Binary relation definition

Definition: When A and B are sets, we say any subset of $A \times B$ is a **binary relation**. A relation R can also be represented as

- A function $f_{TF} : A \times B \to \{T, F\}$ where, for $a \in A$ and $b \in B$, $f_{TF}(\ (a, b)\ ) = \begin{cases} T & \text{when } (a, b) \in R \\ F & \text{when } (a, b) \notin R \end{cases}$
- A function $f_{\mathcal{P}} : A \to \mathcal{P}(B)$ where, for $a \in A$, $f_{\mathcal{P}}(a) = \{b \in B \mid (a, b) \in R\}$

When A is a set, we say any subset of $A \times A$ is a (binary) **relation** on A.

Binary relation examples

Example: For $A = \mathcal{P}(\mathbb{R})$, we can define the relation $EQ_{\mathbb{R}}$ on A as

$$\{(X_1, X_2) \in \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}) \mid |X_1| = |X_2|\}$$

Example: Let $R_{(\textbf{mod } n)}$ be the set of all pairs of integers (a, b) such that (a **mod** $n = b$ **mod** n). Then a is **congruent to** b **mod** n means $(a, b) \in R_{(\textbf{mod } n)}$. A common notation is to write this as $a \equiv b (\textbf{mod } n)$.

$R_{(\textbf{mod } n)}$ is a relation on the set _____

Some example elements of $R_{(\textbf{mod } 4)}$ are:

Reflexive relation definition

A relation R on a set A is called **reflexive** means $(a, a) \in R$ for every element $a \in A$.

Reflexive relation informally

Informally, every element is related to itself.

Graphically, there are self-loops (edge from a node back to itself) at every node.

Symmetric relation definition

A relation R on a set A is called **symmetric** means $(b, a) \in R$ whenever $(a, b) \in R$, for all $a, b \in A$.

Symmetric relation informally

Informally, order doesn't matter for this relation.

Graphically, every edge has a paired "backwards" edge so we might as well drop the arrows and think of edges as undirected.

Transitive relation definition

A relation R on a set A is called **transitive** means whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$, for all $a, b, c \in A$.

Transitive relation informally

Informally, chains of relations collapse.

Graphically, there's a shortcut between any endpoints of a chain of edges.

Antisymmetric relation definition

A relation R on a set A is called **antisymmetric** means $\forall a \in A \, \forall b \in A \; (\; (\; (a, b) \in R \wedge (b, a) \in R \;) \rightarrow a = b \;)$

Antisymmetric relation informally

Informally, the relation has directionality.

Graphically, can organize the nodes of the graph so that all non-self loop edges go up.

Binary relation properties examples

When the domain is $\{a, b, c, d, e, f, g, h\}$ define a relation that is **not reflexive** and is **not symmetric** and is **not transitive**.

When the domain is $\{a, b, c, d, e, f, g, h\}$ define a relation that is **not reflexive** but is **symmetric** and is **transitive**.

When the domain is $\{a, b, c, d, e, f, g, h\}$ define a relation that is **symmetric** and is **antisymmetric**.

Is the relation $EQ_{\mathbb{R}}$ reflexive? symmetric? transitive? antisymmetric?

Is the relation $R_{(\mathbf{mod}\ 4)}$ reflexive? symmetric? transitive? antisymmetric?

Cardinality caution

Caution: we use familiar symbols to define cardinality, like $|\quad| \leq |\quad|$ and $|\quad| \geq |\quad|$ and $|\quad| = |\quad|$, but the meaning of these symbols depends on context. We've seen that vertical lines can mean absolute value (for real numbers), divisibility (for integers), and now sizes (for sets).

Now we see that $\leq$ and $\geq$ can mean comparing numbers or comparing sizes of sets. When the sets being compared are finite, the definitions of $|A| \leq |B|$ agree.

But, properties of numbers cannot be assumed when comparing cardinalities of infinite sets.

In a nutshell: cardinality of sets is defined via functions. This definition agrees with the usual notion of "size" for finite sets.

Cantor schroder bernstein theorem

Cantor-Schroder-Bernstein Theorem: For all nonempty sets,

$$|A| = |B| \quad \text{if and only if} \quad (|A| \leq |B| \text{ and } |B| \leq |A|) \quad \text{if and only if} \quad (|A| \geq |B| \text{ and } |B| \geq |A|)$$

To prove $|A| = |B|$, we can do any **one** of the following

- Prove there exists a bijection $f : A \to B$;
- Prove there exists a bijection $f : B \to A$;
- Prove there exists two functions $f_1 : A \to B$, $f_2 : B \to A$ where each of f_1, f_2 is one-to-one.
- Prove there exists two functions $f_1 : A \to B$, $f_2 : B \to A$ where each of f_1, f_2 is onto.

Countably infinite definition

Definition: A set A is **countably infinite** means it is the same size as $\mathbb{N}$.

Countably infinite examples sets of numbers

Natural numbers $\mathbb{N}$ *List*: $0 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8 \;\; 9 \;\; 10 \ldots$

$identity : \mathbb{N} \to \mathbb{N}$ with $identity(n) = n$

Claim: *identity* is a bijection. *Proof*: Ex. **Corollary**: $|\mathbb{N}| = |\mathbb{N}|$

Positive integers $\mathbb{Z}^+$ *List*: $1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8 \;\; 9 \;\; 10 \;\; 11 \ldots$

$positives : \mathbb{N} \to \mathbb{Z}^+$ with $positives(n) = n + 1$

Claim: *positives* is a bijection. *Proof*: Ex. **Corollary**: $|\mathbb{N}| = |\mathbb{Z}^+|$

Negative integers $\mathbb{Z}^-$ *List*: $-1 \; -2 \; -3 \; -4 \; -5 \; -6 \; -7 \; -8 \; -9 \; -10 \; -11 \ldots$

$negatives : \mathbb{N} \to \mathbb{Z}^-$ with $negatives(n) = -n - 1$

Claim: *negatives* is a bijection. **Corollary**: $|\mathbb{N}| = |\mathbb{Z}^-|$

Proof: We need to show it is a well-defined function that is one-to-one and onto.

- Well-defined?

  Consider an arbitrary element of the domain, $n \in \mathbb{N}$. We need to show it maps to exactly one element of $\mathbb{Z}^-$.

- One-to-one?

  Consider arbitrary elements of the domain $a, b \in \mathbb{N}$. We need to show that

  $$(\; negatives(a) = negatives(b) \;) \to (a = b)$$

- Onto?

