

PART IVa: QUANTIFIERS

Version 2.18: 2004/05/14 14:24:12.087 GMT-5

John Greiner
Ian Barland

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License *

Abstract

Introducing quantifiers, to upgrade from propositional logic to first-order logic.

partIVa: quantifiers

1 Talking about unnamed items

Suppose we want to express a statement like "there is a location which has two neighbors" (which is true, at least for the domain of WaterWorld board locations), or "all actors have co-starred with Kevin Bacon¹" (which isn't true, at least for the domain of all Hollywood actors). As it stands, we can only formulate these in an awkward way – by talking about *specific* (constant) locations like A and G , or specific actors like Ewan McGregor² and Cameron Diaz³. To talk about all locations, or actors, we're forced to make huge formulas such as $(\text{nhbr}(Z, Y) \wedge \neg \text{nhbr}(Z, A) \wedge \neg \text{nhbr}(Z, B) \wedge \dots \wedge \neg \text{nhbr}(Z, X))$, just to express "there is a location which has only one neighbor".

We'll redress this by introducing two **quantifiers**, " \exists " ("there exists") and " \forall " ("for all"). For example, "all actors have co-starred with Kevin Bacon" will be written $\forall a. \text{costarred}(a, \text{Kevin Bacon})$. For "there is a location which has (at least) two neighbors", we'll start with "there exists a location x . . .", written " $\exists x$. . .".

"For all" is really just an abbreviation for a large conjunction, while "exists" is a disjunction. How large? As big as your domain – which actually could be very small, or it could be infinitely large. Even aside from the fact that we can't write down an infinitely large conjunction or disjunction, quantifiers let us form the conjunction without having to select a domain in advance.

To continue with our WaterWorld example, how can we express the concept " x has (at least) two neighbors"? Well, we'll rephrase this as, "there exist distinct locations, y and z , which each neighbor x ", written " $\exists x. \exists y. \exists z. (\neg(y = z) \wedge \text{nhbr}(x, y) \wedge \text{nhbr}(x, z))$ ". We need the condition $\neg(y = z)$ in that formula to ensure that we have distinct locations. Compare to the algebraic equation $(x + y = 4)$ in which one possible solution is $(x = y = 2)$. Variables act the same way in both logic and algebra: different variables can happen to take on the same value.

*<http://creativecommons.org/licenses/by/1.0>

¹<http://us.imdb.com/Name?Bacon,+Kevin>

²<http://us.imdb.com/Name?McGregor,+Ewan>

³<http://us.imdb.com/Name?Diaz,+Cameron>

We use quantifiers all the time in natural language. Consider the following examples, where we provide a natural English wording together with an equivalent phrasing that makes the quantification more explicit. We'll take the translations with a grain of salt, since sometimes people can disagree on the exact details of the intended English meaning. Such ambiguity can sometimes be a rich source of creativity, but it's not tolerable when documenting safety properties of software. While some of these examples are a bit frivolous, in general quantifiers let us precisely capture more interesting concepts in type-checking, data structures such as trees and hash tables, circuit specifications, etc.

Quantification in English

Natural English
"All's well that ends well."
"If you don't love yourself, you can't love anybody else."
"N*Sync is the best band ever!"
A casually subtle line from <i>Something About Mary</i> : "Every day is better than the next."
A buggy line from a song (<i>Everybody Loves My Baby</i> , Jack Palmer & Spencer Willson, 1924): "Everybody loves
"Every neighbor of x is unsafe."
"There is a safe location that is a neighbor of x , if $\text{num}(x) < 3$."
"If you've seen one episode, you've seen 'em all."
"Somebody loves everybody."
"There is someone for everybody."

2 First-order logic: WFFs revisited

We originally defined a well-formed formula (WFF) for propositional logic; we'll extend this to WFFs for **first-order logic**, also known as **predicate logic**. At the same time, we'll more precisely define the binding of variables.

This logic allows use of both functions and relations. Since these functions' outputs are not Booleans (otherwise, we'd call them relations), but rather data than can be used as a relation's input, we separate the syntax into that of **terms** and formulas. Terms are all the possible inputs for a relation.

Definition 1: term

1. A variable.

Example:

a, b, \dots

2. A constant.

Example:

WaterWorld location F , Kevin Bacon, or the number 3.

3. A function applied to one or more terms.

Example:

successor (3)

Definition 2: Well-Formed Formula (WFF) for first-order logic

1. A constant: true or false.
2. An **atomic formula**: a relation symbol applied to one or more terms.

Example:

$$\text{nhbr}(x, F)$$

3. A negation of a WFF, $\neg\phi$.
4. A conjunction of WFFs, $(\phi \wedge \psi)$.
5. A disjunction of WFFs, $(\phi \vee \psi)$.
6. An implication of WFFs, $(\phi \rightarrow \psi)$.
7. A **universal quantification** of a WFF, $\forall x.\phi$.

