universität innsbruck

Discrete structures

Manuel Eberl
Jan Jakubuv
Jamie Hochrainer
Cezary Kaliszyk

http://cl-informatik.uibk.ac.at/

Summary last week

- model questions as problems on discrete structures
- various problems modelled as graph problems
- representing graphs as sets of vertices and edges, and by adjacency matrices
- Floyds shortest path algorithm, stepwise transforming adjacency matrix

Summary last week

Theorem

The following algorithm overwrites the matrix B with the matrix of distances

> For r from 0 to $n-1$ repeat:
>
>   Set $N = B$.
>
>   For i from 0 to $n-1$ repeat:
>
>   For j from 0 to $n-1$ repeat:
>
>   Set $N_{ij} = \min(B_{ij}, B_{ir} + B_{rj})$.
>
>   Set $B = N$.

Summary last week

Theorem

The following algorithm overwrites the matrix B with the matrix of distances

> *For r from 0 to $n-1$ repeat:*
>
>   *Set $N = B$.*
>
>   *For i from 0 to $n-1$ repeat:*
>
>   *For j from 0 to $n-1$ repeat:*
>
>   *Set $N_{ij} = \min(B_{ij}, B_{ir} + B_{rj})$.*
>
>   *Set $B = N$.*

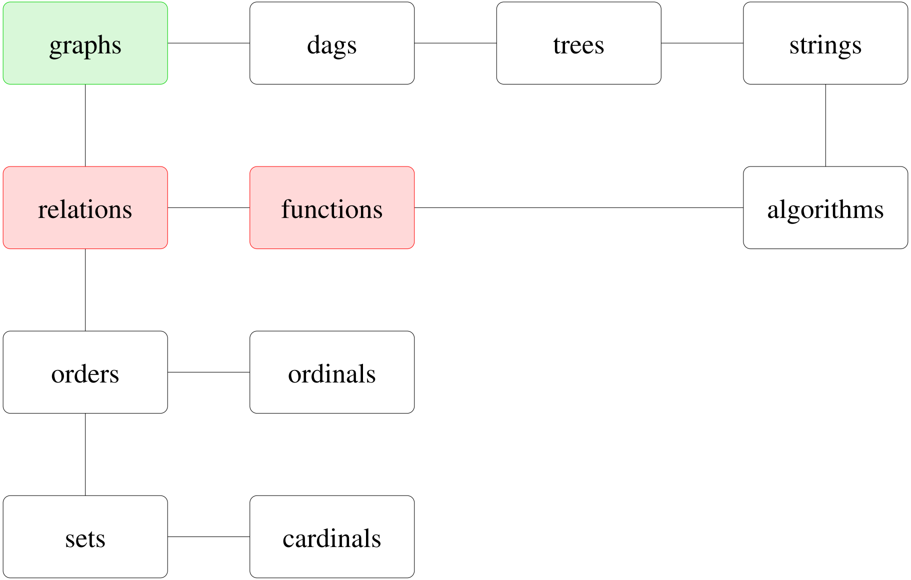
Proof.

today [1] $\blacksquare$

Course themes

- directed and undirected graphs
- relations and functions
- orders and induction
- trees and dags
- finite and infinite counting
- elementary number theory
- Turing machines, algorithms, and complexity
- decidable and undecidable problem

Discrete structures



Indirect proof resp. proof by contradiction

Definition

- To show that a statement A holds, a proof by contradiction assumes that the negation of A holds.
- If from this assumption (that the negation of A holds, that is, that A is false) a contradiction can be deduced, then our assumption itself must have been false, hence A must hold.

Indirect proof resp. proof by contradiction

Definition

- To show that a statement A holds, a proof by contradiction assumes that the negation of A holds.
- If from this assumption (that the negation of A holds, that is, that A is false) a contradiction can be deduced, then our assumption itself must have been false, hence A must hold.

Example

The statement

?There are infinitely many natural numbers.?

is true (and therefore a theorem). To show this, we assume the negation of the statement, that is

?There are only finitely many natural numbers.?

Indirect proof resp. proof by contradiction

Example

A positive natural number is prime if it has no non-trivial divisors.

