CS 410/610, MTH 410/610
Theoretical Foundations of Computing

Fall Quarter 2010

Slides 3

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/

TOC: Turing Computable Functions

Chapter 9 of [Sudkamp 2006].

1. Computation of Functions
2. Numeric Computation
3. Sequential Operation of TMs
4. Composition of Functions
5. Uncomputable Functions

Functions

A function f: X $\rightarrow$ Y is an assignment, to each x$\in$ X, of *at most* one value in Y. (Mathematicians call these: *partial* functions.)

X … domain of f

Y … range of f

We write f(x)$\uparrow$ (or f(x)=$\uparrow$) if no value is assigned to f(x), and say f(x) is undefined.

We write f(x)$\downarrow$ if f(x) is defined (we're not giving the value in this case).

If f(x)$\downarrow$ for all x$\in$ X, we say that f is a *total* function.

TMs for computing functions

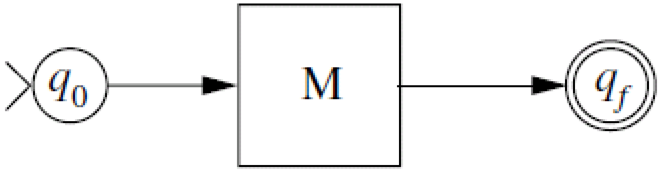
TMs for computing functions have

- **Two distinguished states**
  - **The initial state q_0**
  - **The final state q_f**
- **Input is positioned as usual**
- **Computation always begins with transition from q_0 that positions the tape head at the beginning of the input string.**
- **The initial state is never reentered (there is no transition into q_0).**
- **All computations with output terminate in q_f and with tape head in initial position**
- **There is no transition of the form $\delta(q_f,B)$**
- **Output is given in the same position as the input**
- **The computation does not terminate on input u with $f(u)\uparrow$**
- **The computation yields output v if and only if $f(u)=v$.**

Turing computability

A function f: $\Sigma^* \rightarrow \Sigma^*$ is Turing computable if there is a TM that computes it.

We may depict such a TM schematically as

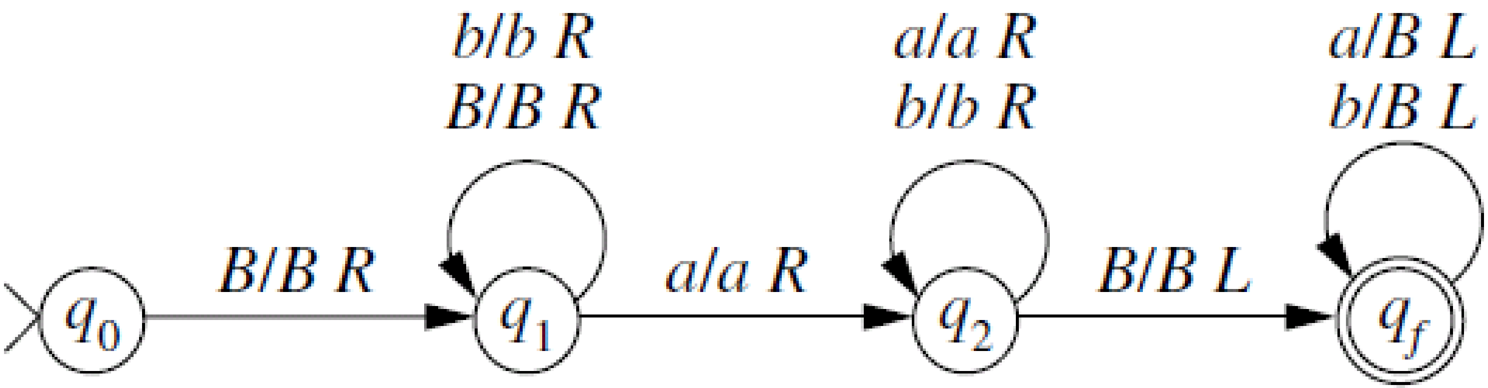


Example 2.1

TM computing f:{a,b}* $\rightarrow$ {a,b}* defined as

f(u) = λ, if u contains an a

f(u) = $\uparrow$, otherwise



Note: on undefined input (say, BbBbBaB) we may still get some "output" (e.g., BbBbq_fB).

Exercise 22 [hand-in]

Make a TM which computes the function

f(n) = n/2 (n divided by 2) if n is a multiple of 2
f(n) = ↑ if n is not a multiple of 2

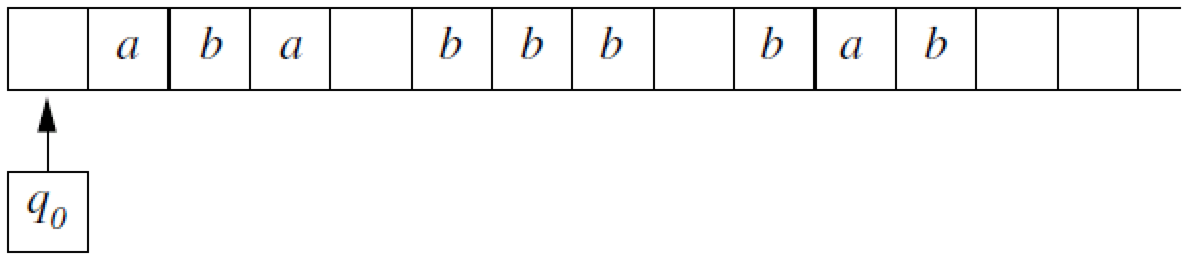
where the input and output strings are non-negative integers in binary representation.

Describe, in words, the strategy of your TM.

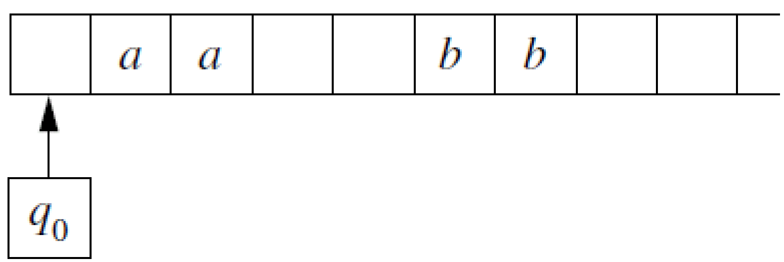
Multiple parameters

The input for functions with more than one argument is given by blank-separated strings, in the sequence of the arguments.

