Cartesian product definition

Definition: The **Cartesian product** of the sets A and B, $A \times B$, is the set of all ordered pairs (a, b), where $a \in A$ and $b \in B$. That is: $A \times B = \{(a, b) \mid (a \in A) \wedge (b \in B)\}$. The Cartesian product of the sets $A_1, A_2, \ldots, A_n$, denoted by $A_1 \times A_2 \times \cdots \times A_n$, is the set of ordered n-tuples $(a_1, a_2, ..., a_n)$, where a_i belongs to A_i for $i = 1, 2, \ldots, n$. That is,

$$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i \text{ for } i = 1, 2, \ldots, n\}$$

Rna mutation insertion deletion example

Trace the pseudocode to find the output of *mutation*($(\texttt{AUC}, 3, \texttt{G})$)

Fill in the blanks so that *insertion*($(\texttt{AUC}, \_, \_)$) $= \texttt{AUCG}$

Fill in the blanks so that *deletion*($(\_\_, \_)$) $= \texttt{G}$

Rna rnalen basecount definitions

Recall the definitions: The set of RNA strands S is defined (recursively) by:

$$\begin{array}{ll} \text{Basis Step:} & \texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B \text{, then } sb \in S \end{array}$$

where sb is string concatenation.

The function *rnalen* that computes the length of RNA strands in S is defined recursively by:

$$\begin{array}{lllll} & & \mathit{rnalen} : S & \to \mathbb{Z}^+ \\ \text{Basis Step:} & \text{If } b \in B \text{ then} & \mathit{rnalen}(b) & = 1 \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B \text{, then} & \mathit{rnalen}(sb) & = 1 + \mathit{rnalen}(s) \end{array}$$

The function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively by:

$$\begin{array}{llll} & & \mathit{basecount} : S \times B & \to \mathbb{N} \\ \text{Basis Step:} & \text{If } b_1 \in B, b_2 \in B & \mathit{basecount}(\ (b_1, b_2)\ ) & = \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases} \\ \text{Recursive Step:} & \text{If } s \in S, b_1 \in B, b_2 \in B & \mathit{basecount}(\ (sb_1, b_2)\ ) & = \begin{cases} 1 + \mathit{basecount}(\ (s, b_2)\ ) & \text{when } b_1 = b_2 \\ \mathit{basecount}(\ (s, b_2)\ ) & \text{when } b_1 \neq b_2 \end{cases} \end{array}$$

Alternating quantifiers order rna examples

Alternating nested quantifiers

$$\forall s \in S\ \exists n \in \mathbb{N}\ (\ \mathit{basecount}(\ (s, \texttt{U})\ ) = n\ )$$

In English: For each strand, there is a nonnnegative integer that counts the number of occurrences of `U` in that strand.

$$\exists n \in\ \ \forall s \in S\ \mathbb{N}\ (\ \mathit{basecount}(\ (s, \texttt{U})\ ) = n\ )$$

In English: There is a nonnnegative integer that counts the number of occurrences of `U` in every strand.

Are these statements true or false?

$$\forall s \in S \; \exists b \in B \; (\; basecount(\; (s,b) \;) = 3 \;)$$

In English: For each RNA strand there is a base that occurs 3 times in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

Proof strategies quantification finite domain

When a predicate $P(x)$ is over a **finite** domain:

- To show that $\forall x P(x)$ is true: check that $P(x)$ evaluates to T at each domain element by evaluating over and over. This is called "Proof of universal by **exhaustion**".
- To show that $\forall x P(x)$ is false: find a **counterexample**, a domain element where $P(x)$ evaluates to F.
- To show that $\exists x P(x)$ is true: find a **witness**, a domain element where $P(x)$ evaluates to T.
- To show that $\exists x P(x)$ is false: check that $P(x)$ evaluates to F at each domain element by evaluating over and over. DeMorgan's Law gives that $\neg \exists x P(x) \equiv \forall x \neg P(x)$ so this amounts to a proof of universal by exhaustion.

