Math 17
Winter 2015
Turing Machines

A Turing machine, considered as if it were a physical object, consists of the following:

1. A linear tape divided into cells, having a leftmost cell, and extending indefinitely (or infinitely) to the right.
2. A read/write head, that can scan one cell at a time, and move to the left or right.
3. Finitely many symbols, which can be written on the tape, one to a cell.
4. Finitely many states, in any one of which the machine may be. One state is designated as an initial state, and one or more as final states.
5. A program, consisting of finitely many instructions of the form

   $$(\text{current state, read symbol} \Longrightarrow \text{write symbol, direction, new state}).$$

   For example, the instruction

   $$(\, q_5,\, 2 \Longrightarrow 1,\, \text{L},\, q_6)$$

   means that when in state q_5 reading the symbol 2 on a cell of the tape, the machine will write 1 on that cell (replacing the 2), move one cell to the left, and shift into state q_6.

   The direction can be L (left), R (right) or S (stay). The write symbol can be the same as the read symbol, and the new state can be the same as the current state. If the read symbol is Λ, that indicates that the cell being scanned is empty (has no symbol in it).

   For each state that is not a final state, and each symbol (including Λ), there is exactly one instruction beginning with that current state and that read symbol. There are no instructions with a final state as the current state.

The specific Turing machines we work with will obey the following conventions:

1. The symbols are $*$, 0, 1, 2, 3, and λ.
2. A cell containing the symbol λ is treated exactly like an empty cell. (We follow the convention of the book here. We could do without this symbol by giving our machine the ability to erase the contents of a cell and leave it empty.)
3. For any given machine, the states are q_1, q_2, q_3, $\ldots$, q_n for some $n \geq 3$, although we may give them other names as well.
   (a) The state q_1 is the initial state.

(b) The final states are q_2 (also called +, YES, or DONE), and q_3 (also called −, NO, or NOTDONE).

(c) The content of the tape always consist of $*$ in the leftmost cell, followed by some number (perhaps zero) of contiguous cells containing symbols 0, 1, 2, or 3, followed by cells that contain λ or are empty.

4. We will also use the following abbreviations.

(a) If an instruction has q as the current state and $()$ as the read symbol, that means this instruction is to be followed when in state q when reading a symbol for which no other instruction is given. (Formally, the machine should be understood to have an instruction of this form for every relevant symbol.)

(b) If an instruction has − as the write symbol, that means the write symbol is the same as the read symbol (or is λ if the read symbol is Λ).

(c) We will not use Λ in instructions. Instead, if a machine has an instruction $(q, \lambda \Longrightarrow s, D, q')$, it is understood also to have the instruction $(q, \Lambda \Longrightarrow s, D, q')$.

We may put the program for a Turing machine into the form of a table, as in the following example.

TURING MACHINE STAR				
current state	read symbol	write symbol	direction	new state
q_1	$*$	$*$	S	q_2
q_1	0	0	L	q_1
q_1	1	1	L	q_1
q_1	2	2	L	q_1
q_1	3	3	L	q_1
q_1	λ	λ	L	q_1
q_1	Λ	λ	L	q_1

TURING MACHINE STAR (USING ABBREVIATIONS)				
current state	read symbol	write symbol	direction	new state
q_1	$*$	$*$	S	q_2
q_1	$()$	−	L	q_1

Turing machine STAR never alters the content of the tape. Started in initial state q_1, with the head positioned on any cell of the tape, this machine moves the head to the left, remaining in state q_1, until the head reaches a cell containing the symbol $*$. (Assuming our conventions are being followed, this will be the leftmost cell of the tape.) When it has reached that cell, the machine leaves the head positioned there, and halts in state q_2 (DONE).

This is one of the "basis machines" described in the textbook, from which more complicated Turing machines are assembled. Its purpose is simply to move the head to the leftmost cell of the tape.

We code a sequence of natural numbers $(a_1, a_2, \ldots, a_n)$ as an input for a Turing machine by starting the machine with the following symbols on the tape (from left to right):

$*$;

0 followed by a_1-many 1's;

0 followed by a_2-many 1's;

...

0 followed by a_n-many 1's;

empty cells (or cells containing λ).

The output is $(a_1, a_2, \ldots, a_n)$ if the machine halts with these symbols on the tape.

For example, if the tape contains symbols $*001011\Lambda\Lambda\Lambda\cdots$, the input (or output) is $(0, 1, 2)$.

If $X \subseteq \mathbb{N}^k$, a Turing machine is a *decision machine* for X if, when started with input $(a_1, \ldots, a_k)$, it halts with output $(a_1, \ldots, a_k)$, in state q_2 (YES) if $(a_1, \ldots, a_k) \in X$, and in state q_3 (NO) if $(a_1, \ldots, a_k) \notin X$.

