Predicates and Quantifiers

Section 1.4

Section Summary

- Predicates
- Variables
- Quantifiers
  - Universal Quantifier （全称量词）
  - Existential Quantifier （存在量词）
- Negating Quantifiers
  - De Morgan's Laws for Quantifiers
- Translating English to Logic
- Logic Programming (*optional*)

Propositional Logic Not Enough

- If we have:
  "All men are mortal."
  "Socrates is a man."
- Does it follow that "Socrates is mortal?"
- Can't be represented in propositional logic. Need a language that talks about objects, their properties, and their relations.
- Later we'll see how to draw inferences.

Introducing Predicate Logic

- Predicate logic uses the following new features:
  - Variables: x, y, z
  - Predicates: $P(x)$, $M(x)$
  - Quantifiers (*to be covered in a few slides*):
- *Propositional functions* are a generalization of propositions.
  - They contain variables and a predicate, e.g., $P(x)$
  - Variables can be replaced by elements from their *domain*.

Propositional Functions

- Propositional functions become propositions (and have truth values) when their variables are each replaced by a value from the *domain* (or *bound* by a quantifier, as we will see later).
- The statement *P(x)* is said to be the value of the propositional function *P* at *x*.
- For example, let *P(x)* denote "$x > 0$" and the domain be the integers. Then:
  P(-3) is false.
  P(0) is false.
  P(3) is true.
- Often the domain is denoted by *U*. So in this example *U* is the integers.

Examples of Propositional Functions

- Let "$x + y = z$" be denoted by $R(x, y, z)$ and U (for all three variables) be the integers. Find these truth values:
  R(2,-1,5)
  Solution: F
  R(3,4,7)
  Solution: T
  R(x, 3, z)
  Solution: Not a Proposition
- Now let "$x - y = z$" be denoted by $Q(x, y, z)$, with U as the integers. Find these truth values:
  Q(2,-1,3)
  Solution: T
  Q(3,4,7)
  Solution: F
  Q(x, 3, z)
  Solution: Not a Proposition

Compound Expressions

- Connectives from propositional logic carry over to predicate logic.
- If *P(x)* denotes "$x > 0$," find these truth values:
  - $P(3) \vee P(-1)$ **Solution**: T
  - $P(3) \wedge P(-1)$ **Solution**: F
  - $P(3) \rightarrow P(-1)$ **Solution**: F
  - $P(-1) \rightarrow P(3)$ **Solution**: T
- Expressions with variables are not propositions and therefore do not have truth values. For example,
  - $P(3) \wedge P(y)$
  - $P(x) \rightarrow P(y)$
- When used with quantifiers (to be introduced next), these expressions (propositional functions) become propositions.

Quantifiers

Charles Peirce (1839-1914)

- We need *quantifiers* to express the meaning of English words including *all* and *some*:
  - "All men are Mortal."
  - "Some cats do not have fur."
- The two most important quantifiers are:
  - *Universal Quantifier*, "For all," symbol: $\forall$
  - *Existential Quantifier*, "There exists," symbol: $\exists$
- We write as in $\forall x\, P(x)$ and $\exists x\, P(x)$.
- $\forall x\, P(x)$ asserts $P(x)$ is true for <u>every</u> x in the *domain*.
- $\exists x\, P(x)$ asserts $P(x)$ is true for <u>some</u> x in the *domain*.
- The quantifiers are said to bind the variable x in these expressions.

Universal Quantifier

- $\forall x\, P(x)$ is read as "For all x, P(x)" or "For every x, P(x)"

  Examples:

  1) If $P(x)$ denotes "$x > 0$" and U is the integers, then $\forall x\, P(x)$ is false.
  2) If $P(x)$ denotes "$x > 0$" and U is the positive integers, then $\forall x\, P(x)$ is true.
  3) If $P(x)$ denotes "x is even" and U is the integers, then $\forall\, x\, P(x)$ is false.

