Quantification

COMP SCI 2LC3

Ryszard Janicki

Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

Introduction

- We discuss quantification for any symmetric and associative operator
- One can express using quantification
    - Summing a set of values (using addition $+$)
    - Making the disjunction of a set of boolean values (using disjunction $\vee$)
- Quantification is important in the predicate calculus and it is used in the rest of the course

On Types

- In programming languages, a type denotes the (nonempty) set of values that can be associated with a variable
- $\mathbb{B}$ is the set of values true and false
- There are other types

Name	Symbol	Type (set of values)
integer	$\mathbb{Z}$	integers: $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$
nat	$\mathbb{N}$	natural numbers: $0, 1, 2, \ldots$
positive	$\mathbb{Z}^+$	positive integers: $1, 2, 3, \ldots$
negative	$\mathbb{Z}^-$	negative integers: $-1, -2, -3, \ldots$
rational	$\mathbb{Q}$	rational numbers i/j for i, j integers, $j \neq 0$
reals	$\mathbb{R}$	real numbers
positive reals	$\mathbb{R}^+$	positive real numbers
bool	$\mathbb{B}$	booleans: *true* , *false*

On Types

- To be an expression, not only must a sequence of symbols satisfy the normal rules of syntax concerning balanced parentheses, etc., it must also be type correct

- Every expression E has a type t
  - which we can declare by writing $E : t$

Example

$1 : \mathbb{Z}$ $\quad$ $\text{true} : \mathbb{B}$ $\quad$ $\pi : \mathbb{R}$

- Similarly, every variable has a type

Example

$x : \mathbb{Z}$ $\quad$ $p : \mathbb{B}$ $\quad$ $y : \mathbb{R}$

On Types

- The type of a variable might be mentioned in the text accompanying an expression that uses the variable
- It might be given in some sort of a declaration, much like a programming-language declaration var x: integer
- When the type of a variable is not important to the discussion, we may omit it
- We may want to declare the type of a subexpression of an expression, in order to make the expression absolutely clear to the reader

Example

- One might write 1^n as $(1 : \mathbb{Z})^{n:\mathbb{N}}$
- A fully typed expression: $((x : \mathbb{N} + y : \mathbb{N}) * x : \mathbb{N}) : \mathbb{N}$

On Types

- Each function has a type, which describes the types of its parameters and the type of its result
- If the parameters $p_1, \cdots, p_n$ of function f have types $t_1, \cdots, t_n$ nd the result of the function has type r, then f has type $t_1 \times \cdots \times t_n \longrightarrow r$
- We indicate this by writing

$$f : t_1 \times \cdots \times t_n \longrightarrow r$$

Example

Funtion	Type	Typical function application
plus	$\mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z}$	plus(1, 3) or $1 + 3$
not	$\mathbb{B} \longrightarrow \mathbb{B}$	not(true) or $\neg$true
sin	$\mathbb{R} \longrightarrow \langle 0, 1 \rangle$	$\sin(\pi/2), \sin(72.5)$
$\Longrightarrow$	$\mathbb{B} \times \mathbb{B} \longrightarrow \mathbb{B}$	$p \Longrightarrow q$ or $\Longrightarrow(p, q)$

On Types

- It is important to recognize that type and type correctness, as we have defined them, are syntactic notions.
- Type correctness depends only on the sequence of symbols in the proposed expression, and not on evaluation of the expression (in a state).
- For example, $(1/(x : \mathbb{Z})) : \mathbb{R}$ is an expression, even though its evaluation is undefined if $x = 0$.

On Types

- If $E : t$, then $E \in t$ evaluates to true in all states in which E is well defined

Example

Evaluate $2 * i + 3 \in \mathbb{N}$.

- In a textual substitution $E[x := F]$, x and F must have the same type

- Equality $b = c$ is defined only if b and c have the same types

If for example $E = 2 \cdot x + 2^x \cdot y$ and $F = x + 5 \cdot z$ then
$E[x := F] = 2 \cdot (x + 5 \cdot z) + 2^{x+5 \cdot z} \cdot y$.

