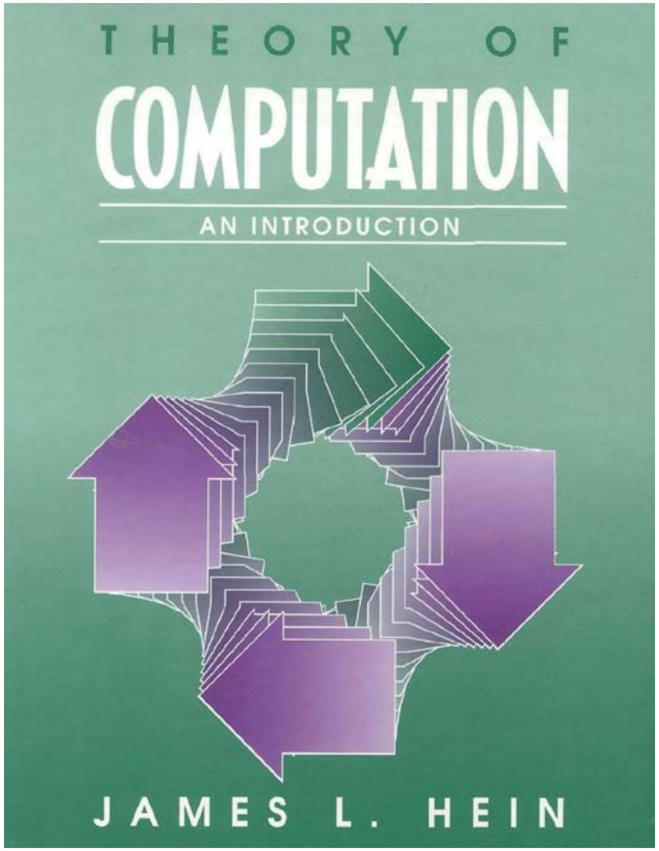
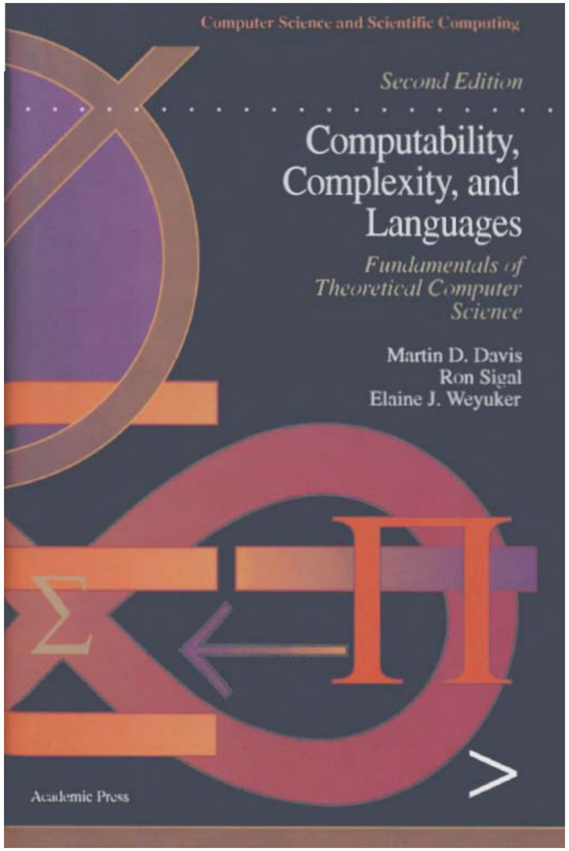
Theory of Computation