Indirect proof resp. proof by contradiction

Example

A positive natural number is prime if it has no non-trivial divisors.

Theorem

There are infinitely many prime numbers.

Indirect proof resp. proof by contradiction

Example

A positive natural number is prime if it has no non-trivial divisors.

Theorem

There are infinitely many prime numbers.

To show this, we assume the negation of the statement. That is,

Indirect proof resp. proof by contradiction

Example

A positive natural number is prime if it has no non-trivial divisors.

Theorem

There are infinitely many prime numbers.

To show this, we assume the negation of the statement. That is,
Suppose there were only finitely many prime numbers, say $p_1, \ldots, p_n$.

Indirect proof resp. proof by contradiction

Example

A positive natural number is prime if it has no non-trivial divisors.

Theorem

There are infinitely many prime numbers.

To show this, we assume the negation of the statement. That is,
Suppose there were only finitely many prime numbers, say $p_1, \ldots, p_n$.
Any number can be written as product of prime numbers (Fundamental Theorem of Arithmetic).

Indirect proof resp. proof by contradiction

Example

A positive natural number is prime if it has no non-trivial divisors.

Theorem

There are infinitely many prime numbers.

To show this, we assume the negation of the statement. That is,
Suppose there were only finitely many prime numbers, say $p_1, \ldots, p_n$.
Any number can be written as product of prime numbers (Fundamental Theorem of Arithmetic).
In particular then $p = p_1 \cdot \ldots \cdot p_n + 1$ could be written as a product of prime numbers.
But per construction p is not divisible by any p_i, so would be prime. Contradiction.

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

By contradiction: Suppose $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ were two distinct v-splits of the path p.

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

By contradiction: Suppose $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ were two distinct v-splits of the path p.
Looking from the left there then must be a first i such that p_i is distinct from q_i (one of them may not exist) as otherwise the v-splits would be the same.

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

By contradiction: Suppose $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ were two distinct v-splits of the path p.
Looking from the left there then must be a first i such that p_i is distinct from q_i (one of them may not exist) as otherwise the v-splits would be the same.
Note that then p_i is a proper prefix of q_i (or vice versa; that case is similar)

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

By contradiction: Suppose $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ were two distinct v-splits of the path p.
Looking from the left there then must be a first i such that p_i is distinct from q_i (one of them may not exist) as otherwise the v-splits would be the same.
Note that then p_i is a proper prefix of q_i (or vice versa; that case is similar)
Then p_i must end in v but that would mean that q_i has some intermediate v-node (since both v-splits must yield p when concatenated).

Indirect proof resp. proof by contradiction

Example

An v-split of a path p in G is a sequence of paths $p_1, \ldots, p_n$ for some n, such that concatenating them yields p, and p_i starts (ends) at v and is non-empty if $i > 1$ ($i < n$), and v only occurs as start/end of the p_i (not as an intermediate node).

Theorem

a path has at most one v-split.

By contradiction: Suppose $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ were two distinct v-splits of the path p.
Looking from the left there then must be a first i such that p_i is distinct from q_i (one of them may not exist) as otherwise the v-splits would be the same.
Note that then p_i is a proper prefix of q_i (or vice versa; that case is similar)
Then p_i must end in v but that would mean that q_i has some intermediate v-node (since both v-splits must yield p when concatenated).
Contradiction with the assumption that $q_1, \ldots, q_m$ was a v-split.

Properties of Floyd’s algorithm

- Does it work? What does that mean, exactly?
- In what language do we express that?
- How do we prove it?
- Why does the algorithm work?
- How fast is it? As a function of what?
- How much memory does it use?
- How do we express this in a computer-independent way?
- . . .

Floyd correctness

Theorem

Input: adjacency matrix of graph G

Output: distance matrix of graph G

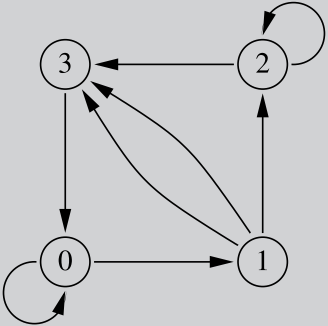
Proof.