  Consider arbitrary element of the codomain $b \in \mathbb{Z}^-$. We need witness in $\mathbb{N}$ that maps to b.

Integers $\mathbb{Z}$ *List*: $0 \; -1 \;\; 1 \; -2 \;\; 2 \; -3 \;\; 3 \; -4 \;\; 4 \; -5 \;\; 5 \ldots$

$$f : \mathbb{Z} \to \mathbb{N} \text{ with } f(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases}$$

Claim: f is a bijection. *Proof*: Ex. **Corollary**: $|\mathbb{Z}| = |\mathbb{N}|$

Countably infinite examples other sets

More examples of countably infinite sets

Claim: S is countably infinite

Similarly: The set of all strings over a specific alphabet is countably infinite.

Bijection using alphabetical-ish ordering (first order by length, then alphabetically among strings of same length) of strands

Claim: L is countably infinite

$$list : \mathbb{N} \to L$$
$$list(n) = (n, [])$$

$$toNum : L \to \mathbb{N}$$
$$toNum([]) = 0$$
$$toNum(\ (n, l)\ ) = 2^n 3^{toNum}(l) \qquad \text{for } n \in \mathbb{N},\ l \in L$$

Claim: $|\mathbb{Z}^+| = |\mathbb{Q}|$

One-to-one function from $\mathbb{Z}^+$ to $\mathbb{Q}$ is $f_1 : \mathbb{Z} \to \mathbb{Q}$ with $f_1(n) = n$ for all $n \in \mathbb{N}$.

$$f_2 : \mathbb{Q} \to \mathbb{Z} \times \mathbb{Z}$$
$$f_2(x) = \begin{cases} (0,1) & \text{if } x = 0 \\ (p,q) & \text{if } x = \frac{p}{q}, \\ & gcd(p,q) = 1,\ q > 0 \end{cases}$$

$$f_3 : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^+ \times \mathbb{Z}^+$$
$$f_3(\ (x,y)\ ) = \begin{cases} (2x+2, 2y+2) & \text{if } x \geq 0, y \geq 0 \\ (-2x-1, 2y+2) & \text{if } x < 0, y \geq 0 \\ (2x+2, -2y+1) & \text{if } x \geq 0, y < 0 \\ (-2x-1, -2y-1) & \text{if } x < 0, y < 0 \end{cases}$$

$$f_4 : \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{Z}^+$$
$$f_4(\ (x,y)\ ) = 2^x 3^y \qquad \text{for } x, y \in \mathbb{Z}^+$$

Cardinality categories

A set A is **finite** means it is empty or it is the same size as $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$.

A set A is **countably infinite** means it is the same size as $\mathbb{N}$. *Notice: all countably infinite sets are the same size as each other.*

A set A is **countable** means it is either finite or countably infinite.

A set A is **uncountable** means it is not countable.

Cardinality countability lemmas

Lemmas about countable and uncountable sets

Lemma: If A is a subset of a countable set, then it's countable.

Lemma: If A is a superset of an uncountable set, then it's uncountable.

Lemma: If A and B are countable sets, then $A \cup B$ is countable and $A \cap B$ is countable.

Lemma: If A and B are countable sets, then $A \times B$ is countable.

Generalize pairing ideas from $\mathbb{Z}^+ \times \mathbb{Z}^+$ *to* $\mathbb{Z}^+$

Lemma: If A is a subset of B , to show that $|A| = |B|$, it's enough to give one-to-one function from B to A or an onto function from A to B.

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*((`AUC`, 3, `G`))

Fill in the blanks so that *insertion*((`AUC`, _, _)) = `AUCG`

Fill in the blanks so that *deletion*((__, _)) = `G`

Rna rnalen basecount definitions

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

Basis Step:	$\mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

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

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

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

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

Alternating quantifiers order rna examples

Alternating nested quantifiers

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

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

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

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 $\neg \exists x P(x) \quad \equiv \quad \forall x \neg P(x)$ 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**: $\{\texttt{A}, \texttt{C}, \texttt{U}, \texttt{G}\} \subseteq \{\texttt{AA}, \texttt{AC}, \texttt{AU}, \texttt{AG}\}$

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 . . . , an **arbitrary** . . . . **Assume** . . . , we **want to show** that . . . . Which is what was needed, so the proof is complete □.

or, in other words:

Let . . . be an **arbitrary** . . . . **Assume** . . . , **WTS** that . . . **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**: $\forall x\ (\ V(x) \rightarrow N(x)\ )$

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:

Basis Step:	$\mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

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

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

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

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

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?

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

In English:

Witness that proves this existential quantification is true:

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

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**?

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 basic proofs operations

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$

Example elements in W are:

Prove or **disprove**: $\forall A \in W \, \forall B \in W \; (A \subseteq B \; \rightarrow \; \mathcal{P}(A) \subseteq \mathcal{P}(B))$

Prove or **disprove**: $\forall A \in W \, \forall B \in W \; (\mathcal{P}(A) = \mathcal{P}(B) \; \rightarrow \; A = B)$

Prove or **disprove**: $\forall A \in W \, \forall B \in W \, \forall C \in W \; (A \cup B = A \cup C \; \rightarrow \; B = C)$

Proof strategies road map

We now have propositional and predicate logic that can help us express statements about any domain. We will develop proof strategies to craft valid argument for proving that such statements are true or disproving them (by showing they are false). We will practice these strategies with statements about sets and numbers, both because they are familiar and because they can be used to build cryptographic systems. Then we will apply proof strategies more broadly to prove statements about data structures and machine learning applications.

Numbers facts

1. Addition and multiplication of real numbers are each commutative and associative.

2. The product of two positive numbers is positive, of two negative numbers is positive, and of a positive and a negative number is negative.

3. The sum of two integers, the product of two integers, and the difference between two integers are each integers.

4. For every integer x there is no integer strictly between x and $x + 1$,

5. When x, y are positive integers, $xy \geq x$ and $xy \geq y$.

Factoring definition

Definition: When a and b are integers and a is nonzero, a **divides** b means there is an integer c such that $b = ac$.

Symbolically, $F(\ (a, b)\ ) =$ and is a predicate over the domain _____

Other (synonymous) ways to say that $F(\ (a, b)\ )$ is true:

a is a **factor** of b a is a **divisor** of b b is a **multiple** of a $a|b$

When a is a positive integer and b is any integer, $a|b$ exactly when b **mod** $a = 0$

When a is a positive integer and b is any integer, $a|b$ exactly when $b = a \cdot (b$ **div** $a)$

Factoring translation examples

Translate these quantified statements by matching to English statement on right.

