Predicates and Quantifiers

Dr Waqar Nabi
(with thanks to Dr Gethin Norman)

School of Computing Science
University of Glasgow

“Logic will get you from A to B.
Imagination will take you everywhere."

A. Einstein

Propositional logic

Propositional Logic is the:

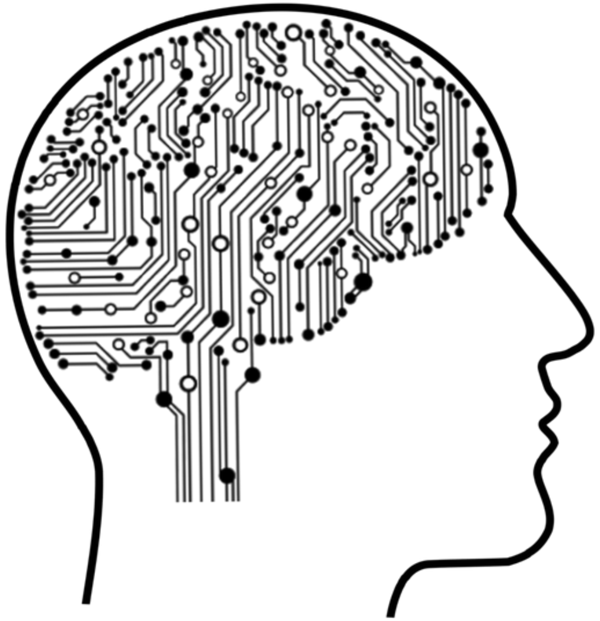
logic of compound statements

built from

simpler statements

using

Boolean connectives



Some applications in computing science

- design of digital electronic circuits
- expressing conditions in programs
- queries to databases and search engines

Predicates and Quantifiers: Outline

Introduction

Predicates

Quantifiers

- free/bound variables and scope
- nesting quantifiers

Logical equivalences

Quantifiers as games

https://upload.wikimedia.org/wikipedia/commons/thumb/a/a4/Free_knowledge_for_all_mankind.PNG/443px-Free_knowledge_for_all_mankind.PNG

Predicate logic – Introduction

Propositional Logic has its limits.

E.g., consider the following two TRUE propositions:

p: Every GA student must study Practical Algorithms

q: Susan is a GA student

It seems like we should be able derive logically the proposition **"Susan must study Practical Algorithms"**, since she belongs a certain SET (she is a GA student). The same should be true for *anyone* who is a GA student. A simple proposition like "Susan studies Practical Algorithms" is does not really express what we wish to state.

What we *do* want to say is this: "X must study Practical Algorithms" is a **true proposition** *only for certain values of X.*

But, with the tools and symbols we have in our hands from Propositional Logic don't enable us to express this.

We need something that allows propositions to be TRUE or FALSE, DEPENDING on (***predicated on***) the value that certain VARIABLES take.

Predicate logic – Introduction – Continues

So, we often want to specify statements which involve variables. Let's look at some examples from Arithmetic:

- e.g. x>3, x=y+3 or x+y=z
- these statements are neither **true** nor **false** when the values of the variables (i.e. x, y and z) are not specified

Predicates allow us to construct propositions which include such statements

Example predicate: **x>3**

- this states "x is greater than 3"
- the variable x is the *subject* of the statement
- the predicate x>3 refers to a *property* the subject can have
- can be expressed **by P(x)** where **P** is the predicate "**is greater than 3**"

Been there, done that!

When we were doing "Proof by induction":

P(n) was a *predicate*

(it's truthness depended on the value of n)

How to write the proof: Template

state clearly what you want to prove, i.e. **P(n)**

basis step

- show that P(1) is true *[or, in general, P(b)]*

Inductive step:

- state clearly P(k), i.e. what you are assuming (*the inductive hypothesis*) *and possible values of k*
- state clearly P(k+1), i.e. what you want to prove
- then show P(k+1) is true *using the inductive hypothesis, i.e. P(k)*
- in many cases this involves show that the lhs = rhs of some equation
  - in such cases start manipulating the lhs keeping an eye on rhs as this is where you want to end up as we as looking for the inductive hypothesis so you can use it