Domain (domain of discourse / universe of discourse): range of the possible values of the variable x

- An element for which $P(x)$ is false is called a counterexample（反例） of $\forall\, x\, P(x)$
- Example:

  Let $P(x)$ be the statement "x<2." In the domain of all real numbers, x=3 is a counterexample for $\forall\, x\, P(x)$.

10

- Many ways to express universal quantification:
  - For all
  - For every
  - All of
  - For each
  - Given any
  - For arbitrary
  - For any

- Examples:

3. What is the truth value of $\forall x\, P(x)$, where $P(x)$ is the statement "$x<3$" and the domain is $\{1,2,3\}$?

Solution:
$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3)$$
Because $P(3)$, which is the statement "3<3," is false, it follows that $\forall x\, P(x)$ is false.

- Remark: Given the domain as $\{x_1, x_2, \cdots, x_n\}$,

$$\forall x P(x) \equiv P(x_1) \wedge P(x_2) \wedge \cdots \wedge P(x_n)$$

Existential Quantifier

- $\exists x\ P(x)$ is read as "For some x, P(x)", or as "There is an x such that P(x)," or "For at least one x, P(x)."

  Examples:
  1. If $P(x)$ denotes "$x > 0$" and U is the integers, then $\exists x\ P(x)$ is true. It is also true if U is the positive integers.
  2. If $P(x)$ denotes "$x < 0$" and U is the positive integers, then $\exists x\ P(x)$ is false.
  3. If $P(x)$ denotes "x is even" and U is the integers, then $\exists x\ P(x)$ is true.

【Definition】A existential quantification of $P(x)$, denoted by $\exists\ x\ P(x)$, is the statement "There exists an element x in the domain such that $P(x)$. "

- $\exists$: existential quantifier
- Other expressions:

  For some x $P(x)$

  There is an x such that $P(x)$

  There is at least one x such that $P(x)$

14

- Examples:

2. What is the truth value of $\exists\, x\, P(x)$, where $P(x)$ is the statement "$x<3$" and the domain is $\{1,2,3\}$?

Solution:

$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3)$$

Because $P(1)$, which is the statement "1<3," is true, it follows that $\exists\, x\, P(x)$ is true.

- Remark: Given the domain as $\{x_1, x_2, \cdots, x_n\}$,

$$\exists x P(x) \equiv P(x_1) \vee P(x_2) \vee \cdots \vee P(x_n)$$

Uniqueness Quantifier（唯一性量词）

- $\exists! x\, P(x)$ means that $P(x)$ is true for <u>one and only one</u> x in the universe of discourse.
- This is commonly expressed in English in the following equivalent ways:
  - "There is a unique x such that $P(x)$."
  - "There is one and only one x such that $P(x)$"
- Examples:
  1. If $P(x)$ denotes "$x + 1 = 0$" and U is the integers, then $\exists! x\, P(x)$ is true.
  2. But if $P(x)$ denotes "$x > 0$," then $\exists! x\, P(x)$ is false.
- The uniqueness quantifier is not really needed as the restriction that there is a unique x such that $P(x)$ can be expressed as:

$$\exists x\, (P(x) \wedge \forall y\, (P(y) \rightarrow y = x))$$

Thinking about Quantifiers

- When the domain of discourse is finite, we can think of quantification as looping through the elements of the domain.
- To evaluate $\forall x\, P(x)$ loop through all x in the domain.
  - If at every step $P(x)$ is true, then $\forall x\, P(x)$ is true.
  - If at a step $P(x)$ is false, then $\forall x\, P(x)$ is false and the loop terminates.
- To evaluate $\exists x\, P(x)$ loop through all x in the domain.
  - If at some step, $P(x)$ is true, then $\exists x\, P(x)$ is true and the loop terminates.
  - If the loop ends without finding an x for which $P(x)$ is true, then $\exists x\, P(x)$ is false.
- Even if the domains are infinite, we can still think of the quantifiers this fashion, but the loops will not terminate in some cases.