$\exists a \in \mathbb{Z}^{\neq 0}\ (\ F(\ (a, a)\ )\ )$	Every nonzero integer is a factor of itself.
$\exists a \in \mathbb{Z}^{\neq 0}\ (\ \neg F(\ (a, a)\ )\ )$	No nonzero integer is a factor of itself.
$\forall a \in \mathbb{Z}^{\neq 0}\ (\ F(\ (a, a)\ )\ )$	At least one nonzero integer is a factor of itself.
$\forall a \in \mathbb{Z}^{\neq 0}\ (\ \neg F(\ (a, a)\ )\ )$	Some nonzero integer is not a factor of itself.

Factoring basic claims

Claim: Every nonzero integer is a factor of itself.

Proof:

Prove or **Disprove**: There is a nonzero integer that does not divide its square.

Prove or **Disprove**: Every positive factor of a positive integer is less than or equal to it.

Factoring basic claims continued

Claim: Every nonzero integer is a factor of itself and every nonzero integer divides its square.

Factoring even odd

Definition: an integer n is **even** means that there is an integer a such that $n = 2a$; an integer n is **odd** means that there is an integer a such that $n = 2a + 1$. Equivalently, an integer n is **even** means $n \mod 2 = 0$; an integer n is **odd** means $n \mod 2 = 1$. Also, an integer is even if and only if it is not odd.

Prime number definition

Definition: An integer p greater than 1 is called **prime** means the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called composite.

Primes basic claims

Extra examples: Use the definition to prove that 1 is not prime, 2 is prime, 3 is prime, 4 is not prime, 5 is prime, 6 is not prime, and 7 is prime.

True or False: The statement "There are three consecutive positive integers that are prime."

Hint: These numbers would be of the form $p, p+1, p+2$ (where p is a positive integer).

Proof: We need to show _____

True or False: The statement "There are three consecutive odd positive integers that are prime."

Hint: These numbers would be of the form $p, p+2, p+4$ (where p is an odd positive integer).

Proof: We need to show _____

Rna rnalen basecount definitions

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

Basis Step:	$\mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

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

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

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

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

Alternating quantifiers proofs rna examples

Which proof strategies could be used to prove each of the following statements?

Hint: first translate the statements to English and identify the main logical structure.

$\forall s \in S$ ($rnalen(s) > 0$)

$\forall b \in B \; \exists s \in S$ ($basecount(\ (s, b)\ )\ > 0$)

$\forall s \in S \; \exists b \in B$ ($basecount(\ (s, b)\ ) > 0$)

$\exists s \in S \, (rnalen(s) = basecount(\ (s, \texttt{A})\ )$

$\forall s \in S \, (rnalen(s) \geq basecount(\ (s, \texttt{A})\ ))$

Structural induction motivating example rna

Claim $\forall s \in S$ ($rnalen(s) > 0$)

Proof: Let s be an arbitrary RNA strand. By the recursive definition of S, either $s \in B$ or there is some strand s_0 and some base b such that $s = s_0 b$. We will show that the inequality holds for both cases.

Case: Assume $s \in B$. We need to show $rnalen(s) > 0$. By the basis step in the definition of $rnalen$,

$$rnalen(s) = 1$$

which is greater than 0, as required.

Case: Assume there is some strand s_0 and some base b such that $s = s_0 b$. We will show *(the stronger claim)* that

$$\forall u \in S \; \forall b \in B \; (\; rnalen(u) > 0 \rightarrow rnalen(ub) > 0 \;)$$

Consider an arbitrary RNA strand u and an arbitrary base b, and assume towards a direct proof, that

$$rnalen(u) > 0$$

We need to show that $rnalen(ub) > 0$.

$$rnalen(ub) = 1 + rnalen(u) > 1 + 0 = 1 > 0$$

as required.

Proof strategies structural induction

Proof by Structural Induction To prove a universal quantification over a recursively defined set:

Basis Step: Show the statement holds for elements specified in the basis step of the definition.

Recursive Step: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

Structural induction example rnalen basecount

Claim $\forall s \in S\,(rnalen(s) \geq basecount(\ (s, \mathtt{A})\ ))$:

Proof: We proceed by structural induction on the recursively defined set S.

Basis Case: We need to prove that the inequality holds for each element in the basis step of the recursive definition of S. Need to show

$$(\ rnalen(\mathtt{A}) \geq basecount(\ (\mathtt{A}, \mathtt{A})\ )\ ) \wedge (\ rnalen(\mathtt{C}) \geq basecount(\ (\mathtt{C}, \mathtt{A})\ )\ )$$
$$\wedge(\ rnalen(\mathtt{U}) \geq basecount(\ (\mathtt{U}, \mathtt{A})\ )\ ) \wedge (\ rnalen(\mathtt{G}) \geq basecount(\ (\mathtt{G}, \mathtt{A})\ )\ )$$

We calculate, using the definitions of *rnalen* and *basecount*:

Recursive Case: We will prove that

$$\forall u \in S\ \forall b \in B\ (\ rnalen(u) \geq basecount(\ (u, \mathtt{A})\ ) \rightarrow rnalen(ub) \geq basecount(\ (ub, \mathtt{A})\ )$$

Consider arbitrary RNA strand u and arbitrary base b. Assume, as the **induction hypothesis**, that $rnalen(u) \geq basecount(\ (u, \mathtt{A})\ )$. We need to show that $rnalen(ub) \geq basecount(\ (ub, \mathtt{A})\ )$.

Using the recursive step in the definition of the function *rnalen*:

$$rnalen(ub) = 1 + rnalen(u)$$

The recursive step in the definition of the function *basecount* has two cases. We notice that $b = \mathtt{A} \vee b \neq \mathtt{A}$ and we proceed by cases.

Case i. Assume $b = \mathtt{A}$.

Using the first case in the recursive step in the definition of the function *basecount*:

$$basecount(\ (ub, \mathtt{A})\ ) = 1 + basecount(\ (u, \mathtt{A})\ )$$

By the **induction hypothesis**, we know that $basecount(\ (u, \mathtt{A})\ ) \leq rnalen(u)$ so:

$$basecount(\ (ub, \mathtt{A})\ ) = 1 + basecount(\ (u, \mathtt{A})\ ) \leq 1 + rnalen(u) = rnalen(ub)$$

and, thus, $basecount(\ (ub, \mathtt{A})\ ) \leq rnalen(ub)$, as required.

Case ii. Assume $b \neq \mathtt{A}$.

Using the second case in the recursive step in the definition of the function *basecount*:

$$basecount(\ (ub, \mathtt{A})\ ) = basecount(\ (u, \mathtt{A})\ )$$

By the **induction hypothesis**, we know that $basecount(\ (u, \texttt{A})\ ) \leq rnalen(u)$ so:

$$basecount(\ (ub, \texttt{A})\ ) = basecount(\ (u, \texttt{A})\ ) \leq rnalen(u) < 1 + rnalen(u) = rnalen(ub)$$

and, thus, $basecount(\ (ub, \texttt{A})\ ) \leq rnalen(ub)$, as required.

Proofs signposting kinds of claims

To organize our proofs, it's useful to highlight which claims are most important for our overall goals. We use some terminology to describe different roles statements can have.

Theorem: Statement that can be shown to be true, usually an important one.

Less important theorems can be called **proposition**, **fact**, **result**, **claim**.

Lemma: A less important theorem that is useful in proving a theorem.

Corollary: A theorem that can be proved directly after another one has been proved, without needing a lot of extra work.

Invariant: A theorem that describes a property that is true about an algorithm or system no matter what inputs are used.

Structural induction example sum of powers

The set $\mathbb{N}$ is recursively defined. Therefore, the function $sumPow : \mathbb{N} \to \mathbb{N}$ which computes, for input i, the sum of the nonnegative powers of 2 up to and including exponent i is defined recursively by

Basis step: $sumPow(0) = 1$

Recursive step: If $x \in \mathbb{N}$, then $sumPow(x+1) = sumPow(x) + 2^{x+1}$

$sumPow(0) =$

$sumPow(1) =$

$sumPow(2) =$

Fill in the blanks in the following proof of

$$\forall n \in \mathbb{N}\ (sumPow(n) = 2^{n+1} - 1)$$

Proof: Since $\mathbb{N}$ is recursively defined, we proceed by _____.

Basis case: We need to show that _____. Evaluating each side: $LHS = sumPow(0) = 1$ by the basis case in the recursive definition of $sumPow$; $RHS = 2^{0+1} - 1 = 2^1 - 1 = 2 - 1 = 1$. Since $1 = 1$, the equality holds.

Recursive case: Consider arbitrary natural number n and assume, as the _____ that $sumPow(n) = 2^{n+1} - 1$. We need to show that _____. Evaluating each side:

$$LHS = sumPow(n+1) \overset{\text{rec def}}{=} sumPow(n) + 2^{n+1} \overset{\text{IH}}{=} (2^{n+1} - 1) + 2^{n+1}.$$

$$RHS = 2^{(n+1)+1} - 1 \overset{\text{exponent rules}}{=} 2 \cdot 2^{n+1} - 1 = \left(2^{n+1} + 2^{n+1}\right) - 1 \overset{\text{regrouping}}{=} (2^{n+1} - 1) + 2^{n+1}$$

Thus, $LHS = RHS$. The structural induction is complete and we have proved the universal generalization. □

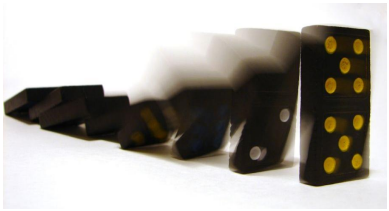
Proof strategy mathematical induction

Proof by Mathematical Induction
To prove a universal quantification over the set of all integers greater than or equal to some base integer b,

Basis Step: Show the property holds for b.

Recursive Step: Consider an arbitrary integer n greater than or equal to b, assume (as the **induction hypothesis**) that the property holds for n, and use this and other facts to prove that the property holds for $n + 1$.

Induction dominos



Proof strategy mathematical induction

Proof by Mathematical Induction
To prove a universal quantification over the set of all integers greater than or equal to some base integer b,

Basis Step: Show the property holds for b.

Recursive Step: Consider an arbitrary integer n greater than or equal to b, assume (as the **induction hypothesis**) that the property holds for n, and use this and other facts to prove that the property holds for $n + 1$.

Proof strategy strong induction

Proof by Strong Induction

To prove that a universal quantification over the set of all integers greater than or equal to some base integer b holds, pick a fixed nonnegative integer j and then:

Basis Step:	Show the statement holds for b, $b+1$, $\ldots$, $b+j$.
Recursive Step:	Consider an arbitrary integer n greater than or equal to $b+j$, assume (as the **strong induction hypothesis**) that the property holds for **each of** b, $b+1$, $\ldots$, n, and use this and other facts to prove that the property holds for $n+1$.

Binary expansions exist proof

Theorem: Every positive integer is a sum of (one or more) distinct powers of 2. *Binary expansions exist!*

Recall the definition for binary expansion:

Definition For n a positive integer, the **binary expansion of** n is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are each 0 or 1, $a_{k-1} \neq 0$, and

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

The idea in the "Least significant first" algorithm for computing binary expansions is that the binary expansion of *half* a number becomes *part* of the binary expansion of the number of itself. We can use this idea in a proof by strong induction that binary expansions exist for all positive integers n.

Proof by strong induction, with $b = 1$ and $j = 0$.

Basis step: WTS property is true about 1.

Recursive step: Consider an arbitrary integer $n \geq 1$.

Assume (as the strong induction hypothesis, IH) that the property is true about each of $1, \ldots, n$.

WTS that the property is true about $n + 1$.

Idea: We will apply the IH to $(n + 1)$ **div** 2.

Why is this ok?

Why is this helpful?

By the IH, we can write $(n+1)$ **div** 2 as a sum of powers of 2. In other words, there are values $a_{k-1}, \ldots, a_0$ such that each a_i is 0 or 1, $a_{k-1} = 1$, and

$$\sum_{i=0}^{k-1} a_i 2^i = (n+1) \textbf{ div } 2$$

Define the collection of coefficients

$$c_j = \begin{cases} a_{j-1} & \text{if } 1 \le j \le k \\ (n+1) \textbf{ mod } 2 & \text{if } j = 0 \end{cases}$$

Calculating:

$$\begin{aligned}
\sum_{j=0}^{k} c_j 2^j &= c_0 + \sum_{j=1}^{k} c_j 2^j = c_0 + \sum_{i=0}^{k-1} c_{i+1} 2^{i+1} && \text{re-indexing the summation} \\
&= c_0 + 2 \cdot \sum_{i=0}^{k-1} c_{i+1} 2^i && \text{factoring out a 2 from each term in the sum} \\
&= c_0 + 2 \cdot \sum_{i=0}^{k-1} a_i 2^i && \text{by definition of } c_{i+1} \\
&= c_0 + 2\,(\ (n+1) \textbf{ div } 2\ ) && \text{by IH} \\
&= (\ (n+1) \textbf{ mod } 2\ ) + 2\,(\ (n+1) \textbf{ div } 2\ ) && \text{by definition of } c_0 \\
&= n+1 && \text{by definition of long division}
\end{aligned}$$

Thus, $n+1$ can be expressed as a sum of powers of 2, as required.

