GB13624 - Computer Science in English B

Lecture 2 – Proofs, Part 2 (Induction and Sets)

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Science

2024-10-09

Last updated October 8, 2024

Lecture 2 – Outline

In the first lecture, we covered the basic concepts of proofs. In this lecture, we first introduce a very important proof method, **Induction.**

After that, we discuss the concepts of **state machines** and **sets.**

- **Section 1:** Proof by Induction – Chapter 5
- **Section 2:** State Machines – Chapter 5
- **Section 3:** Sets and Relations – Chapter 4

I highly recommend that you also study chapters 6 and 7 to learn more advanced topics related to Induction and Sets.

Part 1: Induction

An initial induction (1/2)

Imagine that I want to color the natural numbers ($\mathbb{N} \geq 0$), using the following rules:

- Number 0 is red
- Any integer next to a red number is also red

Using these rules, can you imagine how the set $\mathbb{N}$ looks like?

An initial induction (2/2)

Result: 0,1,2,3,4,...

The "rule of reds" gives us a general idea of induction:

- $R(0)$ is True
- $R(0) \rightarrow R(1); R(1) \rightarrow R(2); R(2) \rightarrow R(3); \ldots$
- $R(n) \rightarrow R(n+1)$ for every $n \in \mathbb{N}$

Induction can be used to prove a predicate that depends on some $n \in \mathbb{N}$.

$$\frac{R(0), R(n) \rightarrow R(n+1), n \in \mathbb{N}}{\forall n, R(n)}$$

Example of proof by Induction

Let's prove that:

$$P(n) : 1 + r + r^2 + r^3 + \ldots + r^n = \frac{r^{n+1} - 1}{r - 1}, r \neq 1, \forall n \in \mathbb{N}$$

Remember the modus ponens rule for induction:

$$\frac{P(0), P(n) \rightarrow P(n+1), n \in \mathbb{N}}{\forall n, P(n)}$$

To prove the bottom part, we need to prove all statements in the top part are true.

- First Step: Prove $P(0)$
- Second Step: Prove $P(n) \rightarrow P(n+1)$

Example of proof by Induction

Proof of $P(n), \forall n \in \mathbb{N}$ by induction on n.

First Step: Prove $P(0)$

- $P(0)$, left side: $r^0 = 1$
- $P(0)$, right side: $\frac{r^{0+1}-1}{r-1} = \frac{r-1}{r-1} = 1$

Second Step: Prove $P(n) \rightarrow P(n+1)$

- $P(n+1)$, left side: $1 + r + r^2 + \ldots + r^n + r^{n+1}$, which is equal to $P(n) + r^{n+1}$
- Because $P(n)$ is True, $P(n) + r^{n+1} = \frac{r^{n+1}-1}{r-1} + r^{n+1} = \frac{(r^{n+1}-1)}{r-1} + \frac{(r^{n+1}(r-1))}{r-1}$
- Algebra: $\frac{(r^{n+1}-1)+(r^{n+1}(r-1))}{r-1} = \frac{r^{n+1}-1+r^{n+2}-r^{n+1}}{r-1} = \frac{r^{n+2}-1+(r^{n+1}-r^{n+1})}{r-1}$
- $\frac{r^{n+2}-1}{r-1} = \frac{r^{(n+1)+1}-1}{r-1}$, which is the right side of $P(n+1)$

$\square$

Review: Proof Template for Induction

Proof by induction on n
Proof hypothesis: $P(n) = \ldots$ for all $n \in \mathbb{N}.n \geq 0$

First we prove $P(0)$.
$\ldots$ *(calculate that P(0) is True)*
$\ldots$

Second we prove that $\forall n \geq 0, P(n) \rightarrow P(n+1)$
$\ldots$ *(calculate P(n+1) using P(n))*
$\ldots$

This completes the proof that $P(n)$ for all $n \in \mathbb{N}$ $\square$

The Statue Park

A more complex proof by induction

The university is making a new park with the following rules:

- The park is square, with side 2^n;
- In the middle of the park, there is a statue, size 1×1;
- Other than that, the park is made of L-shaped tiles, with size $3m^2$;

How can we prove that it is possible to build this park for any n?