Proof strategy universal generalization

New! Proof by universal generalization: To prove that $\forall x \, P(x)$ is true, we can take an arbitrary element e from the domain of quantification and show that $P(e)$ is true, without making any assumptions about e other than that it comes from the domain.
An **arbitrary** element of a set or domain is a fixed but unknown element from that set.

Quiz translating counting quantifiers

Suppose $P(x)$ is a predicate over a domain D.

1. True or False: To translate the statement "There are at least two elements in D where the predicate P evaluates to true", we could write

$$\exists x_1 \in D \, \exists x_2 \in D \, (P(x_1) \wedge P(x_2))$$

2. True or False: To translate the statement "There are at most two elements in D where the predicate P evaluates to true", we could write

$$\forall x_1 \in D \, \forall x_2 \in D \, \forall x_3 \in D \; (\; (\; P(x_1) \wedge P(x_2) \wedge P(x_3) \;) \rightarrow (\; x_1 = x_2 \vee x_2 = x_3 \vee x_1 = x_3 \;) \;)$$

Proof strategies conditionals

New! Proof of conditional by direct proof: To prove that the conditional statement $p \rightarrow q$ is true, we can assume p is true and use that assumption to show q is true.

New! Proof of conditional by contrapositive proof: To prove that the implication $p \rightarrow q$ is true, we can assume q is false and use that assumption to show p is also false.

New! Proof of disjuction using equivalent conditional: To prove that the disjunction $p \vee q$ is true, we can rewrite it equivalently as $\neg p \rightarrow q$ and then use direct proof or contrapositive proof.

Proof strategies proof by cases

New! Proof by Cases: To prove q, we can work by cases by first describing all possible cases we might be in and then showing that each one guarantees q. Formally, if we know that $p_1 \vee p_2$ is true, and we can show that $(p_1 \rightarrow q)$ is true and we can show that $(p_2 \rightarrow q)$, then we can conclude q is true.

Proof strategies ands

New! Proof of conjunctions with subgoals: To show that $p \wedge q$ is true, we have two subgoals: subgoal (1) prove p is true; and, subgoal (2) prove q is true.

To show that $p \wedge q$ is false, it's enough to prove that $\neg p$.
To show that $p \wedge q$ is false, it's enough to prove that $\neg q$.

Sets proof strategies

To prove that one set is a subset of another, e.g. to show $A \subseteq B$:

To prove that two sets are equal, e.g. to show $A = B$:

Sets equality example

Example: $\{43, 7, 9\} = \{7, 43, 9, 7\}$

Sets basic proofs

Prove or **disprove**: $\{\mathtt{A}, \mathtt{C}, \mathtt{U}, \mathtt{G}\} \subseteq \{\mathtt{AA}, \mathtt{AC}, \mathtt{AU}, \mathtt{AG}\}$

Prove or **disprove**: For some set B, $\emptyset \in B$.

Prove or **disprove**: For every set B, $\emptyset \in B$.

Prove or **disprove**: The empty set is a subset of every set.

Prove or **disprove**: The empty set is a proper subset of every set.

Prove or **disprove**: $\{4, 6\} \subseteq \{n \mid \exists c \in \mathbb{Z}(n = 4c)\}$

Prove or **disprove**: $\{4, 6\} \subseteq \{n \textbf{ mod } 10 \mid \exists c \in \mathbb{Z}(n = 4c)\}$

Proofs signposting

Consider $\ldots$, an **arbitrary** $\ldots$. **Assume** $\ldots$, we **want to show** that $\ldots$. Which is what was needed, so the proof is complete $\square$.

or, in other words:

Let $\ldots$ be an **arbitrary** $\ldots$. **Assume** $\ldots$, **WTS** that $\ldots$ **QED**.