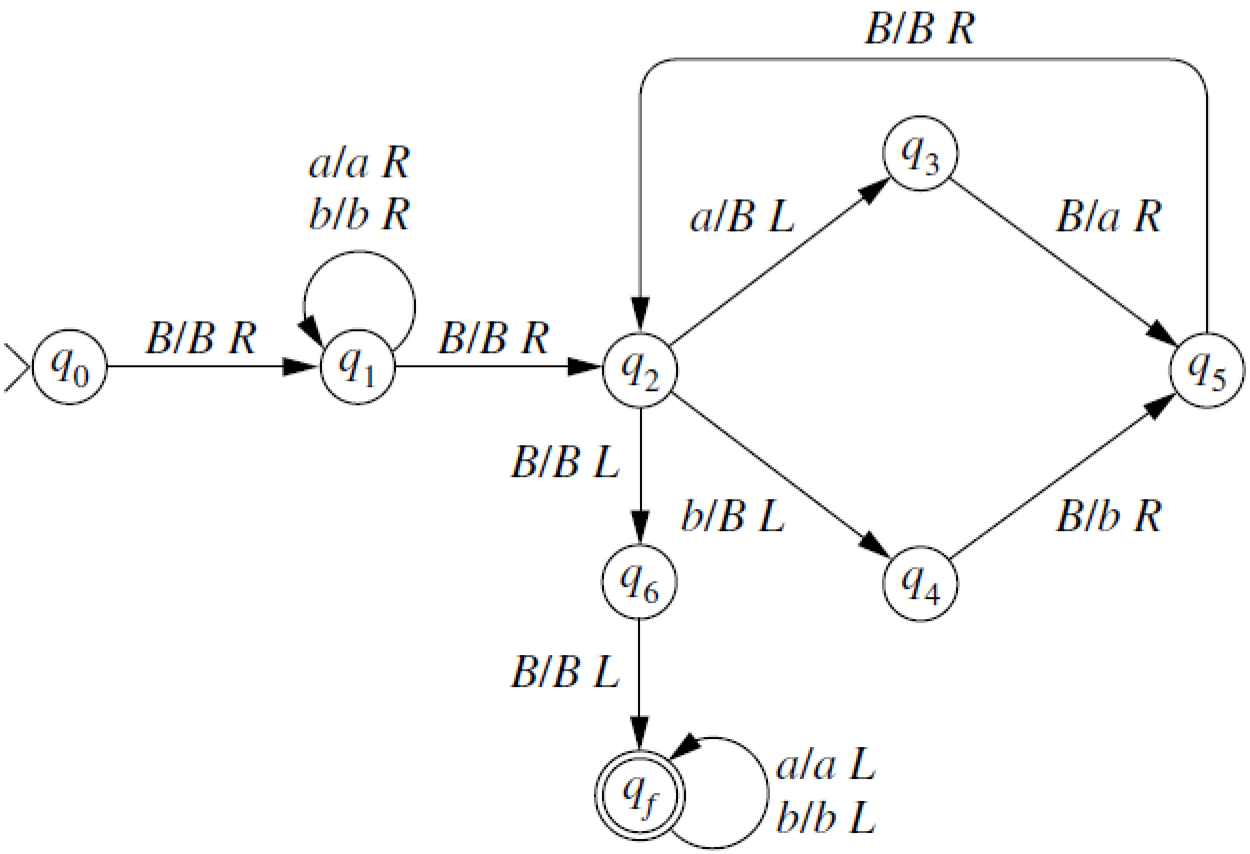
E.g., input (aba,bbb,bab) is given as



Input (aa,λ,bb) is given as



Example 2.2: String concatenation



Characteristic functions

The *characteristic function* of a language L is the function

$c_L: \Sigma^* \to \{0,1\}$ defined by

$c_L(u) = 1$ if $u \in L$

$c_L(u) = 0$ if $u \notin L$

Note: A TM that computes the partial characteristic function

$c_L(u) = 1$ if $u \in L$

$c_L(u) = 0$ or $\uparrow$ if $u \notin L$

shows that L is recursively enumerable.

Exercise 23 [hand-in]

Show for every language L: if there is a TM that computes the partial characteristic function of L, then L is recursively enumerable.

[exercise is due in the first session after the mid-term]

Exercise 24 [hand-in]

Show that, for each recursively enumerable language L, there exists a TM which computes the partial characteristic function of L.

[exercise is due in the first session after the mid-term]

Exercise 25 [hand-in]

Show that a language L is recursive if and only if its (total) characteristic function is Turing computable.

[exercise is due in the first session after the mid-term]

TOC: Turing Computable Functions

Chapter 9 of [Sudkamp 2006].

Number-theoretic functions

A *number-theoretic function* is a function of the form
F: $N \times N \ldots \times N \rightarrow N$,
where N is the set of non-negative integers.

For computing number-theoretic functions by TMs, we assume that non-negative integers are represented by strings of “1” symbols. More precisely, the number n is represented by a string with (n+1) consecutive “1”s. We call this *the unary representation* of numbers.

E.g., “5” is represented as “111111”. “0” is represented as “1”.

For a number a, we write its unary representation as $\bar{a}$.