Predicates and Quantifiers: Outline

Introduction

Predicates

Quantifiers

- free/bound variables and scope
- nesting quantifiers

Logical equivalences

Quantifiers as games

P(x)

https://upload.wikimedia.org/wikipedia/commons/thumb/a/a4/Free_knowledge_for_all_mankind.PNG/443px-Free_knowledge_for_all_mankind.PNG

Predicates – Definition

P(x)

A predicate P is a propositional (or Boolean) *function*

- a **mapping** from some **domain** (or *universe*) U to **truth values** (**true** or **false**)
- `P : U → {true,false}` *("→" is not used as the if-then implication symbol here!)*
- for any element `x` of `U`, we have `P(x)` is either **`true`** or **`false`**

Predicates – Definition

P(x)

A predicate P is a propositional (or Boolean) *function*

- a **mapping** from some **domain** (or *universe*) U to **truth values (true or false)**
- `P : U → {true,false}` *("→" is not used as the if-then implication symbol here!)*
- for any element x of U, we have `P(x)` is either **`true`** or **`false`**

Example: let U equals the set of all students at GU

`U = {Susan, Alan, Molly, ... }`

and let the predicate `P(x)` be given by:

x is a GA students

then

- `P("Susan")` is **`true`**
- `P("Alan")` is (let's say) **`false`**
- `P(x)` is true whenever x belongs to a particular set (GA students)

Predicates – Examples

P(x)

A predicate P is a propositional (or Boolean) *function*

- a **mapping** from some **domain** (or *universe*) U to **truth values (true or false)**
- P : U → {**true**,**false**} *("→" is not used as the if-then implication symbol here!)*
- for any element x of U, we have P(x) is either **true** or **false**

Example: let **U** equals the set of integers

$\mathbb{Z}$ = {…,-2,-1,0,1,2,…}

and let the predicate **P(x)** be given by

x>0

then

- P(-2) is **false**
- P(42) is **true**
- P(0) is **false**

Predicates – Examples

P(x)

A predicate P is a propositional (or Boolean) *function*

- a **mapping** from some **domain** (or *universe*) U to **truth values (true or false)**
- `P : U → {true,false}` *("→" is not used as the if-then implication symbol here!)*
- for any element x of U, we have `P(x)` is either **`true`** or **`false`**

Predicates can have more that one argument

Example: let the predicate **`Q(x,y)`** be given by **`x>y`**

- `Q(1,2)` is **`false`**
- `Q(2,1)` is **`true`**

Predicates – Examples

P(x)

A predicate P is a propositional (or Boolean) ***function***

- a **mapping** from some **domain** (or *universe*) U to **truth values (true or false)**
- P : U → {**true**,**false**} *("→" is not used as the if-then implication symbol here!)*
- for any element x of U, we have P(x) is either **true** or **false**

Predicates can have more that one argument

Example: let the predicate R(x,y,z) be given by x+y+z=4

- R(-2,2,0) and R(8,4,4) are **false**
- R(-2,6,0) and R(1,1,2) are **true**

Predicates – More Examples

P(x)

Examples:

- `isOdd(x), isEven(x)`
- `isMarried(x), isTeenager(x), …`
- `isGreaterThan(x,y)`
- `sumsToOneHundred(a,b,c,d,e)`

Predicates – Free and bounded variables

https://www.pickpik.com/notebook-book-to-write-work-note-take-notes-129623

Predicates – Free and bounded variables

Predicates become propositions (true or false) if

- variables are assigned values or
- variables are **bound** to values from its domain U through quantifiers *(more on this soon)*
  - e.g., recalling an earlier example
    - U is the UNIVERSE of all students at GU
    - and we can bind “x” to certain *set of values* (“x” belongs to the set of GA students)
    - the Predicate P(x) – student “x” must study PA – then becomes a proposition
      - and a TRUE one

If not assigned or bound, then in predicate P(x) the variable x is free or unbounded

- i.e. the value of x is not yet specified
- hence P(x) could be either **true** or **false** depending on the value of y

For example: $P(x) \wedge \neg P(1)$ is *not* a compound proposition

- since the variable x is free in P(x) it is therefore not a proposition

Predicates, Free and bounded variables – Example

Let **U** equals the set of integers $\mathbb{Z}$ `= {…,-2,-1,0,1,2,…}`

– put another way: "let the domain of discourse be the set of all integers"

