BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PREDICATE LOGIC

Propositional Logic – Expressiveness

Predicates

Propositional Logic – Expressiveness

- Consider the following statements:
    - Ravi is a student
    - Kavi is a student
  - If you were to express these in propositional logic as:
    - **ravi_student**
    - **kavi_student**

  you end up with *propositions that are atomic* (i.e. *they do not have a structure*).

- And the consequence is:
  - *we cannot infer the relation – that both are students – between the two propositions*

The need for structure

- What do we need to express such relations?
  - We need predicates.

- A proposition is also a predicate but
  - a ***predicate*** can capture a *property (or a relation) on one or more entities*: e.g.
    - **student(ravi)**
    - **student(kavi)**
    - **friend(ravi, kavi)**
    - **friend(ravi, pavi)**

Predicates – Expressiveness

- Consider the following statement:
    - *A teaching assistant is a student and a teacher*.
  - If you were to express this in propositional logic:
    - **ta <--> student $\wedge$ teacher**
- But now if you want to ask:
    - Is Kavi a **ta**?
  - You need to ask:
    - Is Kavi a student?
    - Is Kavi a teacher?
- You can (try to) fix this by saying:
  - **ta(kavi) <--> student(kavi) $\wedge$ teacher(kavi)**

This does not fix the problem adequately! Why?

BITS Pilani
Pilani Campus

innovate achieve lead

CS/IS F214
Logic in Computer Science

MODULE: PREDICATE LOGIC

Expressiveness of Predicates

- Need for variables and quantification

Predicates – The need for variables

- How do we address this problem?
  - Introduce variables:
    - **ta(X) <--> student(X) $\wedge$ teacher(X)**
- What is X?
  - **X** is a variable which may stand
    - for Kavi, Pavi, or Ravi - in our example or
    - for "any **student** or **teacher**" in a generic context
  - In the latter form there is a ***quantification*** i.e. *who (or what) can **X** range over*?
    - i.e. the statement above should actually be read as
      - <u>for all **X**</u> **ta(X) <--> student(X) $\wedge$ teacher(X)**
- In notation, we write:
    - **$\forall$(X) ta(X) <--> student(X) $\wedge$ teacher(X)**
  - This is referred to as ***universal quantification*.**

Use of Predicates, Variables, and Quantification

- Write each of the following using *predicates, variables,* and *(universal) quantifiers*:
  - *All birds fly*
  - *All humans are mammals*
  - *All students in BITS think they are brighter than others*

Predicates ,Variables, and Quantification - Examples

- *All birds fly*
  - **$\forall$X bird(X) --> fly(X)**
- *All humans are mammals*
  - **$\forall$X human(X) --> mammal(X)**
- *All students in BITS think they are brighter than others*
  - **$\forall$X $\forall$Y student(X, “BITS”) $\wedge$ $\neg$equals(X,Y) --> think_brighter(X, Y)**
    - Question:
      - Why not express it as ***think(brighter(X,Y))***?
        - Relation (think) over a Relation (brighter)!

Predicates – Variables and Quantification

- Consider the following statements:
  1. *There are some students in BITS who are not bright.*
  2. *For every number, there is a larger number.*
  3. *Not all birds can fly*
- What do you need – *in addition to predicates, variables, and universal quantifiers* – to specify these?

Predicates and Variables – Existential Quantification

- Writing the following statements:
  1. *There are some students in BITS who are not bright.*
  2. *For every number, there is a larger number.*
  3. *Not all birds can fly*
- require "existential quantification" of the form
  - there exists **X** : **P(X)**

  in notation, formulas of the form
  - **∃X P(X)**

Predicates and Variables - Existential Quantification: Examples

- Write the following statements formally:
  1. *There are some students in BITS who are not bright.*

     $\exists$X (student(X, "BITS") $\wedge$ $\neg$bright(X))

  2. *For every number, there is a larger number*

     $\forall$X (number(X) --> $\exists$Y number(Y) $\wedge$ larger(Y, X))

  3. *Not all birds can fly*

     $\neg$($\forall$X bird(X) --> fly(X))

Quantifiers: Order of quantifiers and Negation of Quantifiers

1. *For every number, there is a larger number*

   $\forall X\ (number(X) \dashrightarrow \exists Y\ number(Y) \wedge larger(Y, X))$

   - Is this different from each of the following?
     - $\forall X\ \exists Y\ (number(X) \dashrightarrow number(Y) \wedge larger(Y, X))$
     - $\exists Y\ \forall X\ (number(X) \dashrightarrow number(Y) \wedge larger(Y, X))$

2. *Not all birds can fly*

   $\neg(\forall X\ bird(X) \dashrightarrow fly(X))$

   - Is this the same as $\exists X\ bird(X) \wedge \neg fly(X)$?

