INTRODUCTION TO
DISCRETE STRUCTURES

10.1 Program correctness

1.1–1.2

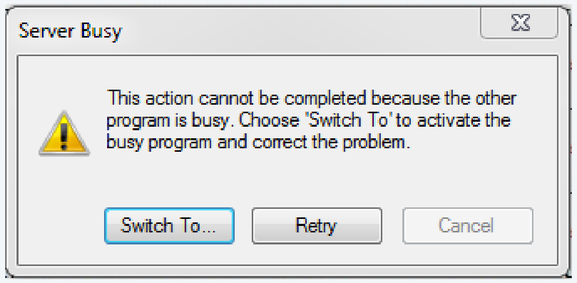
10.1 Program correctness

- Introduction
- Pre and post conditions
- Rules of inference
- Partial correctness of programs
- Loop invariants

Why study program correctness?

Coding disasters can be costly

- AT & T (1990) – could not connect calls for 14 hours (loss of $60 million)
  - Bug. C program that featured a break statement located within an if clause, that was nested within a switch clause.
- Mars Orbiter (1999)
  - Bug. Unit conversion (metric system) failure
- Intel bug (1994)
  - Bug. floating point bug when dividing
- Ariane – 5 disaster (4 billion)
  - Bug. 64-bit to 16-bit error
- Boeing 747 Max 9
  - Bug. software error due to single point of failure


Server Busy

This action cannot be completed because the other program is busy. Choose 'Switch To' to activate the busy program and correct the problem.

Program correctness

Showing that a program is correct means that it does what it is supposed to do.

More formally, our goal is to prove that a program satisfies its specifications

It correctly realizes the prescribed relationship between inputs and outputs.

In other words, for each input, the specification tells us what the program should output as a response.

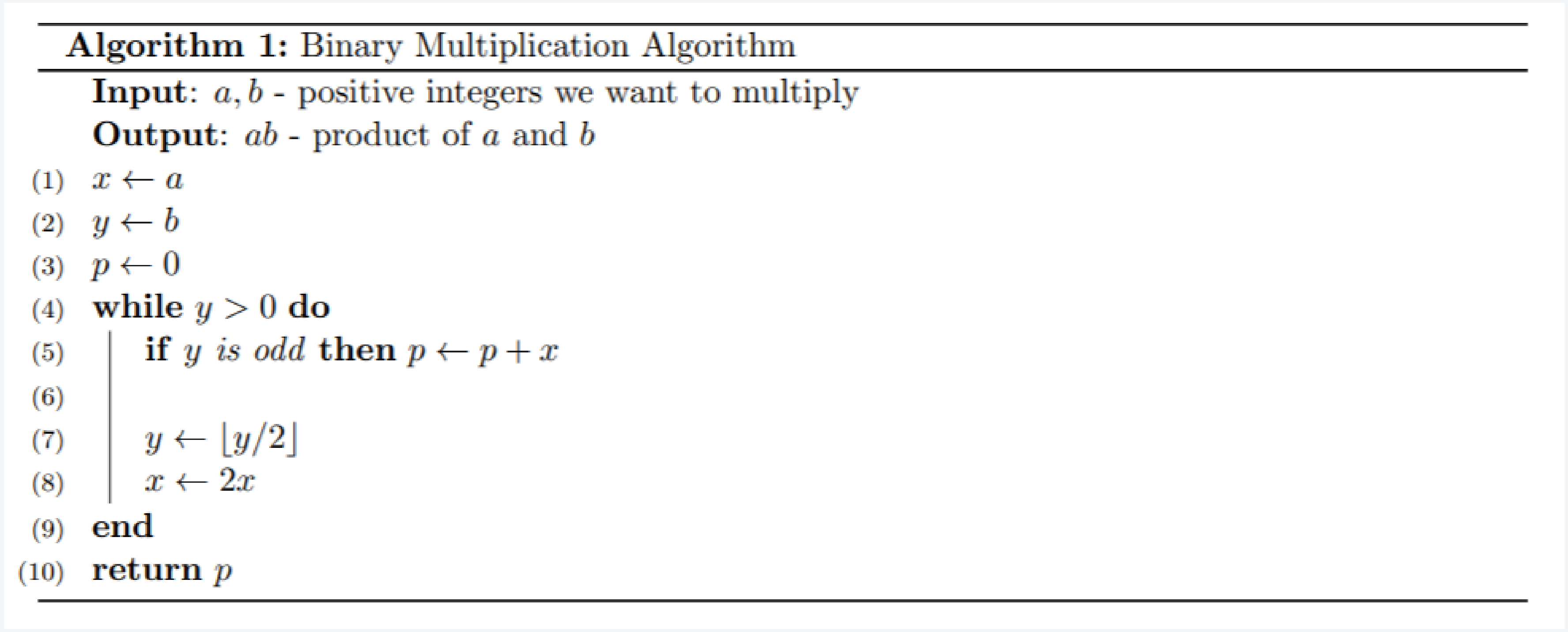
There are two parts to correctness of a program.