Let `R(x,y,z)` denote the statement `x+y = z`

What is the truth value of

– `R(2,-1,3)`

– `R(x,3,z)`

– `R(3,6,9)`

Predicates, Free and bounded variables – Example

Let U equals the set of integers $\mathbb{Z}$ = {…,-2,-1,0,1,2,…}

- put another way: "let the domain of discourse be the set of all integers"

Let R(x,y,z) denote the statement x+y = z

What is the truth value of

- R(2,-1,3) is **false** since $2 + (-1) = 1 \neq 3$
- R(x,3,z)
- R(3,6,9)

Predicates, Free and bounded variables – Example

Let **U** equals the set of integers $\mathbb{Z}$ = **{…,-2,-1,0,1,2,…}**

- put another way: “let the domain of discourse be the set of all integers”

Let **R(x,y,z)** denote the statement **x+y = z**

What is the truth value of

- R(2,-1,3) is **false** since $2 + (-1) = 1 \neq 3$
- R(x,3,z) is unknown since both x and z are free (unbounded)
- R(3,6,9)

Predicates, Free and bounded variables – Example

Let U equals the set of integers $\mathbb{Z}$ = {…,-2,-1,0,1,2,…}

- put another way: "let the domain of discourse be the set of all integers"

Let R(x,y,z) denote the statement x+y = z

What is the truth value of

- R(2,-1,3) is **false** since $2 + (-1) = 1 \neq 3$
- R(x,3,z) is unknown since both x and z are free (unbounded)
- R(3,6,9) is **true** since $3 + 6 = 9$

QUANTIFIERS

Outline

Introduction

Predicates

Quantifiers

- free/bound variables and scope
- nesting quantifiers

Logical equivalences

Quantifiers as games

First, reviewing some definitions...

- "Universe":
  - "In the formal sciences, the **domain of discourse**, also called the **universe of discourse**, **universal set**, or simply **universe**, is the set of entities over which certain variables of interest in some formal treatment may range."

- Predicate `P(x)`
  - is a propositional (or Boolean) *function*
  - a mapping from some domain (or *universe*) U to **truth values** (true or false)
  - `P : U → {true,false}` *("→" is not used as the if-then implication symbol here!)*
  - for any element x of U, we have P(x) is either **`true`** or **`false`**

Motivation for Quantifiers...

- **We want to be able to BIND free variables to a SET of values from the Universe**
  - This will allow is to convert a Predicate to a *Proposition*, that is, something that can be assigned a truth value
  - A lot of problems present themselves such that we are interested in knowing whether a given Predicate is true for *all* values from a given set, or if it's true for *some (at least one)* values.
- **E.g.:**
  - All values from U *(All students at University of Glasgow)*
  - All values from a certain sub-set of U *(All GA students at University of Glasgow)*
  - At least one value from U *(At least one student from University of Glasgow)*
  - At least one value from a certain sub-set of U *(At least one GA student from University of Glasgow)*

Quantifiers – Universal: $\forall$

Quantifiers – Universal: ∀

The universal quantifier asserts that a property holds for all values of a variable in a given "domain of discourse" (i.e., a set)

Quantifiers – Universal: ∀

The universal quantifier asserts that a property holds for all values of a variable in a given "domain of discourse"

$\forall x.P(x)$ means: "for all values of x the predicate $P(x)$ holds (is `true`)"

However, we should also state the domain (universe of discourse)*

$\forall x \in U.P(x)$ "for all values of x in domain U the predicate $P(x)$ holds"

Example: $\forall x \in \{1,2,3\}.P(x)$ is the same as $P(1) \land P(2) \land P(3)$

- note correspondence between *universal quantification* and *conjunction ("and" operation)*

Quantifiers – Existential

Quantifiers – Existential

The existential quantifier asserts that a property holds

for one or more values of a variable in a given domain of discourse

That is: There EXISTS AT LEAST ONE VALUE for which a property holds

Quantifiers – Existential

The existential quantifier asserts that a property holds for <u>one or more</u> values of a variable in a given domain of discourse

That is: There EXISTS AT LEAST ONE VALUE for which a property holds

$\exists x.P(x)$ "for some values of x the predicate $P(x)$ holds (is `true`)"

However, again we should really state the **domain**

$\exists x \in U.P(x)$ "for some values of x in domain U predicate $P(x)$ holds"

