the free variable

KAIST

Interpretations (1/5)

- Def 5.10 Let U be a set of formulas s.t. $\{p_1,\ldots p_m\}$ are all the predicate letters and $\{a_1,\ldots, a_k\}$ are all the constant symbols appearing in U. An interpretation $\mathcal{I}$ is a triple $(D, \{R_1,\ldots R_m\}, \{d_1,\ldots,d_k\})$, where
  - D is a non-empty set,
  - R_i is an n_i-ary relation on D that is assigned to the n_i-ary predicate p_i
    - Notation: $p_i^{\mathcal{I}} = R_i$
  - $d_i \in D$ is an element of D that is assigned to the constant a_i
    - Notation: $a_i^{\mathcal{I}} = d_i$
- Ex 5.11. Three numerical interpretations for $\forall x\, p(a,x)$:
  - $\mathcal{I}_1= (\mathcal{N}, \{\leq\}, \{0\})$, $\mathcal{I}_2=(\mathcal{N}, \{\leq\}, \{1\})$. $\mathcal{I}_3=(\mathcal{Z}, \{\leq\}, \{0\})$.
  - $\mathcal{I}_4=(\mathcal{S}$, {substr},{“”}) where S is the set of strings on some alphabet

Interpretations (2/5)

- Def 5.12 Let $\mathcal{I}$ be an interpretation. An assignment $\sigma_{\mathcal{I}} : \mathcal{V} \rightarrow D$ is a function which maps every variable to an element of the domain of $\mathcal{I}$. $\sigma_{\mathcal{I}}[x_i \leftarrow d_i]$ is an assignment that is the same as $\sigma_{\mathcal{I}}$ except that x_i is mapped to d_i
- Def 5.13 Let A be a formula, $\mathcal{I}$ an interpretation and $\sigma_{\mathcal{I}}$ an assignment. $v_{\sigma_{\mathcal{I}}}(A)$, the truth value of A under $\sigma_{\mathcal{I}}$ is defined by induction on the structure of A:
  - Let $A = p_k(c_1,\ldots,c_n)$ be an atomic formula where each c_i is either a variable x_i or a constant a_i. $v_{\sigma_{\mathcal{I}}}(A) = T$ iff
    - $\langle d_1,\ldots d_n \rangle \in R_k$ where R_k is the relation assigned by $\mathcal{I}$ to p_k and
    - d_i is the domain element assigned to c_i, either
      - by $\mathcal{I}$ if c_i is a constant or
      - by $\sigma_{\mathcal{I}}$ if c_i is variable
  - $v_{\sigma_{\mathcal{I}}}(\neg A)= T$ iff $v_{\sigma_{\mathcal{I}}}(A)= F$
  - $v_{\sigma_{\mathcal{I}}}(A_1 \vee A_2)$ iff $v_{\sigma_{\mathcal{I}}}(A_1)= T$ or $v_{\sigma_{\mathcal{I}}}(A_2)= T$
  - $v_{\sigma_{\mathcal{I}}}(\forall x\, A_1) = T$ iff $v_{\sigma_{\mathcal{I}}[x \leftarrow d]}(A_1)= T$ for all $d \in D$
  - $v_{\sigma_{\mathcal{I}}}(\exists x\, A_1) = T$ iff $v_{\sigma_{\mathcal{I}}[x \leftarrow d]}(A_1)= T$ for some $d \in D$

Interpretations (3/5)

