COMP2111 Week 2
Term 1, 2024
Discrete Mathematics Recap I

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic

Sets

A set is defined by the collection of its elements.
Sets are typically described by:
(a) Explicit enumeration of their elements

$$
\begin{aligned}
S_1 &= \{a, b, c\} = \{a, a, b, b, b, c\} \\
&= \{b, c, a\} = \ldots \quad \text{three elements} \\
S_2 &= \{a, \{a\}\} \quad \text{two elements} \\
S_3 &= \{a, b, \{a, b\}\} \quad \text{three elements} \\
S_4 &= \{\} \quad \text{zero elements} \\
S_5 &= \{\{\}\} \quad \text{one element} \\
S_6 &= \{\, \{\}, \{\{\}\}\, \} \quad \text{two elements}
\end{aligned}
$$

(b) Specifying the properties their elements must satisfy; the elements are taken from some 'universal' domain, $\mathcal{U}$. A typical description involves a **logical** property $P(x)$

$$S = \{\, x : x \in \mathcal{U} \text{ and } P(x) \,\} = \{\, x \in \mathcal{U} : P(x) \,\}$$

We distinguish between an element and the set comprising this single element. Thus always $a \neq \{a\}$.
Set $\{\}$ is empty (no elements);
set $\{\{\}\}$ is nonempty — it has one element.
There is only one empty set; only one set consisting of a single a; only one set of all natural numbers.

(c) Constructions from other sets (already defined)

- Union, intersection, set difference, symmetric difference, complement
- **Power set** $\mathrm{Pow}(X) = \{ A : A \subseteq X \}$
- Cartesian product (below)
- Empty set $\emptyset$
  $\emptyset \subseteq X$ for all sets X.

$S \subseteq T$ S is a **subset** of T; includes the case of $T \subseteq T$

$S \subset T$ a **proper** subset: $S \subseteq T$ and $S \neq T$

NB

Element and subset are two different concepts

$$a \in \{a, b\}, \quad a \not\subseteq \{a, b\}; \qquad \{a\} \subseteq \{a, b\}, \quad \{a\} \notin \{a, b\}$$

Cardinality

Number of elements in a set X (various notations):

$$|X| = \#(X) = \mathrm{card}(X)$$

Fact

Always $|Pow(X)| = 2^{|X|}$ *(for finite* X*)*

$|\emptyset| = 0$ $\quad \mathrm{Pow}(\emptyset) = \{\emptyset\}$ $\quad |\mathrm{Pow}(\emptyset)| = 1$
$\mathrm{Pow}(\mathrm{Pow}(\emptyset)) = \{\emptyset, \{\emptyset\}\}$ $\quad |\mathrm{Pow}(\mathrm{Pow}(\emptyset))| = 2 \quad \ldots$

$|\{a\}| = 1$ $\quad \mathrm{Pow}(\{a\}) = \{\emptyset, \{a\}\}$ $\quad |\mathrm{Pow}(\{a\})| = 2 \quad \ldots$

Sets of Numbers

Natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$
Positive integers $\{1, 2, \ldots\}$
Common notation: $\mathbb{N}^+ = \mathbb{N}_{>0} = \mathbb{Z}_{>0} = \mathbb{N} \setminus \{0\}$

Integers $\mathbb{Z} = \{\ldots, -1, 0, 1, 2, \ldots\}$
Rational numbers $\mathbb{Q} \subseteq \left\{ \frac{m}{n} : m, n \in \mathbb{Z}, n \neq 0 \right\}$
Real numbers $\mathbb{R}$

Intervals of numbers (applies to any type)

$$[a, b] = \{x | a \leq x \leq b\}; \quad (a, b) = \{x | a < x < b\}$$

$$[a, b] \supseteq [a, b), (a, b] \supseteq (a, b)$$

NB

$(a, a) = (a, a] = [a, a) = \emptyset$; *however* $[a, a] = \{a\}$.

Intervals of $\mathbb{N}, \mathbb{Z}$ are finite: if $m \leq n$

$$[m, n] = \{m, m + 1, \ldots, n\} \qquad |[m, n]| = n - m + 1$$

Set Operations

Union $A \cup B$; Intersection $A \cap B$

There is a correspondence between set operations and logical operators (to be discussed later)

We say that A, B are **disjoint** if $A \cap B = \emptyset$

NB

- $A \cup B = B$ *if and only if* $A \subseteq B$
- $A \cap B = B$ *if and only if* $A \supseteq B$

Other set operations

- $A \setminus B$ — **difference**, set difference, relative complement
  It corresponds (logically) to *a* but not *b*
- $A \oplus B$ — **symmetric difference**

$$A \oplus B \stackrel{\text{def}}{=} (A \setminus B) \cup (B \setminus A)$$

  It corresponds to *a* and not *b* or *b* and not *a*; also known as **xor** (**exclusive or**)
- A^c — set **complement** w.r.t. the 'universe' $\mathcal{U}$
  It corresponds to 'not *a*'

Laws of Set Operations

Commutativity	$A \cup B = B \cup A$
	$A \cap B = B \cap A$
Associativity	$(A \cup B) \cup C = A \cup (B \cup C)$
	$(A \cap B) \cap C = A \cap (B \cap C)$
Distribution	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Identity	$A \cup \emptyset = A$
	$A \cap \mathcal{U} = A$
Complementation	$A \cup (A^c) = \mathcal{U}$
	$A \cap (A^c) = \emptyset$

Other useful set laws

The following are all derivable from the previous 10 laws.

Idempotence	$A \cap A = A$
	$A \cup A = A$
Double complementation	$(A^c)^c = A$
Annihilation	$A \cap \emptyset = \emptyset$
	$A \cup \mathcal{U} = \mathcal{U}$
de Morgan's Laws	$(A \cap B)^c = A^c \cup B^c$
	$(A \cup B)^c = A^c \cap B^c$

Example (Idempotence of $\cup$)

$$A = A \cup \emptyset \qquad \text{(Identity)}$$

Example (Idempotence of $\cup$)

$$\begin{aligned} A &= A \cup \emptyset && \text{(Identity)} \\ &= A \cup (A \cap A^c) && \text{(Complementation)} \end{aligned}$$

Example (Idempotence of $\cup$)

$$\begin{aligned} A &= A \cup \emptyset && \text{(Identity)} \\ &= A \cup (A \cap A^c) && \text{(Complementation)} \\ &= (A \cup A) \cap (A \cup A^c) && \text{(Distributivity)} \end{aligned}$$

Example (Idempotence of $\cup$)

$$
\begin{aligned}
A &= A \cup \emptyset && \text{(Identity)} \\
&= A \cup (A \cap A^c) && \text{(Complementation)} \\
&= (A \cup A) \cap (A \cup A^c) && \text{(Distributivity)} \\
&= (A \cup A) \cap \mathcal{U} && \text{(Complementation)}
\end{aligned}
$$

Example (Idempotence of $\cup$)

$$
\begin{aligned}
A &= A \cup \emptyset && \text{(Identity)} \\
&= A \cup (A \cap A^c) && \text{(Complementation)} \\
&= (A \cup A) \cap (A \cup A^c) && \text{(Distributivity)} \\
&= (A \cup A) \cap \mathcal{U} && \text{(Complementation)} \\
&= (A \cup A) && \text{(Identity)}
\end{aligned}
$$

A useful result

Definition

If A is a set defined using $\cap$, $\cup$, $\emptyset$ and $\mathcal{U}$, then dual(A) is the expression obtained by replacing $\cap$ with $\cup$ (and vice-versa) and $\emptyset$ with $\mathcal{U}$ (and vice-versa).

Theorem (Principle of Duality)

If you can prove $A_1 = A_2$ using the Laws of Set Operations then you can prove $dual(A_1) = dual(A_2)$

Example

Absorption law: $A \cup (A \cap B) = A$

Dual: $A \cap (A \cup B) = A$

Application (Idempotence of $\cap$)

Recall Idempotence of $\cup$:

$$\begin{aligned} A &= A \cup \emptyset && \text{(Identity)} \\ &= A \cup (A \cap A^c) && \text{(Complementation)} \\ &= (A \cup A) \cap (A \cup A^c) && \text{(Distributivity)} \\ &= (A \cup A) \cap \mathcal{U} && \text{(Complementation)} \\ &= (A \cup A) && \text{(Identity)} \end{aligned}$$

Application (Idempotence of $\cap$)

Invoke the dual laws!

$$
\begin{aligned}
A &= A \cap \mathcal{U} && \text{(Identity)} \\
&= A \cap (A \cup A^c) && \text{(Complementation)} \\
&= (A \cap A) \cup (A \cap A^c) && \text{(Distributivity)} \\
&= (A \cap A) \cup \emptyset && \text{(Complementation)} \\
&= (A \cap A) && \text{(Identity)}
\end{aligned}
$$

Cartesian Product

$S \times T \stackrel{\text{def}}{=} \{ (s,t) : s \in S,\ t \in T \}$ where (s,t) is an **ordered** pair

$\times_{i=1}^{n} S_i \stackrel{\text{def}}{=} \{ (s_1, \ldots, s_n) : s_k \in S_k, \text{ for } 1 \leq k \leq n \}$

$S^2 = S \times S, \quad S^3 = S \times S \times S, \ldots, \quad S^n = \times_1^n S, \ldots$

$\emptyset \times S = \emptyset$, for every S

$|S \times T| = |S| \cdot |T|, \quad |\times_{i=1}^{n} S_i| = \prod_{i=1}^{n} |S_i|$

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic

Formal Languages

Σ — **alphabet**, a finite, nonempty set

Examples (of various alphabets and their intended uses)

$\Sigma = \{a, b, \ldots, z\}$ for single words (in lower case)
$\Sigma = \{\sqcup, -, a, b, \ldots, z\}$ for composite terms
$\Sigma = \{0, 1\}$ for binary integers
$\Sigma = \{0, 1, \ldots, 9\}$ for decimal integers

The above cases all have a natural ordering; a formal language does not need this.

Definition

word — any finite string of symbols from Σ
empty word — λ (sometimes ϵ)

Example

$w = aba$, $w = 01101\ldots1$, etc.

length(w) — # of symbols in w
length(aaa) $= 3$, length(λ) $= 0$
The only operation on words (discussed here) is **concatenation**, written as juxtaposition $vw, wvw, abw, wbv, \ldots$

NB

$\lambda w = w = w\lambda$
length(vw) = length(v) + length(w)

Notation: Σ^k — set of all words of length k
We often identify $\Sigma^0 = \{\lambda\}$, $\Sigma^1 = \Sigma$
Σ^* — set of all words (of all [finite] lengths)
Σ^+ — set of all nonempty words (of any positive length)

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots; \quad \Sigma^{\leq n} = \bigcup_{i=0}^{n} \Sigma^i$$

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots = \Sigma^* \setminus \{\lambda\}$$

A **language** is a subset of Σ^*. Typically, only the subsets that can be formed (or described) according to certain rules are of interest. Such a collection of 'descriptive/formative' rules is called a **grammar**.

Examples: Programming languages, Database query languages

Example (Decimal numbers)

The "language" of all numbers written in decimal to at most two decimal places can be described as follows:

- $\Sigma = \{-, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Consider all words $w \in \Sigma^*$ which satisfy the following:
  - w contains at most one instance of $-$, and if it contains an instance then it is the first symbol.
  - w contains at most one instance of ., and if it contains an instance then it is preceded by a symbol in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and followed by either one or two symbols in that set.
  - w contains at least one symbol from $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

NB

According to these rules 123, 123.0 *and* 123.00 *are all (distinct) words in this language.*

Example (HTML documents)

Take $\Sigma = \{$“<html>”, “</html>”, “<head>”, “</head>”, “<body>”, $\ldots\}$.

The (language of) **valid HTML documents** is loosely described as follows:

- Starts with “<html>”
- Next symbol is “<head>”
- Followed by zero or more symbols from the set of HeadItems (defined elsewhere)
- Followed by “</head>”
- Followed by “<body>”
- Followed by zero or more symbols from the set of BodyItems (defined elsewhere)
- Followed by “</body>”
- Followed by “</html>”

Set Operations for Languages

Languages are sets, so the standard set operations ($\cap$, $\cup$, $\setminus$, $\oplus$, etc) can be used to build new languages.
Two set operations that apply to languages uniquely:

- Concatenation (written as juxtaposition): $XY = \{xy \ : \ x \in X \text{ and } y \in Y\}$
- Kleene star: X^* is the set of words that are made up by concatenating 0 or more words in X

Set Operations for Languages

Example

Let $A = \{aa, bb\}$ and $B = \{\lambda, c\}$ be languages over $\Sigma = \{a, b, c\}$.

- $A \cup B = \{\lambda, c, aa, bb\}$
- $AB = \{aa, bb, aac, bbc\}$
- $AA = \{aaaa, aabb, bbaa, bbbb\}$

Set Operations for Languages

Example

Let $A = \{aa, bb\}$ and $B = \{\lambda, c\}$ be languages over $\Sigma = \{a, b, c\}$.

- $A \cup B = \{\lambda, c, aa, bb\}$
- $AB = \{aa, bb, aac, bbc\}$
- $AA = \{aaaa, aabb, bbaa, bbbb\}$
- $A^* = \{\lambda, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, \ldots\}$
- $B^* = \{\lambda, c, cc, ccc, cccc, \ldots\}$

Set Operations for Languages

Example

Let $A = \{aa, bb\}$ and $B = \{\lambda, c\}$ be languages over $\Sigma = \{a, b, c\}$.

- $A \cup B = \{\lambda, c, aa, bb\}$
- $AB = \{aa, bb, aac, bbc\}$
- $AA = \{aaaa, aabb, bbaa, bbbb\}$
- $A^* = \{\lambda, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, \ldots\}$
- $B^* = \{\lambda, c, cc, ccc, cccc, \ldots\}$
- $\{\lambda\}^* =$

Set Operations for Languages

Example

Let $A = \{aa, bb\}$ and $B = \{\lambda, c\}$ be languages over $\Sigma = \{a, b, c\}$.

- $A \cup B = \{\lambda, c, aa, bb\}$
- $AB = \{aa, bb, aac, bbc\}$
- $AA = \{aaaa, aabb, bbaa, bbbb\}$
- $A^* = \{\lambda, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, \ldots\}$
- $B^* = \{\lambda, c, cc, ccc, cccc, \ldots\}$
- $\{\lambda\}^* = \{\lambda\}$

Set Operations for Languages

Example

Let $A = \{aa, bb\}$ and $B = \{\lambda, c\}$ be languages over $\Sigma = \{a, b, c\}$.

- $A \cup B = \{\lambda, c, aa, bb\}$
- $AB = \{aa, bb, aac, bbc\}$
- $AA = \{aaaa, aabb, bbaa, bbbb\}$
- $A^* = \{\lambda, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, \ldots\}$
- $B^* = \{\lambda, c, cc, ccc, cccc, \ldots\}$
- $\{\lambda\}^* = \{\lambda\}$
- $\emptyset^* =$

Set Operations for Languages

Example

Let $A = \{aa, bb\}$ and $B = \{\lambda, c\}$ be languages over $\Sigma = \{a, b, c\}$.

- $A \cup B = \{\lambda, c, aa, bb\}$
- $AB = \{aa, bb, aac, bbc\}$
- $AA = \{aaaa, aabb, bbaa, bbbb\}$
- $A^* = \{\lambda, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, \ldots\}$
- $B^* = \{\lambda, c, cc, ccc, cccc, \ldots\}$
- $\{\lambda\}^* = \{\lambda\}$
- $\emptyset^* = \{\lambda\}$

Summary of topics

- Sets
- Formal languages
- Relations
- Functions
- Propositional Logic

Relations and Functions

Relations capture the idea that objects are *related* (well, duh)

- $\leq$ ("less than")
- "is a Facebook friend of"
- "has a different hair colour than"

Functions capture the idea of transforming *inputs* into *outputs*.

In general, functions and relations formalise the concept of interaction among objects from various domains; however, there must be a specified domain for each type of object.

Applications

Relations and functions are ubiquitous in Computer Science

- Databases are collections of relations
- Common data structures (e.g. graphs) are relations
- Any ordering is a relation
- Functions, procedures and programs are relations between their inputs and outputs

Relations are therefore used in most problem specifications and to describe formal properties of programs.
For this reason, studying relations and their properties helps with formalisation, implementation and verification of programs.

Relations

An n-**ary relation** is a subset of the Cartesian product of n sets.

$$R \subseteq S_1 \times S_2 \times \ldots \times S_n$$

$$x \in R \rightarrow x = (x_1, x_2, \ldots x_n) \text{ where each } x_i \in S_i$$

If $n = 2$ we have a **binary** relation $R \subseteq S \times T$.
(mostly we consider binary relations)
equivalent notations: $(x_1, x_2, \ldots x_n) \in R \iff R(x_1, x_2, \ldots x_n)$
for binary relations: $(x, y) \in R \iff R(x, y) \iff xRy$.

Examples

- Equality: $=$
- Inequality: $\leq$, $\geq$, $<$, $>$, $\neq$
- Divides relation: $|$ (recall $m|n$ if $n = km$ for some $k \in \mathbb{Z}$)
- Element of: $\in$
- Subset, superset: $\subseteq$, $\subset$, $\supseteq$, $\supset$
- Size functions (sort of): $|\cdot|$, length$(\cdot)$

Database Examples

Example (Course enrolments)

S = set of CSE students
C = set of CSE courses
E = enrolments = $\{ (s, c) : s \text{ takes } c \}$

$$E \subseteq S \times C$$

In practice, almost always there are various 'onto' (nonemptiness) and 1–1 (uniqueness) constraints on database relations.

Example (Class schedule)

C = CSE courses
T = starting time (hour & day)
R = lecture rooms
S = schedule =

$$\{ (c, t, r) : c \text{ is at } t \text{ in } r \} \subseteq C \times T \times R$$

Example (sport stats)

$$R \subseteq \text{competitions} \times \text{results} \times \text{years} \times \text{athletes}$$

n-ary Relations

Relations can be defined linking $k \geq 1$ domains $D_1, \ldots, D_k$ simultaneously.
In database situations one also allows for *unary* ($n = 1$) relations.
Most common are **binary** relations

$$R \subseteq S \times T; \quad R = \{(s,t) \ : \ \text{“some property that links } s,t\text{”}\}$$

For related s, t we can write $(s,t) \in R$ or sRt; for unrelated items either $(s,t) \notin R$ or $s \not R t$.
R can be defined by

- explicit enumeration of interrelated k-tuples (ordered pairs in case of binary relations);
- properties that identify relevant tuples within the entire $D_1 \times D_2 \times \ldots \times D_k$;
- construction from other relations.

Relation R as Correspondence From S to T

Given $R \subseteq S \times T$, $A \subseteq S$, and $B \subseteq T$.

- $R(A) \stackrel{\text{def}}{=} \{t \in T \ : \ (s,t) \in R \text{ for some } s \in A\}$
- Converse relation $R^{\leftarrow} \subseteq T \times S$:

$$R^{\leftarrow} \stackrel{\text{def}}{=} \{(t,s) \in T \times S \ : \ (s,t) \in R\}$$

- $R^{\leftarrow}(B) = \{s \in S \ : \ (s,t) \in R \text{ for some } t \in B\}$

Observe that $(R^{\leftarrow})^{\leftarrow} = R$.

Binary Relations

A binary relation, say $R \subseteq S \times T$, can be presented as a matrix with rows enumerated by (the elements of) S and the columns by T; eg. for $S = \{s_1, s_2, s_3\}$ and $T = \{t_1, t_2, t_3, t_4\}$ we may have

$$\begin{bmatrix} \bullet & \circ & \bullet & \bullet \\ \circ & \bullet & \bullet & \bullet \\ \bullet & \bullet & \circ & \circ \end{bmatrix}$$

Relations on a Single Domain

Particularly important are binary relationships between the elements of the same set. We say that 'R is a relation on S' if

$$R \subseteq S \times S$$

Such relations can be visualized as a directed graph:

- Vertices: Elements of S
- Edges: Elements of R

Example

$S = \{1, 2, 3\}$
$R = \{(1, 2), (2, 3), (3, 2)\}$

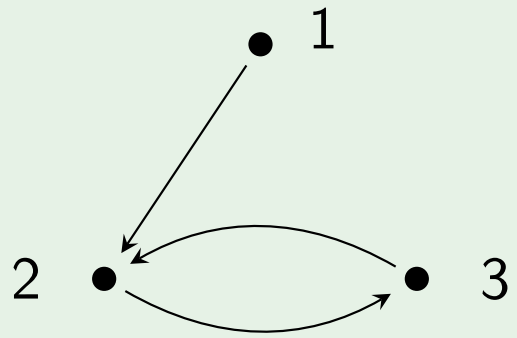
As a matrix:

$$\begin{pmatrix} \circ & \bullet & \circ \\ \circ & \circ & \bullet \\ \circ & \bullet & \circ \end{pmatrix}$$

Example

$S = \{1, 2, 3\}$
$R = \{(1, 2), (2, 3), (3, 2)\}$

As a graph:



Special (Trivial) Relations

(all w.r.t. set S)

Identity (diagonal, equality) $E = \{ (x, x) : x \in S \}$

Empty $\emptyset$

Universal $\mathcal{U} = S \times S$

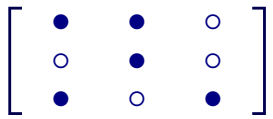
Important Properties of Binary Relations $R \subseteq S \times S$

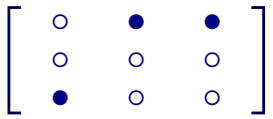
(R)	reflexive	$(x, x) \in R$	$\forall\, x \in S$
(AR)	antireflexive	$(x, x) \notin R$	$\forall\, x \in S$
(S)	symmetric	$(x, y) \in R \rightarrow (y, x) \in R$	$\forall\, x, y \in S$
(AS)	antisymmetric	$(x, y), (y, x) \in R \rightarrow x = y$	$\forall\, x, y \in S$
(T)	transitive	$(x, y), (y, z) \in R \rightarrow (x, z) \in R$	$\forall\, x, y, z \in S$

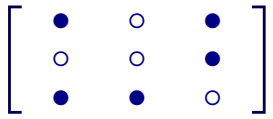
NB

An object, notion etc. is considered to satisfy a property if none of its instances violates any defining statement of that property.

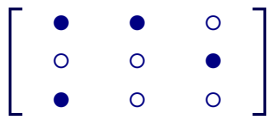
Examples

(R) reflexive $(x, x) \in R$ for all $x \in S$ 

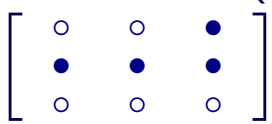
(AR) antireflexive $(x, x) \notin R$ 

(S) symmetric $(x, y) \in R \to (y, x) \in R$ 

(AS) antisymmetric $(x, y), (y, x) \in R \to x = y$



(T) transitive $(x, y), (y, z) \in R \to (x, z) \in R$



Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$					
$\leq$					
$<$					
$\emptyset$					
$\mathcal{U}$					
$\mid$					

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$					
$<$					
$\emptyset$					
$\mathcal{U}$					
$\mid$					

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$	✓			✓	✓
$<$					
$\emptyset$					
$\mathcal{U}$					
$\mid$					

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$	✓			✓	✓
$<$		✓		✓	✓
$\emptyset$					
$\mathcal{U}$					
$\mid$					

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$	✓			✓	✓
$<$		✓		✓	✓
$\emptyset$		✓	✓	✓	✓
$\mathcal{U}$					
$\mid$					

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$	✓			✓	✓
$<$		✓		✓	✓
$\emptyset$		✓	✓	✓	✓
$\mathcal{U}$	✓		✓		✓
$\mid$					

Common relations and their properties

	(R)	(AR)	(S)	(AS)	(T)
$=$	✓		✓	✓	✓
$\leq$	✓			✓	✓
$<$		✓		✓	✓
$\emptyset$		✓	✓	✓	✓
$\mathcal{U}$	✓		✓		✓
$\mid$	✓			✓	✓

Interaction of Properties

A relation *can* be both symmetric and antisymmetric. Namely, when R consists only of some pairs $(x, x), x \in S$.
A relation *cannot* be simultaneously reflexive and antireflexive (unless $S = \emptyset$).

NB

nonreflexive / *nonsymmetric* } *is not the same as* { *antireflexive/irreflexive* / *antisymmetric*

Equivalence Relations and Partitions

Relation R is called an *equivalence* relation if it satisfies (R), (S), (T). Every equivalence R defines *equivalence classes* on its domain S.
The equivalence class $[s]$ (w.r.t. R) of an element $s \in S$ is

$$[s] = \{\ t \in S : tRs\ \}$$

The collection of all equivalence classes $[S]_R = \{\ [s] : s \in S\ \}$ is a partition of S:

$$S = \bigcup_{s \in S} [s]$$

Thus, the equivalence classes are disjoint and jointly cover the entire domain. It means that every element belongs to one (and only one) equivalence class.
For $s, t \in S$ either $[s] = [t]$, when $s \; R \; t$, or $[s] \cap [t] = \emptyset$, when $s \not R \; t$. We write $s \sim_R t$ when s, t are in the same equivalence class.
In the opposite direction, a partition of a set defines an equivalence relation on that set. If $S = S_1 \dot\cup \ldots \dot\cup S_k$, then we specify $s \sim t$ exactly when s and t belong to the same S_i.

Partial Order

A **partial order** $\preceq$ on S satisfies (R), (AS), (T).
We call $(S, \preceq)$ a **poset** — partially ordered set

Examples

Posets:

- $(\mathbb{Z}, \leq)$
- $(\mathrm{Pow}(X), \subseteq)$ for some set X
- $(\mathbb{N}, |)$

Not posets:

- $(\mathbb{Z}, <)$
- $(\mathbb{Z}, |)$

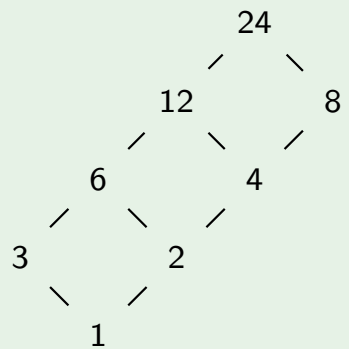
Hasse diagram

Every finite poset $(S, \preceq)$ can be represented with a **Hasse diagram**:

- Nodes are elements of S
- An edge is drawn *upward* from x to y if $x \prec y$ and there is no z such that $x \prec z \prec y$

Example

Hasse diagram for positive divisors of 24 ordered by $|$:



Ordering Concepts

Definition

Let $(S, \preceq)$ be a poset.

- **Minimal** element: x such that there is no y with $y \preceq x$
- **Maximal** element: x such that there is no y with $x \preceq y$
- **Minimum (least)** element: x such that $x \preceq y$ for all $y \in S$
- **Maximum (greatest)** element: x such that $y \preceq x$ for all $y \in S$

NB

- *There may be multiple minimal/maximal elements.*
- *Minimum/maximum elements are the unique minimal/maximal elements if they exist.*
- *Minimal/maximal elements always exist in finite posets, but not necessarily in infinite posets.*

Examples

Examples

- $\mathrm{Pow}(\{a, b, c\})$ with the order $\subseteq$
  $\emptyset$ is minimum; $\{a, b, c\}$ is maximum
- $\mathrm{Pow}(\{a, b, c\}) \setminus \{\{a, b, c\}\}$ (proper subsets of $\{a, b, c\}$)
  Each two-element subset $\{a, b\}, \{a, c\}, \{b, c\}$ is maximal.
  - But there is no maximum

Summary of topics

- Sets
- Formal languages
- Relations
- Functions (tomorrow)
- Propositional Logic