Example: $\exists x \in \{1,2,3\}.P(x)$ is the same as $P(1) \lor P(2) \lor P(3)$

- correspondence between *existential quantification* and *disjunction ("or" operation)*

Side Note

- The world of logic (propositional logic, predicate logic, etc) precedes the world of computer programming (by a long margin).
- Computing is built on top of this world of logic; both the hardware and the software.
- So: you will see strong resonances between the terms and concepts in logic vs computer science, and that is not an accident...
- E.g.:
  - *variables*
  - *free and bound variables*
  - *scope*
  - *functions*
  - *predicates*
  - *truthness of statements/expressions*
  - *etc...*

Outline

Introduction

Predicates

Quantifiers

- binding and scope
- nesting quantifiers

Logical equivalences

Quantifiers as games

Predicates – Free and bounded variables

Predicates become propositions (`true` or `false`) if

- variables are assigned values or
- variables are **bound** with values from its domain U through <u>quantifiers</u>
  - e.g., recalling an earlier example
    - U is the UNIVERSE of all students at GU
    - and we can bound “x” to certain value (“x” belongs to the set of GA students)
    - the Propositon P(x) – student “x” must study PA – then becomes a proposition
      - and a TRUE one
- *quantifiers are coming soon*

If not assigned or bound, then in predicate P(y) the variable y is free or unbounded

- i.e. the value of y is not yet specified
- hence P(y) could be either **`true`** or **`false`** depending on the value of y

For example: $P(y) \wedge \neg P(1)$ is not a compound proposition

- since the variable y is free in P(y) it is therefore not a proposition

Quantifiers – binding and scope

Variables can be **bound** through **quantifiers**

- as we have seen unbound variables are also called free variables

A variable **x** is **bound** to quantifier $\forall x$ or $\exists x$ if

- it appears free **within the** **scope** **of the quantifier**

Quantifiers – binding and scope

Variables can be bound through quantifiers

- as we have seen unbound variables are also called free variables

A variable x is bound to quantifier $\forall x$ or $\exists x$ if

- it appears free within the scope of the quantifier

Examples: $\forall x.(P(y) \wedge Q(x))$ $\exists x.\forall y.(R(y,x) \wedge Q(x))$

for all x, … *there exists at least one x, such that for all y, …*

If a quantifier does not bind any variables it can be removed

Example: $\forall y.\exists x.P(x)$

- since y is not a free variable in $\exists x.P(x)$, the "$\forall y$" quantifier is not used to bind any variables and can be removed

Quantifiers – binding and scope

Variables can be bound through quantifiers

- as we have seen unbound variables are also called free variables

A variable x is bound to quantifier $\forall x$ or $\exists x$ if

- it appears free within the scope of the quantifier

Examples: $\forall x.(P(y) \wedge Q(x))$ — *for all x, ...*

$\exists x.\forall y.(R(y,x) \wedge Q(x))$ — *there exists at least one x, such that for all y, ...*

If a quantifier does not bind any variables it can be removed

Example: $\forall x.\exists x.P(x)$

- since x is not a free variable in $\exists x.P(x)$ (it has been bound by the $\exists x$ quantifier already) so the "$\forall x$" quantifier is not used to bind any variables, and can be removed

Quantifiers – binding and scope

More examples…

For the formula $(\forall x.P(x)) \land Q(x)$ the variable x appearing in $Q(x)$ is outside of the scope of the "$\forall$x" quantifier, and is therefore *free*

However in $\forall x.(P(x) \land Q(x))$ both x's are within the scope of "$\forall$x"

$(\forall x.P(x)) \land (\exists x.Q(x))$ is a valid formula and: <u>the x's are different!</u> (and both are bound)

Take note: <u>parentheses (brackets) are important</u>

Also we often omit "$\in U$" to simplify the presentation

Outline

Introduction

Predicates

Quantifiers

- binding and scope
- nesting quantifiers

Logical equivalences

Quantifiers as games

Nesting of quantifiers – Ordering matters → NEW SLIDE

$\forall x.\exists y.Q(x,y)$ for all x we can find a y such that $Q(x,y)$ holds

$\exists y.\forall x.Q(x,y)$ we can find a y such that $Q(x,y)$ holds for all x

If $\exists y.\forall x.Q(x,y)$ holds, then $\forall x.\exists y.Q(x,y)$ also holds

- if we can find an y such that $Q(x,y)$ holds for all x
  then clearly for any x we can find a y such that $Q(x,y)$ holds

If $\forall x.\exists y.Q(x,y)$ holds, then it does **not** follow $\exists y.\forall x.Q(x,y)$ holds

- if for any x we can find a y such that $Q(x,y)$ holds
  then the y's might be different for each x so it does **not** mean
  we can find a y such that $Q(x,y)$ holds for all x

Motivating Example

Let's say P(x,y) means x (a person) has their birthday on y (a date)

What do these mean? Are they saying the same thing? Which of them are TRUE?

$\forall x . \forall y . P(x,y)$

$\forall x . \exists y . P(x,y)$

$\exists x . \forall y . P(x,y)$

$\exists x . \exists y . P(x,y)$

Motivating Example

Let's say P(x,y) means x (a person) has their birthday on y (a date)

What do these mean? Are they saying the same thing? Which of them are TRUE?

$\forall x. \forall y. P(x,y)$

$\forall x. \exists y. P(x,y)$

$\exists x. \forall y. P(x,y)$

$\exists x. \exists y. P(x,y)$

Nesting of quantifiers – Ordering matters

Let U equals the set of integers $\mathbb{Z}=\{\ldots,-2,-1,0,1,2,\ldots\}$

- put another (way: “let the domain of discourse be the set of all integers”

Let $P(x,y)$ denote the statement $x>y$

- $\forall x.\forall y.P(x,y)$
- $\forall x.\exists y.P(x,y)$
- $\exists x.\forall y.P(x,y)$
- $\exists x.\exists y.P(x,y)$

Each of these as a different meaning…

Nesting of quantifiers – Ordering matters

Let U equals the set of integers $\mathbb{Z}$={…,-2,-1,0,1,2,…}

- put another way: “let the domain of discourse be the set of all integers”

Let P(x,y) denote the statement x>y

- ∀x.∀y.P(x,y) for all integers x and for all integers y we have x>y
  this statement is **false;** take x=y=1, then 1>1 does not hold
  (recall: just one false case is enough to show that the entire proposition is false)

Nesting of quantifiers – Ordering matters

Let U equals the set of integers $\mathbb{Z}=\{\ldots,-2,-1,0,1,2,\ldots\}$

- put another way: “let the domain of discourse be the set of all integers”

Let P(x,y) denote the statement x>y

- ∀x.∀y.P(x,y) for all integers x and for all integers y we have x>y
  this statement is **false** take x=y=1, then 1>1 does not hold
- ∀x.∃y.P(x,y) for all integers x there exists an integer y such that x>y
  this statement is **true** take for example y=x−1

Nesting of quantifiers – Ordering matters

Let U equals the set of integers $\mathbb{Z}$={…,-2,-1,0,1,2,…}

- put another way: “let the domain of discourse be the set of all integers”

Let P(x,y) denote the statement x>y

- ∀x.∀y.P(x,y) for all integers x and y we have x>y
  this statement is **false** take x=y=1, then 1>1 does not hold
- ∀x.∃y.P(x,y) for all integers x there exists an integer y such that x>y
  this statement is **true** take for example y=x-1
- ∃x.∀y.P(x,y) there exists an integer x such that for all integers y, x>y
  this statement is **false** take y=x (or y=x+1), then x≤y
  (This statement is in effect saying: “There exists a largest integer”, which just ain’t so!)

Nesting of quantifiers – Ordering matters

Let U equals the set of integers $\mathbb{Z}=\{\ldots,-2,-1,0,1,2,\ldots\}$

- put another way: "let the domain of discourse be the set of all integers"

Let P(x,y) denote the statement x>y

- $\forall x.\forall y.P(x,y)$ for all integers x and y we have x>y
  this statement is **false** take x=y=1, then 1>1 does not hold
- $\forall x.\exists y.P(x,y)$ for all integers x there exists an integer y such that x>y
  this statement is **true** take for example y=x-1
- $\exists x.\forall y.P(x,y)$ there exists an integer x such that x>y for all integers y
  this statement is **false** take y=x (or y=x+1), then x≤y
- $\exists x.\exists y.P(x,y)$ there exists integer x and there exists an integer y such that x>y
  this statement is **true** take x=2 and y=1, then x>y

Nesting of quantifiers – Ordering matters!

Nesting of quantifiers – Ordering matters

Let $P(x,y)$ denote the statement (predicate) $x>y$

$$\forall x \in \{1,2\}.\forall y \in \{3,4\}.P(x,y) \equiv P(1,3) \land P(1,4) \land P(2,3) \land P(2,4)$$

$\land$ = AND
$\lor$ = OR

Nesting of quantifiers – Ordering matters

Let $P(x,y)$ denote the statement (predicate) $x>y$

$$\forall x \in \{1,2\}.\forall y \in \{3,4\}.P(x,y) \equiv P(1,3) \land P(1,4) \land P(2,3) \land P(2,4)$$

$$\forall x \in \{1,2\}.\exists y \in \{3,4\}.P(x,y) \equiv (P(1,3) \lor P(1,4)) \land (P(2,3) \lor P(2,4))$$

$\land$ = AND
$\lor$ = OR

Nesting of quantifiers – Ordering matters

Let $P(x,y)$ denote the statement (predicate) $x>y$

$\forall x \in \{1,2\}.\forall y \in \{3,4\}.P(x,y) \equiv P(1,3) \land P(1,4) \land P(2,3) \land P(2,4)$

$\forall x \in \{1,2\}.\exists y \in \{3,4\}.P(x,y) \equiv (P(1,3) \lor P(1,4)) \land (P(2,3) \lor P(2,4))$

$\exists x \in \{1,2\}.\forall y \in \{3,4\}.P(x,y) \equiv (P(1,3) \land P(1,4)) \lor (P(2,3) \land P(2,4))$

$\land$ = AND
$\lor$ = OR

Nesting of quantifiers – Ordering matters

Let $P(x,y)$ denote the statement (predicate) $x>y$

$\forall x \in \{1,2\}.\forall y \in \{3,4\}.P(x,y) \equiv P(1,3) \wedge P(1,4) \wedge P(2,3) \wedge P(2,4)$

$\forall x \in \{1,2\}.\exists y \in \{3,4\}.P(x,y) \equiv (P(1,3) \vee P(1,4)) \wedge (P(2,3) \vee P(2,4))$

$\exists x \in \{1,2\}.\forall y \in \{3,4\}.P(x,y) \equiv (P(1,3) \wedge P(1,4)) \vee (P(2,3) \wedge P(2,4))$

$\exists x \in \{1,2\}.\exists y \in \{3,4\}.P(x,y) \equiv P(1,3) \vee P(1,4) \vee P(2,3) \vee P(2,4)$

$\wedge$ = AND
$\vee$ = OR

Examples (self study)

For **P** and **Q** the **universe of discourse** (domain) is set of integers **U**

- $P(x)$ denote the statement $x>3$
- $Q(x,y)$ denote the statement $x+y = 0$

Consider the following:

- $\forall x.P(x)$ this is **false**, for example take $x=2$
- $\forall x.\exists y.Q(x,y)$ this is **true**, for any x take $y = -x$
- $\exists y.\forall x.Q(x,y)$ this is **false**, no single value of y for all values of x

Notice again the ordering of the quantifiers is important

Again have omitted "$\in U$" to simplify the presentation

Outline

Introduction

Predicates

Quantifiers

- binding and scope
- nesting quantifiers

Logical equivalences

Equivalences – Ordering

Above we said that the ordering of quantifiers was important

However, *we can swap the ordering when they are of the same form*:

$\forall x.\forall y.Q(x,y) \equiv \forall y.\forall x.Q(x,y)$

$\exists x.\exists y.Q(x,y) \equiv \exists y.\exists x.Q(x,y)$

Equivalences – Quantifier Negation Laws

$$\neg(\exists x.\neg P(x)) \equiv \forall x.P(x)$$

Equivalences – Quantifier Negation Laws

$$\neg(\exists x.\neg P(x)) \equiv \forall x.P(x)$$

to put it another way

there does not exist an x such that P(x) does not hold ≡ P(x) holds for all x

or, to put it another way

it is not the case that there exists an x such that P(x) does not hold ≡ P(x) holds for all x

Equivalences – Quantifier Negation Laws

$$\neg(\exists x.\neg P(x)) \equiv \forall x.P(x)$$

or, to put it another way

there does not exist an x such that P(x) does not hold ≡ P(x) holds for all x

or, to put it another way

it is not the case that there exists an x such that P(x) does not hold ≡ P(x) holds for all x

$$\neg(\forall x.\neg P(x)) \equiv \exists x.P(x)$$

Equivalences – Quantifier Negation Laws

$$\neg(\exists x.\neg P(x)) \equiv \forall x.P(x)$$

or, to put it another way

there does not exist an x such that P(x) does not hold ≡ P(x) holds for all x

or, to put it another way

it is not the case that there exists an x such that P(x) does not hold ≡ P(x) holds for all x

$$\neg(\forall x.\neg P(x)) \equiv \exists x.P(x)$$

or, to put it another way

it is not the case that for all x, P(x) does not hold ≡ there exists an x such that P(x) holds

or, to put it another way

¬P(x) does not hold for all x ≡ there exists an x such that P(x) holds

Summary

Summary

Predicates

- a Boolean function i.e. returns either **`true`** or **`false`**

Quantifiers

- universal quantifier asserts a property holds for all values of a variable
- existential quantifier asserts a property holds for some value of a variable

Nesting quantifiers and binding

- need to be careful
- order of quantifiers matters

Logical equivalences

- using negation can define one type of quantifier with the other

Laws of Equivalence Summary Sheet

Identity laws:

- $P \wedge \texttt{true} \equiv P$
- $P \vee \texttt{false} \equiv P$

Domination laws:

- $P \vee \texttt{true} \equiv \texttt{true}$
- $P \wedge \texttt{false} \equiv \texttt{false}$

Idempotent laws:

- $P \wedge P \equiv P$
- $P \vee P \equiv P$

Double negation law:

- $\neg(\neg P) \equiv P$

Commutative laws:

- $P \wedge Q \equiv Q \wedge P$
- $P \vee Q \equiv Q \vee P$

Associative laws:

- $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$
- $(P \vee Q) \vee R \equiv P \vee (Q \vee R)$

Distributive laws:

- $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
- $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

De Morgan laws:

- $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
- $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

Contradiction and tautology laws:

- $P \wedge \neg P \equiv \texttt{false}$
- $P \vee \neg P \equiv \texttt{true}$

Implication law:

- $P \rightarrow Q \equiv \neg P \vee Q$

Exclusive or and biconditional laws:

- $P \oplus Q \equiv (P \vee Q) \wedge \neg(P \wedge Q)$
- $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$

Quantifier laws:

- $\forall x.\, \forall y.\, Q(x, y) \equiv \forall y.\, \forall x.\, Q(x, y)$
- $\exists x.\, \exists y.\, Q(x, y) \equiv \exists y.\, \exists x.\, Q(x, y)$
- $\neg(\exists x. \neg P(x)) \equiv \forall x.\, P(x)$
- $\neg(\forall x. \neg P(x)) \equiv \exists x.\, P(x)$
- $\forall x.(P(x) \wedge Q(x)) \equiv \forall x.P(x) \wedge \forall x.\, Q(x)$
- $\exists x.(P(x) \vee Q(x)) \equiv \exists x.P(x) \vee \exists x.\, Q(x)$

References

1. Schneider, G. Michael, Judith Gersting, and Sara Baase. *Invitation to Computer Science : Java Version (2nd Edition).*
2. Goodrich, Michael T., Roberto Tamassia, and Michael H. Goldwasser. *Data structures and algorithms in **Java***. John Wiley & Sons (5th Edition onwards).
3. Goodrich, Michael T., Roberto Tamassia, and Michael H. Goldwasser. *Data structures and algorithms in **Python***.
4. Kent D. Lee, Steve Hubbard, *Data Structures and Algorithms with Python*, Springer, 2015.
5. Cormen, Thomas H., et al. *Introduction to algorithms*. MIT press, 2009
6. Sedgewick, Robert. *Algorithms in Java, Parts 1-4 (Fundamental Algorithms, Data Structures, Sorting, Searching)*. Addison Wesley, 2002.
7. *Discrete Mathematics & its Applications*, Kenneth H. Rosen (5th, 6th, 7th or 8th Edition).
8. Gerard O'Regan, *Concise Guide to Formal Methods*.
9. F Oggier, *Lecture Notes, Discrete Mathematics*.