Set operations union intersection powerset

Cartesian product: When A and B are sets,

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

Example: $\{43, 9\} \times \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} \times \emptyset =$

Union: When A and B are sets,

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

Example: $\{43, 9\} \cup \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} \cup \emptyset =$

Intersection: When A and B are sets,

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

Example: $\{43, 9\} \cap \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} \cap \emptyset =$

Set difference: When A and B are sets,

$$A - B = \{x \mid x \in A \wedge x \notin B\}$$

Example: $\{43, 9\} - \{9, \mathbb{Z}\} =$

Example: $\mathbb{Z} - \emptyset =$

Disjoint sets: sets A and B are disjoint means $A \cap B = \emptyset$

Example: $\{43, 9\}, \{9, \mathbb{Z}\}$ are not disjoint

Example: The sets $\mathbb{Z}$ and $\emptyset$ are disjoint

Power set: When U is a set, $\mathcal{P}(U) = \{X \mid X \subseteq U\}$

Example: $\mathcal{P}(\{43, 9\}) =$

Example: $\mathcal{P}(\emptyset) =$

Quantification definition

The **universal quantification** of predicate $P(x)$ over domain U is the statement "$P(x)$ for all values of x in the domain U" and is written $\forall x P(x)$ or $\forall x \in U\ P(x)$. When the domain is finite, universal quantification over the domain is equivalent to iterated *conjunction* (ands).

The **existential quantification** of predicate $P(x)$ over domain U is the statement "There exists an element x in the domain U such that $P(x)$" and is written $\exists x P(x)$ for $\exists x \in U\ P(x)$. When the domain is finite, existential quantification over the domain is equivalent to iterated *disjunction* (ors).

An element for which $P(x) = F$ is called a **counterexample** of $\forall x P(x)$.

An element for which $P(x) = T$ is called a **witness** of $\exists x P(x)$.

Quantification logical equivalence

Statements involving predicates and quantifiers are **logically equivalent** means they have the same truth value no matter which predicates (domains and functions) are substituted in.

Quantifier version of De Morgan's laws: $\boxed{\neg \forall x P(x) \ \equiv\ \exists x\, (\neg P(x))}$ $\boxed{\neg \exists x Q(x) \ \equiv\ \forall x\, (\neg Q(x))}$

Quantification examples finite domain

Examples of quantifications using $V(x), N(x), Mystery(x)$:

True or **False**: $\exists x\ (\ V(x) \wedge N(x)\ )$

True or **False**: $\forall x\ (\ V(x) \to N(x)\ )$

True or **False**: $\exists x\ (\ N(x) \leftrightarrow Mystery(x)\ )$

Rewrite $\neg \forall x\ (\ V(x) \oplus Mystery(x)\ )$ into a logical equivalent statement.

Notice that these are examples where the predicates have *finite* domain. How would we evaluate quantifications where the domain may be infinite?

Rna rnalen basecount definitions

Recall the definitions: The set of RNA strands S is defined (recursively) by:

$$\begin{array}{ll} \text{Basis Step:} & \texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B \text{, then } sb \in S \end{array}$$

where sb is string concatenation.

The function *rnalen* that computes the length of RNA strands in S is defined recursively by:

$$\begin{array}{lllll} & & \mathit{rnalen} : S & \rightarrow \mathbb{Z}^+ \\ \text{Basis Step:} & \text{If } b \in B \text{ then} & \mathit{rnalen}(b) & = 1 \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B \text{, then} & \mathit{rnalen}(sb) & = 1 + \mathit{rnalen}(s) \end{array}$$

The function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively by:

$$\begin{array}{llll} & & \mathit{basecount} : S \times B & \rightarrow \mathbb{N} \\ \text{Basis Step:} & \text{If } b_1 \in B, b_2 \in B & \mathit{basecount}(\ (b_1, b_2)\ ) & = \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases} \\ \text{Recursive Step:} & \text{If } s \in S, b_1 \in B, b_2 \in B & \mathit{basecount}(\ (sb_1, b_2)\ ) & = \begin{cases} 1 + \mathit{basecount}(\ (s, b_2)\ ) & \text{when } b_1 = b_2 \\ \mathit{basecount}(\ (s, b_2)\ ) & \text{when } b_1 \neq b_2 \end{cases} \end{array}$$