Lecture 04

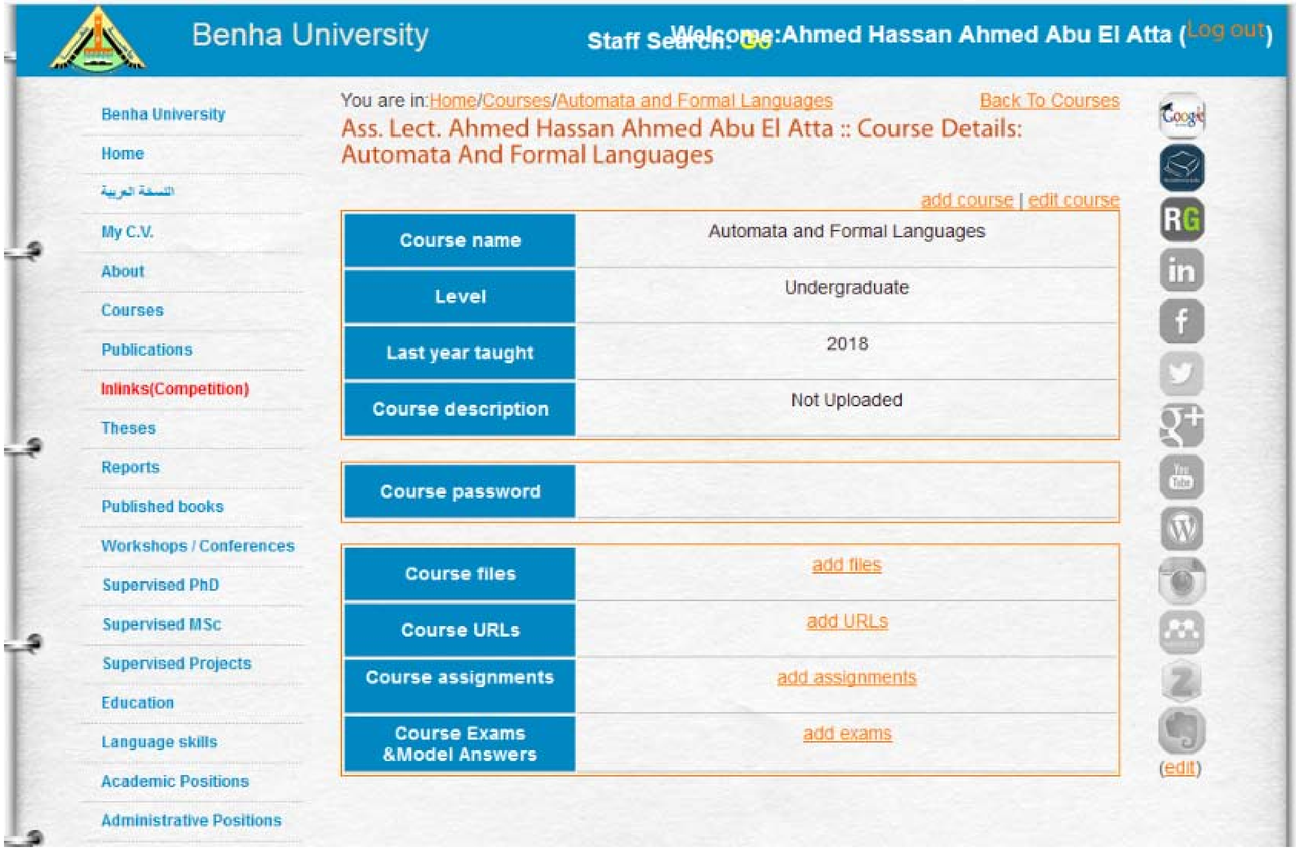
Books


THEORY OF
COMPUTATION
AN INTRODUCTION
JAMES L. HEIN


Computer Science and Scientific Computing
Second Edition
Computability, Complexity, and Languages
Fundamentals of Theoretical Computer Science
Martin D. Davis
Ron Sigal
Elaine J. Weyuker
Academic Press

PowerPoint

http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14767



Programs and Computable Functions

SIMPLE LANGUAGE II

Agenda

- Multiply Two Variables
- The Macro Expansion of $Z_1 \leftarrow X_1 + Y$
- Subtraction
- Syntax
- Snapshot
- Computable Functions

Multiply Two Variables

$$f(x_1, x_2) = x_1 \cdot x_2$$

Since multiplication can be regarded as repeated addition, we are led to the "program"

Multiply Two Variables

$$f(x_1, x_2) = x_1 \cdot x_2$$

$$
\begin{array}{ll}
 & Z_2 \leftarrow X_2 \\
[B] & \text{IF } Z_2 \neq 0 \text{ GOTO } A \\
 & \text{GOTO } E \\
[A] & Z_2 \leftarrow Z_2 \dot{-} 1 \\
 & Z_1 \leftarrow X_1 + Y \\
 & Y \leftarrow Z_1 \\
 & \text{GOTO } B
\end{array}
$$

Multiply Two Variables

Of course, the "instruction" $Z_1 \leftarrow X_1 + Y$ *is not permitted in the language* $\mathscr{S}$.

