Predicate Logic

CS 330: Discrete Structures
Predicate logic adds variables, predicates, and quantifiers to propositional logic.

e.g. propositional logic:

\[ p: \text{"John likes cake."} \quad q: \text{"Jane likes cake"} \]

e.g. predicate logic:

\[ P(x): \text{"x likes cake"} \]

\[ \exists x \ P(x): \text{"there exists x such that x likes cake"} \]
A predicate is a statement that is True or False, depending on the value of one or more variables taken from the domain or "universe of discourse."

A predicate \( P(x, y, \ldots) \) can be thought of as a propositional function that evaluates to T/F, based on its inputs \( x, y, \ldots \).
eq. \( P(x) \): \( x < 10 \) and \( x \) is prime

\[
\begin{align*}
P(3) & \?
\P(9) & ?
\P(11) & ?
P(20) & \?
P(5) \lor P(20) & \? \\
P(7) \land P(10) & \? \\
P(1) \rightarrow P(12) & ?
\end{align*}
\]
e.g. \( Q(x, y, z) : x + y = z \)

\[ Q(1, 2, 3) \]
\[ Q(3, -1, 4) \]
\[ Q(2, 4, 7) \rightarrow Q(3, 4, 7) \]
A quantifier specifies how many values from the domain, when assigned to a particular variable, satisfy a predicate.

Most important quantifiers:
- universal quantifier: ∀ — "for all"
- existential quantifier: ∃ — "there exists"
Eq. 1

\( \forall x \, P(x) \) : "For all \( x \) in the domain, \( P(x) \) is True."

\( \exists x \, P(x) \): "There exists some \( x \) in the domain such that \( P(x) \) is True."

Note: Important to specify the domain!

(We'll discuss different ways of doing this.)
Quantifiers can be thought of as looping over values in the domain.

- $\forall x P(x)$ loops over all $x$ in the domain
  - only true if $P(x)$ is true for all iterations
  - if any $x$ causes $P(x)$ to be false, terminate with false

- $\exists x P(x)$ loops over all $x$ in the domain
  - if we find $P(x)$ true for some $x$, terminate with true
  - if loop terminates without finding $P(x)$ true for any $x$, evaluates to false
Quantifiers can also be considered in terms of logical conjunctions/disjunctions (for finite domains)

- \( \forall x \, P(x) \) is the logical conjunction of \( P(x) \) for all \( x \)
  
  \[ \text{i.e., } \forall x \, P(x) \equiv \bigwedge_x P(x) \]

- \( \exists x \, P(x) \) is the logical disjunction of \( P(x) \) for all \( x \)
  
  \[ \text{i.e., } \exists x \, P(x) \equiv \bigvee_x P(x) \]
On Precedence

∀ and ∃ have higher precedence than all logical operators!

E.g. \( ∀x \, P(x) \lor Q(x) \equiv (∀x \, P(x)) \lor Q(x) \)

\( ∀x \, P(x) \lor Q(x) \neq ∀x \, (P(x) \lor Q(x)) \)
We can come up with other quantifiers ... e.g., "there are exactly \( N \) values ..."

"for the majority of values ..." (assuming finite domain)

"there is a unique value ..."

but we can express most other quantifiers using propositional operators.
Uniqueness Quantifier “\( \exists ! \)”

\( \exists ! x \ P(x) \) : "There is a unique \( x \) such that \( P(x) \)."

e.g. \( P(x) : x + 10 = 0 \), domain is \( \mathbb{Z} \)

\( \exists ! x \ P(x) \) ? \( T \)

e.g. \( P(x) : x < 0 \), domain is \( \mathbb{Z} \)

\( \exists ! x \ P(x) \) ? \( F \)
Express \( \exists! x \) in terms of \( \exists \) and \( \forall \): 

\[
\exists! x \in \mathbb{R} \mid \exists x (P(x) \land \forall y (P(y) \rightarrow y = x))
\]

"needed" quantifier
Translating from English to logic

E.g. "Every student in CS 330 can program in Java"

1. Assume domain of $x$ is students in CS 330
   
   $P(x)$: " $x$ can program in Java"

   \[
   \forall x \ P(x)
   \]

2. Assume domain of $x$ is all students
   
   $Q(x)$: " $x$ is a student in CS 330"

   \[
   \forall x \ (Q(x) \rightarrow P(x)) \quad (\forall x \ (Q(x) \land P(x)))
   \]

   is wrong!
Translating from English to logic

E.g. "Some student in CS 330 can program in Java"

1. Assume domain of $x$ is students in CS 330
   
   $P(x): "x$ can program in Java"
   
   $\exists x \ P(x)$

2. Assume domain of $x$ is all students
   
   $Q(x): "x$ is a student in CS 330"
   
   $\exists x \ (P(x) \land Q(x)) \quad (\forall x \ (P(x) \to Q(x)))$

   is wrong!
Negating Quantifiers.