Predicates example rnalen basecount

Using functions to define predicates:

L with domain $S \times \mathbb{Z}^+$ is defined by, for $s \in S$ and $n \in \mathbb{Z}^+$,

$$L(\ (s,n)\ ) = \begin{cases} T & \text{if } rnalen(s) = n \\ F & \text{otherwise} \end{cases}$$

In other words, $L(\ (s,n)\ )$ means $rnalen(s) = n$

BC with domain $S \times B \times \mathbb{N}$ is defined by, for $s \in S$ and $b \in B$ and $n \in \mathbb{N}$,

$$BC(\ (s,b,n)\ ) = \begin{cases} T & \text{if } basecount(\ (s,b)\ ) = n \\ F & \text{otherwise} \end{cases}$$

In other words, $BC(\ (s,b,n)\ )$ means $basecount(\ (s,b)\ ) = n$

Example where L evaluates to T: ________________ Why?

Example where BC evaluates to T: ________________ Why?

Example where L evaluates to F: ________________ Why?

Example where BC evaluates to F: ________________ Why?

$\exists t\ BC(t)$ $\quad\quad$ $\exists (s,b,n) \in S \times B \times \mathbb{N}\ (basecount(\ (s,b)\ ) = n)$

In English:

Witness that proves this existential quantification is true:

$\forall t\ BC(t)$ $\quad\quad$ $\forall (s,b,n) \in S \times B \times \mathbb{N}\ (basecount(\ (s,b)\ ) = n)$

In English:

Counterexample that proves this universal quantification is false:

Predicates projecting example rna basecount

New predicates from old

1. Define the **new** predicate with domain $S \times B$ and rule

$$basecount(\ (s, b)\ ) = 3$$

Example domain element where predicate is T:

2. Define the **new** predicate with domain $S \times \mathbb{N}$ and rule

$$basecount(\ (s, \mathtt{A})\ ) = n$$

Example domain element where predicate is T:

3. Define the **new** predicate with domain $S \times B$ and rule

$$\exists n \in \mathbb{N}\ (basecount(\ (s, b)\ ) = n)$$

Example domain element where predicate is T:

4. Define the **new** predicate with domain S and rule

$$\forall b \in B\ (basecount(\ (s, b)\ ) = 1)$$

Example domain element where predicate is T:

Nested quantifiers

Nested quantifiers

$$\forall s \in S \ \forall b \in B \ \forall n \in \mathbb{N} \ (basecount(\ (s,b)\ ) = n)$$

In English:

Counterexample that proves this universal quantification is false:

$$\forall n \in \mathbb{N} \ \forall s \in S \ \forall b \in B \ (basecount(\ (s,b)\ ) = n)$$

In English:

Counterexample that proves this universal quantification is false:

Alternating quantifiers strategies rna examples

Alternating nested quantifiers

$$\forall s \in S \ \exists b \in B \ (\ basecount(\ (s,b) \ ) = 3 \ )$$

In English: For each RNA strand there is a base that occurs 3 times in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

$$\exists s \in S \ \forall b \in B \ \exists n \in \mathbb{N} \ (\ basecount(\ (s,b) \ ) = n \ )$$

In English: There is an RNA strand so that for each base there is some nonnegative integer that counts the number of occurrences of that base in this strand.

Write the negation and use De Morgan's law to find a logically equivalent version where the negation is applied only to the BC predicate (not next to a quantifier).

Is the original statement **True** or **False**?

Tautology contradiction contingency examples

Label each of the following as a tautology, contradiction, or contingency.

$p \wedge p$

$p \oplus p$

$p \vee p$

$p \vee \neg p$

$p \wedge \neg p$

Why represent numbers

Modeling uses data-types that are encoded in a computer. The details of the encoding impact the efficiency of algorithms we use to understand the systems we are modeling and the impacts of these algorithms on the people using the systems. Case study: how to encode numbers?