- Thm 5.14 Let A be a closed formula. Then $v_{\sigma_{\mathcal{I}}}(A)$ does not depend on $\sigma_{\mathcal{I}}$. In such cases, we use simply $v_{\mathcal{I}}(A)$ instead of $v_{\sigma_{\mathcal{I}}}(A)$
- (important!) Thm 5.15 Let A' = $A(x_1,\ldots,x_n)$ be a non-closed formula and let $\mathcal{I}$ be an interpretation. Then:
  - $v_{\sigma_{\mathcal{I}}}(A')=T$ for some assignment $\sigma_{\mathcal{I}}$ iff $v_{\mathcal{I}}(\exists x_1 \ldots \exists x_n\, A') = T$
  - $v_{\sigma_{\mathcal{I}}}(A')=T$ for all assignment $\sigma_{\mathcal{I}}$ iff $v_{\mathcal{I}}(\forall x_1 \ldots \forall x_n\, A') = T$
  - Thm 5.15 is important since we have many chances to add or remove quantified variables to and from formula during proofs.
- Def 5.16 A closed formula A is true in $\mathcal{I}$ or $\mathcal{I}$ is a model for A, if $v_{\mathcal{I}}(A) = T$.
  - Notation: $\mathcal{I} \models A$
  - Note that we overload $\models$ with usual logical consequence as in propositional logic
    - $\{A_1, A_2, A_3\} \models A$
- Def 5.18 A closed formula A is satisfiable if for some interpretation $\mathcal{I}$, $\mathcal{I} \models A$. A is valid if for all interpretations $\mathcal{I}$, $\mathcal{I} \models A$
  - Notation: $\models A$.

KAIST

Interpretation (4/5)

- Ex 5.19 $\models (\forall x\ p(x)) \rightarrow p(a)$
  - Suppose that it is not. Then there must be an interpretation $\mathcal{I}$ = (D, {R}, {d}) such that $v_{\mathcal{I}}(\forall x\ p(x))$ = T and $v_{\mathcal{I}}(p(a))$ = F
  - By Thm 5.15, $v_{\sigma_{\mathcal{I}}}(p(x))$ = T for all assignments $\sigma_{\mathcal{I}}$, in particular for the assignment $\sigma'_{\mathcal{I}}$ that assigns d to x (i.e. $v_{\sigma'_{\mathcal{I}}}(p(x))$ = T). But p(a) is closed, so $v_{\sigma'_{\mathcal{I}}}(p(a)) = v_{\mathcal{I}}(p(a))$ = F, a contradiction

Example 5.20 Here is a semantic analysis of the formulas from Example 5.5:

- $\forall x \forall y (p(x, y) \rightarrow p(y, x))$
  The formula is satisfiable in an interpretation where p is assigned a symmetric relation like $=$.
- $\forall x \exists y p(x, y)$
  The formula is satisfiable in an interpretation where p is assigned a relation that is a total function, such as $(x, y) \in R$ iff $y = x + 1$ for $x, y \in \mathcal{Z}$.
- $\exists x \exists y (p(x) \wedge \neg p(y))$
  This formula is satisfiable only in a domain with at least two elements.

Interpretation (5/5)

- $\forall x p(a, x)$
  This expresses the existence of a special element. For example, if p is interpreted by the relation $\leq$ on the domain $\mathcal{N}$, then the formula is true for $a = 0$. If we change the domain to $\mathcal{Z}$ the formula is false for the same assignment of $\leq$ to p. Thus a change of domain alone can falsify a formula.

- $\forall x(p(x) \wedge q(x)) \leftrightarrow (\forall x p(x) \wedge \forall x q(x))$
  The formula is valid. We prove the forward direction and leave the converse as an exercise. Let $\mathcal{I} = (D, \{R_1, R_2\}, \{\,\})$ be an arbitrary interpretation. By Theorem 5.15, $v_{\sigma_\mathcal{I}}(p(x) \wedge q(x)) = T$ for all all assignments $\sigma_\mathcal{I}$, and by the inductive definition of an interpretation, $v_{\sigma_\mathcal{I}}(p(x)) = T$ and $v_{\sigma_\mathcal{I}}(q(x)) = T$ for all assignments $\sigma_\mathcal{I}$. Again by Theorem 5.15, $v_\mathcal{I}(\forall x p(x)) = T$ and $v_\mathcal{I}(\forall x q(x)) = T$, and by the definition of interpretation $v_\mathcal{I}(\forall x p(x) \wedge \forall x q(x)) = T$.