If $f : \mathbb{N}^k \to \mathbb{N}$ is a function, a Turing machine *computes* f if, when started with input $(a_1, \ldots, a_k)$, it halts in state q_2 (DONE) with output $(a_1, \ldots, a_k, f(a_1, \ldots, a_k))$.

We can also describe Turing machines using a kind of flow chart. In this flow chart, non-final states will be represented by empty circles, and final states by circles containing $+$, $-$, YES, NO, DONE or NOTDONE. From a circle containing START, an arrow will point to the initial state.

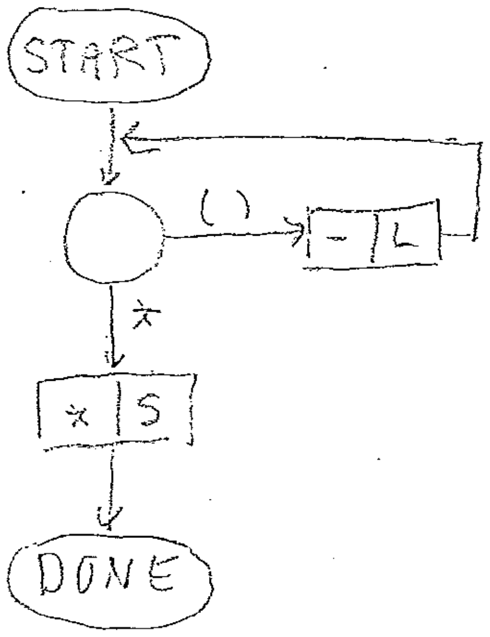
(It is not necessary to label the non-final states. We need to number states only so the program instructions can identify the current and new states. In the flow chart, arrows will indicate the change from state to state.)

From each non-final state (viewed as a current state), arrows labeled with symbols (read symbols) will lead to action boxes, represented by rectangles divided into left and right parts. The left part of an action box will contain a symbol (the write symbol), and the right part will contain a direction (L, R or S). From an action box, an arrow will lead to the new state.

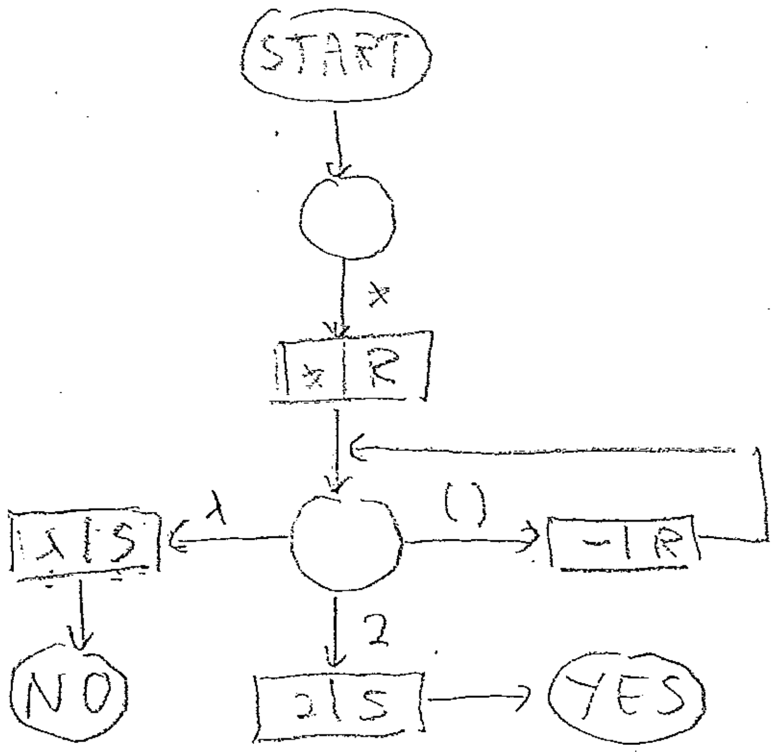
We use our abbreviation conventions in flowchart representations of Turing machines, as well.

On the next page, the machine STAR is pictured in this way, as is another basis machine, THEREIS(2).