1. Partial correctness: If the program ever returns a result, it is the correct result.
2. Termination: The program returns.

Credit: http://pages.cs.wisc.edu

Program correctness

Question. Can you prove the correctness of the grade school multiplication algorithm?



Algorithm 1: Binary Multiplication Algorithm

```
Input: $a, b$ - positive integers we want to multiply
Output: $ab$ - product of $a$ and $b$
$x \leftarrow a$
$y \leftarrow b$
$p \leftarrow 0$
while $y > 0$ do
    if $y$ is odd then $p \leftarrow p + x$

    $y \leftarrow \lfloor y/2 \rfloor$
    $x \leftarrow 2x$
end
return $p$
```

Credit: http://pages.cs.wisc.edu

INTRODUCTION TO
DISCRETE STRUCTURES

9.2 Program correctness

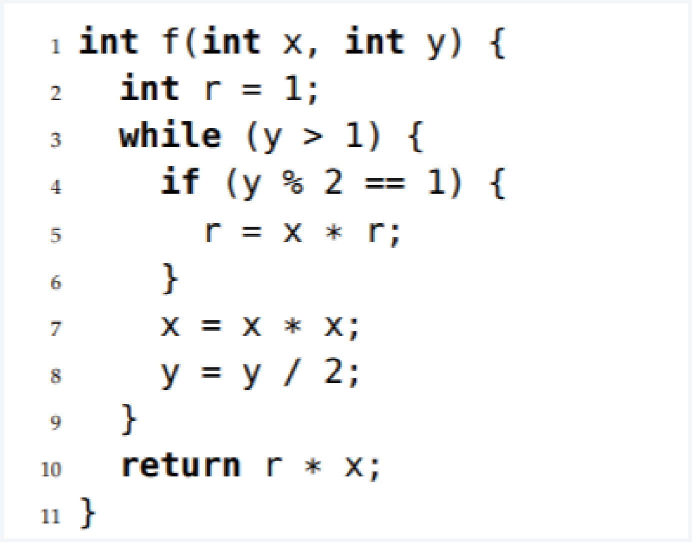
- Introduction
- Pre and post conditions
- Rules of inference
- Partial correctness of programs
- Loop invariants

Pre and post conditions of a program

Pre-Condition. Assumptions about what predicates must be true before executing the program code.

Post Condition. What is predicates are true in and after the program code is executed.

What are pre and post conditions for the following program code?



```
int f(int x, int y) {
  int r = 1;
  while (y > 1) {
    if (y % 2 == 1) {
      r = x * r;
    }
    x = x * x;
    y = y / 2;
  }
  return r * x;
}
```

workshop

Consider the following program code.

```
x = x + 1
y = x + y
```

Assume that the pre-condition predicate is P(x=a, y=b)

Write a post condition predicate after code is executed.

INTRODUCTION TO DISCRETE STRUCTURES

9.2 Program correctness

- Introduction
- Pre and post conditions
- Rules of inference
- Partial correctness of programs
- Loop invariants

Rules of inference

Assume that S : S1; S2

p is the precondition for S1 and q is the post condition

q is the precondition for S2 and r is the post condition

$$\frac{\begin{array}{c} p\{S_1\}q \\ q\{S_2\}r \end{array}}{\therefore\ p\{S_1;\ S_2\}r.}$$

Example.