The Statue Park

Drawing Proof

Remember the rule of induction:

- Prove that $P(0)$ is true.
- Assume that $P(n)$ is true, then prove that $P(n) \implies P(n+1)$

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**
  - Consider the set of n+1 horses: $H = h_1, h_2, \ldots, h_n, h_{n+1}$

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**
  - Consider the set of n+1 horses: $H = h_1, h_2, \ldots, h_n, h_{n+1}$
  - Subset A ($h_1, h_2, \ldots, h_n$): has n horses, so all horses have the same color.

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**
  - Consider the set of n+1 horses: $H = h_1, h_2, \ldots, h_n, h_{n+1}$
  - Subset A $(h_1, h_2, \ldots, h_n)$: has n horses, so all horses have the same color.
  - Subset B $(h_2, \ldots, h_n, h_{n+1})$: also has n horses, so all horses have the same color.

BAD Proof by induction: All horses are of the same color

$P(n)$ ::= for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**
  - Consider the set of n+1 horses: $H = h_1, h_2, \ldots, h_n, h_{n+1}$
  - Subset A $(h_1, h_2, \ldots, h_n)$: has n horses, so all horses have the same color.
  - Subset B $(h_2, \ldots, h_n, h_{n+1})$: also has n horses, so all horses have the same color.
  - Horse h_2 is in subset A **and** in subset B, so subset A and B have the same color.

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**
  - Consider the set of n+1 horses: $H = h_1, h_2, \ldots, h_n, h_{n+1}$
  - Subset A $(h_1, h_2, \ldots, h_n)$: has n horses, so all horses have the same color.
  - Subset B $(h_2, \ldots, h_n, h_{n+1})$: also has n horses, so all horses have the same color.
  - Horse h_2 is in subset A **and** in subset B, so subset A and B have the same color.
- Since we showed that P(n+1) is true if P(n) is true, then all horses for any group size have the same color.

BAD Proof by induction: All horses are of the same color

$P(n) ::=$ for any set with exactly n horses, all horses have the same color.

- **Prove P(1)**: For any set with one horse, all horses have the same (one) color.
- **Assume P(n) is true:** For any set with n horses, all horses have the same color.
- **Show that $P(n) \implies P(n+1)$:**
  - Consider the set of n+1 horses: $H = h_1, h_2, \ldots, h_n, h_{n+1}$
  - Subset A ($h_1, h_2, \ldots, h_n$): has n horses, so all horses have the same color.
  - Subset B ($h_2, \ldots, h_n, h_{n+1}$): also has n horses, so all horses have the same color.
  - Horse h_2 is in subset A **and** in subset B, so subset A and B have the same color.
- Since we showed that P(n+1) is true if P(n) is true, then all horses for any group size have the same color.

QUIZ: What is wrong with this proof?

What is wrong with the horse proof?

The induction step when we show that $P(n) \implies P(n+1)$ is not valid.

- The implication proof depends on "h_i belongs to subsets A and B".
- But is this ALWAYS true?
  - When $n+1=2$, The $n+1$ set is $\{h_1, h_2\}$, set $A = h_1$, set $B = h_2$;
  - But in this case, **there is no h_i that is common to A and B**!
- So the implication proof is not valid when $P(2)$.

Note that this is the only problem with the proof!

Strong Induction

- In regular induction, you assume P(n) to show P(n+1)
- In strong induction, you assume P(0), P(1), P(2) . . . P(n), and use all of them to show P(n+1)

Strong Induction Example: Stacking Game

- Begin with a stack of 10 blocks
- Divide it in two (a, b): for example, 2 and 8 blocks.
- For each division, you get $a \times b$ points: 16 points
- Repeat with the new stacks until all stacks have 1 block.

Which of the two strategies below give you more points?

- Simple strategy: $(1 \times 9) + (1 \times 8) + (1 \times 7) + (1 \times 6)$...
- CS recursive strategy: $(5 \times 5) + (2 \times 3) + (2 \times 3)$, ...

Strong Induction Example: Stacking Game

- Begin with a stack of 10 blocks
- Divide it in two (a, b): for example, 2 and 8 blocks.
- For each division, you get $a \times b$ points: 16 points
- Repeat with the new stacks until all stacks have 1 block.

Which of the two strategies below give you more points?

- Simple strategy: $(1 \times 9) + (1 \times 8) + (1 \times 7) + (1 \times 6)$... 45 points!
- CS recursive strategy: $(5 \times 5) + (2 \times 3) + (2 \times 3)$, ...