Idea: successively compute distances via subsets of nodes.

1. Pre: distance via empty subset $\emptyset$ is
   - 0 from node to itself
   - edge weight if edge between distinct nodes
   - ∞ if no edge
2. (Outer) Loop invariant:

   Input: matrix of distances in G via nodes $\{v_0, \ldots, v_{r-1}\}$
   Output: matrix of distances in G via nodes $\{v_0, \ldots, v_{r-1}, v_r\}$
3. Post: distance via all nodes is distance

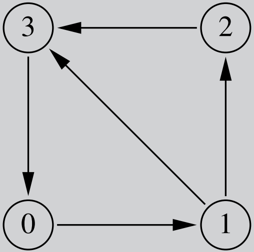
Example



$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

multigraph

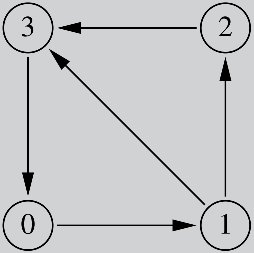
Example



$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

digraph

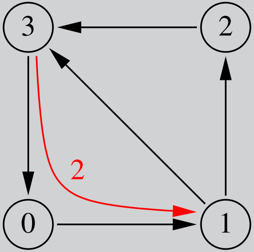
Example



$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & \infty & \infty & 0 \end{pmatrix}$$

shortest paths via $\emptyset$

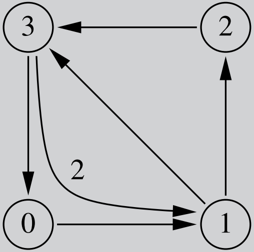
Example



$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & \infty & 0 \end{pmatrix}$$

shortest paths via $\{0\}$

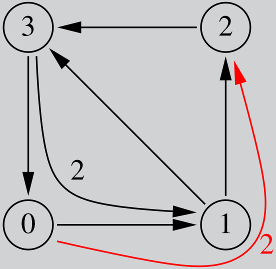
Example



$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & \infty & 0 \end{pmatrix}$$

shortest paths via $\{0\}$

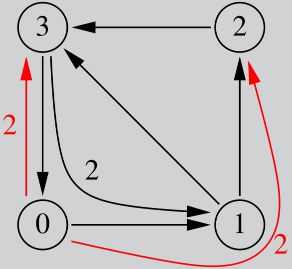
Example



$$\begin{pmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & \infty & 0 \end{pmatrix}$$

shortest paths via $\{0, 1\}$

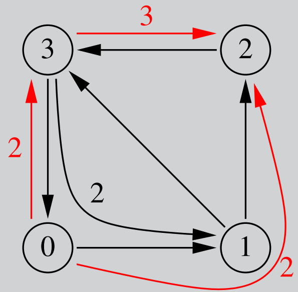
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & \infty & 0 \end{pmatrix}$$

shortest paths via $\{0, 1\}$

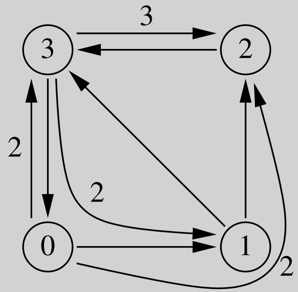
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & \color{red}{3} & 0 \end{pmatrix}$$

shortest paths via $\{0, 1\}$

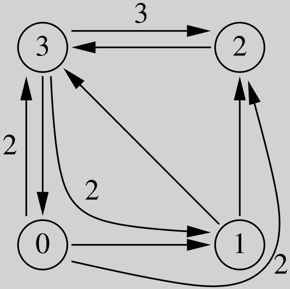
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths via $\{0, 1\}$

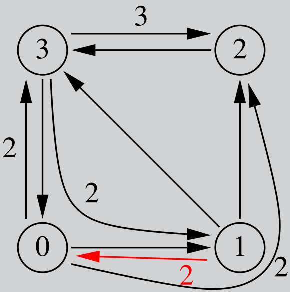
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths via $\{0, 1, 2\}$