S1: x = x + 1

S2: y = x + y

Write the pre and post conditions of S1 and S2

Conditional statements

If condition, then

S

More formally,

$$\begin{array}{l} (p \wedge condition)\{S\}q \\ (p \wedge \neg condition) \rightarrow q \\ \hline \therefore\ p\{\textbf{if}\ condition\ \textbf{then}\ \ S\}q\,. \end{array}$$

Example.

S : if (x < 2 && x != 1) then x=1

(p ^ condition) {S} q

(p^ ~condition) ➔ q

workshop

Verify that the program segment

$$\textbf{if } x < 0 \textbf{ then } x := 0$$

is correct with respect to the initial assertion **T** and the final assertion $x \geq 0$.

Solution:

workshop

Consider the program S: { if $x < y$ then $x = y$}

Find a final assertion for the program

Prove when initial assertion is T, then the final assertion holds

Solution.

Two-way conditional statements

Assume S1 and S2 are program statements, p is a pre-condition and q is a post-condition.

```
if condition then
        $S_1$
else
        $S_2$
```

Program code

$$\begin{array}{l} (p \wedge condition)\{S_1\}q \\ (p \wedge \neg condition)\{S_2\}q \\ \hline \therefore\ p\{\textbf{if}\ condition\ \textbf{then}\ \ S_1\ \textbf{else}\ S_2\}q\,. \end{array}$$

Propositional description

workshop

Consider the program segment

```
  if x < y then
     min = x
   else
     min = y
```

1. If initial assertion is T, write a proper final assertion.
2. Prove the final assertion by cases.

Solution.

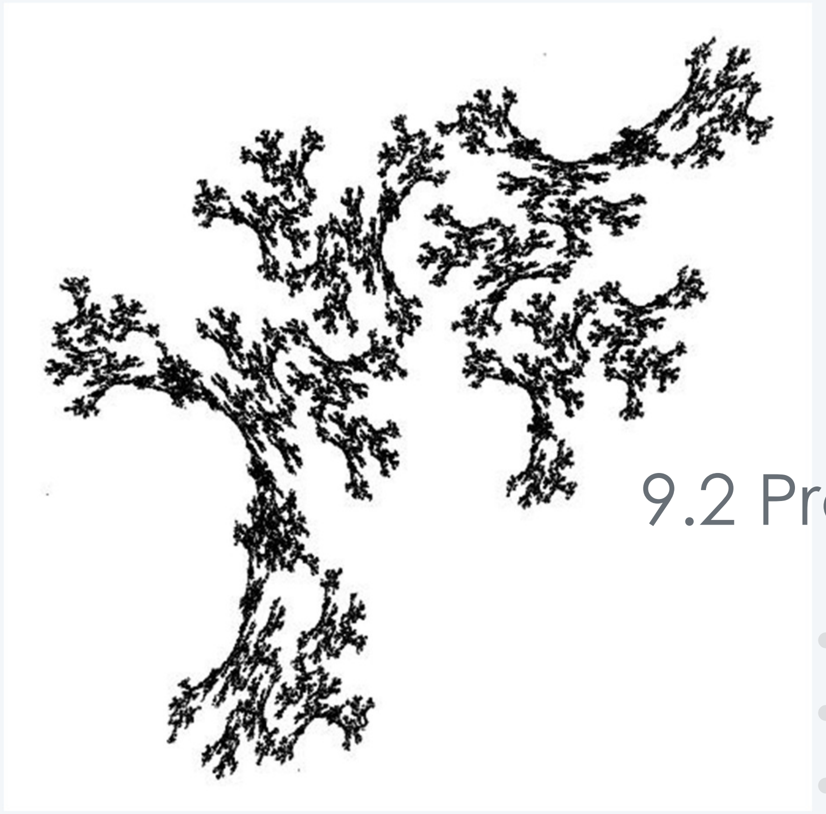
workshop

Show that following program S is correct with p = T and q = {abs = |x|}

```
**if** $x < 0$ **then**
    $abs := -x$
**else**
    $abs := x$
```

Proof.