What we have in mind is that since we already have an addition program, we can replace the macro $Z_1 \leftarrow X_1 + Y$ by a program for computing it, which we will call its macro expansion.

$$
\begin{array}{ll}
 & Z_2 \leftarrow X_2 \\
[B] & \text{IF } Z_2 \neq 0 \text{ GOTO } A \\
 & \text{GOTO } E \\
[A] & Z_2 \leftarrow Z_2 \dot{-} 1 \\
 & Z_1 \leftarrow X_1 + Y \\
 & Y \leftarrow Z_1 \\
 & \text{GOTO } B
\end{array}
$$

$$
\boxed{\begin{array}{l} Z_1 \leftarrow X_1 + Y \\ Y \leftarrow Z_1 \end{array}}
$$

Why not?

$$
\boxed{Y \leftarrow X_1 + Y}
$$

Multiply Two Variables

$$\boxed{Y \leftarrow X_1 + Y}$$

$$
\begin{array}{ll}
 & Y \leftarrow X_1 \\
 & Z \leftarrow Y \\
[B] & \text{IF } Z \neq 0 \text{ GOTO } A \\
 & \text{GOTO } E \\
[A] & Z \leftarrow Z - 1 \\
 & Y \leftarrow Y + 1 \\
 & \text{GOTO } B
\end{array}
$$

What does this program actually compute?

It should not be difficult to see that instead of computing $x_1 + y$ as desired,

this program computes $2x_1$

The Macro Expansion of $Z_1 \leftarrow X_1 + Y$:

$$
\begin{array}{lll}
 & Z_2 \leftarrow X_2 & \\
[B] & \text{IF } Z_2 \neq 0 \text{ GOTO } A & \\
 & \text{GOTO } E & \\
[A] & Z_2 \leftarrow Z_2 - 1 & \\
 & Z_1 \leftarrow X_1 & \\
 & Z_3 \leftarrow Y & \\
[B_2] & \text{IF } Z_3 \neq 0 \text{ GOTO } A_2 & \text{Macro Expansion of} \\
 & \text{GOTO } E_2 & Z_1 \leftarrow X_1 + Y \\
[A_2] & Z_3 \leftarrow Z_3 - 1 & \\
 & Z_1 \leftarrow Z_1 + 1 & \\
 & \text{GOTO } B_2 & \\
[E_2] & Y \leftarrow Z_1 & \\
 & \text{GOTO } B &
\end{array}
$$

The Macro Expansion of $Z_1 \leftarrow X_1 + Y$:

1. The local variable Z_1 in the addition program in (d) must be replaced by another local variable (we have used Z_3) because Z_1 (the other name for Z) is also used as a local variable in the multiplication program.
2. The labels A, B, E are used in the multiplication program and hence cannot be used in the macro expansion. We have used A_2, B_2, E_2 instead.
3. The instruction GOTO E_2 terminates the addition. Hence, it is necessary that the instruction immediately following the macro expansion be labeled E_2.

What is the Function?

If we begin with $X_1 = 5$, $X_2 = 2$, the program first sets Y = 5 and Z = 2.

Successively the program sets Y = 4 Z = 1 and Y = 3, Z = 0. Thus, the computation terminates with Y = 3 = 5 - 2.

Clearly, if we begin with $X_1 = m$, $X_2 = n$, where $m \geq n$, the program will terminate with Y = m - n.

```
      Y ← X1
      Z ← X2
[C]   IF Z ≠ 0 GOTO A
      GOTO E
[A]   IF Y ≠ 0 GOTO B
      GOTO A
[B]   Y ← Y − 1
      Z ← Z − 1
      GOTO C
```

What is the Function?

What happens if we begin with a value of X_1 less than the value of X_2 , e.g., $X_1 = 2$, $X_2 = 5$?