Properties of Quantifiers

- The truth value of $\exists x\, P(x)$ and $\forall x\, P(x)$ depend on both the propositional function $P(x)$ and on the domain U.
- Examples:
  1. If U is the positive integers and $P(x)$ is the statement "$x < 2$", then $\exists x\, P(x)$ is true, but $\forall x\, P(x)$ is false.
  2. If U is the negative integers and $P(x)$ is the statement "$x < 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are true.
  3. If U consists of 3, 4, and 5, and $P(x)$ is the statement "$x > 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are true. But if $P(x)$ is the statement "$x < 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are false.

Precedence of Quantifiers

- The quantifiers $\forall$ and $\exists$ have higher precedence than all the logical operators.
- For example, $\forall x\, P(x) \vee Q(x)$ means $(\forall x\, P(x)) \vee Q(x)$
- $\forall x\, (P(x) \vee Q(x))$ means something different.
- Unfortunately, often people write $\forall x\, P(x) \vee Q(x)$ when they mean $\forall x\, (P(x) \vee Q(x))$.

Binding Variables

- Bound variable: a variable is bound if it is known or quantified.
- Free variable: a variable neither quantified nor specified with a value
- *All the variables in a propositional function must be quantified or set equal to a particular value to turn it into a proposition.*
- Scope（作用域） of a quantifier: the part of a logical expression to which the quantifier is applied
- Examples

  $\exists x\, (x+y)=1$

  $\exists x\, (P(x) \wedge Q(x)) \vee \forall x\, R(x)$

20

Translating from English to Logic

Example 1: Translate the following sentence into predicate logic: "Every student in this class has taken a course in Java."

Solution:

First decide on the domain *U*.

Solution 1: If *U* is all students in this class, define a propositional function J(x) denoting "x has taken a course in Java" and translate as $\forall x\, J(x)$.

Solution 2: But if *U* is all people, also define a propositional function S(x) denoting "x is a student in this class" and translate as $\forall x\, (S(x) \rightarrow J(x))$.

$\forall x\, (S(x) \wedge J(x))$ is not correct. What does it mean?

Translating from English to Logic

Example 2: Translate the following sentence into predicate logic: "Some student in this class has taken a course in Java."

Solution:

First decide on the domain *U*.

Solution 1: If *U* is all students in this class, translate as

$\exists x\, J(x)$

Solution 1: But if *U* is all people, then translate as

$\exists x\, (S(x) \wedge J(x))$

$\exists x\, (S(x) \rightarrow J(x))$ is not correct. What does it mean?

Returning to the Socrates Example

- Introduce the propositional functions *Man(x)* denoting "*x* is a man" and *Mortal(x)* denoting "*x* is mortal." Specify the domain as all people.
- The two premises are: $\forall x (Man(x) \rightarrow Mortal(x))$
  $Man(Socrates)$
- The conclusion is: $Mortal(Socrates)$
- Later we will show how to prove that the conclusion follows from the premises.

Equivalences in Predicate Logic

- Statements involving predicates and quantifiers are *logically equivalent* if and only if they have the same truth value
  - for every predicate substituted into these statements and
  - for every domain of discourse used for the variables in the expressions.
- The notation $S \equiv T$ indicates that S and T are logically equivalent.
- Example: $\forall x \neg\neg S(x) \equiv \forall x\, S(x)$

Logical Equivalences Involving Quantifiers

x is not occurring in A.

$$
\begin{array}{llcl}
(1) & \forall x P(x) \vee A & \equiv & \forall x (P(x) \vee A) \\
(2) & \forall x P(x) \wedge A & \equiv & \forall x (P(x) \wedge A) \\
(3) & \exists x P(x) \vee A & \equiv & \exists x (P(x) \vee A) \\
(4) & \exists x P(x) \wedge A & \equiv & \exists x (P(x) \wedge A) \\
(5) & \forall x (A \to P(x)) & \equiv & A \to \forall x P(x) \\
(6) & \exists x (A \to P(x)) & \equiv & A \to \exists x P(x) \\
(7) & \forall x (P(x) \to A) & \equiv & \exists x P(x) \to A \\
(8) & \exists x (P(x) \to A) & \equiv & \forall x P(x) \to A
\end{array}
$$

25

Logical Equivalences Involving Quantifiers

x is not occurring in A.