Characteristic functions

A k-variable total number-theoretic function
$r: N \times N \ldots \times N \rightarrow \{0,1\}$
defines a k-ary relation R on the domain of the function:

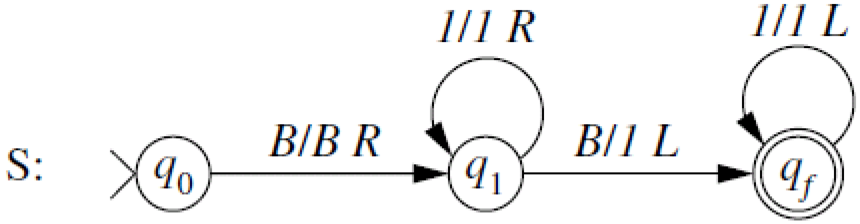
$(n_1,\ldots,n_k) \in R$ if $r(n_1,\ldots,n_k) = 1$
$(n_1,\ldots,n_k) \notin R$ if $r(n_1,\ldots,n_k) = 0$

r is the *characteristic function* of R.

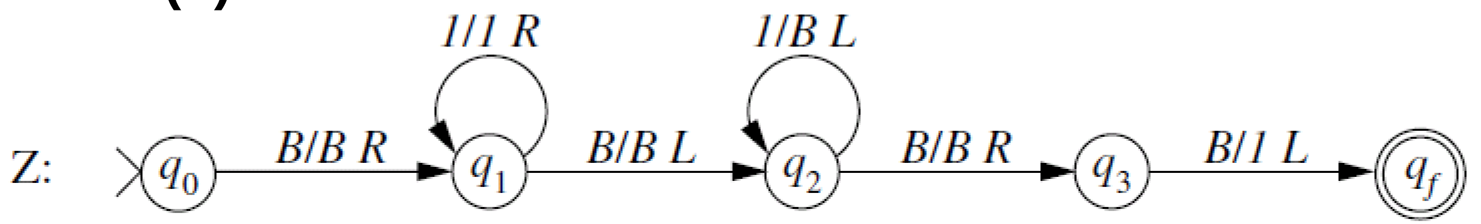
We define: A relation is Turing computable if its characteristic function is Turing computable.

Some TMs for number-theoret. fctns

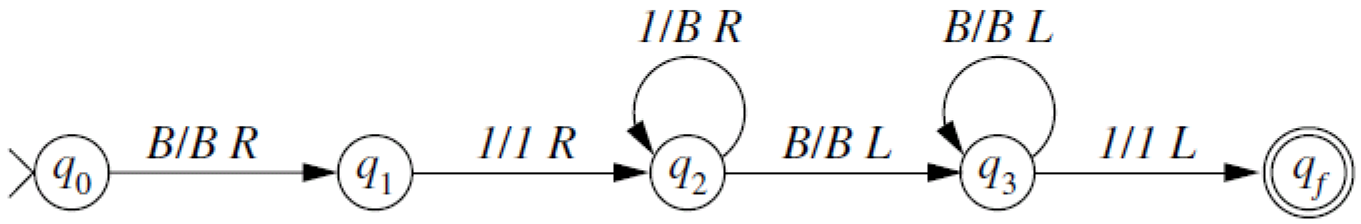
- **Successor function s(n) = n+1**



- **Zero function z(n) = 0**

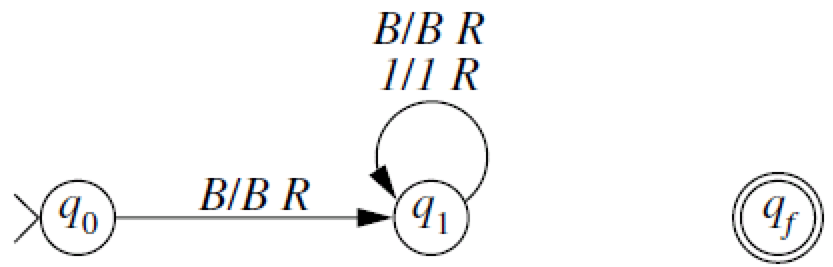


Alternatively:

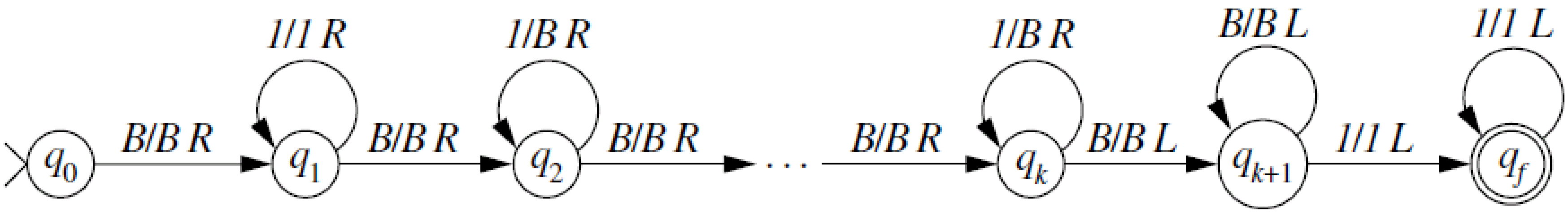


Some TMs for number-theoret. fctns

- **Empty function e(n) = ↑**

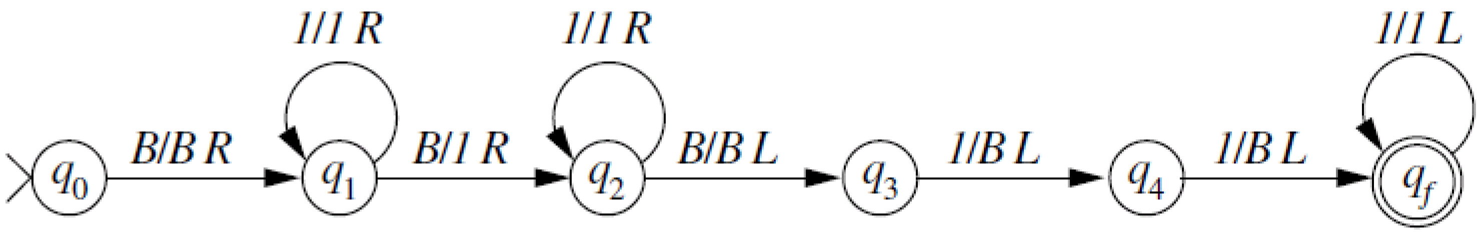
E:


