SC/MATH 1090

8- Predicate Logic

Ref: G. Tourlakis, *Mathematical Logic*, John Wiley & Sons, 2008.

York University

Department of Computer Science and Engineering

Overview

- Why do we need predicate logic?
  - Boolean Logic is totally insufficient. It can not handle objects, such as numbers, sets, strings, etc.
- Alphabet
- Term calculation
- Formula calculation- predicate logic
- Extending definitions :
  - complexity, priority, bracket reduction
  - Tautologies (and abstractions)
  - Substitution

Alphabet for Predicate Logic

- Logical symbols (fixed part of the alphabet) :
  - Boolean Alphabet, i.e. Boolean variables, Boolean constants, brackets, and Boolean connectives
  - Object variables
    x, y, z, u, v, w with or without primes or subscripts
  - Symbol for equality of objects: =
  - The universal quantifier symbol: $\forall$

Alphabet for Predicate Logic- cont.

- Nonlogical symbols (variable part, based on application):
  - Zero or more object constants
    - Examples: 1, 1000 , {1}, {1,2,3}, {}
    - Denoted generally as a, b, c with or without primes or subscripts
  - Zero or more functions
    - Examples: +, -, $\cap$, $\cup$
    - Denoted generally as *f, g, h* with or without primes or subscripts
  - Zero or more predicates
    - Examples: >, $\leq$, $\subset$
    - Denoted generally as φ,ψ with or without primes or subscripts

First order language

- In first order language, we can talk about formulas and objects

  - Well-formed formulae
    - Atomic formulae (complexity =zero)
    - More complex formulae (complexity > 0)

  - Terms
    - Atomic terms (complexity=zero)
    - More complex terms (complexity >0)

Term Calculation (Term-Parse)

- A *term-calculation* is any finite and ordered sequence of strings that we may write respecting the following rules:

  1) At any step, we may write an *object variable* or an *object constant*

  2) At any step, for a function ***f*** of arity ***n***, we may write
     $\boldsymbol{f t_1 t_2 \ldots t_n}$
     provided that $t_1, t_2, \ldots, t_n$ are terms written in a previous step

Complexity of terms

- **Definition**. **(Complexity of terms)** The complexity of a term is defined to be the number of function symbols.
- We will use the symbols ***t*** and ***s*** to denote terms.
- Since $ft_1t_2...t_n$ is not familiar, we sometimes write it as ***f(t1, t2, ...tn)***. But note this is NOT the formal definition.
- Just like the Boolean formula definition, we can define terms **recursively** as well.
- Top-down and bottom-up parse for terms

Atomic formulae

- Atomic formula of predicate logic are:
  - Boolean variables or constants
  - The string $\boldsymbol{s=t}$ for any terms t and s
  - The string $\boldsymbol{\varphi t_1 t_2 .. t_n}$ for some predicate φ with arity n, and terms $t_1, t_2, \ldots t_n$

- Note: The complexity of above formulae is zero.

Formula Calculation (formula-parse)

- A *formula-calculation* is any finite and ordered sequence of strings that we may write respecting the following rules:

1) At any step, we may write any atomic formulae.

2) At any step, we may write $(\neg A)$, provided we have already written A in a previous step.

3) At any step, we may write $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, or $(A \equiv B)$, provided we have already written A and B in a previous step.

4) At any step, we may write **$((\forall x)\ A)$**, provided we have already written A in a previous step.

Well-formed Formula

- **Definition**.**(First-Order Formulae)** A string A over the alphabet introduced for predicate logic is a first-order formula or a well-formed-formula (wff) **iff** it is a string written at some step of some formula calculation.
- WFF is used to denote the set of all wff's.
- **Definition**. The complexity of a wff is the total number of occurrences of $\forall, \neg, \wedge, \vee, \rightarrow, \equiv$ in the formula.
- Note: Recursive definition of WFF

Some abbreviations

- **Definition. (Existential Quantifier)**

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

- Another notation:
- $((\forall x)\, (B \rightarrow A)) \equiv (\forall x)_B\, A$
- $((\exists x)\, (B \wedge A)) \equiv (\exists x)_B\, A$

Priorities, Bracket Reduction

- Order of priorities

$$\left\{ \begin{matrix} (\forall \mathbf{x}) \\ \neg \end{matrix} \right\}, \wedge, \vee, \rightarrow, \equiv$$

  - If same priority, the rightmost acts first.