$$
\begin{array}{llcl}
(1) & \forall x P(x) \vee A & \equiv & \forall x (P(x) \vee A) \\
(2) & \forall x P(x) \wedge A & \equiv & \forall x (P(x) \wedge A) \\
(3) & \exists x P(x) \vee A & \equiv & \exists x (P(x) \vee A) \\
(4) & \exists x P(x) \wedge A & \equiv & \exists x (P(x) \wedge A) \\
(5) & \forall x (A \to P(x)) & \equiv & A \to \forall x P(x)
\end{array}
$$

Proof:

$$
\begin{aligned}
\forall x (A \to P(x)) &\equiv \forall x (\neg A \vee P(x)) \\
&\equiv \neg A \vee \forall x P(x) \\
&\equiv A \to \forall x P(x)
\end{aligned}
$$

26

Thinking about Quantifiers as Conjunctions and Disjunctions

- If the domain is finite, a universally quantified proposition is equivalent to a conjunction of propositions without quantifiers and an existentially quantified proposition is equivalent to a disjunction of propositions without quantifiers.
- If U consists of the integers 1,2, and 3:

$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3)$$

$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3)$$

- Even if the domains are infinite, you can still think of the quantifiers in this fashion, but the equivalent expressions without quantifiers will be infinitely long.

Negating Quantified Expressions

- Consider $\forall x\, J(x)$

  "Every student in your class has taken a course in Java."

  Here $J(x)$ is "x has taken a course in Java" and

  the domain is students in your class.
- Negating the original statement gives "It is not the case that every student in your class has taken Java." This implies that "There is a student in your class who has not taken Java."

  Symbolically $\neg\forall x\, J(x)$ and $\exists x\, \neg J(x)$ are equivalent

Negating Quantified Expressions (*continued*)

- Now Consider $\exists x\, J(x)$

  "There is a student in this class who has taken a course in Java."

  Where $J(x)$ is "x has taken a course in Java."
- Negating the original statement gives "It is not the case that there is a student in this class who has taken Java." This implies that "Every student in this class has not taken Java"

  Symbolically $\neg\exists x\, J(x)$ and $\forall x\, \neg J(x)$ are equivalent

De Morgan's Laws for Quantifiers

- The rules for negating quantifiers are:

TABLE 2 De Morgan's Laws for Quantifiers.

Negation	*Equivalent Statement*	*When Is Negation True?*	*When False?*
$\neg\exists x P(x)$	$\forall x \neg P(x)$	For every x, $P(x)$ is false.	There is an x for which $P(x)$ is true.
$\neg\forall x P(x)$	$\exists x \neg P(x)$	There is an x for which $P(x)$ is false.	$P(x)$ is true for every x.

- The reasoning in the table shows that:

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

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

- These are important. You will use these.

Translating from English into Logical Expressions

- Goal: To produce a logical expression that is simple and can be easily used in subsequent reasoning.
- Steps:
  - Clearly identify the appropriate quantifier(s)
  - Introduce variable(s) and predicate(s)
  - Translate using quantifiers, predicates, and logical operators

There can be many ways to translate a particular sentence.

31

Translation from English to Logic

Examples:

1. "Some student in this class has visited Mexico."

   Solution: Let $M(x)$ denote "x has visited Mexico" and $S(x)$ denote "x is a student in this class," and U be all people.

   $$\exists x\ (S(x) \wedge M(x))$$

2. "Every student in this class has visited Canada or Mexico."