Fixed width definition

Definition For b an integer greater than 1, w a positive integer, and n a nonnegative integer _____________, the **base b fixed-width w expansion of n** is

$$(a_{w-1} \cdots a_1 a_0)_{b,w}$$

where $a_0, a_1, \ldots, a_{w-1}$ are nonnegative integers less than b and

$$n = \sum_{i=0}^{w-1} a_i b^i$$

Fixed width example

Decimal $b = 10$	Binary $b = 2$	Binary fixed-width 10 $b = 2$, $w = 10$	Binary fixed-width 7 $b = 2$, $w = 7$	Binary fixed-width 4 $b = 2$, $w = 4$
$(20)_{10}$				

Fixed width fractional definition

Definition For b an integer greater than 1, w a positive integer, w' a positive integer, and x a real number the **base b fixed-width expansion of x with integer part width w and fractional part width w'** is $(a_{w-1}\cdots a_1 a_0.c_1\cdots c_{w'})_{b,w,w'}$ where $a_0, a_1, \ldots, a_{w-1}, c_1, \ldots, c_{w'}$ are nonnegative integers less than b and

$$x \geq \sum_{i=0}^{w-1} a_i b^i + \sum_{j=1}^{w'} c_j b^{-j} \qquad \text{and} \qquad x < \sum_{i=0}^{w-1} a_i b^i + \sum_{j=1}^{w'} c_j b^{-j} + b^{-w'}$$

3.75 in fixed-width binary, integer part width 2, fractional part width 8	
0.1 in fixed-width binary, integer part width 2, fractional part width 8	

```
welcome $jshell
|  Welcome to JShell -- Version 10.0.1
|  For an introduction type: /help intro

jshell> 0.1
$1 ==>

jshell> 0.2
$2 ==>

jshell> 0.1 + 0.2
$3 ==>

jshell> Math.sqrt(2)
$4 ==>

jshell> Math.sqrt(2)*Math.sqrt(2)
$5 ==>

jshell>
```

Note: Java uses floating point, not fixed width representation, but similar rounding errors appear in both.

Negative int expansions

Representing negative integers in binary: Fix a positive integer width for the representation w, $w > 1$.

	To represent a positive integer n	To represent a negative integer $-n$
Sign-magnitude	$[0a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$ Example $n = 17$, $w = 7$:	$[1a_{w-2} \cdots a_0]_{s,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$ Example $-n = -17$, $w = 7$:
2s complement	$[0a_{w-2} \cdots a_0]_{2c,w}$, where $n = (a_{w-2} \cdots a_0)_{2,w-1}$ Example $n = 17$, $w = 7$:	$[1a_{w-2} \cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2} \cdots a_0)_{2,w-1}$ Example $-n = -17$, $w = 7$:

Calculating 2s complement

For positive integer n, to represent $-n$ in 2s complement with width w,

- Calculate $2^{w-1} - n$, convert result to binary fixed-width $w - 1$, pad with leading 1, or
- Express $-n$ as a sum of powers of 2, where the leftmost 2^{w-1} is negative weight, or
- Convert n to binary fixed-width w, flip bits, add 1 (ignore overflow)

Challenge: use definitions to explain why each of these approaches works.

Representing zero

Representing 0:

So far, we have representations for positive and negative integers. What about 0?

	To represent a **non-negative** integer n	To represent a **non-positive** integer $-n$
Sign-magnitude	$[0a_{w-2}\cdots a_0]_{s,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 0$, $w = 7$:	$[1a_{w-2}\cdots a_0]_{s,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $-n = 0$, $w = 7$:
2s complement	$[0a_{w-2}\cdots a_0]_{2c,w}$, where $n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $n = 0$, $w = 7$:	$[1a_{w-2}\cdots a_0]_{2c,w}$, where $2^{w-1} - n = (a_{w-2}\cdots a_0)_{2,w-1}$ Example $-n = 0$, $w = 7$: