Set Theory

Dr. Isaac Griffith

Idaho State University

Set Theory

- **Set Theory:** One of the fundamental branches of mathematics
- Has a deep connection to Logic, as we'll see
- The notation and terminology is quite useful for describing both *data types* and *algorithms*

Outline

The lecture if structured as follows:

- Set Notation
- Set Operations
- Finite Sets with Equality
- Set Laws (Identities)
- Proofs with Sets
- Advanced Concepts

Set Notation

CS 1187

Set Notation

- A **set** is a collection of objects called *members* or *elements*
- We can describe a set simply by listing all of its elements between branches $\{\ldots\}$, this is called the **roster method**
  - **Example:**

$$
\begin{aligned}
A &= \{\text{dog}, \text{cat}, \text{horse}\} \\
C &= \{0, 1, 2, 3, 4\} \\
E &= \{\} \\
N &= \{0, 1, 2, 3, \ldots\}
\end{aligned}
$$

- An element may only occur once in a given set
  - Thus, we can test membership using the membership operator $\in$ which returns `True` or `False`
  - Similarly, we can test lack of membership with the not a member operator $\notin$
  - **Examples:**

$$
\begin{aligned}
\text{dog} \in A &= \textit{True} \\
\text{dog} \notin A &= \textit{False}
\end{aligned}
$$

Set Notation

- Sets can have any number of elements
  - A has 3 elements
  - C has 5 elements
  - E has 0 elements
  - N has infinite elements
- The empty set, $\{\}$, is special and is denoted as $\varnothing$
- Sets tend to be denoted using a capital letter or as block font (i.e., $\mathbb{S}$)

Some Important Sets

- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, the set of natural numbers
- $\mathbb{Z} = \{\ldots, -2, -1, 0, -1, -2, \ldots\}$, the set of all integers
- $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$, the set of all positive integers
- $\mathbb{Q} = \{p/q \mid p \in \mathbb{Z}, q \in \mathbb{Z}, q \neq 0\}$, the set of all rational numbers
- $\mathbb{R}$ = the set of real numbers
- $\mathbb{Q}^+\{x \in \mathbb{R}^+ \mid x = \frac{p}{q}$, for some positive integers $p, q\}$, the set of positive rational numbers
- $\mathbb{R}^+$, the set of positive real numbers

Set Notation

- Another standard method of set notation is the **set comprehension** or **set builder** notation
  - In its simplest form, it is written as:

$$\{x \mid p\ x\}$$

    Where:
    - *px* is a predicate, which defines those items to be included
    - **Read as:** "The set of *x* such that *px*"
  - **General Form:**

$$\{f\ x \mid p\ x\}$$

    - Set contains values of the results of applying *f* to those values which satisfy $p\ x$

- **Example:**
  - Set of even numbers $\{x \mid x \in \mathbb{N} \land even\ x\}$

Set Notation

Idaho State University | Computer Science

- In calculus, we study sets called **intervals**, which are sets of real numbers between two numbers *a* and *b*, and may include/exclude *a* and *b*.

- If $a, b \in \mathbb{R} \wedge a \leq b$, we denote these intervals by:

$$
\begin{array}{lcll}
[a, b] & = & \{x \mid a \leq x \leq b\} & \Rightarrow \text{ closed interval} \\
[a, b) & = & \{x \mid a \leq x < b\} & \\
(a, b] & = & \{x \mid a < x \leq b\} & \\
(a, b) & = & \{x \mid z < x < b\} & \Rightarrow \text{ open interval}
\end{array}
$$

- In CS, the concept of a *data type* or *type* is based on the set concept
  - **Data type** or **type** is the name of a set, together with a set of operations that can be performed on objects of that set.
  - **Example:** Boolean = {True, False} together with the operators $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$

ROAR

Venn Diagrams

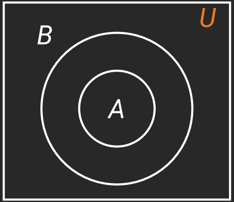
- A graphic notation for sets named after John Venn who introduced these diagrams in 1881
- Starts with a rectangle labeled U, which represents the **universal set** that contains all objects under consideration
- Inside the rectangle we use shapes, typically circles or ellipses, to represent sets
- Inside sets, we can use points to show specific members

Subsets

- **Subset:** the set A is a *subset* of B, and B is a *superset* of A, iff every element of A is also an element of B

$$
\begin{array}{lcl}
A \subseteq B & - & \text{A is a subset of B} \\
B \supseteq A & - & \text{B is a superset of A} \\
A \subseteq B & \equiv & B \supseteq A \\
A \subseteq B & \leftrightarrow & \forall\, x.\ (x \in A \rightarrow x \in B)
\end{array}
$$

- To show that A is a subset of B
  - Show that if x belongs to A, then x also belongs to B
- To show that A is not a subset of B
  - To show $A \not\subseteq B$, find a single $x \in A$ such that $x \notin B$
- For every set S:
  - $\varnothing \subseteq S$
  - $S \subseteq S$



Set Equality

Idaho State University | Computer Science

- **Set Equality:** To show that two sets A and B are equal, show that $A \subseteq B$ and $B \subseteq A$

- If we have two sets A and B, where A is a subset of B but where $A \neq B$, then we call A a **proper subset** of B, denoted as:

$$A \subset B$$

For $A \subset B$ to be true, then

$$\forall\, x.\ (x \in A \rightarrow x \in B) \land \exists\, x.\ (x \in B \land x \notin A)$$

- **Note:** Sets may also contain other sets as members
  $A = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$ and $B = \{x \mid x \subseteq \{a, b\}\}$
  $A = B$

ROAR

Cardinality

- **Cardinality:** Let S be a set. If there are exactly n distinct elements in S where n is a nonnegative integer, we say S is a **finite set** and that n is the **cardinality** of S.
  - We denote the cardinality of a set S as: $|S|$

- **Example:**
  - The set, A, of odd positive integers < 10. $|A| = 5$
  - The set, S, of letters in the English alphabet. $|S| = 26$
  - The empty set. $|\varnothing| = 0$

- A set is said to be **infinite** if it is not finite.
  - $\mathbb{Z}^+$ is infinite

Power Sets

Idaho State University | Computer Science

- **Powerset:** Let A be a set. The *powerset*, written $\mathcal{P}(A)$, is the set of all subsets of A:

$$\mathcal{P}(A) = \{s \mid s \subseteq A\}$$

- **Examples:**
  - $\mathcal{P}(\varnothing) = \{\varnothing\}$
  - $\mathcal{P}(\{a\}) = \{\varnothing, \{a\}\}$
  - $\mathcal{P}(\{a, b\}) = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$
  - $\mathcal{P}(\{a, b, c\}) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
- If $|A| = n$, then $|\mathcal{P}(A)| = 2^n$

ROAR

Cartesian Products

- Often order of elements is important, but sets are unordered, so we often need something else

- **Ordered n-tuple:** $(a_1, a_2, \ldots, a_n)$ is an ordered collection that has a_1 as its first element, a_2 as its second element, . . ., and a_n as its n^{th}
  - we say two ordered n-tuples are equal iff each corresponding pair is equal
  - Ordered 2-tuples are called **ordered pairs**
    - The ordered pairs (a, b) and (c, d) are equal iff $a = b$ and $c = d$

- **Cartesian Product:** Let A and B be sets. The *cartesian product* of A and B, denoted $A \times B$, is the set of all ordered pairs (a, b), where $a \in A$ and $b \in B$

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

- **Example:** $A = \{1, 2\}$, $B = \{a, b, c\}$
  $A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\} \Rightarrow$ The *zip* function comes to mind

Cartesian Products

- The **cartesian product** of the sets $A_1, A_2, \ldots, A_n$, denoted by $A_1 \times A_2 \times \ldots \times A_n$, is the set of ordered n-tuples $(a_1, a_2, \ldots, a_n)$, where a_i belongs to A_i for $i = 1, 2, \ldots, n$

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

- **Example:** $A = \{0, 1\}$, $B = \{1, 2\}$, $C = \{0, 1, 2\}$

$$\begin{aligned} A \times B \times C \quad = \quad & \{(0,1,0),(0,1,1),(0,1,2),(0,2,1),(0,2,1),(0,2,2), \\ & (1,1,0),(1,1,1),(1,1,2),(1,2,0),(1,2,1),(1,2,2),\} \end{aligned}$$

- A subset R of the Cartesian product $A \times B$ is called a **relation** from the set A to the set B, where the elements of R are ordered pairs, with the first element belonging to A and the second to B.

Sets and Quantifiers

- Often we restrict the domain of a quantified statement

  $\forall\ x \in S(P(x))$, which is shorthand for $\forall\ x.(x \in S \rightarrow P(x))$

  "Universal quantification of $P(x)$ over all elements in S"

  $\exists\ x \in S(P(x))$, short hand for $\exists\ x.(x \in S \land P(x))$

  "Existential quantification of $P(x)$ over all elements in S"

- **Truth Set:** Given a predicate P, and a domain D, the *truth set* of P is the set of elements $x \in D$ for which $P(x)$ is true.

  That is the Truth Set of $P(x) = \{x \in D \mid P(x)\}$

  - $\forall\ x.P(x)$ is true over the domain U iff the truth set of P is U
  - $\exists\ x.P(x)$ is true over the domain U iff the truth set of P is not empty.

Set Operations

CS 1187

ROAR

Union, Intersection, and Difference

- **Union ($\cup$):** The *union* of two sets A and B, written $A \cup B$, is the set that contains all elements that are in either A or B, or both

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

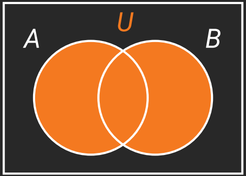
- **Intersection ($\cap$):** The *intersection* of two sets A and B, written $A \cap B$, is the set that contains all elements that are in *both* A and B.

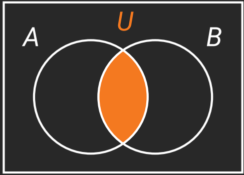
$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

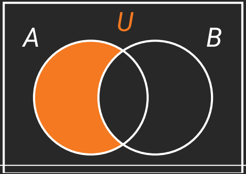
- **Difference ($-$):** The *difference* of two sets A and B, written $A - B$, is the set of all elements that are in A but not in B

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

- $|A \cup B| = |A| + |B| - |A \cap B|$
  - *Note:* $|A| + |B|$ counts elements twice hence the need to subtract $|A \cap B|$







Union, Intersection, and Difference

- **Example:** $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{4, 5, 6\}$

$$\begin{aligned} A \cup B &= \{1, 2, 3, 4, 5\} \\ A \cap B &= \{3\} \\ A - B &= \{1, 2\} \end{aligned} \qquad \begin{aligned} A \cup C &= \{1, 2, 3, 4, 5, 6\} \\ A \cap C &= \varnothing \\ A - C &= \{1, 2, 3\} \end{aligned}$$

- **Example:** Let

$$\begin{aligned} \mathbb{Z} &= \{\ldots, -2, -1, 0, 1, 2, \ldots\} &&\Rightarrow \text{Set of integers} \\ \mathbb{N} &= \{0, 1, 2, \ldots\} &&\Rightarrow \text{Set of natural numbers} \\ \mathbb{H} &= \{-2^{15}, \ldots, -2, -1, 0, 1, 2, \ldots, 2^{15} - 1\} &&\Rightarrow \text{Integers representable using 16-bit word} \\ \mathbb{W} &= \{-2^{31}, \ldots, -2, -1, 0, 1, 2, \ldots, 2^{31} - 1\} &&\Rightarrow \text{Integers representable using 32-bit word} \end{aligned}$$

$\mathbb{I} - \mathbb{W}$ is the set of integers not representable in a word

Symmetric Difference

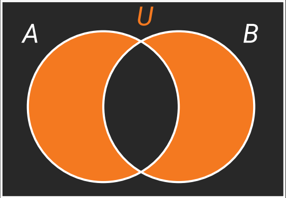
- **Symmetric Difference:** The symmetric difference of two sets A and B, written $A \oplus B$ is the set containing those elements in either A or B, but not is both A and B

$$A \oplus B = \{x | (x \in A \land x \notin B) \lor (x \notin A \land x \in B)\}$$

Identities:

- $A \oplus A = \varnothing$
- $A \oplus U = \overline{A}$
- $A \oplus \varnothing = A$
- $A \oplus \overline{A} = U$
- $A \oplus B = B \oplus A$
- $(A \oplus B) \oplus B = A$
- $A \oplus B = (A \cup B) - (A \cap B)$
- $A \oplus B = (A - B) \cup (B - A)$

Venn Diagram:

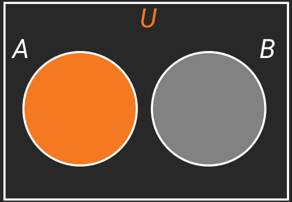


ROAR

Disjoint Sets

- **Disjoint Sets:** For any two sets A and B, if $A \cap B = \varnothing$ then A and B are *disjoint sets*
- **Example:** $A = \{1, 3, 5, 7, 9\}$ and $B = \{2, 4, 6, 8, 10\}$

  $A \cap B = \varnothing$, thus A and B are disjoint
- **Venn Diagram:**



Set Complement

- **Complement:** Let U be the *universal set* and A be a set. The *complement* of A, written A' or $\overline{A}$, is the set $U - A$

$$\overline{A} = \{x \in U | x \notin A\}$$

- **Note:** $A - B = A \cup \overline{B}$

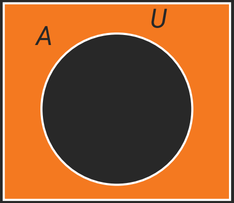
Example:

$$\begin{aligned} D &= \{0, 1, 2, \ldots, 9\} \\ L &= \{a, b, \ldots, z\} \\ U &= L \cup D \\ \overline{D} &= L \\ \overline{L} &= D \end{aligned}$$

Example: $U = \{1, 2, 3, 4, 5\}$

$\overline{\{1, 2\}} = \{3, 4, 5\}$

Venn Diagram:



Finite Sets with Equality

CS 1187

Finite Sets with Equality

- **Finite Set with Equality:** a set with a finite number of elements and for which we have a function to test the equality of two elements from the universe
  - These are important in computation as they can ensure computation over finite sets may terminate

- We can represent sets using a list, but there are important differences between lists and sets:
  1. Lists can have duplicate items
  2. These is a fixed order to the elements of a list
  3. All elements in a list must be of the same type

Finite Sets with Equality

- To perform any useful computations involving sets, we must be able to determine if an element is in the set
  - This requires the ability to test if two values are the same (using ==)
    - Simple for elementary types, but difficult for compound types and functions
  - In Haskell, we can express the fact that it is possible to compare elements for equality, by using type restrictions:

```haskell
Eq a => [a] -- as we use a list to represent a set
```

- Additionally, we want the ability to print the set, so we add the following additional restriction:

```haskell
(Eq, Show) a => [a]
```

Finite Sets with Equality

- Using lists to represent sets requires some care, specifically because
  - There is a possibility of duplicates
  - There is an ordering of the elements

- To ensure we do not allow duplicates, we need a means by which we can represent sets using a **normal form**, which contains no duplicates
  - All operations will then ensure their results are in normal form

- However, because order matters in lists, but not in sets the list `[3, 2, 1]` is different from the list `[1, 2, 3]`, but as sets these are the same.
  - Thus, to alleviate this issue, we will ensure the sets are similarly ordered

Finite Sets With Equality

- An ordered list, requires that the contained elements are comparable using the (<, =, >) operators
  - This requires we add another type constraint:

```
Ord a => [a]
```

  - This says that there must be an ordering on the element type a, which can be used to determine the relations $<, \leq, =, \neq, >, \geq$

- The methods to define lists can also be used to define sets
  - Enumerated set: defined by simply listing the elements (roster method)
  - **Sequence:** when enumeration is too tedious: $\{0, 1, 2, \ldots, 1000\} \Rightarrow$ `[0,1..1000]`
  - **Set Comprehension:** $\{x^2 \mid x \in \{0, 1, \ldots, n\}\} \Rightarrow$ `[x^2 | x <- [0..n]]`

Computing with Sets

- We can define a *set type* as:

```
type Set a = [a]
```

- The universe of discourse

```
universe -- global var
```

Operations:

- The following are functions we can use on finite sets with equality. Each of these functions always returns a set in *normal form*

```
normalForm :: (Eq a, Show a) => [a] -> Bool -- checks if in normal form
normalizeSet :: (Eq a, Show a) => [a] -> Bool -- normalizes a set
```

Computing with Sets

- Symbolic operators for set operations

$$
\begin{aligned}
\texttt{A+++B} &= A \cup B \\
\texttt{A***B} &= A \cap B \\
\texttt{A\~{}\~{}\~{}B} &= A - B
\end{aligned}
$$

```
(+++) :: (Eq a, Show a) => Set a -> Set a -> Set a
(***) :: (Eq a, Show a) => Set a -> Set a -> Set a
(~~~) :: (Eq a, Show a) => Set a -> Set a -> Set a
```

- **Other Operations**

```
subset, properSubset :: (Eq a, Show a) => Set a -> Set a -> Bool
setEq :: (Eq a, Show a) => Set a -> Set a -> Bool
complement S = universe ~~~ S
powerset :: (Eq a, Show a) => Set a -> Set (Set a)
crossproduct :: (Eq a, Show a, Eq b, Show b) => Set a -> Set b -> Set (a, b)
```

Other Representations

- There are many ways to represent sets using computers
  - For example, ti may be tempting to store a set in an ad hoc unordered way
  - However, this is inefficient due to the large number of searches required to perform the various basic set operations

- Another way is to use an arbitrary ordering of elements on the universal set

- This requires a few assumptions
  1. The universe is finite
  2. The $|U| <$ memory size of the computer

Other Representations

Idaho State University | Computer Science

- We first specify the arbitrary ordering (i.e., ascending in value)
  - This creates the sequence in $U : a_1, a_2, \ldots, a_n$
- We then represent a subset A of U with a length n bit string
  - where the *ith* bit is 1 if a_i belongs to A and is 0 otherwise
- **Example:**

$$
\begin{aligned}
U &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\
O &= \text{the odd numbers in U} = \{1, 3, 5, 7, 9\} \\
E &= \text{the even numbers in U} = \{2, 4, 6, 8, 10\}
\end{aligned}
$$

O is represented as: $10\ 1010\ 1010$

E is represented as: $01\ 0101\ 0101$

Other Representations

Operations:

- **Complement:** of a set S is performed by taking the bitwise `NOT` of each bit in the bit string
- **Union:** of sets S and T is performed by taking the bitwise `OR` of S and T's' bit string representations
- **Intersection:** of sets S and T is performed by taking the bitwise `AND` of S and T's bit string representations

Example:

$E = \overline{O} = \texttt{NOT}\ 1010101010 = 0101010101$

$E \cup O$

```
    0101010101
    1010101010
    ----------
OR  1111111111
```

$E \cap O$

```
     0101010101
     1010101010
     ----------
AND  0000000000
```

Sets Laws

CS 1187

ROAR

Set Laws

Idaho State University | Computer Science

- Often in carrying out set operations or in describing the properties of algorithms, we often need to use several operators together

- Fortunately, set operations satisfy a number of basic laws that simplify their use
  - The first of which is:

    $\subseteq$ **Transitivity:** Let A, B, and C be sets. If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$

Proof: Let x be any element of the universe

1.	$A \subseteq B$	{Premise}
2.	$x \in A \to x \in B$	{Def. $\subseteq$}
3.	$B \subseteq C$	{Premise}
4.	$x \in B \to x \in C$	{Def. $\subseteq$}
5.	$x \in A \to x \in C$	{2, 4, chain rule}
6.	$\forall x. (x \in A \to x \in C)$	{$\forall$ introduction}
7.	$A \subseteq C$	{Def. $\subseteq$}

Basic Laws

Laws: For any set A in universe U

Identity Laws

$$
\begin{aligned}
A \cap U &= A \\
A \cup \varnothing &= A
\end{aligned}
$$

Domination Laws

$$
\begin{aligned}
A \cup U &= U \\
A \cap \varnothing &= \varnothing
\end{aligned}
$$

Idempotent Laws

$$
\begin{aligned}
A \cup A &= A \\
A \cap A &= \varnothing
\end{aligned}
$$

Double Complementation Law

$$
\overline{(\overline{A})} = A
$$

Commutative and Associative

Laws: For all sets A, B, and C

Associative Laws

$$\begin{aligned} A \cup (B \cup C) &= (A \cup B) \cup C \\ A \cap (B \cap C) &= (A \cap B) \cap C \end{aligned}$$

Set Difference

$$A - B = A \cap \overline{B}$$

Commutative Laws

$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$

Example: Prove $A \cap B = B \cap A$

1.	$x \in A \cap B$	{Premise}
2.	$x \in A \wedge x \in B$	{Def. $\cap$}
3.	$x \in B \wedge x \in A$	{Comm. $\wedge$}
4.	$x \in B \cap A$	{Def. $\cap$}
5.	$\forall x \in U . x \in A \cap B \leftrightarrow x \in B \cap A$	{{$\forall I$}}
6.	$A \cap B = B \cap A$	{Def. set eq.}

Distribution and DeMorgan's

Idaho State University | Computer Science

Laws: For any sets *A*, *B*, *C* and universe *U*

Distributive Laws:

$$
\begin{aligned}
A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \\
A \cup (B \cap C) &= (A \cup B) \cap (A \cup C)
\end{aligned}
$$

DeMorgan's Laws

$$
\begin{aligned}
\overline{A \cap B} &= \overline{A} \cup \overline{B} \\
\overline{A \cup B} &= \overline{A} \cap \overline{B}
\end{aligned}
$$

Absorption Laws

$$
\begin{aligned}
A \cup (A \cap B) &= A \\
A \cap (A \cup B) &= A
\end{aligned}
$$

Complement Laws

$$
\begin{aligned}
A \cup \overline{A} &= U \\
A \cap \overline{A} &= \varnothing
\end{aligned}
$$

Proofs with Sets

CS 1187

ROAR

Using Membership Tables

- We can prove set identities using set membership tables
  - Here, we consider each combination of atomic sets (original sets used to produce the sets on each side of an identity) that an element can belong to.
  - We then verify that elements on the same combinations belong to both the sets in the identity
  - To indicate an element is in a set we us a 1, otherwise a 0
- **Example:** Show $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

A	B	C	$B \cup C$	$A \cap (B \cup C)$	$A \cap B$	$A \cap C$	$(A \cap B) \cup (A \cap C)$
1	1	1	1	**1**	1	1	**1**
1	1	0	1	**1**	1	0	**1**
1	0	1	1	**1**	0	1	**1**
1	0	0	0	**0**	0	0	**0**
0	1	1	1	**0**	0	0	**0**
0	1	0	1	**0**	0	0	**0**
0	0	1	1	**0**	0	0	**0**
0	0	0	0	**0**	0	0	**0**

Using Identities

- Once we have proved set identities (laws), we can use them to prove new identities through equational reasoning

- **Example:** Let A, B, and C be sets

  Show that $\overline{A \cup (B \cap C)} = (\overline{C} \cup \overline{B}) \cap \overline{A}$

$$
\begin{aligned}
&\overline{A \cup (B \cap C)} \\
&\quad = \overline{A} \cap \overline{(B \cap C)} && \{\text{DeMorgan's law}\} \\
&\quad = \overline{A} \cap (\overline{B} \cup \overline{C}) && \{\text{DeMorgan's law}\} \\
&\quad = (\overline{B} \cup \overline{C}) \cap \overline{A} && \{\text{Commutative law}\} \\
&\quad = (\overline{C} \cup \overline{B}) \cap \overline{A} && \{\text{Commutative law}\}
\end{aligned}
$$

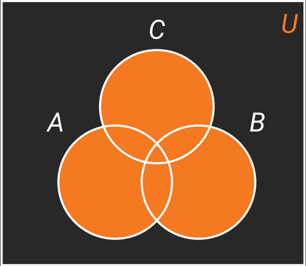
Advanced Concepts

CS 1187

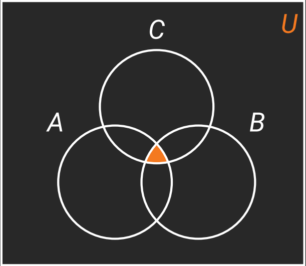
ROAR

Generalized Union and Intersection

- We can calculate the union of several sets using the $\cup$ operator.
  - Because it is associative, statements such as $A \cup B \cup C$ are unambiguous
- Similarly we can also find the intersection of multiple sets using a statement such as $A \cap B \cap C \cap D$
- However, attempting to visualize the union or intersection of 4+ sets starts to get difficult.



$A \cup B \cup C$



$A \cap B \cap C$

Generalized Union and Intersection

- Sometimes it becomes necessary to compute the union or intersection of a collection of sets.
- The corresponding operations which handle this are often called *big union* and *big intersection*
- Let C be a non-empty collection of subsets of U. Let I be a non-empty set, and for each $i \in I$ let $A_i \subseteq C$, then
  $\bigcup_{i \in I} A_i = \{x \mid \exists\, i \in I.\ x \in A_i\}$ $\quad$ $\bigcap_{i \in I} A_i = \{x \mid \forall\, i \in I.\ x \in A_i\}$
- We could also consider writing these same definitions as follows:
  $\bigcup_{A \in C} A = \{x \mid \exists\, A \in C.\ x \in A\}$ $\quad$ $\bigcap_{A \in C} A = \{x \mid \forall\, A \in C.\ x \in A\}$
- In either case
  $A_1 \cup A_2 \cup \ldots \cup A_n = \bigcup_{i=1}^{n} A_i$ $\quad$ $A_1 \cap A_2 \cap \ldots \cap A_n = \bigcap_{i=1}^{n} A_i$

Multisets

- **Multiset:** An unordered collection of elements, where an element can occur as a member more than once
  - *Notation:* $\{m_1 \cdot a_1, m_2 \cdot a_2, \ldots, m_r \cdot a_r\}$ denotes the multiset with element a_1 occurring m_1 times, element a_2 occurring m_2 times and so on.
    - The numbers m_i, $i = 1, 2, \ldots, r$ are called **multiplicities** of the elements a_i, $i = 1, 2, \ldots, r$
    - Elements not in the multiset have a multiplicity of 0

Cardinality: The cardinality of a multiset is defined as the sum of the multiplicities of its elements

- **Examples:**

$$\begin{aligned} P &= \{4 \cdot a, 1 \cdot b, 3 \cdot c\} \\ |P| &= 4 + 1 + 3 = 8 \end{aligned}$$

ROAR

Multiset Operations

- **Union:** the *union* of multisets P and Q is the multiset in which the multiplicity of an element is the maximum of its multiplicities in P and Q. Written as $P \cup Q$

- **Intersection:** the *intersection* of multisets P and Q is the multiset in which the multiplicity of an element is the minimum of its multiplicities in P and Q. Written as $P \cap Q$

- **Difference:** the *difference* of multisets P and Q is the multiset in which the multiplicity of an element is the multiplicity of the element in P less its multiplicity in Q unless the difference is negative, in which case the multiplicity is 0. Written as $P - Q$

- **Sum:** the *sum* of multisets P and Q is the multiset in which the multiplicity of an element is the sum of the multiplicities in P and Q. Written as $P + Q$

ROAR

Multiset Operation Examples

- **Example:** $P = \{4 \cdot a, 1 \cdot b, 3 \cdot c\}$ and $Q = \{3 \cdot a, 4 \cdot b, 2 \cdot d\}$

$$\begin{aligned}
P \cup Q &= \{\max(4,3)a, \max(1,4)b, \max(3,0)c, \max(0,2)d\} \\
&= \{4 \cdot a, 4 \cdot b, 3 \cdot c, 2 \cdot d\}
\end{aligned}$$

$$\begin{aligned}
P \cap Q &= \{\min(4,3)a, \min(1,4)b, \min(3,0)c, \min(0,2)d\} \\
&= \{3 \cdot a, 1 \cdot b, 0 \cdot c, 0 \cdot d\}
\end{aligned}$$

$$\begin{aligned}
P - Q &= \{\max(4-3,0)a, \max(1-4,0)b, \max(3-0,0)c, \max(0-2,0)d\} \\
&= \{1 \cdot a, 3 \cdot c\}
\end{aligned}$$

$$\begin{aligned}
P + Q &= \{(4+3)a, (1+4)b, (3+0)c, (0+2)d\} \\
&= \{7 \cdot a, 5 \cdot b, 3 \cdot c, 2 \cdot d\}
\end{aligned}$$

ROAR

Fuzzy Sets

- *Fuzzy sets* are a type of set typically used in AI and ML
- Each element in the universe *U* has a degree of membership, in fuzzy set *S*
  - *Degree of membership* is a real number $[0, 1]$
- A fuzzy set is denoted by listing the elements with their degree
  - elements with degree 0 are not listed
- **Example:** $\{0.6\ \mathit{Alice}, 0.9\ \mathit{Brian}, 0.4\ \mathit{Fred}, 0.1\ \mathit{Oscar}, 0.5\ \mathit{Rita}\} = F$
- A traditional, or **crisp set**, is a fuzzy set where all elements that are members have a degree of 1.0 and all other elements have a degree of 0.0

Fuzzy Set Operations

Idaho State University | Computer Science

- **Union ($\cup$):** The union of two fuzzy sets S and T is the fuzzy set $S \cup T$ where the degree of membership of an element in $S \cup T$ is the maximum of the degrees of membership of this element in S and T
- **Intersection ($\cap$):** The intersection of two fuzzy sets S and T is the set $S \cap T$, where the degree of membership of an element in $S \cap T$ is the minimum of the degrees of membership of this element in S and in T.
- **Complement:** The complement of a fuzzy set S is the set $\overline{S}$, with the degree of membership of an element in $\overline{S}$ equal to 1.0 minus the degree of membership fo the element in S

ROAR

For Next Time

- Review DMUC Chapter 8
- Review DMA Chapter 2.1 – 2.2
- Review this Lecture
- Read DMUC Chapter 3

Are there any questions?