Example:

$$\forall x.\text{nhbr}(x, F)$$

8. An **existential quantification** of a WFF, $\exists x.\phi$.

Example:

$$\exists x.\text{nhbr}(x, F)$$

While a formula is just a piece of syntax, the meaning of its connectives, including the quantifiers, is part of the definition of a WFF. However, as previously discussed, the meaning of a WFF also depends on the interpretation⁴ we give to its relations.

2.1 Examples**Example 7:**

The following formula is a simple application of symmetry. $((\forall x.\forall y. (\text{near}(x, y) \rightarrow \text{near}(y, x)) \wedge \text{near}(\text{Sue}, \text{Joe}))$

While it is certainly true under the intended interpretation, it is also true under *any* interpretation. Such formulas are called **valid**. Valid first-order formulas are the natural analogue of tautological propositional formulas.

Example 8:

$\forall x. ((\text{even}(x) \wedge \text{prime}(x)) \rightarrow (x = 2))$ is a mathematical fact, in the standard interpretation of arithmetic.

While technically not allowed by our term (Definition 1) and formula (Definition 2) syntax, we'll continue using infix notation for common mathematical functions and relations, as in the previous example (Example 8).

Exercise 1:

The previous example (Example 8) used the relations *even* and *prime*. Of course, to use such relations, they must either be defined directly by the interpretation, or be defined in terms of functions and relations provided by the interpretation.

How would you define these two relations in terms of the basic numerical functions (addition, multiplication, ...) and relations ($=$, $<$, $>$)?

⁴<http://cnx.rice.edu/content/m10726/latest/>

Solution:

”Evenness” is a straightforward translation from ”A number n is even, iff it is twice some other number k ”: $even(n) \equiv \exists k. (n = (2 * k))$

There are many equivalent ways to define primality, just as there many algorithms for checking primality. One straightforward solution is $noncomposite(n) \equiv \forall j. \forall k. (((j * k) = z) \rightarrow ((j = 1) \vee (k = 1)))$. Well, this almost expresses ”prime”, except that $n=1$ satisfies this formula. A mathematician points out that just as 0 is neither positive nor negative, 1 is neither prime nor composite; as stated this formula actually captures ”noncomposite”, oops. There are several ways to upgrade this to exactly capture ”prime”.

ASIDE: 1 is called a ”unit”. If we consider the domain of all integers (not just natural numbers), the idea of primality still makes sense; -17 is also prime; and -1 is also another unit. Similarly, considering the domain of ”complex integers” $\{ a+bi \mid a, b \text{ in } \mathbb{Z} \}$ (could be written ” $\mathbb{Z}+Zi$ ”), then i and $-i$ are also units. How might we generalize our definition of prime, to work in these further interpretations?

A similar, equivalent formula to the above is $noncomposite(n) \equiv \neg \exists j. \exists k. (((j * k) = z) \wedge \neg (j = 1) \wedge \neg (k = 1))$.

Exercise 2:

A hypothesis about natural numbers is known as Goldbach’s Conjecture⁵. It states that all even numbers greater than two are the sum of two primes. It is one of the oldest still-unsolved problems about numbers. How would you write this conjecture as a WFF?

Solution:

$\forall n. ((even(n) \wedge (n > 2)) \rightarrow \exists p. \exists q. (prime(p) \wedge prime(q) \wedge ((p + q) = n)))$

Enough about number theory. Let’s look at some examples about common data structures and some about our favorite problem, WaterWorld.

Example 9:

If your program uses binary search trees, you need to know $\forall node. (data(left(node)) \leq data(node) \wedge data(right(node)) \geq data(node))$.
 . If these trees are also balanced, you need to know $\forall node. (height(left(node)) = height(right(node)) \vee height(left(node)) = height(right(node)) + 1)$.
 . Again, these assume the implied interpretations.

Example 10:

We would like to be able to state that the output of a sorting routine is, in fact, sorted. Let’s assume we’re sorting arrays into ascending order.

To talk about the elements of an array, we need a function that gets the i -th element. Instead of a function written, say, $ith(A, i)$, we’ll use the more common programming notation, $A[i]$.

Basically, we want to state that each element is greater than or equal to the previous one. However, just like in a program, we need to ensure our formula doesn’t index outside the bounds of the array. For this example, we’ll assume that an array’s indices are zero to (but not including) size(A).

$sorted(A) \equiv \forall i. ((1 \leq i \wedge i < size(A)) \rightarrow A[i - 1] < A[i])$

⁵<http://www.wikipedia.org/wiki/Goldbachs.conjecture>

When proving things about programs, it's often useful to realize there are alternate ways of defining things. So, let's see a couple more definitions.

We could change or indexing slightly: $sorted(A) \equiv \forall i. ((0 \leq i \wedge i < size(A) - 1) \rightarrow A[i] < A[i + 1])$

Or we could state that the ordering holds on every pair of elements: $sorted(A) \equiv \forall i. \forall j. ((0 \leq i \wedge i < size(A) \wedge 0 \leq j \wedge j < size(A) \wedge i < j) \rightarrow A[i] \leq A[j])$. This definition isn't any stronger, but it makes an additional property explicit. Generally, you'd find it harder to prove that this formula was true, but once you did, you'd find it easier to use this formula to prove other statements.