INTRODUCTION TO
DISCRETE STRUCTURES



9.2 Program correctness

- Introduction
- Pre and post conditions
- Rules of inference
- Partial correctness of programs
- Loop invariants

Definition of Correctness

A program S is correct, if S produces the correct output q for input p. We write p{S}q [*Hoare triple*]

A program is said to be **correct** if it produces the correct output for all possible inputs

To prove a program is correct

(partial correctness) Prove that the correct answer is obtained if the program terminates.

(Termination) Prove that program always terminates for all inputs

workshop

Consider the following program. What is the post condition q?

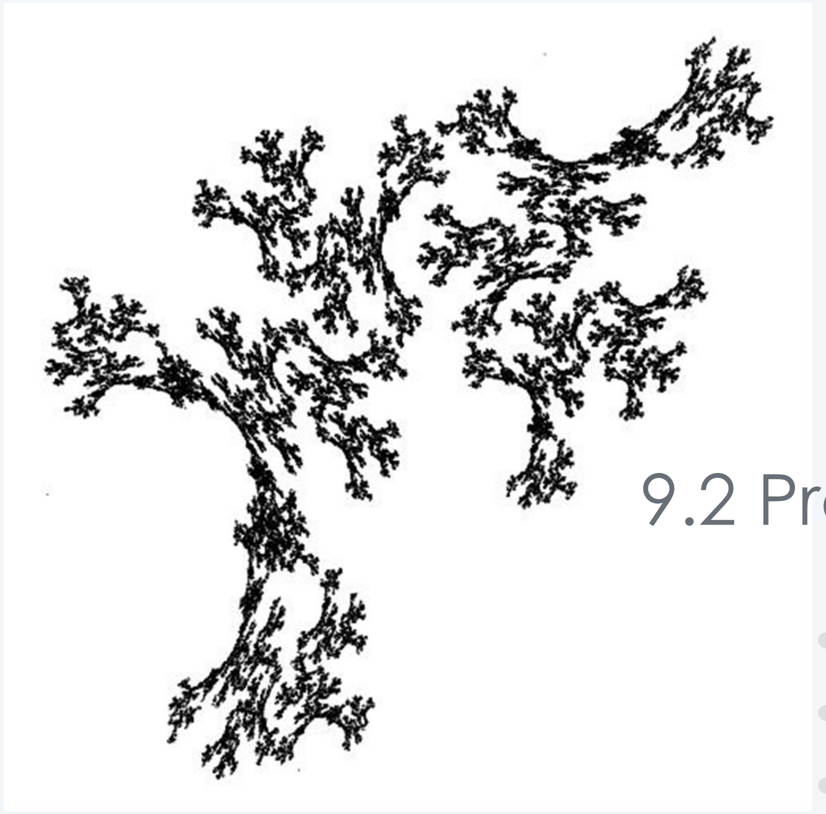
p : {i==0, x=0}

S: while (i < n) {i++;   x = x + i;}

q:

How do we prove partial correctness? That is, correct post condition is met, if the program terminates.

How do we prove termination? That is, the program in fact terminates.



INTRODUCTION TO
DISCRETE STRUCTURES

9.2 Program correctness

- Introduction
- Pre and post conditions
- Rules of inference
- Partial correctness of programs
- Loop invariants

Loop invariants

A Boolean condition that must be true immediately before every evaluation of a loop

A Boolean condition that is true immediately after loop terminates (if the loop terminates)

Example. What is a loop invariant for the following code?

```
while (i < n)  {i++;   x = x + i;}
```

workshop

Find loop invariants for the following code segment and prove that they are indeed invariants

p: {i==1, x=1}

S: while (i < n) {i++;   x = x * i;}

q: x = 1*2* …*n

INTRODUCTION TO
DISCRETE STRUCTURES

9.2 Program correctness

- Introduction
- Pre and post conditions
- Rules of inference
- Partial correctness of programs
- Loop invariants

INTRODUCTION TO
DISCRETE STRUCTURES

10.1 Program correctness

1.1–1.2