Linked lists definition

Definition The set of linked lists of natural numbers L is defined recursively by

Basis Step:	$[] \in L$
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$, then $(n, l) \in L$

Linked lists examples

Visually:

Example: the list with two nodes whose first node has 20 and whose second node has 42

Linked list length definition

Definition: The length of a linked list of natural numbers L, $length : L \to \mathbb{N}$ is defined by

Basis Step:		$length(\ []\ )$	$= 0$
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$, then	$length(\ (n, l)\ )$	$= 1 + length(l)$

Linked lists prepend definition

Definition: The function $prepend : L \times \mathbb{N} \to L$ that adds an element at the front of a linked list is defined by

Linked list append definition

Definition The function $append : L \times \mathbb{N} \to L$ that adds an element at the end of a linked list is defined by

Basis Step:	If $m \in \mathbb{N}$ then
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$ and $m \in \mathbb{N}$, then

Linked list append length claim proof

Claim: $\forall l \in L$ ($length($ $append($ $(l, 100)$)) $> length(l)$)

Proof: By structural induction on L, we have two cases:

Basis Step

1. **To Show** $length($ $append($ $([], 100)$)) $> length($ [])	Because [] is the only element defined in the basis step of L, we only need to prove that the property holds for [].
2. **To Show** $length($ $(100, [])$) $> length($ [])	By basis step in definition of *append*.
3. **To Show** $(1 + length($ [])) $> length($ [])	By recursive step in definition of *length*.
4. **To Show** $1 + 0 > 0$	By basis step in definition of *length*.
5. T	By properties of integers
QED	Because we got to T only by rewriting **To Show** to equivalent statements, using well-defined proof techniques, and applying definitions.

Recursive Step

Consider an arbitrary: $l' \in L$, $n \in \mathbb{N}$, and we assume as the **induction hypothesis** that:

$$length($ $append($ $(l', 100$)) $> length(l')$$

Our goal is to show that $length($ $append($ ($(n, l'), 100$))) $> length($ (n, l')) is also true. We start by working with one side of the candidate inequality:

$$\begin{aligned}
LHS &= length($ $append($ ($(n, l'), 100$))) \\
&= length($ $(n, append($ $(l', 100)$))) \qquad \text{by the recursive definition of } append \\
&= 1 + length($ $append($ $(l', 100)$)) \qquad \text{by the recursive definition of } length \\
&> 1 + length(l') \qquad \text{by the induction hypothesis} \\
&= length((n, l')) \qquad \text{by the recursive definition of } length \\
&= RHS
\end{aligned}$$

Linked list example each length

Prove or disprove: $\forall n \in \mathbb{N}\ \exists l \in L\ (\ length(l) = n\ )$

Proof strategy proof by contradiction

New! Proof by Contradiction
To prove that a statement p is true, pick another statement r and once we show that $\neg p \to (r \wedge \neg r)$ then we can conclude that p is true.
Informally The statement we care about can't possibly be false, so it must be true.

Rational numbers definition

The **set of rational numbers**, $\mathbb{Q}$ is defined as

$$\left\{ \frac{p}{q} \mid p \in \mathbb{Z} \text{ and } q \in \mathbb{Z} \text{ and } q \neq 0 \right\} \quad \text{or, equivalently,} \quad \{x \in \mathbb{R} \mid \exists p \in \mathbb{Z} \exists q \in \mathbb{Z}^{+} (p = x \cdot q)\}$$

Extra practice: Use the definition of set equality to prove that the definitions above give the same set.

Proof by contradiction irrational

Goal: The square root of 2 is not a rational number. In other words: $\neg\exists x \in \mathbb{Q}(x^2 - 2 = 0)$

Attempted proof: The definition of the set of rational numbers is the collection of fractions p/q where p is an integer and q is a nonzero integer. Looking for a **witness** p and q, we can write the square root of 2 as the fraction $\sqrt{2}/1$, where 1 is a nonzero integer. Since the numerator is not in the domain, this witness is not allowed, and we have shown that the square root of 2 is not a fraction of integers (with nonzero denominator). Thus, the square root of 2 is not rational.

The problem in the above attempted proof is that _____

Lemma 1: For every two integers a and b, not both zero, with $gcd(\ (a, b)\ ) = 1$, it is not the case that both a is even and b is even.

Lemma 2: For every integer x, x is even if and only if x^2 is even.

Proof: Towards a proof by contradiction, we will define a statement r such that $\sqrt{2} \in \mathbb{Q} \to (r \wedge \neg r)$.

Assume that $\sqrt{2} \in \mathbb{Q}$. Namely, there are positive integers p, q such that

$$\sqrt{2} = \frac{p}{q}$$

Let $a = \frac{p}{gcd(\ (p,q)\ )}$, $b = \frac{q}{gcd(\ (p,q)\ )}$, then

$$\sqrt{2} = \frac{a}{b} \qquad \text{and} \qquad gcd(\ (a, b)\ ) = 1$$

By Lemma 1, a and b are not both even. We define r to be the statement "a is even and b is even", and we have proved $\neg r$.

Squaring both sides and clearing denominator: $2b^2 = a^2$.

By definition of even, since b^2 is an integer, a^2 is even.

By Lemma 2, this guarantees that a is even too. So, by definition of even, there is some integer (call it c), such that $a = 2c$.

Plugging into the equation:

$$2b^2 = a^2 = (2c)^2 = 4c^2$$

and dividing both sides by 2

$$b^2 = 2c^2$$

and since c^2 is an integer, b^2 is even. By Lemma 2, b is even too. Thus, a is even and b is even and we have proved r.

In other words, assuming that $\sqrt{2} \in \mathbb{Q}$ guarantees $r \wedge \neg r$, which is impossible, so $\sqrt{2} \notin \mathbb{Q}$. QED

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 $-n = 0$, $w = 7$:

Equivalence relation definition

A relation is an **equivalence relation** means it is reflexive, symmetric, and transitive.

Partial order definition

A relation is a **partial ordering** (or partial order) means it is reflexive, antisymmetric, and transitive.