- **Projection $p_i^{(k)}$ defined as $p_i^{(k)}(n_1,\ldots,n_k) = n_i$**
  We give the TM for $p_1^{(k)}$:

$P_1^{(k)}$:


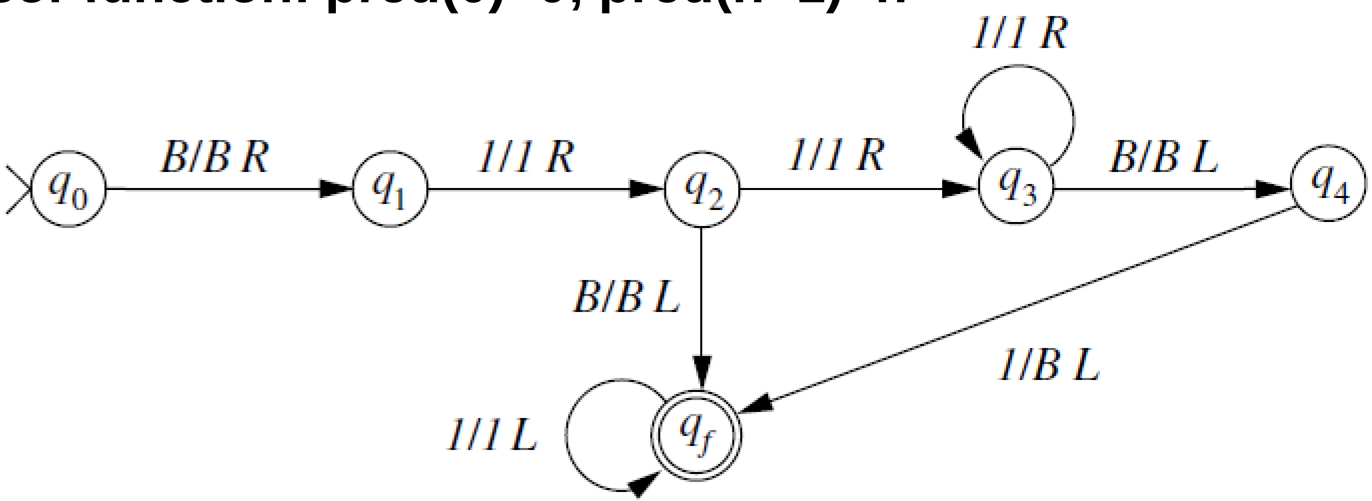
- **Binary addition:**

A:



- **Predecessor function: pred(0)=0; pred(n+1)=n**

D:



TOC: Turing Computable Functions

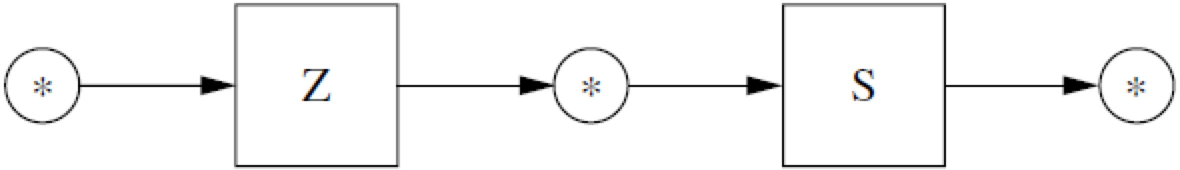
Chapter 9 of [Sudkamp 2006].

1. **Computation of Functions**
2. **Numeric Computation**
3. **Sequential Operation of TMs**
4. **Composition of Functions**
5. **Uncomputable Functions**

Sequential composition

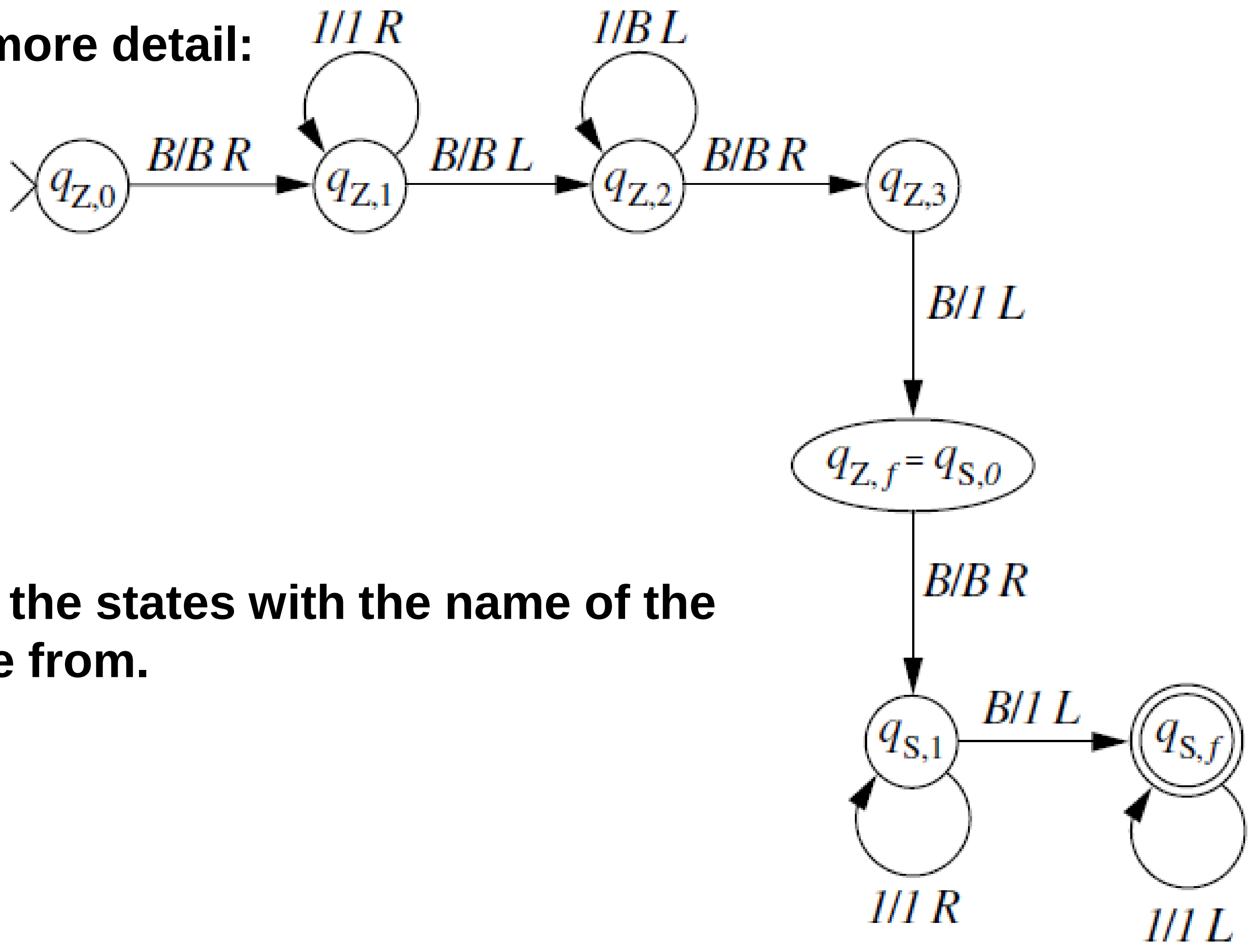
- **E.g., first run "zero" TM, then run "successor" TM**
  Result: Put value "one" on tape.