Strong Induction Example: Stacking Game

- Begin with a stack of 10 blocks
- Divide it in two (a, b): for example, 2 and 8 blocks.
- For each division, you get $a \times b$ points: 16 points
- Repeat with the new stacks until all stacks have 1 block.

Which of the two strategies below give you more points?

- Simple strategy: $(1 \times 9) + (1 \times 8) + (1 \times 7) + (1 \times 6)$... 45 points!
- CS recursive strategy: $(5 \times 5) + (2 \times 3) + (2 \times 3)$, ... 45 points!

Proof: All strategies have the same score (Part I)

Let us prove by strong inductions that **ALL** strategies for the stack game with “n” blocks have the same final score:

$$C(n) = \frac{n(n-1)}{2}$$

Base Cases: 0, 1

- When the stack has 0 blocks, I have no moves, so 0 points.
- When the stack has 1 block, I have no moves, so 0 points.

$$C(0) = \frac{0(0-1)}{2}, C(1) = \frac{1(1-1)}{2} = 0$$

Proof: All strategies have the same score (Part II)

Inductive Case $C(n+1)$
By strong induction, we assume that all $C(0) \ldots C(n)$ are true.

1. A stack with $n+1$ blocks can be split into two: k and $n+1-k$
2. The score is: $C(n+1) = (k \times (n+1-k)) + C(k) + C(n+1-k)$
3. Using the strong inductive assumption: $\forall m \leq n, C(m) = \frac{m(m-1)}{2}$
4. Transforming (2): $C(n+1) = \frac{2k(n+1-k)}{2} + \frac{k(k-1)}{2} + \frac{(n+1-k)(n-k)}{2}$

... You can finish the calculation from here ;-)

Part 2: State Machines

What are state machines?

State machines are used to represent "step-by-step" processes. They contain:

- A description of each possible state in the machine;
- How the machine transition from one state to another;

State machines are often used to describe algorithms, programs, logic circuit, decision processes, etc.

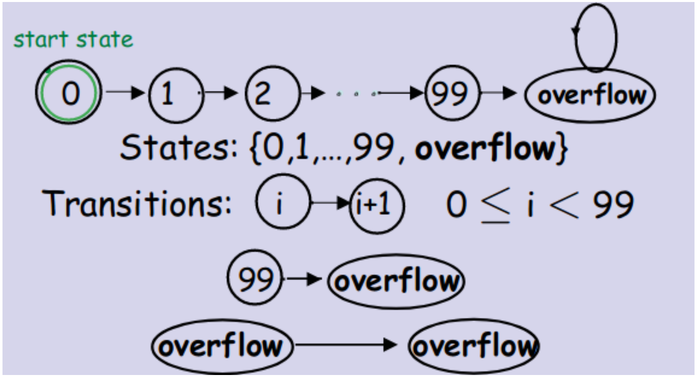
State machines are a formal description that can be used to prove the correctness of an algorithm.

State Machines: Simple Example

State machine for counting from 0 to 99:

- **States:** 0 to 99, overflow.
- **Start State:** 0
- **Transitions:**
  $i \rightarrow i+1$ if $i < 99$
  $99 \rightarrow$ overflow
  overflow $\rightarrow$ overflow

Note how we can represent the State Machine many different ways.



State Machine for Proofs

Robot 1.0

Imagine a robot moving forwards and backwards on a street. The robot has two speeds:

- The robot can move exactly **five squares** forwards.
- The robot can move exactly **three squares** backwards.

If the robot starts from position 0, is it possible for it to reach position 4?

What do you think? Can you show this with a state machine?

State Machine for Proofs

Robot 1.1

Imagine a robot moving forwards and backwards on a street. The robot has two speeds:

- The robot can move exactly **nine squares** forwards.
- The robot can move exactly **three squares** backwards.

If the robot starts from position 0, is it possible for it to reach position 4?

What do you think? Is the state machine very different in this case?

State Machine for Proofs

Preserved Invariants

Preserved Invariants are propositions that continue to always true after **any** transition of the state machine.

We can use preserved invariants to prove which squares the robots can/cannot reach.

Robot 1.0

The position of robot 1.0 is always: $s_0 + 5a - 3b$

Robot 1.1

The position of robot 1.1 is always: $s_0 + 9a - 3b$

- $s_0 + 9a - 3b = s_0 + 3(3a - b)$
- The position of robot 1.1 is always s_0 plus a multiple of 3; (Preserved Invariant)
- So it is impossible for robot 1.1 to reach 4 from 0.