Hasse diagram definition

For a partial ordering, its **Hasse diagram** is a graph representing the relationship between elements in the ordering. The nodes (vertices) of the graph are the elements of the domain of the binary relation. The edges do not have arrow heads. The directionality of the partial order is indicated by the arrangements of the nodes. The nodes are arranged so that nodes connected to nodes above them by edges indicate that the relation holds between the lower node and the higher node. Moreover, the diagram omits self-loops and omits edges that are guaranteed by transitivity.

Hasse diagram example

Draw the Hasse diagram of the partial order on the set $\{a, b, c, d, e, f, g\}$ defined as

$$\{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(g,g),\\ (a,c),(a,d),(d,g),(a,g),(b,f),(b,e),(e,g),(b,g)\}$$

Partition definition

A **partition** of a set A is a set of non-empty, disjoint subsets $A_1, A_2, \cdots, A_n$ such that

$$A = \bigcup_{i=1}^{n} A_i = \{x \mid \exists i(x \in A_i)\}$$

Equivalence class definition

An **equivalence class** of an element $a \in A$ with respect to an equivalence relation R on the set A is the set

$$\{s \in A \mid (a, s) \in R\}$$

We write $[a]_R$ for this set, which is the equivalence class of a with respect to R.

Congruence classes mod four

Recall: We say a is **congruent to** b **mod** n means $(a, b) \in R_{(\mathbf{mod}\ n)}$. A common notation is to write this as $a \equiv b (\mathbf{mod}\ n)$.

We can partition the set of integers using equivalence classes of $R_{(\mathbf{mod}\ 4)}$

$$
\begin{aligned}
[0]_{R_{(\mathbf{mod}\ 4)}} &= \\
[1]_{R_{(\mathbf{mod}\ 4)}} &= \\
[2]_{R_{(\mathbf{mod}\ 4)}} &= \\
[3]_{R_{(\mathbf{mod}\ 4)}} &= \\
[4]_{R_{(\mathbf{mod}\ 4)}} &= \\
[5]_{R_{(\mathbf{mod}\ 4)}} &= \\
[-1]_{R_{(\mathbf{mod}\ 4)}} &=
\end{aligned}
$$

$$\mathbb{Z} = [0]_{R_{(\mathbf{mod}\ 4)}} \cup [1]_{R_{(\mathbf{mod}\ 4)}} \cup [2]_{R_{(\mathbf{mod}\ 4)}} \cup [3]_{R_{(\mathbf{mod}\ 4)}}$$

Modular arithmetic motivation

Integers are useful because they can be used to encode other objects and have multiple representations. However, infinite sets are sometimes expensive to work with computationally. Reducing our attention to a *partition of the integers* based on congrunce mod n, where each part is represented by a (not too large) integer gives a useful compromise where many algebraic properties of the integers are preserved, and we also get the benefits of a finite domain. Moreover, modular arithmetic is well-suited to model any cyclic behavior.

Congruence mod n lemma

Lemma: For $a, b \in \mathbb{Z}$ and positive integer n, $(a, b) \in R_{(\mathbf{mod}\ n)}$ if and only if $n|a - b$.

Proof:

Modular arithmetic

Modular arithmetic:

Lemma: For $a, b, c, d \in \mathbb{Z}$ and positive integer n, if $a \equiv b$ (**mod** n) and $c \equiv d$ (**mod** n) then $a + c \equiv b + d$ (**mod** n) and $ac \equiv bd$ (**mod** n). **Informally**: can bring mod "inside" and do it first, for addition and for multiplication.

$(102 + 48)$ **mod** $10 =$ _____

$(7 \cdot 10)$ **mod** $5 =$ _____

(2^5) **mod** $3 =$ _____

Modular arithmetic cycling examples

Application: Cycling

How many minutes past the hour are we at? *Model with* +15 **mod** 60

Time:	12:00pm	12:15pm	12:30pm	12:45pm	1:00pm	1:15pm	1:30pm	1:45pm	2:00pm
"Minutes past":	0	15	30	45	0	15	30	45	0

Replace each English letter by a letter that's fifteen ahead of it in the alphabet *Model with* +15 **mod** 26

Original index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Original letter:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Shifted letter:	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Shifted index:	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Equivalence relations partitions

Recall:

A relation is an **equivalence relation** means it is reflexive, symmetric, and transitive.

An **equivalence class** of an element $a \in A$ with respect to an equivalence relation R on the set A is the set

$$\{s \in A \mid (a, s) \in R\}$$

We write $[a]_R$ for this set, which is the equivalence class of a with respect to R.

A **partition** of a set A is a set of non-empty, disjoint subsets $A_1, A_2, \cdots, A_n$ such that

$$A = \bigcup_{i=1}^{n} A_i = \{x \mid \exists i(x \in A_i)\}$$

Claim: For each $a \in U$, $[a]_E \neq \emptyset$.

Proof: Towards a _____ consider an arbitrary element a in U. We will work to show that $[a]_E \neq \emptyset$, namely that $\exists x \in [a]_E$. By definition of equivalence classes, we can rewrite this goal as

$$\exists x \in U \ (\ (a, x) \in E\ )$$

Towards a _____, consider $x = a$, an element of U by definition. By _____ of E, we know that $(a, a) \in E$ and thus the existential quantification has been proved.

Claim: For each $a \in U$, there is some $b \in U$ such that $a \in [b]_E$.

Towards a _____ consider an arbitrary element a in U. By definition of equivalence classes, we can rewrite the goal as

$$\exists b \in U \ (\ (b, a) \in E\ )$$

Towards a _____, consider $b = a$, an element of U by definition. By _____ of E, we know that $(a, a) \in E$ and thus the existential quantification has been proved.

Claim: For each $a, b \in U$, ($(a, b) \in E \ \rightarrow \ [a]_E = [b]_E$) and ($(a, b) \notin E \ \rightarrow \ [a]_E \cap [b]_E = \emptyset$)

Corollary: Given an equivalence relation E on set U, $\{[x]_E \mid x \in U\}$ is a partition of U.

Equivalence relations examples ratings

Recall that in a movie recommendation system, each user's ratings of movies is represented as a n-tuple (with the positive integer n being the number of movies in the database), and each component of the n-tuple is an element of the collection $\{-1, 0, 1\}$.

We call Rt_5 the set of all ratings 5-tuples.

