MATHEMATICS FOR INFORMATION SCIENCE

NO.1 WHILE PROGRAM

Tatsuya Hagino

hagino@sfc.keio.ac.jp

Slides URL

https://vu5.sfc.keio.ac.jp/slide/

Course Summary

- A program can be seen as a mathematical function which calculates output value for a given input. In this lecture, we will look into the property of functions which correspond to programs.
- Firstly, in order to understand what we can calculate using programs, we compare three models of programs: recursive functions, Turing machines and lambda calculi. We will show that those three models are equivalent.
- Secondly, we will study complete partial order sets which give the model of lambda calculi and programs.
- Thirdly, in order to understand data types of programs, we will look into category theory which is the abstraction of functions and has an ability to reveal the beauty behind data types.

Course Schedule

1. While Program
2. Primitive Recursive Function
3. Recursive Function
4. Turing Machine
5. Turing Machine and Computability
6. Lambda Calculus
7. Lambda Calculus and Computability
8. Complete Partial Ordered Set
9. CPO and Data Type
10. Continuous Function
11. Denotational Semantics*
12. Introduction to Category Theory
13. Limits and Adjunctions
14. Category Theory and Data Type
15. Summary*

*not in-class lecture

What is Computation?

- Computation = what computers can calculate
- Focus only on computation for Natural Numbers.
  - $N = \{0, 1, 2, 3, 4, 5, 6, 7, \cdots\}$
- Computers can calculate four arithmetic operations (add, subtract, multiply, divide) on natural numbers.
  - For subtraction of a bigger number from a small number, the result is 0.
    - e.g. $3 - 5 = 0$
  - For division, the result is rounded down to natural numbers (no fraction).
    - e.g. $5 \div 2 = 2$
- What computers can do:
  - Store the result of arithmetic operations into variables (Assignment Statement).
  - Use values stored in variables in arithmetic operations.
  - Process arithmetic and others one by one based on prescribed steps.
  - Depending on values of variables, do different steps (Conditional Statement).

Computation and Algorithm

- Computation:
  - Store several natural numbers in variables
  - Process arithmetic and others based on prescribed steps.
  - The result of computation is stored in a variable.
  - Algorithm can be represented as a flow chart.

- Mathematically
  - Computation = what computers can calculate
  - Computers can be seen as functions.
  - What kind of functions can computer calculate?
  - Computability

- Algorithm = description of computation steps
  - Algorithm can be represented as a flow chart.

Greatest Common Divisor

- Calculate the greatest common divisor of two natural numbers
  - the biggest common divisor
  - the biggest number which can divide both numbers
  - for natural numbers m and n, let $\gcd(m, n)$ be the greatest common divisor

- Example： the greatest common divisor of 315 and 231
  - Divisors of 315
    -
  - Divisors of 213
    -
  - Common divisors of 315 and 231
    -
  - The greatest common divisor of 315 and 231
    -

Euclidean Algorithm

- The oldest algorithm by Euclid
  - Euclid: BC330 -- BC275
  - Euclid's Elements

- Euclidean algorithm of caluculating the greatest common divisor of two natural numbers n and m:
  1. Calculate the remainder r of n divided by m.
     - $n = q \times m + r$
     - $\mathrm{gcd}(n, m)$ is equal to $\mathrm{gcd}(m, r)$
  2. Replace n, m by m, r, and do 1 again.
  3. Repeat until n becomes divisible by m.
  4. When the remainder is 0, n is the answer.
     - $\mathrm{gcd}(n, 0) = n$

Euclidean Algorithm Example

- Example: $\gcd(315{,}231)$
  - $\gcd(315{,}231)$
    - $315 \div 231 = 1 \cdots$
  - $\gcd(315{,}231) = \gcd(231, \quad)$
    - $231 \div \quad = 2 \cdots$
  - $\gcd(231, \quad) = \gcd(\quad , \quad)$
    - $\quad \div \quad = 1 \cdots$
  - $\gcd(\quad , \quad) = \gcd(\quad , \quad)$
    - $\quad \div \quad = 3 \cdots 0$
  - $\gcd(\quad , \quad) = \gcd(\quad , 0)$
  - $\gcd(\quad , 0) =$

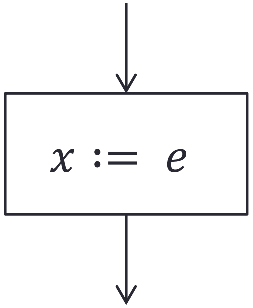
Euclidean Algorithm

1. Calculate the remainder r of n divided by m.
   - $\gcd(n, m) = \gcd(m, r)$
2. Replace n, m by m, r
3. Repeat until n becomes divisible by m.
4. When the remainder is 0
   - $\gcd(n, 0) = n$

$$\frac{231}{315} = \frac{231 \div \quad}{315 \div \quad} = \frac{\quad}{\quad}$$

