

Functions and Relations ^{*a*}

Teodor Rus

rus@cs.uiowa.edu

The University of Iowa, Department of Computer Science

^{*a*}These lecture notes have been developed by Teodor Rus. They are copyrighted materials and may not be used in other course settings outside of the University of Iowa in their current form or modified form without the express written permission of one of the copyright holders. During this course, students are prohibited from selling notes to or being paid for taking notes by any person or commercial firm without the express written permission of one of the copyright holders.

Functions (informal)

- A function is a mathematical object that sets up an input-output relationship.
- A function takes an input and produces an output.
However note: the same input always produces the same output.
- If f is a function whose output value is b when the input value is a , we write $f(a) = b$.

Observations

- A function is also called a *mapping*;
- If $f(a) = b$ we say that f maps a to b and may write $a \xrightarrow{f} b$.
- If $a \xrightarrow{f} b$ is a function then the mechanism that determines b from the given a is referred to as the **algorithm for function computation**.

Question: is there any difference between a function and the algorithm which computes that function?

Examples:

- The absolute value function *abs* takes a number x as input and returns x if $x > 0$, $-x$ if $x < 0$, and 0 if $x = 0$.

Example: $abs(2) = abs(-2) = 2$

- The function *add* takes two numbers and returns their sum. Thus, $add(2, -2) = 0$
- Function *sub* takes two numbers and returns their difference. Thus, $sub(2, -2) = 4$

Domain and range

- The set of possible inputs to a function is called its *domain*;
- The outputs of a function come from a set called *range*;
- **Notation:** to denote that f is a function with domain D and range R we write $f : D \rightarrow R$

Examples:

1. $abs : \mathbb{Z} \rightarrow \mathbb{Z}$,
2. $add : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$,
3. $sub : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$

Describing functions

- By giving the computation rule implemented by the function;
- By a table that lists all possible inputs and give the output for each input.

Example:

$f : \{0, 1, 2, 3, 4\} \rightarrow \{0, 1, 2, 3, 4\}$ is described in Figure 1

Example description

n	$f(n)$
0	1
1	2
2	3
3	4
4	0

Figure 1: Example function description

Note: this function adds 1 to its input and then outputs the result modulo 5.

Modular arithmetic

- A number n modulo m is the remainder after division of n by m , i.e., if $n = qm + r$ then $n \text{ modulo } m = r$
- When we do modular arithmetic we define $\mathcal{Z}_m = \{0, 1, 2, \dots, m - 1\}$

Example:

$0 \text{ modulo } 5 = 0$, $1 \text{ modulo } 5 = 1$, $2 \text{ modulo } 5 = 2$, $3 \text{ modulo } 5 = 3$,
 $4 \text{ modulo } 5 = 4$, $5 \text{ modulo } 5 = 0$, $6 \text{ modulo } 5 = 1$, etc.

Note: f in Figure 1 has the form $f : \mathcal{Z}_5 \rightarrow \mathcal{Z}_5$

Application

Minute hand of a clock face counts modulo 60

More on function description

- When the domain of a function f is a Cartesian product of two sets, a two-dimensional table may be used to describe f .
- Table in Figure 2 describes function $g : \mathcal{Z}_4 \times \mathcal{Z}_4 \rightarrow \mathcal{Z}_4$ that performs the addition modulo 4.

Question: is it always possible to describe a function by table?

Function description

A function of two-arguments described by a two-dimensional table:

g		0	1	2	3
0		0	1	2	3
1		1	2	3	0
2		2	3	0	1
3		3	0	1	2

Figure 2: Addition modulo 4

Formal characterization

Definition: a function $f : A \rightarrow B$ is a binary relation $f \subseteq A \times B$ (i.e., a set of tuples) that satisfies the property:

$$(x, y), (x, z) \in f \Rightarrow y = z$$

Formally:

$$f = \{(x, y) \in A \times B \mid (x, y), (x, z) \in f \Rightarrow y = z\}.$$

Questions:

1. What is the property of the informal concept of a function which when formalized led to the mathematical definition of a function?
2. What is the set theory axiom that justifies this definition?

Terminology

- **Injection:** if $f : A \rightarrow B$ is a function and $\text{dom}(f) = A$ then f is also called an injection.
- **Surjection:** if $f : A \rightarrow B$ is a function and $\text{ran}(f) = B$ then f is also called a surjection.
- **Bijection:** if $f : A \rightarrow B$ is an injection and a surjection then f also called a bijection.