- **Schematically:**



Sequential composition

- **"one" TM in more detail:**



We subscript the states with the name of the TM they come from.

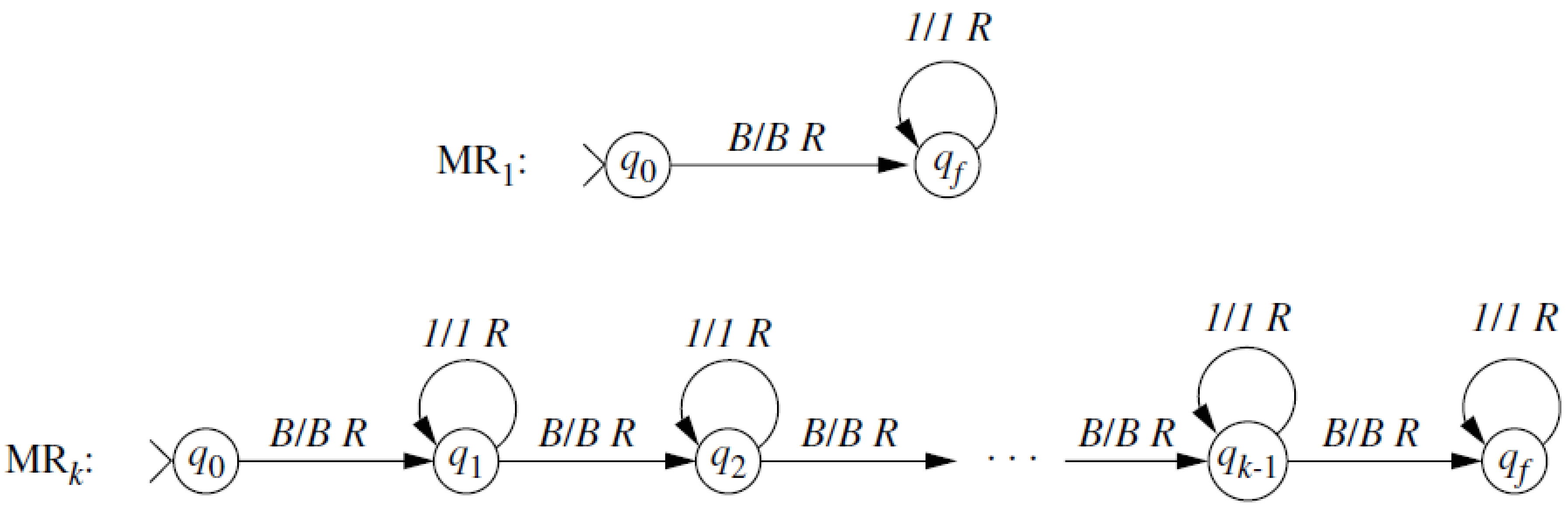
Macros

KNO.E.SIS

- **We call a machine constructed to perform a single simple task a *macro*.**
- **Conditions on TMs for computing functions are slightly relaxed**
  - **Computation does not necessarily start with tape head at position zero.**
  - **First tape symbol read must be a blank.**
  - **Input to be found to the immediate left or right of the starting position.**
  - **There may be several halting states in which a computation may terminate.**
  - **There are no transitions away from any halting state.**

Macros – Examples

- **Move head right through several consecutive natural numbers .**



Macros – Examples

- **Macros can also be described by their effect on the tape.**
  Tape head location: underscore

ML_k (move left):

$$B\overline{n}_1 B\overline{n}_2 B \ldots B\overline{n}_k \underline{B} \qquad k \geq 0$$

$$\updownarrow \qquad\qquad \updownarrow$$

$$\underline{B}\overline{n}_1 B\overline{n}_2 B \ldots B\overline{n}_k B$$

FR (find right):

$$\underline{B} B^i \overline{n} B \qquad i \geq 0$$

$$\updownarrow \qquad \updownarrow$$

$$B^i \underline{B} \overline{n} B$$

Macros – Examples

FL (find left):

$$\begin{array}{ll} B\overline{n}B^i\underline{B} & i \geq 0 \\ \updownarrow \quad \updownarrow & \\ \underline{B}\overline{n}B^iB & \end{array}$$

E_k (erase):

$$\begin{array}{ll} \underline{B}\overline{n}_1B\overline{n}_2B \ldots B\overline{n}_kB & k \geq 1 \\ \updownarrow \qquad\qquad \updownarrow & \\ \underline{B}B \quad \ldots \quad BB & \end{array}$$