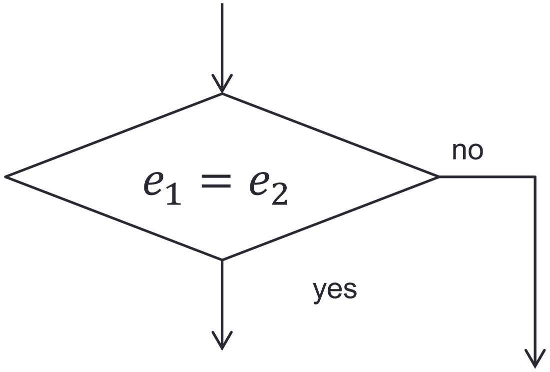
Flow Chart

- Assignment



where e is an expression of variables, natural numbers and arithmetic operations.

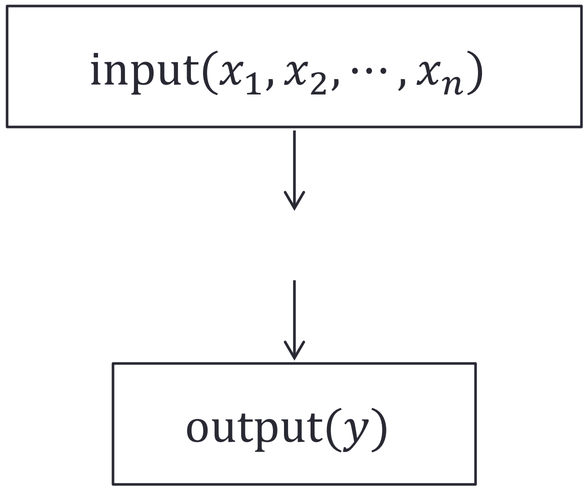
- Conditional branch



where e_1 and e_2 are expressions of variables, natural numbers and arithmetic operations.

Input and Output

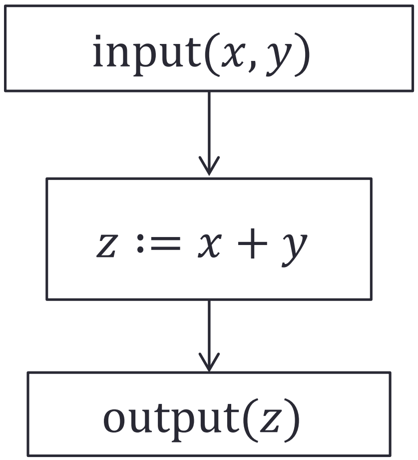
- Input
- Output



- Flow chart program
  - Start from input box, connect assignment and conditional boxes and end with output box.
  - Output box specifies the result of the function

$$f : \underbrace{N \times N \times \cdots \times N}_{\text{input}} \rightarrow \underbrace{N}_{\text{output}}$$