Note

- When the domain of a function f is $A_1 \times A_2 \times \dots \times A_k$ for some sets A_1, A_2, \dots, A_k , the input to f is a k -tuple (a_1, a_2, \dots, a_k) and $a_i, i = 1, 2, \dots, k$, are called the *arguments* of f ;
- A function f with k arguments is called k -ary function and k is called the arity of f ;
- If $k = 1$, f is called a *unary function*; if $k = 2$, f is called a binary function.

Notations

- **Infix notation:** binary function symbol placed between its two arguments;
Example: familiar binary functions such as addition, subtraction, multiplication $a + b$, $a - b$, $a \times b$.
- **Prefix notation:** function symbol precedes the arguments;
Example: $+(a, b)$, $-(a, b)$, $\times(a, b)$.
- **Postfix notation:** function symbol follows the arguments.
Example: $(a, b)+$, $(a, b)-$, $(a, b)\times$.

Question: do we need parentheses when we use prefix or postfix notation ? Justify your answer.

Observation

A special case of infix notation used in PL is the *distributed notation* where function symbol embraces function arguments, as in:

if E then S else S endif

Predicates or properties

- A predicate or property is a function whose range is the set $\{true, false\}$

Example: $even : \mathcal{N} \rightarrow \{true, false\}$ with:

(a) $even(n) = true$ if $n = 2k$ for some k ;

(b) $even(n) = false$ if $n = 2k + 1$ for some k .

- A property whose domain is a set of k -tuples $A \times \dots \times A$ is called a *relation*, a k -ary relation, or a k -ary relation on A

Example relations

- A common case of k -ary relations are 2-ary relations called *binary relations*
- Expressions involving binary relations, R , usually use infix-notation, i.e., aRb rather than $R(a, b)$ or $(a, b)R$
- Example binary relations are:
 - equality* denoted $=$,
 - less than*, denoted $<$,
 - less than or equal*, denoted \leq ,
 - greater than*, denoted $>$,
 - greater than or equal*, denoted \geq

Facts

If R is a k -ary relation, the statement $R(a_1, \dots, a_k)$ means that $R(a_1, \dots, a_k) = \text{true}$;

Similarly for binary relations.

A game as a relation

In a children's game Scissors-Paper-Stone the two players simultaneously select a member of the set $\{SCISSORS, PAPER, STONE\}$ and indicate their selections with hand signals

- If the two selections are the same, the game starts over;
- If the selections differ, one player win according to the relation *beats* in Figure 3.

Relation representation

<i>beats</i>	SCISSORS	PAPER	STONE
SCISSORS	false	true	false
PAPER	false	false	true
STONE	true	false	false

Figure 3: Game rule representation

Facts

- Sometimes describing predicates with sets instead of functions is more convenient;
- The predicate $P : D \rightarrow \{true, false\}$ may be written (D, S) , i.e. $D \times S$, where $S = \{a \in D \mid P(a) = true\}$.

Example: relation *beats* may be written:

$\{(SCISSORS, PAPER), (PAPER, STONE), (STONE, SCISSORS)\}$

Equivalence relation

- An equivalence relation captures the notion of two objects being equal in some feature
- A binary relation R is an equivalence relation if R satisfies:
 1. R is reflexive, i.e., for every x , xRx
 2. R is symmetric, i.e., for every x, y , xRy iff yRx
 3. R is transitive, i.e., for every x, y, z , xRy and yRz implies xRz

An equivalence on \mathcal{N}

- Define an equivalence relation on \mathcal{N} written \equiv_7 by:

for $i, j \in \mathcal{N}$, we say that $i \equiv_7 j$ if $i - j$ is a multiple of 7.

Note: this is an equivalence because:

1. It is reflexive, because $i - i = 0$, which is a multiple of 7
2. It is symmetric, because if $i - j$ is a multiple of 7 then $j - i$ is a multiple of 7 and vice-versa
3. It is transitive, because whenever $i - j$ is a multiple of 7 and $j - k$ is a multiple of 7 then $i - k = (i - j) + (j - k)$ which is a sum of two multiple of 7, i.e., $i - k$ is a multiple of 7