MATH 433

Applied Algebra

Lecture 15:
Relations (continued).
Finite state machines.

Relations

Definition. Let X and Y be sets. A **relation** R from X to Y is given by specifying a subset of the Cartesian product: $S_R \subset X \times Y$.

If $(x, y) \in S_R$, then we say that x **is related to** y (in the sense of R or by R) and write xRy.

Remarks. • Usually the relation R is identified with the set S_R.

• In the case $X = Y$, the relation R is called a **relation on** X.

Examples. • "is equal to"
$xRy \iff x = y$
Equivalently, $R = \{(x, x) \mid x \in X \cap Y\}$.

• "is not equal to"
$xRy \iff x \neq y$

• "is mapped by f to"
$xRy \iff y = f(x)$, where $f : X \to Y$ is a function.
Equivalently, R is the graph of the function f.

• "is the image under f of"
(from Y to X) $yRx \iff y = f(x)$, where $f : X \to Y$ is a function. If f is invertible, then R is the graph of f^{-1}.

• reversed R'
$xRy \iff yR'x$, where R' is a relation from Y to X.

• not R'
$xRy \iff$ not $xR'y$, where R' is a relation from X to Y.
Equivalently, $R = (X \times Y) \setminus R'$ (set difference).

Relations on a set

- "is equal to"

$xRy \iff x = y$

- "is not equal to"

$xRy \iff x \neq y$

- "is less than"

$X = \mathbb{R},\ xRy \iff x < y$

- "is less than or equal to"

$X = \mathbb{R},\ xRy \iff x \leq y$

- "is contained in"

$X =$ the set of all subsets of some set Y,

$xRy \iff x \subset y$

- "is congruent modulo n to"

$X = \mathbb{Z},\ xRy \iff x \equiv y \mod n$

- "divides"

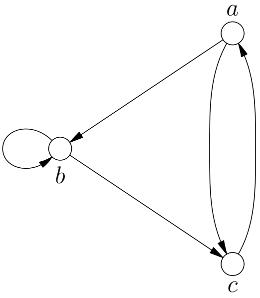
$X = \mathbb{P},\ xRy \iff x | y$

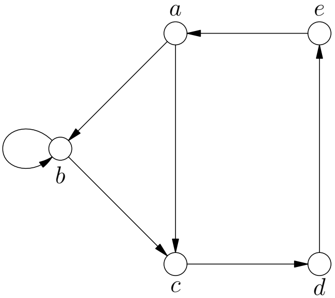
A relation R on a finite set X can be represented by a **directed graph**.

Vertices of the graph are elements of X, and we have a directed edge from x to y if and only if xRy.

Another way to represent the relation R is the **adjacency table**.

Rows and columns are labeled by elements of X. We put 1 at the intersection of a row x with a column y if xRy. Otherwise we put 0.





	a	b	c
a	0	1	1
b	0	1	1
c	1	0	0

	a	b	c	d	e
a	0	1	1	0	0
b	0	1	1	0	0
c	0	0	0	1	0
d	0	0	0	0	1
e	1	0	0	0	0

Properties of relations

Definition. Let R be a relation on a set X. We say that R is

- **reflexive** if xRx for all $x \in X$,
- **symmetric** if, for all $x, y \in X$, xRy implies yRx,
- **antisymmetric** if, for all $x, y \in X$, xRy and yRx cannot hold simultaneously,
- **weakly antisymmetric** if, for all $x, y \in X$, xRy and yRx imply that $x = y$,
- **transitive** if, for all $x, y, z \in X$, xRy and yRz imply that xRz.

Partial ordering

Definition. A relation R on a set X is a **partial ordering** (or **partial order**) if R is reflexive, weakly antisymmetric, and transitive:

- xRx,
- xRy and $yRx \implies x = y$,
- xRy and $yRz \implies xRz$.

A relation R on a set X is a **strict partial order** if R is antisymmetric and transitive:

- $xRy \implies$ not yRx,
- xRy and $yRz \implies xRz$.

Examples. "is less than or equal to", "is contained in", "is a divisor of" are partial orders. "is less than" is a strict order.

Hasse diagrams

Any partial order $\preceq$ on a finite set S can be visualized by a directed graph called the **Hasse diagram**. Nodes of the graph are elements of S. The graph contains an edge from a node x to a node y if $x \preceq y$, $x \neq y$, and there is no z different from x and y such that $x \preceq z \preceq y$.

Given $x, y \in S$, we have $x \preceq y$ if and only if there is a directed path from x to y in the Hasse diagram (the path may not be unique).

The Hasse diagram is usually drawn so that y is located higher than x whenever $x \preceq y$ and $x \neq y$. That way it is easier to find directed paths. Besides, the edges need not be drawn as arrows since directions are clear from the picture.