- Bracket reduction
  - Outermost brackets are redundant.
  - Any pair of brackets is redundant, if its existence can be understood from the above priorities.

Bound and Free Occurrences of object variables

- **Scope of a quantifier**
  - Example: B $\rightarrow$ ($\forall$x) A
  - Any x occurring in A is in the scope of ($\forall$x).
  - Any x occurring in B is not in the scope of ($\forall$x).

- An occurrence of a variable x in a formula is **bound** iff
  - It occurs in a substring ($\forall$x), or
  - It occurs in the scope of some ($\forall$x)
- An occurrence of a variable x in a formula is **free** iff it is not bound.

Subformulae & Abstraction

- **Definition. (Subformulae)** B is a subformula of A iff
  1) B is identical to A, or
  2) A is (¬C) and B is subformula of C, or
  3) A is (C ◦ D) and B is a subformula of C or D or both, or
  4) A is ((∀x) C) and B is a subformula of C.

- Note: A subformula is a (well-formed) formula.

- To write the **abstraction** of a formula, we need to (iteratively)
  - identify the shortest possible subformulae of A than contains non-Boolean symbols and
  - replace it by fresh Boolean variables.

Tautologies and Tautological Implications

- The formula A is a tautology (or $\models_{\text{taut}}$ A) iff the **abstraction** of A is a tautology.
- Examples:

  $\models_{\text{taut}}$ x=y ∨ ¬x=y

  $\models_{\text{taut}}$ φx ∨ q ≡ q ∨ φx

- We write $\Gamma \models_{\text{taut}}$ A iff the *abstractions* of the formula in Γ tautologically imply the *abstraction* of A.
  - For example x=y $\models_{\text{taut}}$ x=y ∨ (∀x) φx

Substitution

Substituting terms into (object) variables in terms

$s[\mathbf{x} := t]$ is :

$$\begin{cases} s & \text{if } s \text{ is a constant or a variable other than } \mathbf{x} \\ t & \text{if } s \text{ is } \mathbf{x} \\ f(s_1[\mathbf{x} := t], \ldots, s_n[\mathbf{x} := t]) & \text{if } s \text{ is } f(s_1, \ldots, s_n) \end{cases}$$

- Example:

  g(f(x+1, y*6)) [x:=2]

  is g(f(2+1, y*6))

Substitution

Substituting terms into (object) variables in formulae

$A[\mathbf{x} := t]$ is :

$\phi(s_1[\mathbf{x} := t], \ldots, s_n[\mathbf{x} := t])$	if A is $\phi(s_1, \ldots, s_n)$
$s_1[\mathbf{x} := t] = s_2[\mathbf{x} := t]$	if A is $s_1 = s_2$
$\neg C[\mathbf{x} := t]$	if A is $\neg C$
$C[\mathbf{x} := t] \circ D[\mathbf{x} := t]$	if A is $C \circ D$
A	if A is one of $p, \top, \bot, (\forall \mathbf{x})B$
$(\forall \mathbf{y})B[\mathbf{x} := t]$	if A is $(\forall \mathbf{y})B$, where $\mathbf{y}(\neq \mathbf{x})$ does *not* occur in t *or* $\mathbf{x}$ is not free in B
undefined	if A is $(\forall \mathbf{y})B$, where $\mathbf{y}(\neq \mathbf{x})$ *does* occur in t *and* $\mathbf{x}$ is free in B

- In other words, substitute unless a variable in t gets captured!