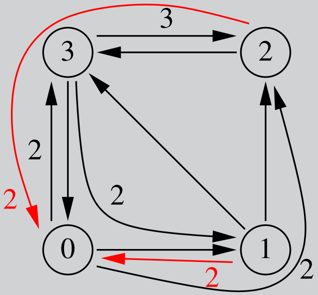
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ \infty & \infty & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths via $\{0, 1, 2, 3\}$

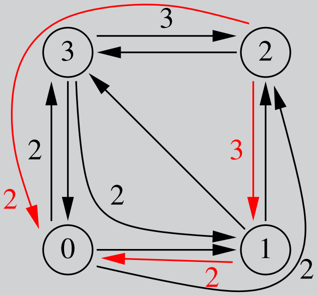
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ {\color{red}2} & \infty & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths via $\{0, 1, 2, 3\}$

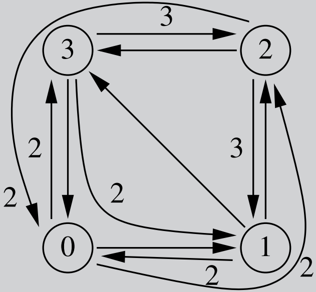
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ 2 & 3 & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths via $\{0, 1, 2, 3\}$

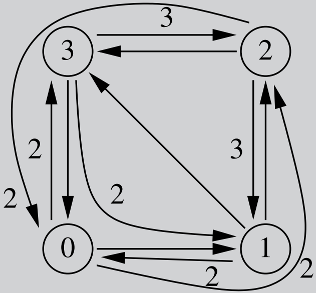
Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ 2 & 3 & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths via $\{0, 1, 2, 3\}$

Example



$$\begin{pmatrix} 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ 2 & 3 & 0 & 1 \\ 1 & 2 & 3 & 0 \end{pmatrix}$$

shortest paths

Correctness of middle and inner loop

Lemma

Let G be a directed multigraph. If there is a non-empty path p from node c to node d, then there is a simple path from c to d, obtained by omitting edges

Observation

Shortest paths are simple

Correctness of middle and inner loop

Auxiliary definition

For $r \in \{0, 1, \ldots, n\}$ let P_r be the set of all shortest paths in the graph of R that only have intermediate nodes in the set $\{v_0, v_1, \ldots, v_{r-1}\}$. Then

Correctness of middle and inner loop

Auxiliary definition

For $r \in \{0, 1, \ldots, n\}$ let P_r be the set of all shortest paths in the graph of R that only have intermediate nodes in the set $\{v_0, v_1, \ldots, v_{r-1}\}$. Then

Lemma

1. *P_0 is the set of all edges of G and empty paths;*

Correctness of middle and inner loop

Auxiliary definition

For $r \in \{0, 1, \ldots, n\}$ let P_r be the set of all shortest paths in the graph of R that only have intermediate nodes in the set $\{v_0, v_1, \ldots, v_{r-1}\}$. Then

Lemma

1. *P_0 is the set of all edges of G and empty paths;*
2. *Assume $r < n$. For a path p in P_{r+1} there are two cases:*

Correctness of middle and inner loop

Auxiliary definition

For $r \in \{0, 1, \ldots, n\}$ let P_r be the set of all shortest paths in the graph of R that only have intermediate nodes in the set $\{v_0, v_1, \ldots, v_{r-1}\}$. Then

Lemma

1. *P_0 is the set of all edges of G and empty paths;*
2. *Assume $r < n$. For a path p in P_{r+1} there are two cases:*
   - *v_r is not an intermediate node of p. Then p in P_r.*

Correctness of middle and inner loop

Auxiliary definition

For $r \in \{0, 1, \ldots, n\}$ let P_r be the set of all shortest paths in the graph of R that only have intermediate nodes in the set $\{v_0, v_1, \ldots, v_{r-1}\}$. Then

Lemma

1. *P_0 is the set of all edges of G and empty paths;*
2. *Assume $r < n$. For a path p in P_{r+1} there are two cases:*
   - *v_r is not an intermediate node of p. Then p in P_r.*
   - *v_r is an intermediate node of p. Then we can write the path p from e to d as the composition of a path u from e to v_r, and a path v from v_r to d, which are both in P_r;*