Macros – Examples

CPY_k (copy):

$$\underline{B}\bar{n}_1 B\bar{n}_2 B \ldots B\bar{n}_k B\,B\,B \quad \ldots \quad B\,B \qquad k \geq 1$$
$$\updownarrow \qquad\qquad \updownarrow \qquad\qquad \updownarrow$$
$$\underline{B}\bar{n}_1 B\bar{n}_2 B \ldots B\bar{n}_k B\bar{n}_1 B\bar{n}_2 B \ldots B\bar{n}_k B$$

$\text{CPY}_{k,i}$ (copy through i numbers):

$$\underline{B}\bar{n}_1 B\bar{n}_2 B \ldots B\bar{n}_k B\bar{n}_{k+1} \ldots B\bar{n}_{k+i} B\,B \quad \ldots \quad B\,B \qquad k \geq 1$$
$$\updownarrow \qquad\qquad \updownarrow \qquad\qquad \updownarrow \qquad\qquad \updownarrow$$
$$\underline{B}\bar{n}_1 B\bar{n}_2 B \ldots B\bar{n}_k B\bar{n}_{k+1} \ldots B\bar{n}_{k+i} B\bar{n}_1 B\bar{n}_2 B \ldots B\bar{n}_k B$$

Macros – Examples

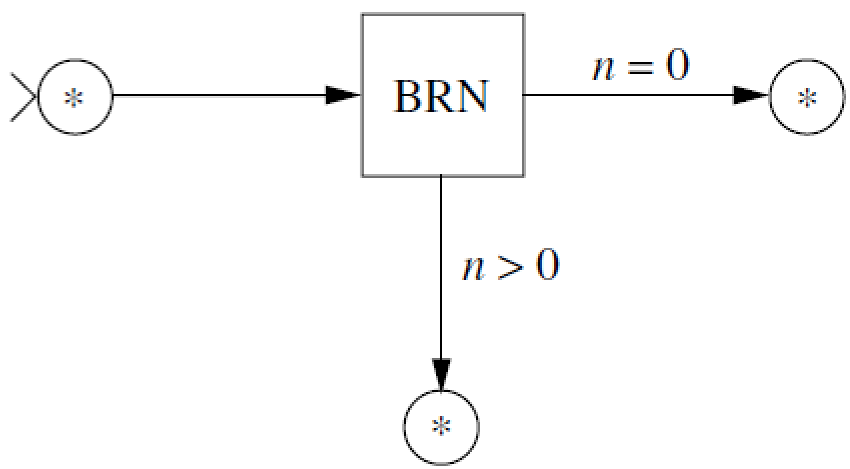
T (translate):

$$\underline{B}\,B^i\,\overline{n}\,B \qquad i \geq 0$$

$$\updownarrow \qquad \updownarrow$$

$$\underline{B}\,\overline{n}\,B^i\,B$$

BRN (branch on zero):

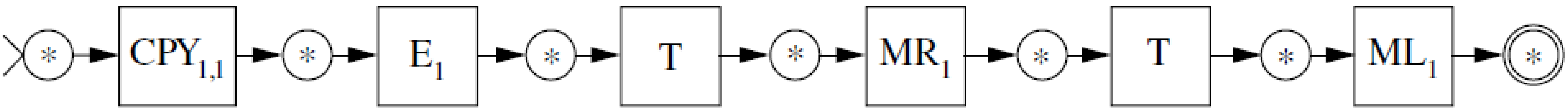


Exercise 26 [no hand-in]

Give a TM for the BRN macro.

Macro composition

INT:



Interchanges the order of two numbers:

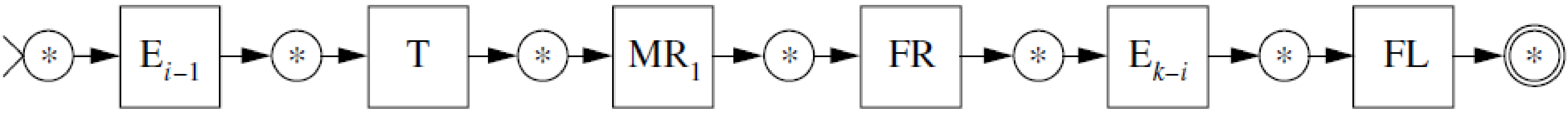
$$\underline{B}\overline{n}B\overline{m}BB^{n+1}B$$

$$\updownarrow \qquad\qquad \updownarrow$$

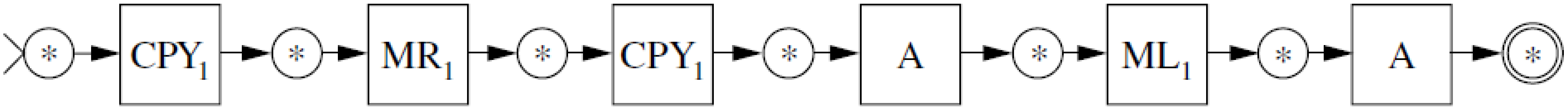
$$\underline{B}\overline{m}B\overline{n}BB^{n+1}B$$