The program sets Y = 2 and Z = 5 and successively sets Y = 1, Z = 4 and Y = 0, Z = 3. At this point the computation enters the "loop":

$[A]$	IF $Y \neq 0$ GOTO B
	GOTO A

	$Y \leftarrow X_1$
	$Z \leftarrow X_2$
$[C]$	IF $Z \neq 0$ GOTO A
	GOTO E
$[A]$	IF $Y \neq 0$ GOTO B
	GOTO A
$[B]$	$Y \leftarrow Y - 1$
	$Z \leftarrow Z - 1$
	GOTO C

Subtraction

Since y = 0, there is no way out of this loop and the computation will continue "forever.“

Thus, if we begin with X_1 =m, X_2 =n, where m < n, the computation will never terminate.

In this case (and in similar cases) we will say that the program computes the partial function.

$$g(x_1, x_2) = \begin{cases} x_1 - x_2 & \text{if} \quad x_1 \geq x_2 \\ \uparrow & \text{if} \quad x_1 < x_2 \end{cases}$$

Syntax

A state of a program $\mathscr{P}$ is a list of equations of the form V = m, where V is a variable and m is a number,

including an equation for each variable that occurs in $\mathscr{P}$ and including no two equations with the same variable.

Syntax

VALID STATES

INVALID STATES

let $\mathscr{P}$ be the program, which contains the variables X, Y, and Z.

The list

X= 4, Y= 3, z = 3

is thus a state of $\mathscr{P}$.

$$X_1 = 4, \quad X_2 = 5, \quad Y = 4, \quad Z = 4$$

$$\boxed{X = 3, \quad Z = 3}$$

$$\boxed{X = 3, \quad X = 4, \quad Y = 2, \quad Z = 2}$$

Snapshot

Suppose we have a program $\mathscr{P}$ and a state σ of $\mathscr{P}$.

In order to say what happens "next," we also need to know which instruction of $\mathscr{P}$ is about to be executed.

We therefore define a snapshot or instantaneous description of a program $\mathscr{P}$ of length n to be a pair (i, σ) where $1 \leq i \leq n + 1$, and σ is a state of $\mathscr{P}$.

Intuitively the number i indicates that it is the i^{th} instruction which is about to be executed; $i = n + 1$ corresponds to a "stop" instruction.

Snapshot

If $s = (i, \sigma)$ is a snapshot of $\mathscr{P}$ and V is a variable of $\mathscr{P}$, then the *value of V at s* just means the value of V at σ.

A snapshot (i, σ) of a program $\mathscr{P}$ of length n is called *terminal* if $i = n + 1$. If (i, σ) is a nonterminal snapshot of $\mathscr{P}$, we define the *successor* of (i, σ) to be the snapshot (j, τ) defined as follows:

Case 1. The ith instruction of $\mathscr{P}$ is $V \leftarrow V + 1$ and σ contains the equation $V = m$. Then $j = i + 1$ and τ is obtained from σ by replacing the equation $V = m$ by $V = m + 1$ (i.e., the value of V at τ is $m + 1$).

Snapshot

Case 2. The ith instruction of $\mathscr{P}$ is $V \leftarrow V - 1$ and σ contains the equation $V = m$. Then $j = i + 1$ and τ is obtained from σ by replacing the equation $V = m$ by $V = m - 1$ if $m \neq 0$; if $m = 0$, $\tau = \sigma$.

Case 3. The ith instruction of $\mathscr{P}$ is $V \leftarrow V$. Then $\tau = \sigma$ and $j = i + 1$.

Case 4. The ith instruction of $\mathscr{P}$ is IF $V \neq 0$ GOTO L. Then $\tau = \sigma$, and there are two subcases:

Case 4a. σ contains the equation $V = 0$. Then $j = i + 1$.

Case 4b. σ contains the equation $V = m$ where $m \neq 0$. Then, if there is an instruction of $\mathscr{P}$ labeled L, j is the *least number* such that the jth instruction of $\mathscr{P}$ is labeled L. Otherwise, $j = n + 1$.