Correctness of middle and inner loop

Auxiliary definition

For $r \in \{0, 1, \ldots, n\}$ let P_r be the set of all shortest paths in the graph of R that only have intermediate nodes in the set $\{v_0, v_1, \ldots, v_{r-1}\}$. Then

Lemma

1. *P_0 is the set of all edges of G and empty paths;*
2. *Assume $r < n$. For a path p in P_{r+1} there are two cases:*
   - *v_r is not an intermediate node of p. Then p in P_r.*
   - *v_r is an intermediate node of p. Then we can write the path p from e to d as the composition of a path u from e to v_r, and a path v from v_r to d, which are both in P_r;*
3. *P_n is the set of all shortest paths in G.*

Complexity of Floyd's algorithm

Parameter
number of nodes n of graph

Complexity of Floyd's algorithm

Parameter

number of nodes n of graph

Time

a single operation: unit time 1

1. Pre: initialisation n^2
2. Loop:
   - assignment (Set N_{ij}): 1
   - inner loop (j) n times assignment: $n \cdot 1 = n$
   - middle loop (i) n times inner loop: $n \cdot n = n^2$
   - outer loop (r) n times middle loop: $n \cdot n^2 = n^3$

   copy matrix twice: $2n^2$
3. Post: –

total time: $3n^2 + n^3 \in O(n^3)$ (detailed later)

polynomial, cubic $O(n^3)$

Complexity of Floyd's algorithm

Parameter

number of nodes n of graph

Space

a single distance: unit space 1

1. matrices B and N of distances: $2 \cdot n^2 \cdot 1 = 2n^2$
2. variables i, j, r: 3

total space: $2n^2 + 3 \in O(n^2)$
polynomial, quadratic $O(n^2)$

Number of paths by matrix multiplication

Lemma

Let (V, E, src, tgt) be a directed multigraph having finitely many nodes- and edges with adjacency matrix $A_{ij} := \#(\{e \in E \mid src(e) = v_i \text{ and } tgt(e) = v_j\})$ for nodes $v_0, \ldots, v_{n-1}$.

Number of paths by matrix multiplication

Lemma

Let (V, E, src, tgt) be a directed multigraph having finitely many nodes- and edges with adjacency matrix $A_{ij} := \#(\{e \in E \mid src(e) = v_i \text{ and } tgt(e) = v_j)\}$ for nodes $v_0, \ldots, v_{n-1}$.

- *For $\ell \in \mathbb{N}$ and $i, j = 0, 1, \ldots, n-1$ is $(A^\ell)_{ij}$ the number of paths from v_i to v_j of length ℓ*

Number of paths by matrix multiplication

Lemma

Let (V, E, src, tgt) be a directed multigraph having finitely many nodes- and edges with adjacency matrix $A_{ij} := \#(\{e \in E \mid src(e) = v_i \text{ and } tgt(e) = v_j\})$ for nodes $v_0, \ldots, v_{n-1}$.

- *For $\ell \in \mathbb{N}$ and $i, j = 0, 1, \ldots, n-1$ is $(A^\ell)_{ij}$ the number of paths from v_i to v_j of length ℓ*

Proof.

How could we prove this? ∎

Relations motivation

Mathematical relations

used to model . . . relations

Example

- friend
- enemy
- taller than
- sitting next to
- superclass
- . . .

Relations definitions

Definition

$R \subseteq M \times M$ is a relation on M; R is

- reflexive, if for all $x \in M$, $(x, x) \in R$
- irreflexive, if for all $x \in M$, $(x, x) \notin R$
- symmetric, if for all $x, y \in M$
  $(x, y) \in R \Rightarrow (y, x) \in R$
- anti-symmetric, if for all $x, y \in M$
  $(x, y) \in R$ and $(y, x) \in R \Rightarrow x = y$
- transitive, if for all $x, y, z \in M$
  $(x, y) \in R$ and $(y, z) \in R \Rightarrow (x, z) \in R$