Exercise 3:

One simple WaterWorld fact is that if a location has no unsafe neighbors, then its number of adjacent pirates is zero. Furthermore, the implication goes both ways. How would you state that as a WFF?

Solution:

$$\forall x. (\forall y. (nhbr(x, y) \rightarrow safe(y)) \leftrightarrow num(x) = 0)$$

Exercise 4:

How would you make a similar statement about the number of adjacent pirates being one?

Solution:

There are various solutions, but they all must capture the same idea: there exists exactly one unsafe neighbor. This solution states that in two parts:

- There exists an unsafe neighbor, u .
- Every unsafe neighbor is u .

Together, these two parts imply there is only one such u .

$$\forall x. (\exists u. (nhbr(x, u) \wedge unsafe(u) \wedge \forall y. (nhbr(x, y) \rightarrow (unsafe(y) \leftrightarrow y = u))) \leftrightarrow num(x) = 1)$$

Statements like these are provable from a set of first-order WaterWorld domain axioms⁶.

In the above examples we often re-used variable names, even within the same formula. This shouldn't be surprising or confusing, since we do the same thing in programs (another formal language). In fact, the same notions of **bound** and **free** variables occur in both situations. An occurrence of variable x is bound if it is in the body of a quantifier $\forall x \dots$ or $\exists x \dots$. Otherwise, the occurrence is free.

For example, in $\forall x. likes(x, y)$, the variable y is free but x is not. So this is a statement about y ; we can't evaluate this to true/false until we get some context for y . It's useful as a subpart for some bigger formula.

A given variable name can have both bound and free occurrences within the same formula, as in $(x \wedge \forall x. R(x))$. In essence, these are two different variables who simply happy to have the same name, but from context it's clear which one is meant. (In programming language terms, we'd say that the inner x **shadows** the outer x .)

Looking back at our previous examples, we can see that many of the formulas we made had no free variables – all variables were bound by some quantifier in the formula. The truth of such formulas depends only on the interpretation and not on any additional knowledge

⁶<http://cnx.rice.edu/content/new0/latest/>

about what any free variables refer to. Thus, these formulas are common and important enough that we give them a special name, **sentences**.

The choice of variable names is arbitrary, except possibly for their mnemonic value. So, it is always possible to consistently rename all instances of a bound variable. (This is called **α -renaming**.) Furthermore, we can repeat this as necessary to ensure all quantifiers in a formula have distinct names. This is useful to avoid potential confusion, especially in definitions based on the syntax of WFFs, such as the upcoming inference rules.

Example 11:

So, $(\forall x.A(x) \wedge \exists x.B(x) \wedge \forall x.C(x))$ is equivalent to the more readable $(\forall x.A(x) \wedge \exists y.B(y) \wedge \forall z.C(z))$.

2.2 WFFs about subdomains

In some of the above examples, we wanted to ensure that a quantified variable ranged over some set S . In shorthand, people often write " $\forall x \in S$, [blah]". However, this isn't allowed in most definitions, including ours, of first-order logic. Our definition of WFFs allows " $\forall x \dots$ ", but there's no "is an element of" built-in.

There are three ways to express what we want:

- We may be able to use set S as our domain. Then, the formula does not need to mention S , since this knowledge is built-in to our interpretation.
- If we have a unary relation $S()$, interpreted as "is in the set S ", then we can just write " $\forall x.(S(x) \rightarrow \text{[blah]})$ ". This has exactly the intended meaning.
- Sometimes we don't have the relation S though, in which case we can try faking it, seeing if there's some other way of characterizing things in that set, still using \rightarrow . For example, suppose our domain is the natural numbers, and we want to say "for all even numbers x , [kerblooey]". Using the definition seen in a previous exercise (Exercise 1), we can write $\forall x.(\exists k.x = 2 * k \rightarrow \text{[kerblooey]})$

2.3 CNF and DNF revisited (Optional)

In first-order logic, normal forms are still useful for providing a notion of a canonical form. However, their other benefit of corresponding closely to truth tables does not apply here, since truth tables aren't useful for first-order logic.

A formula in **Prenex Conjunctive Normal Form**, or **Prenex CNF**, has a body in CNF preceded by a series of quantifiers. Similarly, a formula in **Prenex Disjunctive Normal Form**, or **Prenex DNF**, has a body in DNF preceded by a series of quantifiers.

Example 12:

Assuming ϕ is in CNF, then the following are each in prenex CNF. On the other hand, if ϕ is in DNF, these are in prenex DNF.

- ϕ
- $\forall x.\phi$
- $\exists x.\forall y.\exists z.\phi$

Every formula has an equivalent prenex CNF formula and equivalent prenex DNF formula. For brevity, we'll skip proving this.