Unconditional substitution

Substituting formulae into (Boolean) variables in formulae

$A[\mathbf{p} \setminus B]$ is :

$$\begin{cases} B & \text{if } A \text{ is } \mathbf{p} \\ A & \text{if } A \text{ is in } \mathbf{AF} \text{ but is not } \mathbf{p} \\ \neg C[\mathbf{p} \setminus B] & \text{if } A \text{ is } \neg C \\ C[\mathbf{p} \setminus B] \circ D[\mathbf{p} \setminus B] & \text{if } A \text{ is } C \circ D \\ (\forall \mathbf{x}) C[\mathbf{p} \setminus B] & \text{if } A \text{ is } (\forall \mathbf{x}) C \end{cases}$$

- This is simply substitution, without any conditions!
- Example: $((\forall x)\ (p \equiv \varphi x))[p \setminus x=y]$

  is $((\forall x)\ (x=y \equiv \varphi x))$

Conditional Substitution

Substituting formulae into (Boolean) variables in formulae

$A[\mathbf{p} := B]$ is :

$$\begin{cases} B & \text{if } A \text{ is } \mathbf{p} \\ A & \text{if } A \text{ is in } \mathbf{AF} \text{ but is not } \mathbf{p} \\ \neg C[\mathbf{p} := B] & \text{if } A \text{ is } \neg C \\ C[\mathbf{p} := B] \circ D[\mathbf{p} := B] & \text{if } A \text{ is } C \circ D \\ (\forall \mathbf{x}) C[\mathbf{p} := B] & \text{if } A \text{ is } (\forall \mathbf{x})C \textit{ and } \mathbf{x} \text{ is not free in } B \\ & \text{else } \textit{undefined} \end{cases}$$

- Substitute unless an object variable gets captured!
- Example: $((\forall x)\,(p \equiv \varphi x))[p := x=y]$ is undefined

Partial Generalization

- **Definition. (Partial Generalization)** B is a partial generalization of A if B is A prefixed by *zero or more* universal quantifiers (with any bound variables).

- Example: Some partial generalizations of $(p \equiv \varphi x)$

  $(p \equiv \varphi x)$, $(\forall x)(p \equiv \varphi x)$, $(\forall y)(p \equiv \varphi x)$,

  $(\forall x)(\forall y)(p \equiv \varphi x)$, $(\forall x)(\forall x)(\forall y)(p \equiv \varphi x)$,

  ...

Logical Axiom of First-Order Logic (Λ_1)

The logical axioms of Predicate logic consist of all possible partial generalizations of any formulae in the following forms:

Ax1. All *tautologies*.

Ax2. *Specialization* axiom or *substitution* axiom:

$(\forall x)A \rightarrow A[x:=t]$

Ax3. $(\forall x)(A \rightarrow B) \rightarrow (\forall x)A \rightarrow (\forall x)B$

Ax4. $A \rightarrow (\forall x)A$, *where x is not free in A.*

Ax5. Identity axiom: x=x, for any variable x

Ax6. Leibniz axiom for equality:

$t=s \rightarrow (A[x:=t] \equiv A[x:=s])$

First-order Rules of inference

$$\frac{A,\ A \equiv B}{B}$$ (Eqn)

$$\frac{A \equiv B}{C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]}$$ (Boolean Leibniz or BL)

provided *p is not in the scope of a quantifier in C*

- Due to above condition, there won't be any captures.

Theorem- Calculations (or proofs)

- Let Γbe a given set of formulae (our assumptions)
- A theorem-calculation (or proof) from Γ is any finite (ordered) sequence of formulae that can be written following these rules:
  1. We may write a formula from Λ_1 or Γ at any step
  2. We may write the denominator of an *instance of an inference rule* (Eqn or BL), provided all formulae in the numerator (of the same instance) have been written in a previous step.
- A proof from Γ is also called a **Γ-proof.**
- Any formula A appearing in a Γ-proof is a **Γ-theorem**.