   Solution: Add $C(x)$ denoting "x has visited Canada."

   $$\forall x\ (S(x) \rightarrow (M(x) \vee C(x)))$$

Example

- $C(x)$: x is a CS student, $E(x)$: x is a Math student, $S(x)$: x is a smart student, and the domain consists of all students in our class

1) Everyone is a CS student.

$$\forall \boldsymbol{x}\, \boldsymbol{C}(\boldsymbol{x})$$

2) Nobody is a Math student.

$$\forall \boldsymbol{x} \neg \boldsymbol{E}(\boldsymbol{x}) \quad \textbf{or} \quad \neg \exists \boldsymbol{x}\, \boldsymbol{E}(\boldsymbol{x})$$

3) All CS students are smart students.

$$\forall \boldsymbol{x}\, (\boldsymbol{C}(\boldsymbol{x}) \rightarrow \boldsymbol{S}(\boldsymbol{x}))$$

4) Some CS students are smart students.

$$\exists \boldsymbol{x}\, (\boldsymbol{C}(\boldsymbol{x}) \wedge \boldsymbol{S}(\boldsymbol{x}))$$

Example

- $C(x)$: x is a CS student, $E(x)$: x is an Math student, $S(x)$: x is a smart student, and the domain consists of all students in our class

5) No CS student is an Math student.

- If x is a CS student, then that student is not a Math student.

$$\forall x\, (C(x) \rightarrow \neg E(x))$$

- There does not exist a CS student who is also a Math student.

$$\neg \exists x\, [C(x) \wedge E(x)]$$

6) If any Math student is a smart student then he is also a CS student.

$$\forall x\, ((E(x) \wedge S(x)) \rightarrow C(x))$$

34

Some Fun with Translating from English into Logical Expressions

- U = {fleegles, snurds, thingamabobs}

  $F(x)$: x is a fleegle

  $S(x)$: x is a snurd

  $T(x)$: x is a thingamabob

  Translate "Everything is a fleegle"

  Solution: $\forall x\, F(x)$

Translation (cont)

- U = {fleegles, snurds, thingamabobs}

  $F(x)$: x is a fleegle

  $S(x)$: x is a snurd

  $T(x)$: x is a thingamabob

  "Nothing is a snurd."

  Solution: $\neg\exists x\, S(x)$ What is this equivalent to?

  Solution: $\forall x \neg\, S(x)$

Translation (cont)

- U = {fleegles, snurds, thingamabobs}

  $F(x)$: x is a fleegle

  $S(x)$: x is a snurd

  $T(x)$: x is a thingamabob

"All fleegles are snurds."

Solution: $\forall x\ (F(x) \rightarrow S(x))$

Translation (cont)

- U = {fleegles, snurds, thingamabobs}

  $F(x)$: x is a fleegle

  $S(x)$: x is a snurd

  $T(x)$: x is a thingamabob

"Some fleegles are thingamabobs."

Solution: $\exists x\ (F(x) \wedge T(x))$

Translation (cont)

- U = {fleegles, snurds, thingamabobs}

  $F(x)$: x is a fleegle

  $S(x)$: x is a snurd

  $T(x)$: x is a thingamabob

"No snurd is a thingamabob."

Solution: $\neg\exists x\ (S(x) \wedge T(x))$ What is this equivalent to?

Solution: $\forall x\ (\neg S(x) \vee \neg T(x))$

Translation (cont)

- U = {fleegles, snurds, thingamabobs}

  $F(x)$: x is a fleegle

  $S(x)$: x is a snurd

  $T(x)$: x is a thingamabob

"If any fleegle is a snurd then it is also a thingamabob."

Solution: $\forall x\ ((F(x) \wedge S(x)) \rightarrow T(x))$

System Specification Example

- Predicate logic is used for specifying properties that systems must satisfy.
- For example, translate into predicate logic:
  - "Every mail message larger than one megabyte will be compressed."
  - "If a user is active, at least one network link will be available."
- Decide on predicates and domains (left implicit here) for the variables:
  - Let $L(m, y)$ be "Mail message m is larger than y megabytes."
  - Let $C(m)$ denote "Mail message m will be compressed."
  - Let $A(u)$ represent "User u is active."
  - Let $S(n, x)$ represent "Network link n is state x".
- Now we have: $\forall m(L(m,1) \rightarrow C(m))$

$$\exists u\, A(u) \rightarrow \exists n\, S(n, available)$$

Lewis Carroll Example

Charles Lutwidge Dodgson
(AKA Lewis Caroll)
(1832-1898)

- The first two are called *premises* and the third is called the *conclusion*.
  1. "All lions are fierce."
  2. "Some lions do not drink coffee."
  3. "Some fierce creatures do not drink coffee."
- Here is one way to translate these statements to predicate logic. Let P(x), Q(x), and R(x) be the propositional functions "x is a lion," "x is fierce," and "x drinks coffee," respectively.
  1. $\forall x\, (P(x) \rightarrow Q(x))$
  2. $\exists x\, (P(x) \land \neg R(x))$
  3. $\exists x\, (Q(x) \land \neg R(x))$
- Later we will see how to prove that the conclusion follows from the premises.