Define $d : Rt_5 \times Rt_5 \to \mathbb{N}$ by

$$d(\ (\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ )\ ) = \sum_{i=1}^{5} |x_i - y_i|$$

Consider the following binary relations on Rt_5.

$$E_{proj} = \{(\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ ) \in Rt_5 \times Rt_5 \ \mid\ (x_1 = y_1) \wedge (x_2 = y_2) \wedge (x_3 = y_3)\}$$

Example ordered pair in E_{proj}:

Reflexive? Symmetric? Transitive? Antisymmetric?

$$E_{dist} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid\ d(\ (u, v)\ ) \leq 2\}$$

Example ordered pair in E_{dist}:

Reflexive? Symmetric? Transitive? Antisymmetric?

$$E_{circ} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid\ d(\ (\ (0, 0, 0, 0, 0)\ , u)\ ) = d(\ (\ (0, 0, 0, 0, 0), v\ )\ )\}$$

Example ordered pair in E_{circ}:

Reflexive? Symmetric? Transitive? Antisymmetric?

The partition of Rt_5 defined by ____ is

{ { $(-1,-1,-1,-1,-1), (-1,-1,-1,-1,0), (-1,-1,-1,-1,1), (-1,-1,-1,0,-1), (-1,-1,-1,0,0), (-1,-1,-1,0,1), (-1,-1,-1,1,-1), (-1,-1,-1,1,0), (-1,-1,-1,1,1)$ },
{ $(-1,-1,0,-1,-1), (-1,-1,0,-1,0), (-1,-1,0,-1,1), (-1,-1,0,0,-1), (-1,-1,0,0,0), (-1,-1,0,0,1), (-1,-1,0,1,-1), (-1,-1,0,1,0), (-1,-1,0,1,1)$ },
{ $(-1,-1,1,-1,-1), (-1,-1,1,-1,0), (-1,-1,1,-1,1), (-1,-1,1,0,-1), (-1,-1,1,0,0), (-1,-1,1,0,1), (-1,-1,1,1,-1), (-1,-1,1,1,0), (-1,-1,1,1,1)$ },
{ $(-1,0,-1,-1,-1), (-1,0,-1,-1,0), (-1,0,-1,-1,1), (-1,0,-1,0,-1), (-1,0,-1,0,0), (-1,0,-1,0,1), (-1,0,-1,1,-1), (-1,0,-1,1,0), (-1,0,-1,1,1)$ },
{ $(-1,0,0,-1,-1), (-1,0,0,-1,0), (-1,0,0,-1,1), (-1,0,0,0,-1), (-1,0,0,0,0), (-1,0,0,0,1), (-1,0,0,1,-1), (-1,0,0,1,0), (-1,0,0,1,1)$ },
{ $(-1,0,1,-1,-1), (-1,0,1,-1,0), (-1,0,1,-1,1), (-1,0,1,0,-1), (-1,0,1,0,0), (-1,0,1,0,1), (-1,0,1,1,-1), (-1,0,1,1,0), (-1,0,1,1,1)$ },
{ $(-1,1,-1,-1,-1), (-1,1,-1,-1,0), (-1,1,-1,-1,1), (-1,1,-1,0,-1), (-1,1,-1,0,0), (-1,1,-1,0,1), (-1,1,-1,1,-1), (-1,1,-1,1,0), (-1,1,-1,1,1)$ },
{ $(-1,1,0,-1,-1), (-1,1,0,-1,0), (-1,1,0,-1,1), (-1,1,0,0,-1), (-1,1,0,0,0), (-1,1,0,0,1), (-1,1,0,1,-1), (-1,1,0,1,0), (-1,1,0,1,-1)$ },
{ $(-1,1,1,-1,-1), (-1,1,1,-1,0), (-1,1,1,-1,1), (-1,1,1,0,-1), (-1,1,1,0,0), (-1,1,1,0,1), (-1,1,1,1,-1), (-1,1,1,1,0), (-1,1,1,1,1)$ },
{ $(0,-1,-1,-1,-1), (0,-1,-1,-1,0), (0,-1,-1,-1,1), (0,-1,-1,0,-1), (0,-1,-1,0,0), (0,-1,-1,0,1), (0,-1,-1,1,-1), (0,-1,-1,1,0), (0,-1,-1,1,1)$ },
{ $(0,-1,0,-1,-1), (0,-1,0,-1,0), (0,-1,0,-1,1), (0,-1,0,0,-1), (0,-1,0,0,0), (0,-1,0,0,1), (0,-1,0,1,-1), (0,-1,0,1,0), (0,-1,0,1,1)$ },
{ $(0,-1,1,-1,-1), (0,-1,1,-1,0), (0,-1,1,-1,1), (0,-1,1,0,-1), (0,-1,1,0,0), (0,-1,1,0,1), (0,-1,1,1,-1), (0,-1,1,1,0), (0,-1,1,1,1)$ },
{ $(0,0,-1,-1,-1), (0,0,-1,-1,0), (0,0,-1,-1,1), (0,0,-1,0,-1), (0,0,-1,0,0), (0,0,-1,0,1), (0,0,-1,1,-1), (0,0,-1,1,0), (0,0,-1,1,1)$ },
{ $(0,0,0,-1,-1), (0,0,0,-1,0), (0,0,0,-1,1), (0,0,0,0,-1), (0,0,0,0,0), (0,0,0,0,1), (0,0,0,1,-1), (0,0,0,1,0), (0,0,0,1,1)$ },
{ $(0,0,1,-1,-1), (0,0,1,-1,0), (0,0,1,-1,1), (0,0,1,0,-1), (0,0,1,0,0), (0,0,1,0,1), (0,0,1,1,-1), (0,0,1,1,0), (0,0,1,1,1)$ },
{ $(0,1,-1,-1,-1), (0,1,-1,-1,0), (0,1,-1,-1,1), (0,1,-1,0,-1), (0,1,-1,0,0), (0,1,-1,0,1), (0,1,-1,1,-1), (0,1,-1,1,0), (0,1,-1,1,1)$ },
{ $(0,1,0,-1,-1), (0,1,0,-1,0), (0,1,0,-1,1), (0,1,0,0,-1), (0,1,0,0,0), (0,1,0,0,1), (0,1,0,1,-1), (0,1,0,1,0), (0,1,0,1,-1)$ },
{ $(0,1,1,-1,-1), (0,1,1,-1,0), (0,1,1,-1,1), (0,1,1,0,-1), (0,1,1,0,0), (0,1,1,0,1), (0,1,1,1,-1), (0,1,1,1,0), (0,1,1,1,1)$ },
{ $(1,-1,-1,-1,-1), (1,-1,-1,-1,0), (1,-1,-1,-1,1), (1,-1,-1,0,-1), (1,-1,-1,0,0), (1,-1,-1,0,1), (1,-1,-1,1,-1), (1,-1,-1,1,0), (1,-1,-1,1,1)$ },
{ $(1,-1,0,-1,-1), (1,-1,0,-1,0), (1,-1,0,-1,1), (1,-1,0,0,-1), (1,-1,0,0,0), (1,-1,0,0,1), (1,-1,0,1,-1), (1,-1,0,1,0), (1,-1,0,1,1)$ },
{ $(1,-1,1,-1,-1), (1,-1,1,-1,0), (1,-1,1,-1,1), (1,-1,1,0,-1), (1,-1,1,0,0), (1,-1,1,0,1), (1,-1,1,1,-1), (1,-1,1,1,0), (1,-1,1,1,1)$ },
{ $(1,0,-1,-1,-1), (1,0,-1,-1,0), (1,0,-1,-1,1), (1,0,-1,0,-1), (1,0,-1,0,0), (1,0,-1,0,1), (1,0,-1,1,-1), (1,0,-1,1,0), (1,0,-1,1,1)$ },
{ $(1,0,0,-1,-1), (1,0,0,-1,0), (1,0,0,-1,1), (1,0,0,0,-1), (1,0,0,0,0), (1,0,0,0,1), (1,0,0,1,-1), (1,0,0,1,0), (1,0,0,1,1)$ },
{ $(1,0,1,-1,-1), (1,0,1,-1,0), (1,0,1,-1,1), (1,0,1,0,-1), (1,0,1,0,0), (1,0,1,0,1), (1,0,1,1,-1), (1,0,1,1,0), (1,0,1,1,1)$ },
{ $(1,1,-1,-1,-1), (1,1,-1,-1,0), (1,1,-1,-1,1), (1,1,-1,0,-1), (1,1,-1,0,0), (1,1,-1,0,1), (1,1,-1,1,-1), (1,1,-1,1,0), (1,1,-1,1,1)$ },
{ $(1,1,0,-1,-1), (1,1,0,-1,0), (1,1,0,-1,1), (1,1,0,0,-1), (1,1,0,0,0), (1,1,0,0,1), (1,1,0,1,-1), (1,1,0,1,0), (1,1,0,1,-1)$ },
{ $(1,1,1,-1,-1), (1,1,1,-1,0), (1,1,1,-1,1), (1,1,1,0,-1), (1,1,1,0,0), (1,1,1,0,1), (1,1,1,1,-1), (1,1,1,1,0), (1,1,1,1,1)$ }
}

