CMPSCI 250: Introduction to Computation

Lecture 7: Quantifiers and Languages
David Mix Barrington
18 September 2013

Quantifiers and Languages

- Quantifier Definitions
- Translating Quantifiers
- Types and the Universe of Discourse
- Some Quantifier Rules
- Multiple Quantifiers
- Languages and Language Operations
- Language Concatenation and Kleene Star

The Existential Quantifier

- Suppose that P(x) is a predicate, where x is a variable of type T. For example, T might be a set of dogs and P(x) might mean "dog x is a poodle".
- The **quantified statement** ∃x: P(x) means "there exists a dog x such that x is a poodle", or "there is at least one poodle in T". The symbol "∃" is called the **existential quantifier**.

Universal Quantifiers and Binding

- The quantified statement ∀x: P(x) means "for all dogs x, x is a poodle" or "every dog in T is a poodle". The symbol ∀ is the **universal quantifier**.
- Each quantifier **binds** a free variable, making it a **bound variable**. Both the statements ∃x: P(x) and ∀x: P(x) are propositions, as they have no free variables -- they are either true or false once T and P are defined.

Translating Quantifiers

- We translate quantified statements into English very carefully and mechanically -- after making a first translation we can adapt to something that sounds more natural.
- In translating "∃x: P(x)", we say "there exists an x" for "∃x", "such that" for the colon, and then translate P(x). If we want to emphasize the type of x, we might say "there exists an x of type T such that P(x) is true". In our example, this was "there exists a dog x such that x is a poodle".

Translating Quantifiers

- In translating "$\forall x: P(x)$", we say "for all x" for "$\forall x$", nothing for the colon (it becomes a comma), and then translate P(x). Again we may emphasize the type -- "for all x of type T, P(x) is true". In the example, "for all dogs x, x is a poodle".
- If there are multiple quantifiers the rules for translating the colon change a bit. We translate "$\exists x: \exists y: P(x) \wedge P(y)$" as "there exist a dog x and a dog y such that both are poodles".

Types and the Universe of Discourse

- The type of the bound variable is an important part of the meaning of a quantified statement.
- Every variable is typed, and "there exist" and "for all" refer to the type, whether or not we state this in our translation.
- Traditionally logicians have referred to the type as the **universe of discourse** for the variable.

Types and Universal Quantifiers

- This is particularly important for universal quantifiers.
- The statements "all deer have antlers" and "all animals have antlers" have different meanings but might both be written $\forall x: A(x)$ -- the difference would be the type of the variable x. In the first the type of x is "deer", in the second it is "animals".

Quantifiers and Empty Types

- We can quantify over types that contain *no* elements -- let's take the set U of unicorns as our example.
- Any statement of the form $\exists x: P(x)$ is false if the type of x is U, as it says "there exists a unicorn such that" something. But any statement of the form $\forall x: P(x)$ is true.
- It is true that all unicorns are green, and also true that all unicorns are not green. (For that matter, it is true that all unicorns are both green and not green -- $\forall x: G(x) \wedge \neg G(x)$ in symbols.)

Some Rules for Quantifiers

- Whenever our original predicate has more than one free variable, we need more than one quantifier to bind them and form a proposition. Let D be a set of dogs and C be a set of colors, and let H(d, c) mean “dog d has color c”.
- If I say ∃d: ∃c: H(d, c), this means “there exists a dog c and a color c such that d has c”. Note that the first colon translates as “and” rather than as “such that”.

Quantifiers of the Same Kind

- If instead of $\exists d: \exists c: H(d, c)$, we said $\exists c: \exists d: H(d, c)$, this would mean exactly the same thing. Similarly $\forall d: \forall c: H(d, c)$ and $\forall c: \forall d: H(d, c)$ both mean "every dog has every color".
- We can switch *similar* adjacent quantifiers, but we will soon see that switching *dissimilar* quantifiers changes the meaning.

Quantifer DeMorgan Rules

- We have two "Quantifier DeMorgan" rules to relate quantifiers to negation. We can simplify $\neg\exists x: P(x)$ as $\forall x: \neg P(x)$, and $\neg\forall x: P(x)$ as $\exists x: \neg P(x)$.
- A universal statement is true if and only if there is not a **counterexample** to it.
- This rule explains the convention about empty types: "All unicorns are green" is equivalent to "there does not exist a non-green unicorn" which is clearly true.

Clicker Question #1

- Consider the statement "All dogs like to go for walks". Which of the following statements is *equivalent* to it?
- (a) There exists a dog that does not like walks.
- (b) If a dog does not exist, it does not like walks.
- (c) It is not the case that there exists a dog that does not like walks.
- (d) There exists a dog that likes to go for walks.

Answer #1

- Consider the statement "All dogs like to go for walks". Which of the following statements is *equivalent* to it?
- (a) There exists a dog that does not like walks.
- (b) If a dog does not exist, it does not like walks.
- (c) *It is not the case that there exists a dog that does not like walks.*
- (d) There exists a dog that likes to go for walks.

Multiple Quantifiers

- Let's look more closely at the effect of multiple dissimilar quantifiers. Let x and y be of type `natural` and consider $x \leq y$, which has two free variables.
- If we say $\exists x: x \leq y$, this statement still has y as a free variable, so its meaning depends on y. It says that there is a natural less than or equal than y, and this statement is true for any y. (For example, x could be y itself).