Induction with Preserved Invariants

Preserved Invariants can be used together with Proof by Induction to prove things about state machines:

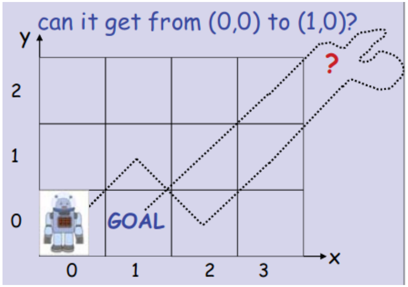
- Prove that $P(s)$ is a preserved invariant. This means that if $P(s)$ is true for some state s, then it will continue to be true after any transition.
- Prove that $P(s)$ is true for the initial state, s_0.
- Conclude that $P(s)$ is always true for the entire state machine.

If $P(s)$ is a "correctness condition" of an algorithm, this method can be used to prove that an algorithm is correct.

State Machine for Proofs

Robot 2.0

Robot 2.0 can move on the diagonals of $\mathbb{Z}^2$: (+1, +1), (-1,-1), (+1,-1), (-1,+1). Starting from (0,0), is it possible for the robot to reach position (1,0)?



Practice: Construct a state machine for this robot, and define the relevant preserved invariant about its position.

State Machine for Proofs

Robot 2.0 – Solution

We can show that a preserved invariant of robot 2.0 is that "the parity (odd or even) of the sum of its coordinates never changes":

- P(0,0) is true (0+0 is even).
- The steps of the robot are:
  - $+1+1=+2$: even + 2 is still even;
  - $-1-1=-2$: even - 2 is still even;
  - $+1-1=0$: even + 0 is still even;
  - $-1+1=0$: even + 0 is still even;

So we can see that the parity of the position is a preserved invariant. Because the parities of (0,0) and (1,0) are different, it is impossible for robot 2.0 to go from (0,0) to (1,0).

State Machines for Proofs

Fast Exponentiation

In the original MIT OCW website there is a lecture video about using State Machines to prove the "Fast Exponentiation" algorithm (1.9.1). Please see that video by yourself.

Important Points: To prove that an algorithm is correct, we need to prove the following:

1. Prove that if the algorihm is in a "correct state", it will always stay in the correct state;
   - The correct state is a "Preserved Invariant";
2. Prove that the algorithm can reach a "correct state" from the initial position;
3. Prove that if the algorithm is in a "correct state", it will stop at some point.
   - In other words, it is not an infinite loop.
   - You can usually prove this by showing that some variable in the state always decreases.

Part 3: Sets and Relations

Mathematical Set

Mathematical Sets are useful when talking about proofs.

We have already used some sets:

- $\mathbb{N}$ – Set of natural (non negative) numbers;
- $\mathbb{Z}$ – Set of integer numbers;
- $\mathbb{R}$ – Set of real numbers;

Characteristics of Mathematical Sets

- Sets can mix different "types" of objects:
  - {7, “Aranha”, $\pi/2$, TRUE}
- Sets do not have a concept of "order":
  - {7, “Aranha”, $\pi/2$, TRUE} is the same as
  - {TRUE, 7, $\pi/2$, “Aranha”}
- Sets do not contain duplicates:
  - {7, π} = {7, π, 7}
- Sets can be Infinite: $\mathbb{N}, \mathbb{R}, \ldots$
- The empty set: $\varnothing = \{\}$

Set Membership

The fundamental property of a set is membership, represented by the symbol $\in$.

Set A = {7, TRUE, π}

- $7 \in A$
- 7 is an element of A,
- 7 is in A,
- $3 \notin A$

Set B = {7, $\mathbb{Z}$, 3}

- $7 \in B$
- $7 \in \mathbb{Z}$
- $\mathbb{Z} \in B$
- $1 \notin B$

Note that membership is not recursive!

1 belongs to the set $\mathbb{Z}$, but 1 does not belong to the set B.

Set Builder Notation

We can use Predicates to construct sets. The set is composed of all value for which the predicate is true. For example:

- $A ::= \{n \in \mathbb{N} | n \text{ is a prime and } n = 4k + 1 \text{ for } k \in \mathbb{Z}\}$
- $B ::= \{x \in \mathbb{R} | x^3 - 3x + 1 > 0\}$
- $C ::= \{a^2 | a = b^2, a, b \in \mathbb{N}\}$

Please list a few elements of each set. This looks like python's comprehension lists!