E.g. \( P(x) \): "\( x \) loves honey" ; domain of \( x \) is all bees

\[ \forall x \ P(x) : \text{"all bees love honey"} \]

\[ \neg \forall x \ P(x) : \text{"it is not true that all bees love honey"} \]

\[ \not\equiv \text{"all bees do not love honey"} \]

\[ \equiv \text{"there is a bee that doesn't love honey"} \]

i.e., \( \neg \forall x \ P(x) \equiv \exists x (\neg P(x)) \)
De Morgan’s Laws for Quantifiers

\[ \neg \forall x P(x) \equiv \exists x \neg P(x) \]

\[ \neg \exists x P(x) \equiv \forall x \neg P(x) \]
Example — domain = \{ \text{fleegles, snurds, thingamabobs} \}

\begin{align*}
F(x) & : x \text{ is a fleegle} \\
S(x) & : x \text{ is a snurd} \\
T(x) & : x \text{ is a thingamabob}
\end{align*}

Translate “Everything is a fleegle.”

\[ \forall x \ F(x) \]
Example — domain = \{\text{fleegles, snurds, thingamabobs}\}

\begin{align*}
F(x) & : x \text{ is a fleegle} \\
S(x) & : x \text{ is a snurd} \\
T(x) & : x \text{ is a thingamabob}
\end{align*}

Translate "Nothing is a snurd:"

\[
\forall x \neg S(x) \\
\equiv \neg \exists x S(x)
\]
Example — domain = \{fleegles, snurds, thingamabobs\}

\[ F(x) : x \text{ is a fleegle} \]
\[ S(x) : x \text{ is a snurd} \]
\[ T(x) : x \text{ is a thingamabob} \]

Translate “All fleegles are snurds”

\[ \forall x \,(F(x) \rightarrow S(x)) \]
Example — domain = \{ \text{fieegles, snurds, thingamabobs} \}

F(x) : x \text{ is a fieegle}

S(x) : x \text{ is a snurd}

T(x) : x \text{ is a thingamabob}

Translate "Some fieegles are thingamabobs"

\exists x (F(x) \land T(x))
Example — domain = \{ \text{fleegles, snurds, thingamabobs} \}

\begin{align*}
F(x) & : x \text{ is a fleegle} \\
S(x) & : x \text{ is a snurd} \\
T(x) & : x \text{ is a thingamabob}
\end{align*}

Translate "No snurd is a thingamabob"!

\[
\neg \exists x (S(x) \land T(x)) \quad \forall x (S(x) \Rightarrow \neg T(x)) \\
\quad \equiv \quad \forall x (\neg S(x) \lor \neg T(x))
\]
Example — domain = \{ \text{fiegeles, snurds, thingamabobs} \}

F(x) : x is a fiegle
S(x) : x is a snurd
T(x) : x is a thingamabob

Translate “If any fiegle is a snurd then it is also a thingamabob”

\forall x ( (F(x) \land S(x)) \rightarrow T(x))
Validity vs Satisfiability

- an assertion involving predicates + quantifiers is valid if it is true for all domains, and all possible predicates
  \[ \forall x \neg P(x) \iff \neg \exists x P(x) \]

- an assertion is satisfiable if it is only true for some domains and predicates
  \[ \forall x (P(x) \iff Q(x)) \]

- otherwise, it is unsatisfiable
  \[ \forall x (P(x) \land \neg P(x)) \]
Nested Quantifiers

- quantifiers that appear within the scope of other quantifiers

E.g.: \( \forall x \exists y P(x, y) \)

"for every x, there exists a y for which \( P(x, y) \) is true"
Nested Quantifiers

- can be thought of as nested loops — for each value of the variable bound by the outside quantifier, step through all values of inner quantifiers

— order matters!

e.g. $P(x, y) : x + y = 0 \ ; x, y$ from TR

\[ \forall x \exists y \ P(x, y) = T \]

\[ \exists y \forall x \ P(x, y) = F \]
Given: \( L(x, y) : "x \text{ loves } y" \)

Express the following using quantifiers:

- "Everybody loves somebody":
  \[ \forall x \exists y L(x, y) \]

- "There is someone whom everybody loves."
  \[ \exists x \forall y L(y, x) \]

- "Nobody loves everybody"
  \[ \neg \exists x \forall y L(x, y) \text{ or } \forall x \forall y L(x, y) \]
Negating nested quantifiers:

- rewrite the following so that negations only appear directly in front of predicates

\[
\neg \forall x \forall y \, P(x, y) \\
\exists x \exists y \, \neg P(x, y) \\
\neg (\exists x \exists y \, \neg P(x, y) \land \forall x \forall y \, (Q(x, y) \lor \neg R(x, y))) \\
\forall x \forall y \, P(x, y) \lor \exists x \exists y \, (\neg Q(x, y) \land R(x, y))
\]
Given: \( T(s, c) \): student \( s \) has taken class \( c \)

- domain of \( s \) = all IIT students
- domain of \( c \) = all CS classes

Translate to English:

\[
\exists x (T(Michael, x) \land T(Shannon, x))
\]

“There is a class that both Michael and Shannon have taken.”
Given: \( T(s,c) \): student \( s \) has taken class \( c \)

- domain of \( s \) = all IIT students
- domain of \( c \) = all CS classes

Translate to English:

\[
\exists x \forall y \left( x \neq \text{Michael} \land (T(\text{Michael}, y) \implies T(x, y)) \right)
\]

"There is a student who has taken all the classes Michael has taken."
Given: \( T(s,c) \): student \( s \) has taken class \( c \)

- domain of \( s \) = all IIT students
- domain of \( c \) = all CS classes

Translate to English:

\[ \exists x \exists y \forall z \left( (x \neq y) \land (T(x, z) \leftrightarrow T(y, z)) \right) \]

"There are two separate students that have taken precisely the same classes."