Snapshot

$$
\begin{array}{ll}
[A] & \text{IF } X \neq 0 \text{ GOTO } B \\
 & Z \leftarrow Z + 1 \\
 & \text{IF } Z \neq 0 \text{ GOTO } E \\
[B] & X \leftarrow X - 1 \\
 & Y \leftarrow Y + 1 \\
 & Z \leftarrow Z + 1 \\
 & \text{IF } Z \neq 0 \text{ GOTO } A
\end{array}
$$

For an example, we return to the program of (b), Section 2. Let σ be the state

$$X = 4, \qquad Y = 0, \qquad Z = 0$$

and let us compute the successor of the snapshots (i, σ) for various values of i.

For $i = 1$, the successor is $(4, \sigma)$ where σ is as above. For $i = 2$, the successor is $(3, \tau)$, where τ consists of the equations

$$X = 4, \qquad Y = 0, \qquad Z = 1.$$

For $i = 7$, the successor is $(8, \sigma)$. This is a terminal snapshot.

A computation of a program

A *computation* of a program $\mathscr{P}$ is defined to be a sequence (i.e., a list) $s_1, s_2, \ldots, s_k$ of snapshots of $\mathscr{P}$ such that s_{i+1} is the successor of s_i for $i = 1, 2, \ldots, k - 1$ and s_k is terminal.

Computable Functions

Thus, let $\mathscr{P}$ be any program in the language $\mathscr{S}$ and let $r_1, \ldots, r_m$ be m given numbers. We form the state σ of $\mathscr{P}$ which consists of the equations

$$X_1 = r_1, \qquad X_2 = r_2, \qquad \ldots, \qquad X_m = r_m, \qquad Y = 0$$

together with the equations $V = 0$ for each variable V in $\mathscr{P}$ other than $X_1, \ldots, X_m, Y$. We will call this the *initial state*, and the snapshot $(1, \sigma)$, the *initial snapshot*.

Case 1. *There is a computation* $s_1, s_2, \ldots, s_k$ of $\mathscr{P}$ *beginning with the initial snapshot*. Then we write $\psi_{\mathscr{P}}^{(m)}(r_1, r_2, \ldots, r_m)$ for the value of the variable Y at the (terminal) snapshot s_k.

Case 2. *There is no such computation;* i.e., there is an *infinite* sequence $s_1, s_2, s_3, \ldots$ beginning with the initial snapshot where each s_{i+1} is the successor of s_i. In this case $\psi_{\mathscr{P}}^{(m)}(r_1, \ldots, r_m)$ is undefined.

Example

$$
\begin{aligned}
&(1, \{X = r, Y = 0, Z = 0\}),\\
&(4, \{X = r, Y = 0, Z = 0\}),\\
&(5, \{X = r - 1, Y = 0, Z = 0\}),\\
&(6, \{X = r - 1, Y = 1, Z = 0\}),\\
&(7, \{X = r - 1, Y = 1, Z = 1\}),\\
&(1, \{X = r - 1, Y = 1, Z = 1\}),\\
&\ \vdots\\
&(1, \{X = 0, Y = r, Z = r\}),\\
&(2, \{X = 0, Y = r, Z = r\}),\\
&(3, \{X = 0, Y = r, Z = r + 1\}),\\
&(8, \{X = 0, Y = r, Z = r + 1\}).
\end{aligned}
$$

[A]	IF $X \neq 0$ GOTO B	(1)
	$Z \leftarrow Z + 1$	(2)
	IF $Z \neq 0$ GOTO E	(3)
[B]	$X \leftarrow X - 1$	(4)
	$Y \leftarrow Y + 1$	(5)
	$Z \leftarrow Z + 1$	(6)
	IF $Z \neq 0$ GOTO A	(7)

$$\psi_{\mathscr{P}}^{(1)}(x) = x$$

Example a

```
[A]     X ← X − 1
        Y ← Y + 1
        IF X ≠ 0 GOTO A
```

(a) $\psi^{(1)}(r) = \begin{cases} 1 & \text{if } \quad r = 0 \\ r & \text{otherwise,} \end{cases}$

Example b, c

[A]	IF $X \neq 0$ GOTO B
	$Z \leftarrow Z + 1$
	IF $Z \neq 0$ GOTO E
[B]	$X \leftarrow X - 1$
	$Y \leftarrow Y + 1$
	$Z \leftarrow Z + 1$
	IF $Z \neq 0$ GOTO A