Some Predicate Calculus Definitions (*optional*)

- An assertion involving predicates and quantifiers is *valid* if it is true
  - for all domains
  - every propositional function substituted for the predicates in the assertion.

  Example: $\forall x \neg S(x) \leftrightarrow \neg \exists x S(x)$
- An assertion involving predicates is *satisfiable* if it is true
  - for some domains
  - some propositional functions that can be substituted for the predicates in the assertion.

  Otherwise it is *unsatisfiable*.

  Example: $\forall x(F(x) \leftrightarrow T(x))$ not valid but satisfiable

  Example: $\forall x(F(x) \wedge \neg F(x))$ unsatisfiable

More Predicate Calculus Definitions (*optional*)

- The *scope* of a quantifier is the part of an assertion in which variables are bound by the quantifier.

  Example: $\forall x(F(x) \vee S(x))$ *x* has wide scope

  Example: $\forall x(F(x)) \vee \forall y(S(y))$ *x* has narrow scope

Logic Programming (optional)

- Prolog (from *Pro*gramming in *Log*ic) is a programming language developed in the 1970s by researchers in artificial intelligence (AI).
- Prolog programs include *Prolog facts* and *Prolog rules*.
- As an example of a set of Prolog facts consider the following:

```
instructor(chan, math273).
instructor(patel, ee222).
instructor(grossman, cs301).
enrolled(kevin, math273).
enrolled(juna, ee222).
enrolled(juana, cs301).
enrolled(kiko, math273).
enrolled(kiko, cs301).
```

- Here the predicates *instructor(p,c)* and *enrolled(s,c)* represent that professor *p* is the instructor of course *c* and that student *s* is enrolled in course *c*.

Logic Programming (cont)

- In Prolog, names beginning with an uppercase letter are variables.
- If we have apredicate *teaches(p,s)* representing "professor *p* teaches student *s*," we can write the rule:

```
teaches(P,S) :- instructor(P,C), enrolled(S,C).
```

- This Prolog rule can be viewed as equivalent to the following statement in logic (using our conventions for logical statements).

$$\forall p\ \forall c\ \forall s(I(p,c) \land E(s,c)) \rightarrow T(p,s))$$

Logic Programming (cont)

- Prolog programs are loaded into a *Prolog interpreter*. The interpreter receives *queries* and returns answers using the Prolog program.
- For example, using our program, the following query may be given:

```
?enrolled(kevin,math273).
```

- Prolog produces the response:

```
yes
```

- Note that the `?` is the prompt given by the Prolog interpreter indicating that it is ready to receive a query.

Logic Programming (cont)

- The query:

```
?enrolled(X,math273).
```

produces the response:

```
X = kevin;
X = kiko;
no
```

- The query:

```
?teaches(X,juana).
```

produces the response:

```
X = patel;
X = grossman;
no
```

The Prolog interpreter tries to find an instantiation for `X`. It does so and returns `X = kevin.` Then the user types the `;` indicating a request for another answer. When Prolog is unable to find another answer it returns `no`.

Logic Programming (cont)

- The query:
```
?teaches(chan,X).
```
produces the response:
```
X = kevin;
X = kiko;
no
```
- A number of very good Prolog texts are available. *Learn Prolog Now!* is one such text with a free online version at http://www.learnprolognow.org/
- There is much more to Prolog and to the entire field of logic programming.

Homework

- 第8版：Section 1.4 6(c,d,e,f), 9(b,d), 20(e), 24(b,d), 36， 42(b), 46, 51(a)