Predicates, Variables, and Quantification

- ***Predicates** capture **relations*** (*on **entities** or **values***)

- ***Variables** range over entities* (*or values*)
  - i.e. they may not denote a specific entity but
    - can be substituted by an entity from a ***collection*** (that is not explicitly stated)

- ***Quantifiers** quantify the **range***
  - universal quantifier denotes that the *range is the universal set* (or collection)
  - existential quantifier denotes that *there exists at least one such element (in the universe*)

innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PREDICATE LOGIC

Expressiveness of Predicates:
Arguments of Predicates

Predicate Logic: Nature of relations

- *Predicates capture relations:*
  - A (binary) relation can be:
    - one to one
      - **successor(X, Y)** *on natural numbers*
        - X's successor is Y
    - many to one
      - **mother(X,Y)** *on persons*
        - X's mother is Y
    - one to many
      - **characterOf(N, A)** *on novels and persons*
        - characters of N include A

Predicate Logic: Nature of relations

- *Predicates may capture properties*
  - **lawyer(X)**
    - X is a lawyer
  - **god()**
    - *which is TRUE, if you are a monotheist*
    - *FALSE, otherwise*

N-ary Predicates

- Predicates may N-ary (for some positive integer N) e.g.
  - plus(X,Y,Z)
    - TRUE if $X + Y = Z$
  - Example Usage:
    - $\forall X \; \forall Z \; (less(X,Z) \leftrightarrow \exists Y \; plus(X,Y,Z))$
      - definition of *less than* on natural numbers (in fact, on all positive numbers).

N-ary Predicates [2]

- div(S,D,Q,R)
  - TRUE if S when divided by D yields the quotient Q and the remainder R

- Sample usage:
  - $\forall X \; \forall Z \; (\exists Q \; \exists R \; div(X,Z,Q,R) \;\texttt{<-->}\; (\exists P \; mult(Q,Z,P) \wedge plus(P,R,X))$
    - relation between ***division*** and ***multiplication***

N-ary Predicates [2]

- royalty(Pub,W,Amt,Y)
  - TRUE if published Pub owes writer W a royalty payment of Amt in year Y.
- Sample usage:
  - wrote(A, T, P,Y) --> ∀Y1 (greater(Y1,Y) --> royalty(P,A,R,Y1))
    - if writer A authored a book titled T published by P, in year Y,
    - then P owns A a royalty payment of R in all years Y1 > Y.

innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PREDICATE LOGIC

Expressiveness of Predicates

- **Need for function terms**
- **Using function terms to specify Deterministic Relations**
- **Non-deterministic Relations and Partial Relations**

Predicate Logic: Deterministic Relations

- *Predicates capture relations*
  - In that sense a predicate *may not be deterministic*.
  - Consider the following examples:
    1. *Every child is younger than its mother*
    2. *Ravi and Kavi have the same maternal grandmother.*

Predicate Logic: Need for function terms – Example 1

- A predicate may not be deterministic: e.g.

  Every child is younger than its mother

  - **$\forall C\ \forall M$ child(C) $\wedge$ mother(M,C) --> younger(C, M)**
    - This states that for all children C, for all mothers M of C ...
      - *This is a bit awkward and redundant*!
  - **$\forall C\ \exists M$ child(C) $\wedge$ mother(M,C) --> younger(C, M)**
    - This reads better but it still leaves the possibility that there is more than one M !

Predicate Logic: Need for function terms – Example 2

- A predicate may not be deterministic: e.g.

  Ravi and Kavi have the same maternal grandmother.

  - **$\exists X \, \exists Y \, \exists G$ mother(ravi, X) $\wedge$ mother (kavi, Y) $\wedge$ mother(X, G) $\wedge$ mother(Y,G)**
    - This is similarly awkward.

Predicate Logic: Function terms

- How do you specify something that is deterministic?
  - Use ***function terms***!
    - Function terms are *syntactic forms* to denote a function (i.e. *a deterministic relation*) e.g.
      - **mother(X)**
        - to denote X's mother (whoever that is)
      - **successor(N)**
        - to denote N's successor (whatever that is)
  - Function terms are values

Predicate Logic: Using Function terms – Example 1

- Use function terms to specify deterministic relations:

  e.g. *Every child is younger than its mother*

  - This was specified as:

    $\forall C\ \exists M$ child(C) $\wedge$ mother(M,C) --> younger(C, M)

  but it is better specified using function terms as

    $\forall C$ child(C) --> younger(C, mother(C))

Predicate Logic: Using Function terms – Example 2

- Use function terms to specify deterministic relations:
- e.g. *Ravi and Kavi have the same maternal grandmother.*
  - This was specified as:

    $$\exists X\ \exists Y\ \exists G\ \ \mathbf{mother(ravi,\ X) \wedge mother\ (kavi,\ Y) \wedge mother(X,\ G) \wedge mother(Y,G)}$$

  but it can be better specified using function terms as:

  equals(mother(mother(ravi)), mother(mother(kavi)))

Predicate Logic: Function Terms: Composability

- Note that function terms are ***composable***
  - ***mother(mother(ravi))*** is well-formed
  - ***successor(successor(N))*** is well-formed
- This makes predicates simpler and more readable!
  - e.g. Assume **succ** and **pred** denote the *"successor"* and the *"predecessor"* relations respectively on **integers**.
    - Then the statement "**X's successor's predecessor is X**" can be specified as

      ∀X equals(pred(succ(X)), X)
    - Without function terms, this could be specified as:

      ∀X ∃Y ∃Z succ(Y, X) ∧ pred(Z,Y) --> equals(X,Z)

Predicate Logic: Non-deterministic Relations - Example

- It may be inappropriate to use function terms for relations that are not deterministic:
  - e.g. Ravi's brother is Kavi's friend.
    - **equals(brother(ravi), friend(kavi))**
      - This statement is ambiguous to begin with (*which brother*? *which friend*?)
  - The following is a better specification:

    ∃X brother(ravi,X) ∧ friend(kavi,X)
    - Is this different from the following specification?
      - **∃X brother(ravi,X) --> friend(kavi,X)**

Predicate Logic: Non-deterministic Relations – Examples

- e.g. Ravi's brother is Kavi's friend.
  - It was not intended that *all brothers of Ravi are friends of Kavi*:

∀X brother(ravi,X) --> friend(kavi,X)

[*Construct an example statement (in English) where the latter would be the right form*!]

Predicate Logic: Partial Functions and Function Terms

- Consider this statement:
  - *All of Ravi's friends and their brothers and sisters are welcome to the party*
- This can be specified as:

$\forall$X friend(ravi, X) -->

welcomeToParty(X) $\wedge$ welcomeToParty(brother(X)) $\wedge$ welcomeToParty(sister(X))

- Is there a problem with this specification?

Predicate Logic: Partial Functions and Function Terms

- Consider this statement in English and in Predicate Logic:
  - *All of Ravi's friends and their brothers and sisters are welcome to the party*
  - **∀X friend(ravi, X) -->**

    welcomeToParty(X) ∧ welcomeToParty(brother(X)) ∧ welcomeToParty(sister(X))
- **brother** and **sister** are neither *total* nor *deterministic:*
  - i.e. brother(X) *may not be defined for a given X* and
  - *if defined it may be ambiguous*.
- So this is better specified as:

  ∀X ∀Y ((friend(ravi, X) ∧ (brother(X, Y) ∨ sister(X, Y))) --> (welcomeToParty(X) ∧ welcomeToParty(Y)))

Predicate Logic: Partial Functions

- Note that:
  - ***Universal quantification does not assume existence***!
    - i.e. *the universal set may be empty*!
  - In the above example,
    - *Ravi may have not friends*.
    - *Some of his friends may not have a brother nor a sister*.