Relations definitions

Definition

$R \subseteq M \times M$ is a relation on M; R is

- reflexive, if for all $x \in M$, $(x, x) \in R$
- irreflexive, if for all $x \in M$, $(x, x) \notin R$
- symmetric, if for all $x, y \in M$
  $(x, y) \in R \Rightarrow (y, x) \in R$
- anti-symmetric, if for all $x, y \in M$
  $(x, y) \in R$ and $(y, x) \in R \Rightarrow x = y$
- transitive, if for all $x, y, z \in M$
  $(x, y) \in R$ and $(y, z) \in R \Rightarrow (x, z) \in R$

Remark

Homogeneous binary relations (heterogeneous n-ary relation $\subseteq M_1 \times \ldots \times M_n$)

Example

- $R_1 :=$ friend on set of people
- $R_2 :=$ enemy on set of people
- $R_3 :=$ taller than on set of people
- $R_4 :=$ sitting next to on set of people in classroom
- $R_5 :=$ being superclass of in Java program $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$?	$\times$?	$\checkmark$?	$\times$	$\times$
R_2					
R_3					
R_4					
R_5					

Example

- $R_1 :=$ friend on set of people
- $R_2 :=$ enemy on set of people
- $R_3 :=$ taller than on set of people
- $R_4 :=$ sitting next to on set of people in classroom
- $R_5 :=$ being superclass of in Java program $\emptyset$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$?	$\times$?	$\checkmark$?	$\times$	$\times$
R_2	$\times$?	$\checkmark$?	$\checkmark$?	$\times$	$\times$
R_3					
R_4					
R_5					

Example

- R_1 := friend on set of people
- R_2 := enemy on set of people
- R_3 := taller than on set of people
- R_4 := sitting next to on set of people in classroom
- R_5 := being superclass of in Java program $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	✓?	×?	✓?	×	×
R_2	×?	✓?	✓?	×	×
R_3	×	✓	×	✓	✓
R_4					
R_5					

Example

- R_1 := friend on set of people
- R_2 := enemy on set of people
- R_3 := taller than on set of people
- R_4 := sitting next to on set of people in classroom
- R_5 := being superclass of in Java program $\emptyset$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$?	$\times$?	$\checkmark$?	$\times$	$\times$
R_2	$\times$?	$\checkmark$?	$\checkmark$?	$\times$	$\times$
R_3	$\times$	$\checkmark$	$\times$	$\checkmark$	$\checkmark$
R_4	$\times$	$\checkmark$	$\checkmark$	$\times$	$\times$
R_5					

Example

- R_1 := friend on set of people
- R_2 := enemy on set of people
- R_3 := taller than on set of people
- R_4 := sitting next to on set of people in classroom
- R_5 := being superclass of in Java program $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$?	$\times$?	$\checkmark$?	$\times$	$\times$
R_2	$\times$?	$\checkmark$?	$\checkmark$?	$\times$	$\times$
R_3	$\times$	$\checkmark$	$\times$	$\checkmark$	$\checkmark$
R_4	$\times$	$\checkmark$	$\checkmark$	$\times$	$\times$
R_5	$\checkmark$?	$\times$?	$\times$	$\checkmark$	$\checkmark$

Example

- $R_1 := \{(0,0),(1,1),(2,2)\}$ on $\{0,1,2\}$
- $R_2 := \varnothing$ on $\{0\}$
- $R_3 := \{(0,0),(2,1)\}$ on $\{0,1,2\}$
- $R_4 := \{(0,0),(1,2),(2,1)\}$ on $\{0,1,2\}$
- $R_5 := \varnothing$ on $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	✓	×	✓	✓	✓
R_2					
R_3					
R_4					
R_5					

Example

- $R_1 := \{(0,0),(1,1),(2,2)\}$ on $\{0,1,2\}$
- $R_2 := \varnothing$ on $\{0\}$
- $R_3 := \{(0,0),(2,1)\}$ on $\{0,1,2\}$
- $R_4 := \{(0,0),(1,2),(2,1)\}$ on $\{0,1,2\}$
- $R_5 := \varnothing$ on $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$
R_2	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
R_3					
R_4					
R_5					