Examples 2.3 and 2.4

- **Projection function $p_i^{(k)}$**

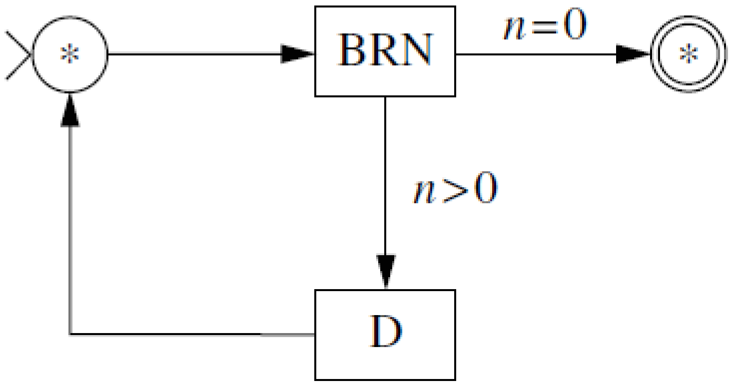


- **f(n) = 3n**



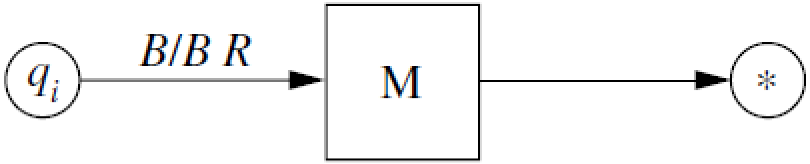
Examples 2.5 and 2.6

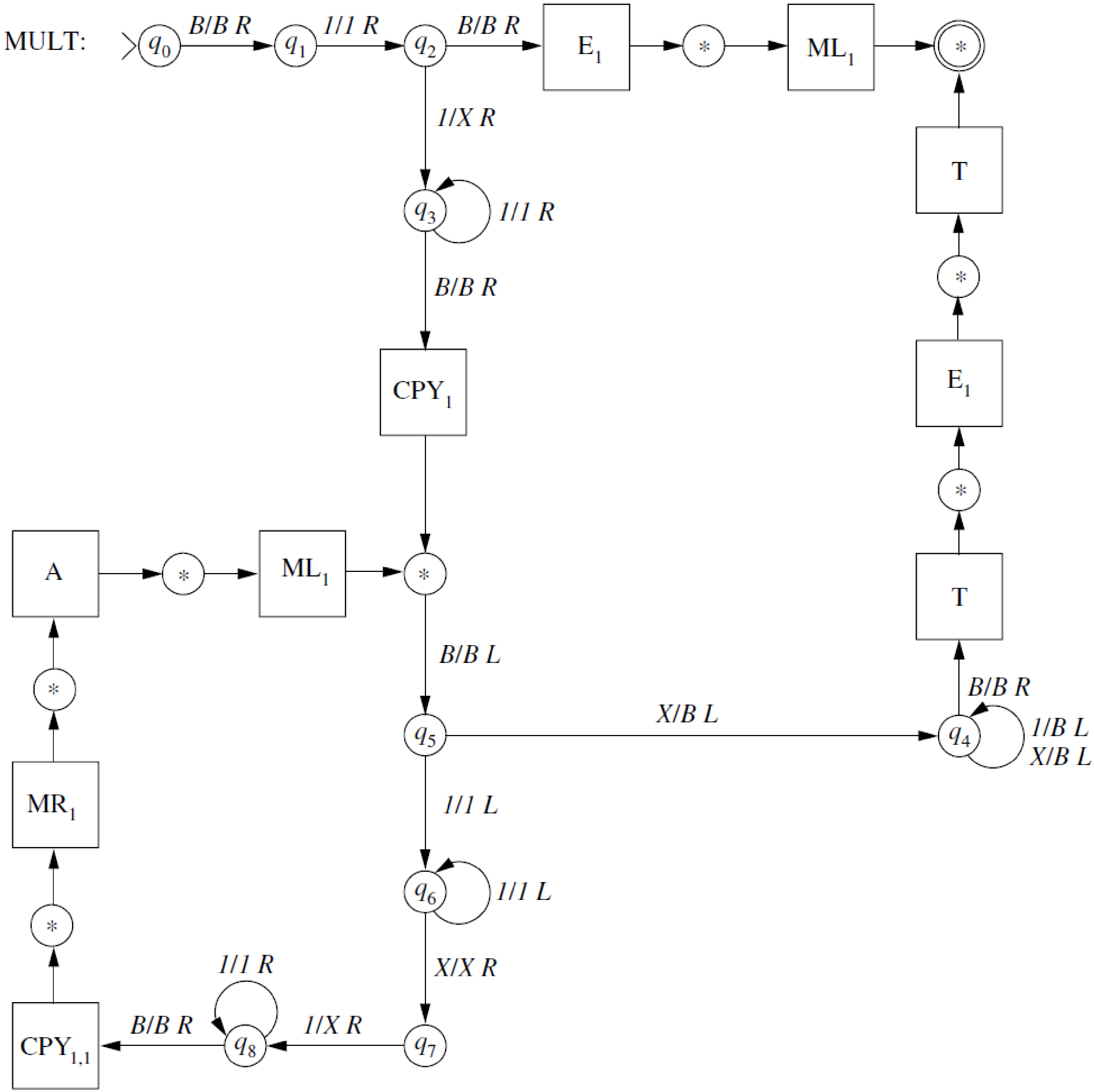
- **One-variable zero function z(n) = 0**



- **MULT (multiplication of natural numbers):**

  We need to mix macros with standard TM transitions for this. Schematically, e.g. identify macro start state with q_i:




MULT:
q0
B/B R
q1
1/1 R
q2
B/B R
E1
*
ML1
*
1/X R
q3
1/1 R
B/B R
CPY1
T
*
E1
*
T
A
*
ML1
*
B/B L
B/B R
q5
X/B L
q4
1/B L
X/B L
*
MR1
1/1 L
q6
1/1 L
*
X/X R
1/1 R
CPY1,1
B/B R
q8
1/X R
q7

TOC: Turing Computable Functions

Chapter 9 of [Sudkamp 2006].

1. **Computation of Functions**
2. **Numeric Computation**
3. **Sequential Operation of TMs**
4. **Composition of Functions**
5. **Uncomputable Functions**

Composition of unary functions

Let g, h be unary number-theoretic functions.

The composition of g with h, written h∘g, is the unary function f: N → N defined by

$$f(x) = \begin{cases} \uparrow & \text{if } g(x) \uparrow \\ \uparrow & \text{if } g(x) = y \text{ and } h(y) \uparrow \\ h(y) & \text{if } g(x) = y \text{ and } h(y) \downarrow \end{cases}$$

Note h∘g(x) = h(g(x)) – which is defined whenever g(x) is defined and h(y) is defined for y=g(x).

Composition of n-ary functions

Let $g_1,\ldots,g_n$ be k-ary number-theoretic functions.
Let h be an n-ary number-theoretic function.