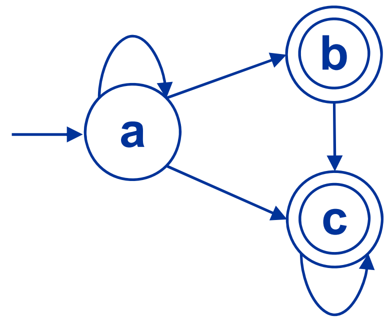
  Show that $\forall$ does not distribute over disjunction by constructing a falsifying interpretation for $\forall x(p(x) \vee q(x)) \leftrightarrow (\forall x p(x) \vee \forall x q(x))$.

- $\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))$
  This is a valid formula, but its converse is not.

KAIST

Example: finite automata

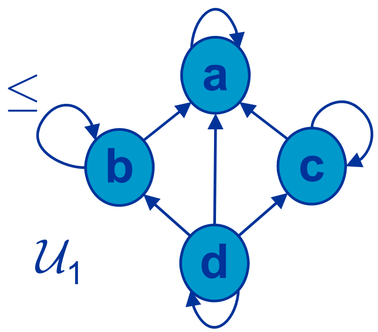
- For an interpretation $\mathcal{I} = (\mathcal{D},\mathcal{R},\mathcal{F},\mathcal{C})$ where
  - $\mathcal{D}$ = {a,b,c}
  - $\mathcal{R}$= {Trans, Final, Equality} where
    - Trans = {(a,a),(a,b),(a,c),(b,c),(c,c)}
    - Final = {b,c}
    - Equality={(a,a),(b,b),(c,c)}
  - $\mathcal{F}$={}
  - $\mathcal{C}$={a}



- Formulas for $\mathcal{I}$ where $R^{\mathcal{I}}$=Trans, $F^{\mathcal{I}}$=Final, $=^{\mathcal{I}}$ =Equality, $i^{\mathcal{I}}$=a
  - $\mathcal{I} \models \exists y\, R(i,y)$
  - $\mathcal{I} \models \neg F(i)$
  - $\mathcal{I} \not\models \forall x \forall y \forall z\, (R(x,y) \wedge R(x,z) \rightarrow y = z)$
  - $\mathcal{I} \models \forall x \exists y\, R(x,y)$

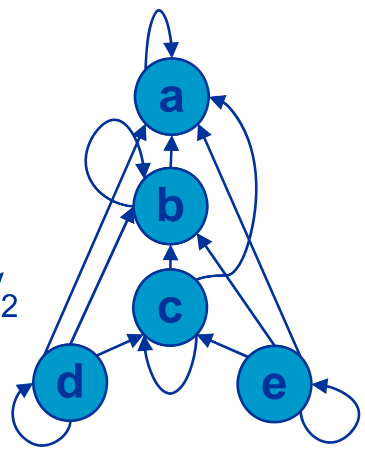
Example: partial order set (POSET)

- Def. $\mathcal{U}$ is a partially ordered set (poset) if $\mathcal{U}$ is a model of
  - $\forall xyz\ (x \leq y \wedge y \leq z \rightarrow x \leq z)$ (transitivity)
  - $\forall xy\ (x \leq y \wedge y \leq x \leftrightarrow x = y)$ (anti-symmetry)
- $\mathcal{U}_1 \vDash \exists x\ \forall y\ (x \leq y)$
  - i.e., $\mathcal{U}_1$ has a least element
- $\mathcal{U}_3 \vDash \forall x \neg \exists y\ (x < y)$
  - i.e., in $\mathcal{U}_3$ no element is strictly less than another element
- Def. $\mathcal{U}$ is a totally ordered set if $\mathcal{U}$ is a poset and $\mathcal{U} \vDash \forall x\ \forall y\ (x \leq y \vee y \leq x)$
- Def. $\mathcal{U}$ is densely ordered if $\mathcal{U} \vDash \forall x\ \forall y\ (x < y \rightarrow \exists z\ (x<z \wedge z< y)$
- We can distinguish $\mathcal{U}_3$ and $\mathcal{U}_4$ by $A(x)=\forall y\ (y \neq x \rightarrow \neg(y \leq x) \wedge \neg(x \leq y))$
  - $\mathcal{U}_4 \vDash \forall x\ \forall y\ (A(x) \wedge A(y) \rightarrow x = y)$
  - $\mathcal{U}_3 \vDash \neg\forall x\ \forall y\ (A(x) \wedge A(y) \rightarrow x = y)$