Example

- $R_1 := \{(0,0),(1,1),(2,2)\}$ on $\{0,1,2\}$
- $R_2 := \varnothing$ on $\{0\}$
- $R_3 := \{(0,0),(2,1)\}$ on $\{0,1,2\}$
- $R_4 := \{(0,0),(1,2),(2,1)\}$ on $\{0,1,2\}$
- $R_5 := \varnothing$ on $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	✓	×	✓	✓	✓
R_2	×	✓	✓	✓	✓
R_3	×	×	×	✓	✓
R_4					
R_5					

Example

- $R_1 := \{(0,0),(1,1),(2,2)\}$ on $\{0,1,2\}$
- $R_2 := \varnothing$ on $\{0\}$
- $R_3 := \{(0,0),(2,1)\}$ on $\{0,1,2\}$
- $R_4 := \{(0,0),(1,2),(2,1)\}$ on $\{0,1,2\}$
- $R_5 := \varnothing$ on $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$
R_2	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
R_3	$\times$	$\times$	$\times$	$\checkmark$	$\checkmark$
R_4	$\times$	$\times$	$\checkmark$	$\times$	$\times$
R_5					

Example

- $R_1 := \{(0,0),(1,1),(2,2)\}$ on $\{0,1,2\}$
- $R_2 := \varnothing$ on $\{0\}$
- $R_3 := \{(0,0),(2,1)\}$ on $\{0,1,2\}$
- $R_4 := \{(0,0),(1,2),(2,1)\}$ on $\{0,1,2\}$
- $R_5 := \varnothing$ on $\varnothing$

	reflexive	irreflexive	symmetric	anti-symmetric	transitive
R_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$
R_2	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
R_3	$\times$	$\times$	$\times$	$\checkmark$	$\checkmark$
R_4	$\times$	$\times$	$\checkmark$	$\times$	$\times$
R_5	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$

Closures

Definition

Let P be a property of relations. The P-closure of R is the least relation R' such that $R \subseteq R'$ and R' has property P.

Closures

Definition

Let P be a property of relations. The P-closure of R is the least relation R' such that $R \subseteq R'$ and R' has property P.

Remark

Only well-defined if it exists and is unique. E.g. if a relation is reflexive an irreflexive extension does not exist, and extending the empty relation on $\{a, b\}$ such that a and b are related in some way is not *unique* (two choices).

Closures

Definition

Let P be a property of relations. The P-closure of R is the least relation R' such that $R \subseteq R'$ and R' has property P.

Notations

$R^{=}$ is the reflexive closure of R, R^{+} its transitive closure, R^{*} its reflexive–transitive closure.

Closures

Definition

Let P be a property of relations. The P-closure of R is the least relation R' such that $R \subseteq R'$ and R' has property P.

Notations

$R^{=}$ is the reflexive closure of R, R^{+} its transitive closure, R^{*} its reflexive–transitive closure.

Example

The transitive closure of friendship and enemy relates everyone to everyone?, of being taller than and superclass are the relation themselves, and of sitting next to is sitting in the same row.

Algorithm of Warshall, transitive closure

Theorem

1. *Let R be a relation on a set M with n elements and let A be its adjacency matrix*
2. *The following algorithm with $\mathrm{O}(n^3)$ bit operations overwrites A with the adjacency matrix of the transitive closure of R*

For r from 0 to $n-1$ repeat:
Set $N = A$.
For i from 0 to $n-1$ repeat:
For j from 0 to $n-1$ repeat:
Set $N_{ij} = max(A_{ij}, A_{ir} \cdot A_{rj})$
Set $A = N$.

Example

The transitive closure of the relation $R = \{(0,2),(1,0),(2,1)\}$ on the set $\{0,1,2\}$ is

$$T = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$$

Example

The transitive closure of the relation $R = \{(0,2),(1,0),(2,1)\}$ on the set $\{0,1,2\}$ is

$$T = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$$

Adjacency matrix and first iteration ($r = 0$)

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad A_1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Second ($r = 1$) and third ($r = 2$) iteration

$$A_2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad A_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Relations as digraphs

Definition

Let R be a relation on a set M. Then the digraph of R is given by:

- the set of nodes M
- the set of edges R
- the functions $src((x, y)) = x$ and $tgt((x, y)) = y$

Graph notions apply to relation

Notions for graphs apply to a relation R via its graph.