The k-ary function f defined by

$$F(x_1,\ldots,x_k) = h(\, g_1 (x_1,\ldots,x_k), \ldots, g_n(x_1,\ldots,x_k)\,)$$

is called the *composition* of h with $g_1,\ldots,g_n$, written
$f = h \circ (g_1,\ldots,g_n)$.

Example 2.7

Let the following functions be defined as indicated:

$g_1(x,y) = x+y$

$g_2(x,y) = xy$

$g_3(x,y) = x^y$

$h(x,y,z) = x\,(y+z)$

Then $f(x,y) = h \circ (g_1,g_2,g_3) = (x+y)(xy+x^y)$.

Composition by TMs

Assume we have

g_1, a ternary function computed by the TM G_1

g_2, a ternary function computed by the TM G_2

h, a binary function computed by the TM H

$h \circ (g_1,g_2)$ is computed by a TM as follows – we give a trace on input n_1, n_2, n_3.

Trace – composition example

	$\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B$
CPY_3	$\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B\overline{n}_1B\overline{n}_2B\overline{n}_3B$
MR_3	$B\overline{n}_1B\overline{n}_2B\overline{n}_3\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B$
G_1	$B\overline{n}_1B\overline{n}_2B\overline{n}_3\underline{B}\overline{g_1(n_1, n_2, n_3)}B$
ML_3	$\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B\overline{g_1(n_1, n_2, n_3)}B$
$CPY_{3,1}$	$\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B\overline{g_1(n_1, n_2, n_3)}B\overline{n}_1B\overline{n}_2B\overline{n}_3B$
MR_4	$B\overline{n}_1B\overline{n}_2B\overline{n}_3B\overline{g_1(n_1, n_2, n_3)}\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B$
G_2	$B\overline{n}_1B\overline{n}_2B\overline{n}_3B\overline{g_1(n_1, n_2, n_3)}\underline{B}\overline{g_2(n_1, n_2, n_3)}B$
ML_1	$B\overline{n}_1B\overline{n}_2B\overline{n}_3\underline{B}\overline{g_1(n_1, n_2, n_3)}B\overline{g_2(n_1, n_2, n_3)}B$
H	$B\overline{n}_1B\overline{n}_2B\overline{n}_3\underline{B}\overline{h(g_1(n_1, n_2, n_3), g_2(n_1, n_2, n_3))}B$
ML_3	$\underline{B}\overline{n}_1B\overline{n}_2B\overline{n}_3B\overline{h(g_1(n_1, n_2, n_3), g_2(n_1, n_2, n_3))}B$
E_3	$\underline{B}B \quad \ldots \quad B\overline{h(g_1(n_1, n_2, n_3), g_2(n_1, n_2, n_3))}B$
T	$\underline{B}\overline{h(g_1(n_1, n_2, n_3), g_2(n_1, n_2, n_3))}B$

Composition of functions by TMs

Theorem 2.8

The Turing computable functions are closed under the operation of composition.

Proof: skipped.

Example 2.9

The binary function (sum-of-squares)

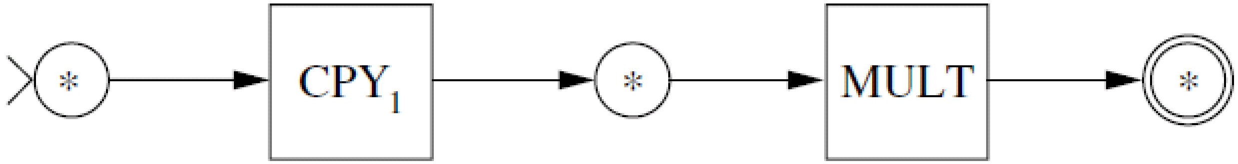
$smsq(n,m) = n^2 + m^2$

is Turing computable.

Proof: It can be written as

$smsq = add \circ (sq \circ p_1^{(2)}, sq \circ p_2^{(2)})$,

where sq is defined by $sq(n) = n^2$. The function add has been shown to be Turing computable earlier. The function sq is computed by the following TM:



Exercise 27 [hand-in]

Show that the relation {(n,m) | n>m} on non-negative integers is Turing-computable.

Exercise 28 [hand-in]

Let F be a TM that computes the total unary number-theoretic function f.

Design a TM that computes the function

$$g(n) = \sum_{i=0}^{n} f(i).$$

TOC: Turing Computable Functions

Chapter 9 of [Sudkamp 2006].

Uncomputable functions

Theorem 2.10

The set of all Turing computable number-theoretic functions is countable.

Proof idea?

Proof

Note: If a set A is countable, then any subset of A is also countable. [Enumerate by skipping the elements which are not in the subset.]

We already know that the set A of all Turing Machines is countable.

Hence, the subset B of A of all Turing Machines which compute number-theoretic functions is countable, say as $M_1, M_2, \ldots$. The function computed by M_i is denoted $f(M_i)$.

By definition, for every computable function there is a TM in B computing it.

Define a subset C of B as follows: M_i is in C if and only if there is no M_j with j>I such that M_i and M_j compute the same function.

C can be enumerated as $N_1, N_2, \ldots$

Hence, all computable functions can be enumerated as $f(N_1), f(N_2), \ldots$

Uncomputable functions

Theorem 2.11

There is a total unary number-theoretic function that is not Turing computable.

Proof idea?

Proof

We show that the set of all a total unary number-theoretic functions is uncountable.

Assume it is countable: f_1, f_2,…

Now define a function by setting $f(n)= f_n(n)+1$.

Then f is a unary number-theoretic function which does not appear in the list. This contradicts the assumption, which, hence, must be wrong.

Thus, the set of all total unary number-theoretic functions is uncountable.

Book chapter 9.6 (omitted)

Chapter 9.6 gives further arguments why high-level programming languages have the same computational power as Turing Machines.

It should be evident from the material which we already covered, so we omit details.

What’s up next

- **We briefly talk about the Chruch-Turing Thesis. [Chap 11]**
- **We talk about undecidability. In particular we give a number of undecidable problems – including the famous Halting Problem. [Chap 12]**
- **We find a mathematical characterization of the functions which are Turing computable. [Chap 13]**