A Simple Flow Chart Program

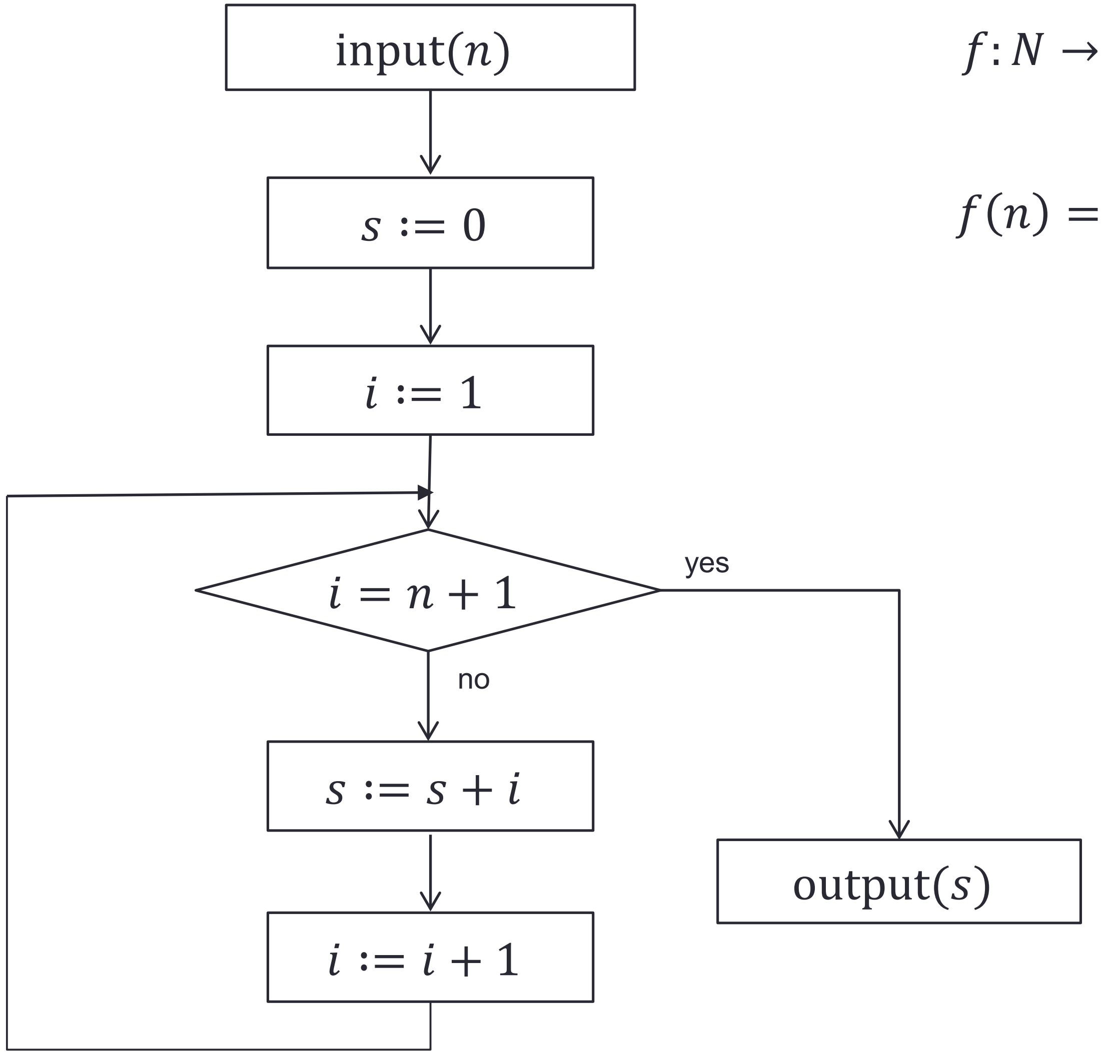


$f: N \times N \rightarrow N$ (input: $N \times N$; output: N)

input output

$f(x, y) = x + y$

Flow Char of Calculating $1 + 2 + \cdots + n$



$$f: N \to N$$

$$f(n) = \sum_{i=1}^{n} i$$

Flow Chart for Euclidean Algorithm

Write a flow chart for Euclidean algorithm.

While Program

- Programming Language
  - For computers, it is difficult to specify flow charts which are two dimensional graphs.
  - Want to express them as one dimensional language.

- While Programs
  - $\text{input}(x_1, x_2, \cdots, x_n)$
  - $\text{output}(y)$
  - $x := e$
  - $\{P_1; P_2; \cdots; P_n\}$
  - if $(e_1 = e_2)$ then P else Q
  - while $(e_1 = e_2)$ P

Example: While Program

- Calculating $1 + 2 + \cdots + n$

```
input(n);
s := 0;
i := 1;
while (i <= n) {
  s := s + i;
  i := i + 1
}
output(s);
```

```
input(n);
s := 0;
i := 1;
while (1 - (i - n) = 1) {
  s := s + i;
  i := i + 1
}
output(s);
```

Example of While Program

- Write a while program for Euclidean algorithm.