Relations as digraphs

Graph notions apply to relation

Notions for graphs apply to a relation R via its graph.

Example

Let G be the graph of the relation R

- R is reflexive iff all nodes of G have a loop
- R is reflexive and transitive iff for every path from a to b there is an edge from a to b in G
- R is symmetric iff for every edge from a to b in G there is an edge from b to a
- ...

Relations as digraphs, Warshall as Floyd

Graph notions apply to relation

Notions for graphs apply to a relation R via its graph.

Theorem

Let R be a relation. R^* can be obtained from the distance matrix of R by mapping ∞ to 0 and natural numbers to 1.

Proof.

The correspondence holds for every stage of Warshall's algorithm applied to $R^=$ and Floyd's algorithm applied to the adjacency matrix of R. ∎

Functions as relations

Definition

A function on M is a relation R on M such that

1. for all $x \in M$, there exists y such that $x \; R \; y$ (totality)
2. for all $x, y, y' \in M$ if $x \; R \; y$ and $x \; R \; y'$ then $y = y'$, i.e. R relates uniquely.

we then write $R(x)$ to denote y.

Functions as relations

Definition

A function on M is a relation R on M such that

1. for all $x \in M$, there exists y such that $x\ R\ y$ (totality)
2. for all $x, y, y' \in M$ if $x\ R\ y$ and $x\ R\ y'$ then $y = y'$, i.e. R relates uniquely.

we then write $R(x)$ to denote y.

Example

- The squaring function on natural numbers is the relation $\{(0,0),(1,1),(2,4),(3,9),(4,16),\ldots\}$.
- Taking the square root is not a function on natural numbers, since, e.g., the square root of 2 is not a natural number (existence fails)
- Taking the square root is not a function on the real numbers either, since, e.g., both -2 and 2 are square roots of 4 (uniqueness (also) fails)

Functions as relations

Definition

A function on M is a relation R on M such that

1. for all $x \in M$, there exists y such that $x \; R \; y$ (totality)
2. for all $x, y, y' \in M$ if $x \; R \; y$ and $x \; R \; y'$ then $y = y'$, i.e. R relates uniquely.

we then write $R(x)$ to denote y.

Specification of functions

A function is said to be defined by some specification this expresses that there exists a unique relation satisfying the specification and the relation is a function.

Example

The function f on natural numbers defined by

- $f(n) = n$? ✓or $f(n) = -1$? $\times$or $f(n) = f(n)$? $\times$
- $f(0) = 10$ and $f(1) = 2$? $\times$or $f(0) = 0$ and $f(n+1) = f(n)$? ✓ ...

Representing and specifying functions

Remark

Relation can be represented by graphs. Every node of the graph of a function on M has out-degree exactly 1. If it is injective, then every node has in-degree at most 1. If it is bijective, then every node has both in- and out-degree 1.

Representing and specifying functions

Remark

Relation can be represented by graphs. Every node of the graph of a function on M has out-degree exactly 1. If it is injective, then every node has in-degree at most 1. If it is bijective, then every node has both in- and out-degree 1.

Remark

Functions in functional programming and functions in the mathematical sense do not always correspond.

Representing and specifying functions

Remark

Relation can be represented by graphs. Every node of the graph of a function on M has out-degree exactly 1. If it is injective, then every node has in-degree at most 1. If it is bijective, then every node has both in- and out-degree 1.

Remark

Functions in functional programming and functions in the mathematical sense do not always correspond.
The function that maps a program to 1 if it halts on every input, and to 0 otherwise, is a mathematical function, but not a Haskell function.
There cannot even exist a (Haskell) program for this mathematical function.

Representing and specifying functions

Remark

Relation can be represented by graphs. Every node of the graph of a function on M has out-degree exactly 1. If it is injective, then every node has in-degree at most 1. If it is bijective, then every node has both in- and out-degree 1.

Remark

Functions in functional programming and functions in the mathematical sense do not always correspond.
The function that maps a program to 1 if it halts on every input, and to 0 otherwise, is a mathematical function, but not a Haskell function.
There cannot even exist a (Haskell) program for this mathematical function.
Defining `f x = f x` is allowed in Haskell (does type-check) but does not define a mathematical function.