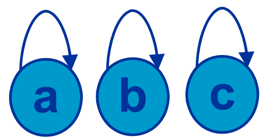


$\mathcal{U}_1$

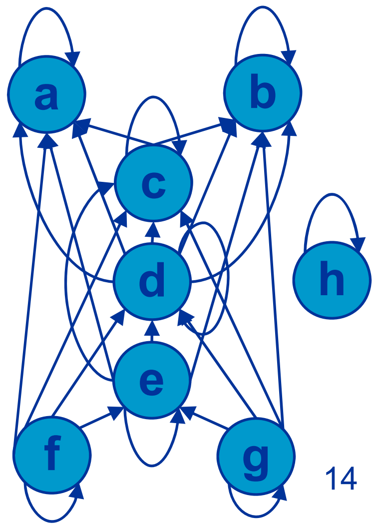
$\mathcal{U}_2$



$\mathcal{U}_3$



$\mathcal{U}_4$



Exercise: POSET (cont.)

- Define formulas for
  - x is the maximum (the largest element in a target domain)
    - $\forall y\ y \leq x$
  - x is maximal (not smaller than any other elements)
    - $\neg\exists y\ x < y \equiv \forall y\ \neg(x < y)$
    - Note the difference between $\forall y\ y \leq x$ and $\forall y\ \neg(x < y)$.
      - For totally ordered set, these two formulas are same, but for POSET, they are different.
  - There is no element between x and y
    - $\neg\exists z\ ((x \leq z \wedge z \leq y) \vee (y \leq z \wedge z \leq x))$
  - x is an immediate successor of y
    - $(x > y) \wedge \neg\exists z\ (y \leq z \wedge z \leq x)$
  - z is the infimum of x and y (the greatest element less than or equal to x and y)
    - $\forall st\ ((s\leq x \wedge t\leq y) \rightarrow (s\leq z \wedge t\leq z) \wedge (z\leq x \wedge z\leq y))$
- Give a formula ϕ s.t. $\mathcal{U}_2 \vDash \phi$ and $\mathcal{U}_4 \vDash \neg\phi$
- Let $\phi = \exists x \forall y\ (x \leq y \vee y \leq x)$. Find posets $\mathcal{U}_1$ and $\mathcal{U}_2$ s.t. $\mathcal{U}_1 \vDash \phi$ and $\mathcal{U}_2 \vDash \neg \phi$

KAIST

A formula represents a set of models

- A formula ϕ describes characteristics of target structures in a compact way.
  - ex. deterministic automata, partial order sets, binary trees, relational database, etc
- In other words, a formula ϕ designates a set of models (i.e., interpretations) that satisfies ϕ
  - $\forall x \forall y \forall z\ (R(x,y) \wedge R(x,z) \rightarrow y = z)$ represents all deterministic graphs
  - $\forall x \forall y \forall z\ (R(x,y) \wedge R(y,z) \rightarrow R(x,z))$ represents all transitive graphs.
- Validity, satisfiability, and provability of a predicate formula is all undecidable. However, checking formulas on concrete interpretations is practical
  - ex. SQL queries over relational database
  - ex. XQueries over XML documents
  - ex. Model checking of a program

KAIST