Set: Subsets

Subset

- $A \subset B$ means that every element of A is also an element of B
- $A \subset B \equiv \forall x, x \in A \rightarrow x \in B$
- $\mathbb{Z} \subset \mathbb{R}, \mathbb{R} \subset \mathbb{C}, \{3\} \subset \{5, 3, 7\}$

Important!

- $A \subset A$
- Every set contains the empty subset: $\forall X, \varnothing \subset X$

Difference between Membership and Subset

Membership ($\in, \notin$) indicates if one member is part of a set. Subset ($\subset, \not\subset$) indicates if one set contains the members of other sets.

- $3 \in \{3, 5, 6\}$
- $3 \not\subset \{3, 5, 6\}$

- $\{3\} \subset \{3, 5, 6\}$
- $\{3\} \notin \{3, 5, 6\}$

- $\{3\} \in \{5, 6, \{3\}\}$

Power Set

The Power Set of A is a special set composed of ALL subsets of A.

$$POW(A) = \forall x \subset A, x \in POW(A)$$

For example: $POW(\{T, F\}) = \{\{T\}, \{F\}, \{T, F\}, \varnothing\}$

Check the following: If a set A is a subset of B, then A is a member of POW(B)

$$\mathbb{N} \subset \mathbb{R}, \mathbb{N} \notin \mathbb{R}, \mathbb{N} \in POW(\mathbb{R})$$

Operations on Sets

Finally, You should be familiar with the regular operations on sets:

- Union: $A \cup B \rightarrow x \in A \vee x \in B$
- Intersection: $A \cap B \rightarrow x \in A \wedge x \in B$
- Subtraction: $A - B \rightarrow x \in A \wedge x \notin B$
- Complement: $\overline{A} = D - A$, where D is the domain (the "everything" set or "parent" set of interest).

Example of proofs using sets (1)

Prove that the empty set is a subset of every set.

Proof by construction:

1. $A \subset B$ means that $\forall x, x \in A \to x \in B$
2. If $A = \varnothing$ then $x \in A$ is FALSE for $\forall x$, so we can replace "$\forall x \in A$" with FALSE in **(1)**
3. The statement FALSE $\to x \in B$ is always TRUE.
   (remember that FALSE $\to X$ is always TRUE)
4. Therefore, $\varnothing \subset B$ is TRUE $\forall B$

□

Sets and Proofs Example (2)

$$A \cup (B \cap C) \iff (A \cup B) \cap (A \cup C)$$

Proof that Union and Intersection are Distributive.

Proof by sequence of "IFF"s:

1. $x \in A \cup (B \cap C)$ **iff**
2. $x \in A \vee x \in (B \cap C)$ **iff** (definition of union)
3. $x \in A \vee (x \in B \wedge x \in C)$ **iff** (definition of intersection)
4. $(x \in A \vee x \in B) \wedge (x \in A \vee x \in C)$ **iff** (distributive prop.)
5. $(x \in A \cup B) \wedge (x \in A \cup C)$ **iff** (definition of union)
6. $x \in (A \cup B) \cap (A \cup C)$ (defintion of intersection)

□

Sequences and Sets

A sequence is another kind of mathematical object. It is similar to a set, but different in important ways.

- Sequences are represented by parenthesis: $(1, 2, 1), (T, F)$
- Sequences can have repeated elements: (T, T, T, F, F)
- The order of elements in a sequence is important: $(0, 1) \neq (1, 0)$

We often see sequences as the products of multiplication between sets:

- All binary sequences with 3 elements: $\{0, 1\}^3 = (0, 0, 0), (0, 0, 1), \dots (1, 1, 1)$
- All pairs of two natural numbers: $\mathbb{N}^2 = (0, 0), (0, 1), (0, 2), (0, 3) \dots (100, 7) \dots$

Definition of Binary Relations

Binary Relations define an association of elements from one set (the **domain**) to another set (the **co-domain**). We see (binary) relations in many different situations:

- Functions are a special case of binary relations: f(x) = y.
  - A function associates the set of inputs with the sets of outputs;
- Operations such as set membership can be expressed as binary relations.
  - For example, the predicate $P(x) : x$ is prime defines a binary relation from $\mathbb{N}$ to {TRUE, FALSE}
- "Relational Databases" (for example, SQL) are also based on the idea of binary relations
  - Key of X, member of a table, joint key, etc;