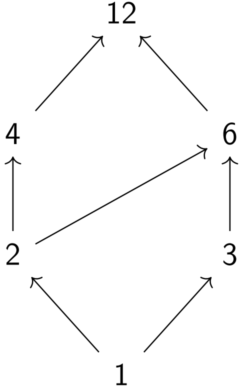
Examples of Hasse diagrams

- Subsets of $\{a, b, c\}$

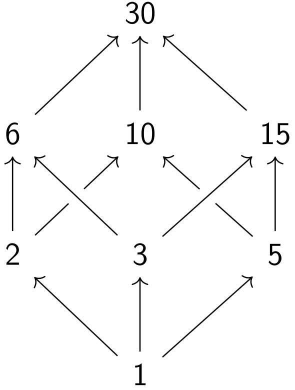
$$
\begin{tikzcd}
 & \{a, b, c\} & \\
\{a, b\} \arrow[ur] & \{a, c\} \arrow[u] & \{b, c\} \arrow[ul] \\
\{a\} \arrow[u] \arrow[ur] & \{b\} \arrow[ul] \arrow[ur] & \{c\} \arrow[ul] \arrow[u] \\
 & \emptyset \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}
$$

Examples of Hasse diagrams

- Divisors of 12



- Divisors of 30



Equivalence relation

Definition. A relation R on a set X is an **equivalence relation** if R is reflexive, symmetric, and transitive:

- xRx,
- $xRy \implies yRx$,
- xRy and $yRz \implies xRz$.

Examples. "is equal to", "is congruent modulo n to" are equivalence relations.

Given an equivalence relation R on X, the **equivalence class** of an element $x \in X$ relative to R is the set of all elements $y \in X$ such that yRx.

Theorem The equivalence classes form a **partition** of the set X, which means that

- any two equivalence classes either coincide, or else they are disjoint,
- any element of X belongs to some equivalence class.

Finite state machine

A **finite state machine** is a triple $M = (S, A, t)$, where S and A are nonempty finite sets and $t : S \times A \to S$ is a function.

- Elements of S are called **states**.
- There is one distinguished element of S called the **initial state**.
- The set A is called the **input alphabet**, its elements are called **letters**.
- The function t is called the **state transition function**.

Notation in the textbook: the states are denoted by natural numbers; the initial state is denoted 0; the alphabet is a subset of the Roman alphabet.

We think of the finite state machine $M = (S, A, t)$ as a formal description of a certain device that operates in a discrete manner. At any moment it is supposed to be in some state $s \in S$. The machine reads an input letter $a \in A$. Then it makes transition to the state $t(s, a)$. After that the machine is ready to accept another input letter.

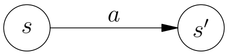
A machine's job looks as follows. We prepare an **input word** w, i.e., a sequence $a_1 a_2 \dots a_n$ of letters from A. Then we set the machine to the initial state s_0 and start inputting the word w into it, letter by letter. The machine's job results in a sequence of states $s_0, s_1, \dots, s_n$, which describes the internal work of the device.

current state s
input letter a

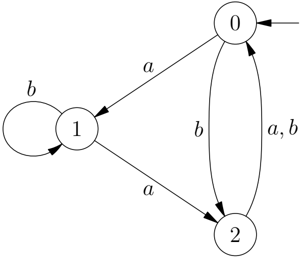
$\downarrow$

transition to the state $s' = t(s, a)$

One way to define (or picture) a finite state machine is the **Moore diagram**. This is a directed graph with labeled vertices and edges. Vertices are labeled by states, edges show possible transition routes (labeled by letters). The initial state is marked by an arrow pointing at it from nowhere.



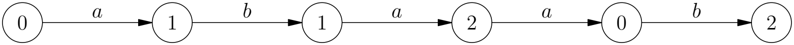
Example



	a	b
0	1	2
1	2	1
2	0	0

(The table is another way to define the transition function.)

Suppose that the input word is $w = abaab$. The internal work of the machine on this input is determined by a path in the graph such that **(i)** the path starts at the initial state 0 and **(ii)** the word w is read off the labels along the path.



Actions

The finite state machine as defined above is simply a **transition machine**. To make use of it, we need to add **actions**, which means activities triggered by the machine's work.

There are four types of actions:

- **entry action** is performed upon entering a state,
- **exit action** is performed upon exiting a state,
- **transition action** is performed upon specific transition,
- **input action** is performed depending on present state and input.

The input action is the most general type of action. Actions of the other types can be simulated by input actions.

A **Moore machine** is a finite state machine that uses only entry actions.

A **Mealy machine** is a finite state machine that uses only input actions.

Both kinds of machines are equivalent in terms of functionality. The Moore machines have simpler behaviour while the use of the Mealy machines allows to reduce the number of states.