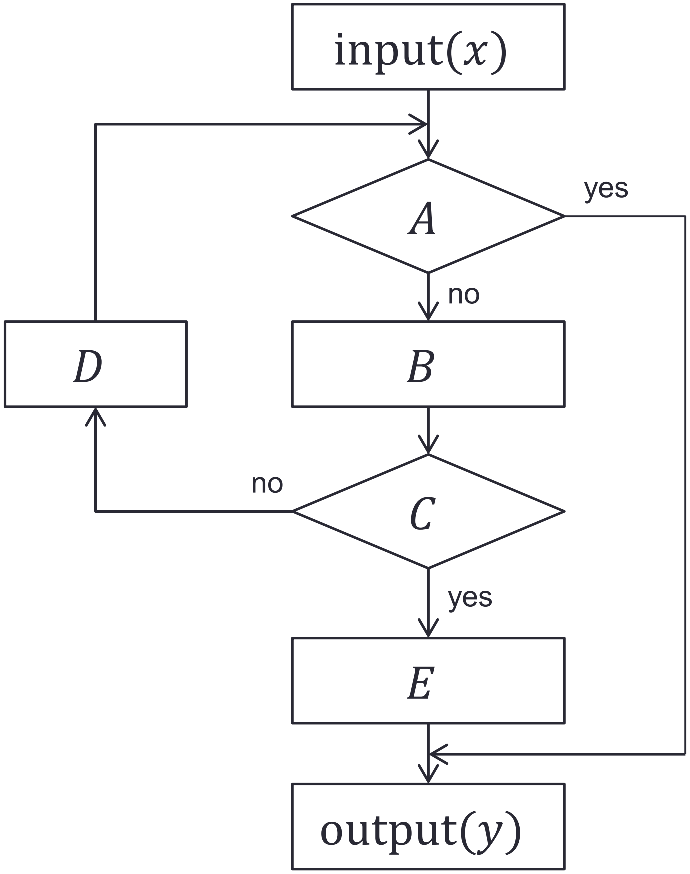
```
input(n,m);











output(n);
```

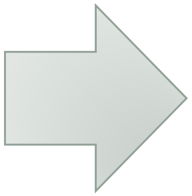
Flow Chart and While Program

- Theorem:
  - Any while program can be expressed as a flow chart program.
  - Any flow chart program can be expressed as a while program.

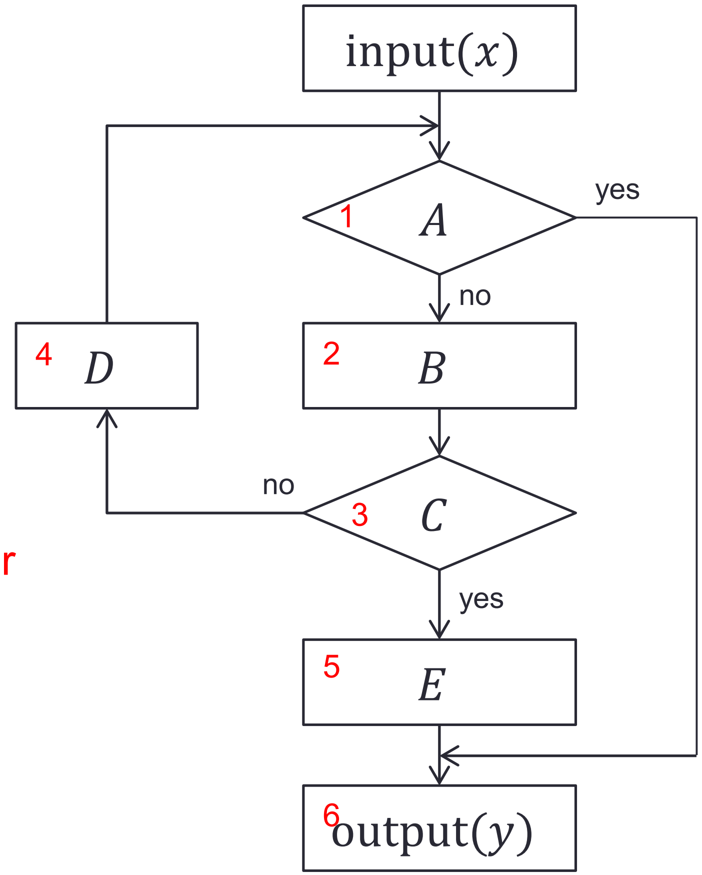
- Proof:
  - It is obvious that any while program can be expressed as a flow chart program.
  - Inverse
    - Put a number to each box (except input box) in the flow chart.
    - Introduce a new variable to manage the box number.
    - Use box numbers instead of arrows in the flow chart.
    - Write a while program which manages the box number.