Example of Binary Relation

Relation $P(x)$: x is prime:

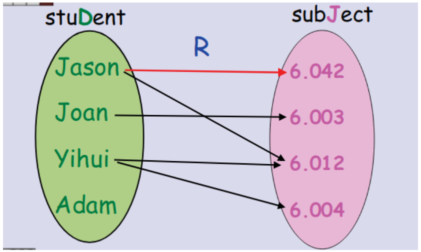
- Domain: $x \in \mathbb{N}$
- Co-Domain: $P(x) \in \{True, False\}$
- Examples:
  $P(1) = False$
  $P(3) = True$
  $P(13) = True$
  $P(20) = False$

It is common to represent binary relations as a graph between the domain and the co-domain.

Binary Relations as a Graph

Let's consider the binary relation **R** from a set of students **(D)** that are registered to a set of courses **(J)**.

- Domain: Set of students;
- Co-domain: Set of courses;
- Vertices: Domain and Co-domain are represented as vertices;
- Edges: Directed edges from Domain to Co-domain;



Notation to describe the relation:

- $R(\text{Jason}) = \{6.042, 6.012\}$
- Jason R 6.042
- $R(\{\text{Jason, Yihui}\}) = \{6.042, 6.012, 6.004\}$

Relations and Inverse Relations

If we think of a binary relation as a directed graph, the Inverse Relation is the relation defined by the same graph when the edges are reversed:

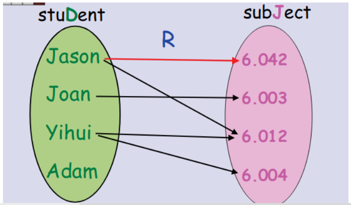
Relation R:

$$R(X) ::= j \in J | \exists d \in X, dRj$$

Reverse Relation R^{-1}:

$$R^{-1}(Y) ::= d \in S | \exists j \in Y, dRj$$

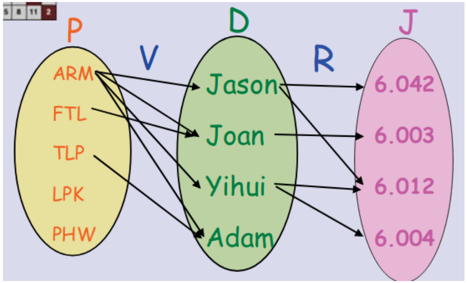
- $R(Jason) = \{6.042, 6.012\}$
- $R^{-1}(6.012) = \{Jason, Yihui\}$



Composite Relations

If we have a relation V from set P to set D, and a relation R from set D to set J, then we can define a composite relation $R \circ V$ from set P to set J (we can also use $R(V)$).

- $R(V(X))$ or $(R \circ V)(X)$
- $R(V(\text{FTL})) = \{6.003\}$
    - professor FTL super**V**ises Joan;
    - Joan is **R**egistered to class 6.003;
    - professor FTL **VR** 6.003 (superregister?)



Types of Binary Relations

We can classify a binary relation based on the number of degrees (arrows) in the relation graph. Imagine a relation R from X to Y (i.e.: $R(X) = Y$)

Classification of R based on Y:

- **Surjection**: Every element in Y has ≥ 1 in-arrows. (Every Y has **one or more** X)
- **Injection**: Every element in Y has ≤ 1 in-arrows. (Every Y has **one or less** X)

Classification of R based on X:

- **Total**: Every element in X has ≥ 1 out arrows. (Every X has **one or more** Y)
- **Function**: Every element in X has ≤ 1 out arrows. (Every X has **one or less** Y)

An important definition:

- **Bijection**: Every element of X has **exactly one** element of Y, **and vice versa.**

Types of Binary Relations: Examples

Example 1: $g : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}, g(x,y) = 1/(x-y)$

- This is a **function**, because each pair (x, y) has at most one output.
- This is not **total**, because not every pair (x, y) has an output: $(x = y)$ has no output.

Example 2: $g_o : \mathbb{R} \times \mathbb{R} - \{x, y | x = y\} \rightarrow \mathbb{R}, g_o(x,y) = 1/(x-y)$

- g and g_o are similar relations, but defined on different domains.
- The domain of g_o removes all (x, y) where $x = y$
- Because of this, g_0 is **function** (every pair has at most one output) and **total** (every pair has at least one output).

Slide Credits

Image Credits I

1. Relation Graph Image from MIT OCS