Multiple Quantifiers

- Similarly $\exists y: x \leq y$ has one free variable, x, and is true for any x.
- We can also form $\forall x: x \leq y$, which is never true for any y, and finally $\forall y: x \leq y$ which is true if $x = 0$ but false for any other x.
- Now we can make propositions from any of these four statements by quantifying the remaining free variable.

Making Propositions

- The statements $\exists x: \exists y: x \leq y$ and $\forall x: \forall y: x \leq y$ are true and false respectively, and can have their quantifier order switched.
- More interesting are $\forall y: \exists x: x \leq y$ (true), $\forall x: \exists y: x \leq y$ (true), $\exists y: \forall x: x \leq y$ (false), and $\exists x: \forall y: x \leq y$ (true, as x could be 0).
- The last two examples show that switching dissimilar quantifiers can change the meaning.

Clicker Question #2

- Let's now change our data type from natural numbers to integers, which may be negative. Which of the following four quantified statements is true?
- (a) $\forall x: \forall y: x \leq y$
- (b) $\neg\exists x: \exists y: x \leq y$
- (c) $\exists x: \forall y: x \leq y$
- (d) $\forall x: \exists y: x \leq y$

Answer #2

- Let's now change our data type from natural numbers to integers, which may be negative. Which of the following four quantified statements is true?
- (a) $\forall x: \forall y: x \leq y$
- (b) $\neg\exists x: \exists y: x \leq y$
- (c) $\exists x: \forall y: x \leq y$
- (d) $\forall x: \exists y: x \leq y$

Languages, Language Operations

- Recall that for any finite alphabet Σ we have defined the set Σ* of all **strings** made up of a finite sequence of letters from Σ, and defined a **language** over Σ to be any subset of Σ*, that is, any set of strings. Here we'll have Σ = {a, b}.
- Because languages are sets, we can use any of our set operators on them.

Set Operators on Languages

- If X is all strings beginning with a, and Y is all strings ending in b, then $X \cup Y$ is the set of all strings that begin with a or end in b, and $X \cap Y$ is the set of all strings that both begin with a and end in b.
- Similarly, we can define $X \Delta Y$, $X \setminus Y$, and the complements of X and Y respectively. For example, the complement of X is the set of all strings that don't begin with a (including the empty string λ).

More Language Operations

- Now that we have quantifiers, we will be able to define two more operations on languages, called **concatenation** and **Kleene star**.
- In the last third of the course, we'll use these two operations, along with the union operation, to define **regular expressions** and thus define the class of **regular languages**.

Language Concatenation

- We'll now define the **concatenation product** (or just **concatenation**) of two languages.
- Remember that the concatenation of two strings is what we get by writing the second string after the first.
- In general, XY is the language $\{w: \exists u: \exists v: (w = uv) \wedge (u \in X) \wedge (v \in Y)\}$. A string w is in XY if it is possible to split it as a string in X followed by a string in Y.

Concatenation Example

- Again let X = {w: w begins with a} and Y = {w: w ends in b}.
- The product XY is the set of all strings that we can make by writing a string in X followed by a string from Y.
- In this example, XY is the same language as X ∩ Y. Any string in XY must both begin with a and end with b, and any string with these two properties can be split into a string in X and a string in Y.

Properties of Concatenation

- Unlike most "multiplication" operations, concatenation is *not commutative*. The language YX is $\{w: \exists u: \exists v: (w = uv) \wedge (u \in Y) \wedge (v \in X)\}$. Strings in YX need not begin with a or end in b -- in fact a string is in YX if and only if it has a b that is immediately followed by an a.
- If we let "a" and "b" denote the languages {a} and {b}, with one string each, what is the language $a\Sigma^*b$? Or $\Sigma^*ba\Sigma^*$?

Clicker Question #3

- Which of the following strings is not in the language $\Sigma^* aa \Sigma^*$?
- (a) abbaab
- (b) bbbaaa
- (c) ababa
- (d) aa

Answer #3

- Which of the following strings is not in the language $\Sigma^{*}aa\Sigma^{*}$?
- (a) abbaab
- (b) bbbaaa
- (c) *ababa*
- (d) aa

Powers of Languages

- In algebra we say "x^k" to denote the product of k copies of x. Similarly in language theory, if X is a language, we abbreviate the concatenation product XX as "X^2", XXX as "X^3", and so forth.
- It turns out that if we treat concatenation as "multiplication" and union as "addition", the distributive law holds, and we can use algebraic rules to get facts like $(X + Y)^2 = X^2 + XY + Y^2$. (We don't say "2XY" because "XY + XY" just equals XY -- the union of a language with itself is just itself.)

The Kleene Star Operation

- X^0 is a special case -- "not multiplying" gives us the multiplicative identity, which turns out to be the language $\{\lambda\}$. (Check that $\{\lambda\}X = X$ for any language X.)
- It's convenient sometimes to talk about the language $X^0 + X^1 + X^2 + X^3 + \ldots$, which is the set of all strings that can be made by concatenating together *any number* of strings from X. We call this language X^*, the **Kleene star** of X. We've used this notation already when we defined Σ^* to be the set of all strings from Σ.