Example of conversion

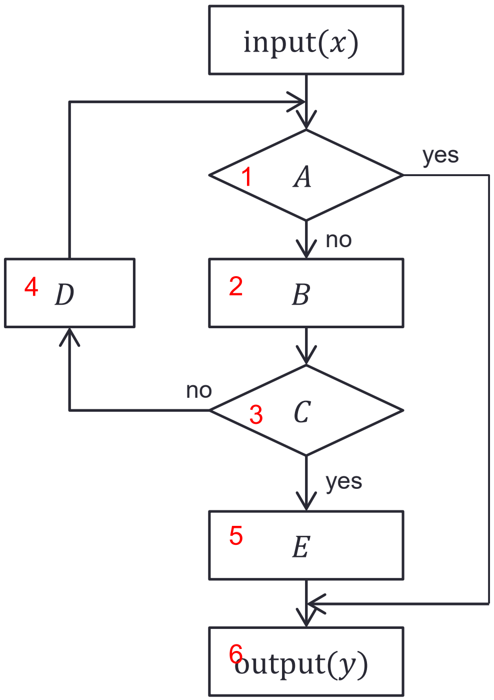




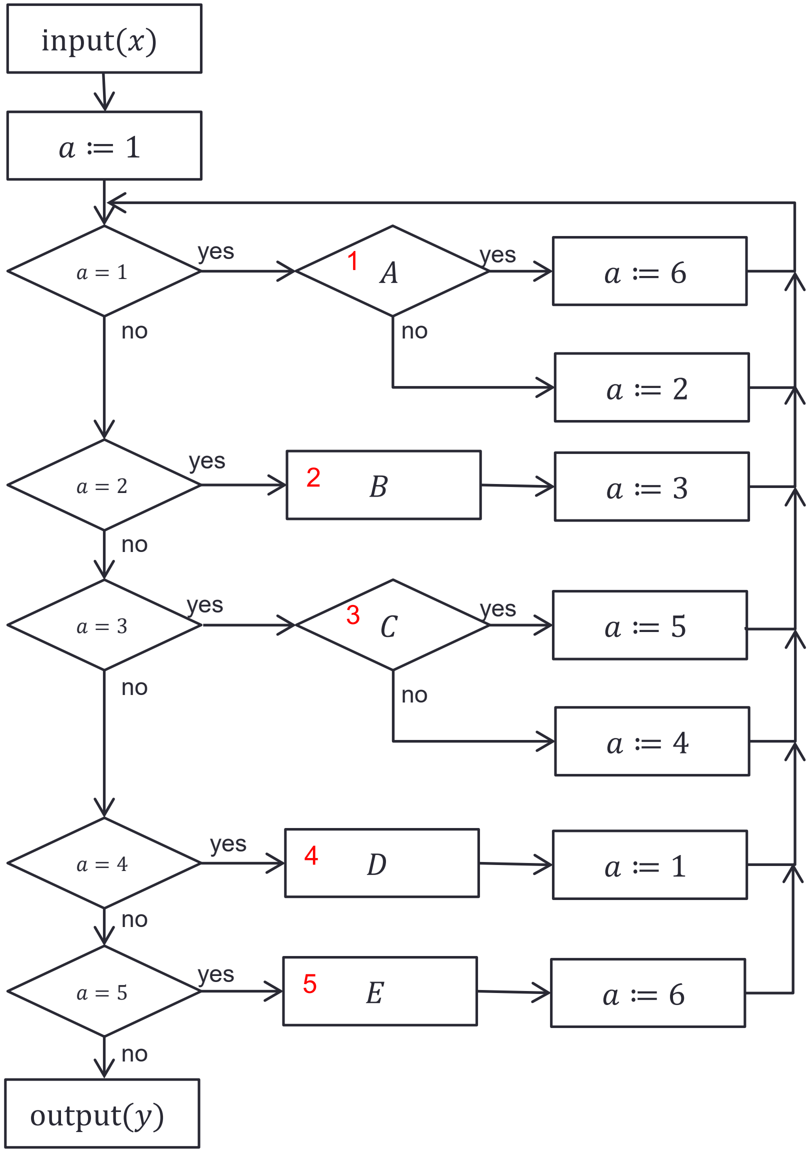
Put a number
to each box



Example of conversion



Introduce a new variable
Use box number
to connect



Example of Conversion

- Write as a While Program

```
input(x);



 a:=1;
 while (a-5=0) {
   if (a=1) then { if (A) then a:=6 else a:=2 }
   else if (a=2) then { B; a:=3 }
   else if (a=3) then { if (C) then a:=5 else a:=4 }
   else if (a=4) then { D; a:=1 }
   else if (a=5) then { E; a:=6 }
 }




output(y);
```

Corollary

- Corollary:
  - Any while program can be converted into a program with one while statement.

- Proof:
  - Express a given while program to a flow chart program.
  - Convert the flow chart program to a while program.

Summary

- Computation = what computers can calculate
- Computable functions = mathematical functions which computers can calculate
- Computability = whether mathematical functions are computable or not
  - Not all the mathematical functions on natural numbers are computable.
  - There are mathematical functions which cannot be calculated by computers.