Other issues have been glossed over

- a notion of subtypes: for example, the natural numbers $\mathbb{N}$ are a subset of the integers $\mathbb{Z}$, so $I : \mathbb{Z}$ and $I : \mathbb{N}$ are both suitable declarations.

- a notion overloading: we need a notion of subtypes, as well as a notion of overloading of both constants and operators, so that the same constants and operators can be used in more than one way.

- a notion of polymorphism: we also need a notion of polymorphism; as an example function $= \ : t \times t \rightarrow \mathbb{B}$ is polymorphic because it is defined for any type t.

Syntax and interpretation of quantification

- $\sum_{i=1}^{n} e$, where e is any expression
- $\sum_{i=1}^{3} i^2$, where i^2 is the expression
- $\sum(i \mid 1 \le i \le n : e)$

  or one can write $+(i \mid 1 \le i \le n : e)$
- This notation is called linear notation

Problem

Use linear notation to write the following expressions:

1. $2+4+6$:
   $+(i \mid 1 \le i \le 3 : 2 \cdot i)$, *or* $+(i \mid 1 \le i \le 7 \wedge even(i) : i)$

2. $2*1+2*3+2*5+2*7$:
   $+(i \mid 0 \le i \le 3 : 2 \cdot (2 \cdot i + 1) i)$, *or*
   $+(i \mid 1 \le i \le 7 \wedge odd(i) : 2 \cdot i)$

3. $1^3+1^4+2^3+2^4$: $+(i,j \mid 1 \le i \le 2 \wedge 3 \le j \le 4 : i^j)$

4. $2*(1^3+1^4+2^3+2^4)+4*(1^3+1^4+2^3+2^4)+6*(1^3+1^4+2^3+2^4)$:
   $+(i,j,k \mid 1 \le i \le 2 \wedge 3 \le j \le 4 \wedge 1 \le k \le 3 : 2 \cdot k \cdot i^j)$

5. $2*1+2*3+2*5+2*7+1^3+1^4+2^3+2^4$:
   $+(i,j,k,l \mid$
   $1 \le i \le 2 \wedge 3 \le j \le 4 \wedge 0 \le k \le 3 \wedge 0 \le l \le 1 :$
   $l \cdot i^j + (1-l) \cdot 2 \cdot (2 \cdot k + 1))$

Syntax and interpretation of quantification

- Let $*$ be any binary operator that satisfy:
  - Symmetry/Commutativity: $b * c = c * b$
  - Associativity: $(b * c) * d = b * (c * d)$
  - Identity u : $u * b = b = b * u$
- A set of values together with an operator * that satisfy the above is called an Abelian monoid

Example

For $*$ and u, we could choose

- $+$ and 0
- $\cdot$ and 1
- $\wedge$ and true
- $\vee$ and false

Syntax and interpretation of quantification

The general form of a quantification over $*$ is exemplified by

$$*(x : t_1, y : t_2 \mid R : P)$$

where:

- Variables x and y are distinct
  - They are called the bound variables or dummies of the quantification
  - There may be one or more dummies
- t_1 and t_2 are the types of dummies x and y
  - If t_1 and t_2 are the same type, we may write $*(x, y : t_1 \mid R : P)$
  - We usually omit the type when it is obvious from the context, writing simply $*(x, y \mid R : P)$

The general form of a quantification over $*$ is exemplified by

$$*(x : t_1, y : t_2 \mid R : P)$$

where:

- R, a boolean expression, is the range of the quantification
  - R may refer to dummies x and y
  - If the range is omitted, as in $*(x, y : t_1 \mid: P)$, then the range *true* is meant
- P, an expression, is the body of the quantification
  - P may refer to dummies x and y
- The type of the result of the quantification is the type of P

- Expression $*(x : X \mid R : P)$ denotes the application of operator * to the values P for all x in X for which range R is true