[A]	If $X \neq 0$ GOTO B
	GOTO C
[B]	$X \leftarrow X - 1$
	$Y \leftarrow Y + 1$
	$Z \leftarrow Z + 1$
	GOTO A
[C]	IF $Z \neq 0$ GOTO D
	GOTO E
[D]	$Z \leftarrow Z - 1$
	$X \leftarrow X + 1$
	GOTO C

(b), (c) $\quad \psi^{(1)}(r) = r,$

Example d

$$
\begin{array}{ll}
 & Y \leftarrow X_1 \\
 & Z \leftarrow X_2 \\
[B] & \text{IF } Z \neq 0 \text{ GOTO } A \\
 & \text{GOTO } E \\
[A] & Z \leftarrow Z - 1 \\
 & Y \leftarrow Y + 1 \\
 & \text{GOTO } B
\end{array}
$$

(d) $\psi^{(2)}(r_1, r_2) = r_1 + r_2$,

Example e

$$
\begin{array}{ll}
 & Z_2 \leftarrow X_2 \\
[B] & \text{IF } Z_2 \neq 0 \text{ GOTO } A \\
 & \text{GOTO } E \\
[A] & Z_2 \leftarrow Z_2 \dot{-} 1 \\
 & Z_1 \leftarrow X_1 + Y \\
 & Y \leftarrow Z_1 \\
 & \text{GOTO } B
\end{array}
$$

(e) $\psi^{(2)}(r_1, r_2) = r_1 \cdot r_2$,

Example f

```
        Y ← X1
        Z ← X2
[C]     IF Z ≠ 0 GOTO A
        GOTO E
[A]     IF Y ≠ 0 GOTO B
        GOTO A
[B]     Y ← Y − 1
        Z ← Z − 1
        GOTO C
```

$$Y \leftarrow X_1$$
$$Z \leftarrow X_2$$
$$[C] \quad \text{IF } Z \neq 0 \text{ GOTO } A$$
$$\text{GOTO } E$$
$$[A] \quad \text{IF } Y \neq 0 \text{ GOTO } B$$
$$\text{GOTO } A$$
$$[B] \quad Y \leftarrow Y - 1$$
$$Z \leftarrow Z - 1$$
$$\text{GOTO } C$$

(f) $$\psi^{(2)}(r_1, r_2) = \begin{cases} r_1 - r_2 & \text{if} \quad r_1 \geq r_2 \\ \uparrow & \text{if} \quad r_1 < r_2 \end{cases}$$

Example

(c) $\psi_{\mathscr{P}}^{(2)}(r_1, r_2) = r_1,$

(d) $\psi_{\mathscr{P}}^{(1)}(r_1) = r_1 + 0 = r_1,$

$\psi_{\mathscr{P}}^{(3)}(r_1, r_2, r_3) = r_1 + r_2$

Total and Partial Functions

As an example, let f be the set of ordered pairs (n, n^2) for $n \in N$. Then, for each $n \in N$, $f(n) = n^2$. The domain of f is N. The range of f is the set of perfect squares.

- A partial function on a set S is simply a function whose domain is a subset of S.
- An example of a partial function on N is given by g(n) = √n, where the domain of g is the set of perfect squares.
- If a partial function on S has the domain S, then it is called total.

Computable Functions

For any program $\mathscr{P}$ and any positive integer m, the function $\psi_{\mathscr{P}}^{(m)}(x_1, \ldots, x_m)$ is said to be *computed* by $\mathscr{P}$. A given partial function g (of one or more variables) is said to be *partially computable* if it is computed by some program. That is, g is partially computable if there is a program $\mathscr{P}$ such that

$$g(r_1, \ldots, r_m) = \psi_{\mathscr{P}}^{(m)}(r_1, \ldots, r_m)$$

for all $r_1, \ldots, r_m$.

Computable Functions

A given function g of m variables is called total if $g(r_1, \ldots, r_m)$ is defined for all $r_1, \ldots, r_m$.

A function is said to be computable if it is both partially computable and total.

Our examples from Section 2 give us a short list of partially computable functions, namely: $x, x + y, x \cdot y$, and $x - y$. Of these, all except the last one are total and hence computable.