The partition of Rt_5 defined by $E =$ ____ is

$$\{ \; [\,(0,0,0,0,0)\,]_E \; , [\,(0,0,0,0,1)\,]_E \; , [\,(0,0,0,1,1)\,]_E \; , [\,(0,0,1,1,1)\,]_E \; , [\,(0,1,1,1,1)\,]_E \; , [\,(1,1,1,1,1)\,]_E \; \}$$

How many elements are in each part of the partition?

Netflix clustering scenario

Scenario: Good morning! You're a user experience engineer at Netflix. A product goal is to design customized home pages for groups of users who have similar interests. Your manager tasks you with designing an algorithm for producing a clustering of users based on their movie interests, so that customized homepages can be engineered for each group.

Your idea: equivalence relations!

$$E_{id} = \{(\ (x_1, x_2, x_3, x_4, x_5), (x_1, x_2, x_3, x_4, x_5)\ ) \mid (x_1, x_2, x_3, x_4, x_5) \in Rt_5\}$$

Describe how each homepage should be designed . . .

$$E_{proj} = \{(\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ ) \in Rt_5 \times Rt_5 \ \mid \ (x_1 = y_1) \wedge (x_2 = y_2) \wedge (x_3 = y_3)\}$$

Describe how each homepage should be designed . . .

$$E_{circ} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid \ d(\ (\ (0, 0, 0, 0, 0)\ , u)\ ) = d(\ (\ (0, 0, 0, 0, 0), v\ )\ )\}$$

Describe how each homepage should be designed . . .

Set construction final review

The bases of RNA strands are elements of the set $B = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. The set of RNA strands S is defined (recursively) by:

Basis Step:	$\texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

Each of the sets below is described using set builder notation. Rewrite them using the roster method.

- $\{s \in S \mid$ the leftmost base in s is the same as the rightmost base in s and s has length $3\}$

- $\{s \in S \mid$ there are twice as many $\texttt{A}$s as $\texttt{C}$s in s and s has length $1\}$

Certain sequences of bases serve important biological functions in translating RNA to proteins. The following recursive definition gives a special set of RNA strands: The set of RNA strands $\hat{S}$ is defined (recursively) by

Basis step:	$\texttt{AUG} \in \hat{S}$
Recursive step:	If $s \in \hat{S}$ and $x \in R$, then $sx \in \hat{S}$

where $R = \{\texttt{UUU}, \texttt{CUC}, \texttt{AUC}, \texttt{AUG}, \texttt{GUU}, \texttt{CCU}, \texttt{GCU}, \texttt{UGG}, \texttt{GGA}\}$.

Each of the sets below is described using set builder notation. Rewrite them using the roster method.

- $\{s \in \hat{S} \mid s$ has length less than or equal to $5\}$

- $\{s \in S \mid$ there are twice as many $\texttt{C}$s as $\texttt{A}$s in s and s has length $6\}$

Set operations final review

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$. Consider the statement

$$\forall A \in W \ \forall B \in W \ \forall C \in W \ ((A \cap B = A \cap C) \to (B = C))$$

Translate the statement to English. Negate the statement and translate this negation to English. Decide whether the original statement or its negation is true and justify your decision.

Function properties final review

The set of linked lists of natural numbers L is defined by

Basis step:	$[] \in L$
Recursive step:	If $l \in L$ and $n \in \mathbb{N}$, then $(n, l) \in L$

The function $length : L \to \mathbb{N}$ that computes the length of a list is

Basis step:	$length([]) = 0$
Recursive step:	If $l \in L$ and $n \in \mathbb{N}$, then $length((n, l)) = 1 + length(l)$

Prove or disprove: the function *length* is onto.

Prove or disprove: the function *length* is one-to-one.

Cardinality final review

Suppose A and B are sets and $A \subseteq B$:

True or False? If A is infinite then B is finite.

True or False? If A is countable then B is countable.

True or False? If B is infinite then A is finite.

True or False? If B is uncountable then A is countable.

Modular arithmetic final review

Compute the last digit of

$$(42)^{2024}$$

Extra Describe the pattern that helps you perform this computation and prove it using mathematical induction.