Example

1. $+(i \mid 0 \leq i < 4 : i \cdot 8)$
2. $\cdot(i \mid 0 \leq i < 3 : i + (i+1))$
3. $\wedge(i \mid 0 \leq i < 2 : i \cdot d \neq 6)$
4. $\vee(i \mid 0 \leq i < 21 : b[i] = 0)$

Syntax and interpretation of quantification

$$\wedge\,(i \;\;|\!:\;\; x \cdot i = 0\,) \tag{1}$$

- (1) asserts that x multiplied by any integer equals 0
- This fact is true only if $x = 0$, so (1) $\iff x = 0$
- The value of (1) in a state depends on the value of x in the state but not on the value of i
- The meaning of (1) does not change when dummy i is renamed:

$$\wedge\,(i \;\;|\!:\;\; x \cdot i = 0\,) \quad = \quad \wedge\,(k \;\;|\!:\;\; x \cdot k = 0\,) \tag{2}$$

Free and *bound* occurrences of variables

Definition

The occurrence of i in the expression i is free.
Suppose an occurrence of i in expression E is free. Then that same occurrence of i is free in

- (E),
- in function application $f(\cdots, E, \cdots)$, and
- in $*(x \mid E : F)$ and $*(x \mid F : E)$

provided i is not one of the dummies in list x.

Definition

Let an occurrence of i be free in an expression E.
That occurrence of i is bound (to dummy i) in the expression $*(x \mid E : F)$ and $*(x \mid F : E)$ if i is one of the dummies in list x.
Suppose an occurrence of i is bound in expression E. Then it is also bound (to the same dummy) in

- (E),
- in function application $f(\cdots, E, \cdots)$, and
- in $*(x \mid E : F)$ and $*(x \mid F : E)$

Example

Which occurrences of i and j are free and which ones are bound?

1. Consider the expression

$$i > 0 \quad \vee \quad \wedge\,(i \mid 0 \leq i \;:\; x \cdot i = 0\,)$$

2. Consider the expression

$$i + j \quad + \quad \Sigma(i \mid 1 \leq i \leq 10 \;:\; b[i]^{j}\,) \\ + \,\Sigma(i \mid 1 \leq i \leq 10 \;:\; \Sigma(j \mid 1 \leq j \leq 10 \;:\; c[i,j]\,)\,)$$

Textual Substitution

Provided $\neg\text{occurs}('y', 'x, F')$, i.e. a dummy of list y will have to be replaced by a fresh variable if that dummy occurs free in x or F.

$$*(y \mid R : P)[x := F] = *(y \mid R[x := F] : P[x := F])$$

Example

1. $+(x \mid 1 \leq x \leq 2 : y)[y := y + z] \quad = \quad ??????$
2. $\wedge \ (i \mid 0 \leq i < n : b[i] = n)[n := m] \quad = \quad ??????$
3. $\vee \ (y \mid 0 \leq y < n : b[y] = n)[n := y] \quad = \quad ??????$
4. $\wedge \ (y \mid 0 \leq y < n : b[y] = n)[y := m] \quad = \quad ??????$

Solution

In the last two examples, dummy y was first replaced by fresh variable j ;- as required by the caveat. Changing the dummy ensures that a free occurrence of Y. in the textual substitution x := F does not become bound.

1. $+(x \mid 1 \leq x \leq 2 \; : \; y\,)[y := y + z] \quad = \quad +\,(x \mid 1 \leq x \leq 2 \; : \; y + z\,)$
2. $\wedge \; (i \mid 0 \leq i < n \; : \; b[i] = n\,)[n := m] \quad = \quad +\,(i \mid 0 \leq i < m \; : \; b[i] = m\,)$
3. $\vee\,(y \mid 0 \leq y < n \; : \; b[y] = n\,)[n := y] \quad = \quad +\,(j \mid 0 \leq j < y \; : \; b[j] = y\,)$
4. $\wedge \; (y \mid 0 \leq y < n \; : \; b[y] = n\,)[y := m] \quad = \quad +\,(j \mid 0 \leq j < n \; : \; b[i] = n\,)$

Rules about quantification

Assume that the operator $*$ is symmetric and associative and has an identity u

- Two additional inferences rules allow substitution of equals for equals in the range and body of a quantification (Leibniz)

1. $$\frac{P=Q}{*(x \mid E[z{:=}P] : S\,) \;=\; *(x \mid E[z{:=}Q] : S\,)}$$

2. $$\frac{R \Longrightarrow P=Q}{*(x \mid R : E[z{:=}P]\,) \;=\; *(x \mid R : E[z{:=}Q]\,)}$$

Rules about quantification

- Axiom, Empty range:

$$*(x \mid \text{false} : P) = u \text{ (the identity of } *)$$

- Axiom, One-point rule: Provided $\neg\text{occurs}('x', 'E')$,

$$*(x \mid x = E : P) = P[x := E]$$

- Axiom, Distributivity: Provided each quantification is defined,

$$*(x \mid R : P) \ * \ *(x \mid R : Q) = *(x \mid R : P * Q)$$

Rules about quantification

- Axiom, Range split: Provided $R \wedge S \iff$ false and each quantification is defined,

$$*(x \mid R \vee S : P) = *(x \mid R : P) * *(x \mid S : P)$$

- Axiom, Range split: Provided each quantification is defined,

$$*(x \mid R \vee S : P) * *(x \mid R \wedge S : P)$$

$=$

$$*(x \mid R : P) * *(x \mid S : P)$$

- Axiom, Range split for idempotent $*$: Provided each quantification is defined,

$$*(x \mid R \vee S : P) = *(x \mid R : P) * *(x \mid S : P)$$

Operation $*$ is idempotent iff $x * x = x$ for all x.
Quantifiers '$\vee$', '$\wedge$', '$\cup$', '$\cap$' are idempotent, while '$+$' and '$\cdot$' are not.

Rules about quantification

- Axiom, Interchange of dummies: Provided each quantification is defined, $\neg occurs('y', 'R')$ and $\neg occurs('x', 'Q')$,

$$*(x \mid R : *(y \mid Q : P)) = *(y \mid Q : *(x \mid R : P))$$

- Axiom, Nesting: Provided $\neg occurs('y', 'R')$,

$$*(x, y \mid R \wedge Q : P) = *(x \mid R : *(y \mid Q : P))$$

- Axiom, Dummy renaming: Provided $\neg occurs('y', 'R, P')$,

$$*(x \mid R : P) = *(y \mid R[x := y] : P[x := y])$$

Example

Show the following for $n : \mathbb{N}$ and dummies $i : \mathbb{N}$

1. $$+(i \mid 0 \leq i < n+1 \;:\; b[i])$$
   $$=$$
   $$+(i \mid 0 \leq i < n \;:\; b[i]) \;+\; b[n]$$

2. $$\cdot(i \mid 0 \leq i < n+1 \;:\; b[i])$$
   $$=$$
   $$b[0] \cdot \;\cdot(i \mid 0 < i \leq n \;:\; b[i])$$

3. $$\forall(i \mid 0 \leq i < n \;:\; b[i] = 0)$$
   $$\Longleftrightarrow \quad \forall(i \mid 0 < i < n \;:\; b[i] = 0) \;\wedge\; b[n] = 0$$

Rules about quantification

Problem

Show the following for $n : \mathbb{N}$ *and dummies* $i : \mathbb{N}$

1. $$*(i \mid 0 \leq i < n+1 \; : \; P\,) = *(i \mid 0 \leq i < n \; : \; P\,) \, * \, P[i := n]$$

2. $$*(i \mid 0 \leq i < n+1 \; : \; P\,) = *(i \mid 0 < i < n+1 \; : \; P\,) \, * \, P[i := 0]$$

3. $$\begin{aligned} & +(i,j \mid 0 \leq i \leq j < n+1 \; : \; c[i,j]\,) \\ = \; & +(i,j \mid 0 \leq i \leq j < n \; : \; c[i,j]\,) \, + \\ & +(i,j \mid 0 \leq i \leq n \; : \; c[i,n]\,) \end{aligned}$$