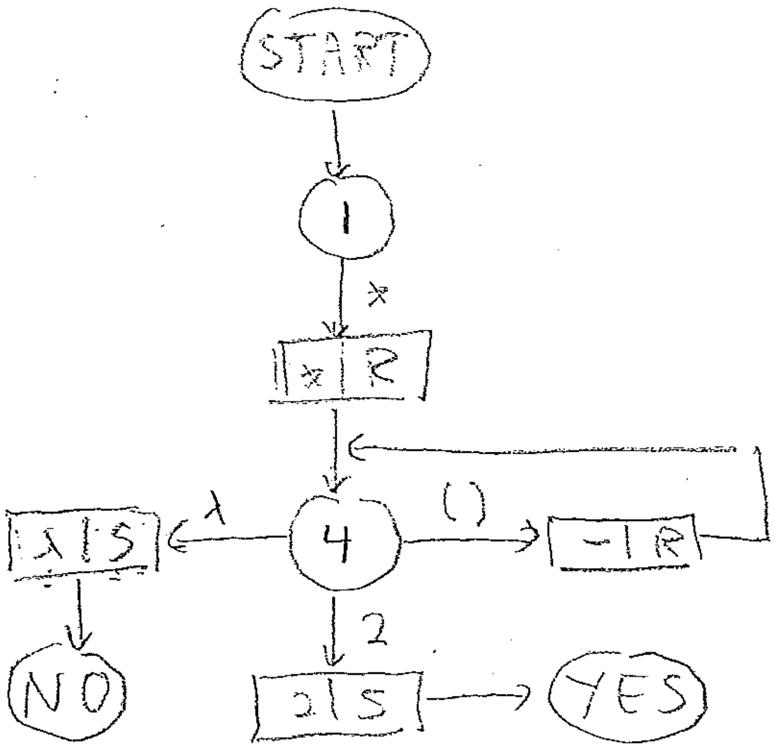
Flowchart Representation of STAR



Flowchart Representation of THEREIS(2)



From the flowchart representation of THEREIS(2), we can make a table representation. We begin by writing the number 1 in the circle to which the arrow from the START circle leads, and writing numbers larger than 3 in the other empty circles. (See the picture below.)



Now, for example, there is an arrow, leading from the circle labeled 1 (representing current state q_1), labeled * (representing the read symbol), to an action block containing * on the left (representing the write symbol) and R on the right (representing the direction), and an arrow leading from that action block to the circle labeled 4 (representing the new state q_4). This tells us that an instruction

$$(q_1, * \Longrightarrow *, \text{R}, q_4)$$

is in the program for THEREIS(2), or that the line

q_1	*	*	R	q_4

is in the table for THEREIS(2).

Similarly, there is an arrow, leading from the circle labeled 4, labeled 2, to an action block containing 2 on the left and S on the right, and an arrow leading from that action block to a circle labeled YES. This gives the instruction

$$(\ q_4,\ 2 \Longrightarrow 2,\ \mathrm{S},\ q_2)$$

or the line in a table

q_4	2	2	S	q_2

Here is the table representation for THEREIS(2).

THEREIS(2)				
current state	read symbol	write symbol	direction	new state
q_1	$*$	$*$	R	q_4
q_1	()	$-$	L	q_1
q_4	λ	λ	S	q_3
q_4	2	2	S	q_2
q_4	()	$-$	R	q_4

By examining either the table or the flowchart representation, we can see what THEREIS(2) does. It does not alter the contents of the tape. First it moves the head to the leftmost cell (always assuming our conventions are followed, and this is the only cell containing $*$). Then it moves the head right until it encounters either a cell containing 2, in which case it stops in state YES, or an empty cell, in which case it stops in state NO.

That is, THEREIS(2) positions the head at the first cell containing 2, unless there is no such cell, in which case it positions the head at the first empty cell. It indicates whether or not it found a cell containing 2 by halting in state YES or NO.

Written Exercises due Friday, January 30

Here is a table representing a Turing machine, Turing machine A.

TURING MACHINE A				
current state	read symbol	write symbol	direction	new state
q_1	$*$	$*$	R	q_4
q_1	$()$	$-$	L	q_1
q_4	0	0	R	q_6
q_4	$()$	$-$	S	q_4
q_6	1	1	R	q_5
q_6	λ	λ	S	q_2
q_5	1	1	R	q_6
q_5	λ	λ	S	q_3

On the last page is a flowchart representation of a Turing machine, Turing machine B.

1. Draw a flowchart representation of Turing machine A.

2. Turing machine A is a decision machine for a subset $X \subseteq \mathbb{N}$. What is X?

3. Assuming Turing machine A is started with an input that is a sequence of length 1, where is the head when the machine halts?

   What does Turing machine A do if started with an input that is not a sequence of length 1? (Does it halt? If so, in what state, with the head in what position, and with what written on the tape? If not, how does the head move, and what, if anything, does it write on the tape? Do the answers to these questions depend on what is initially written on the tape, or on where the head is initially positioned?)

4. Give a table representation of Turing machine B.

5. Turing machine B computes a function $f : \mathbb{N}^3 \longrightarrow \mathbb{N}$. What is that function?

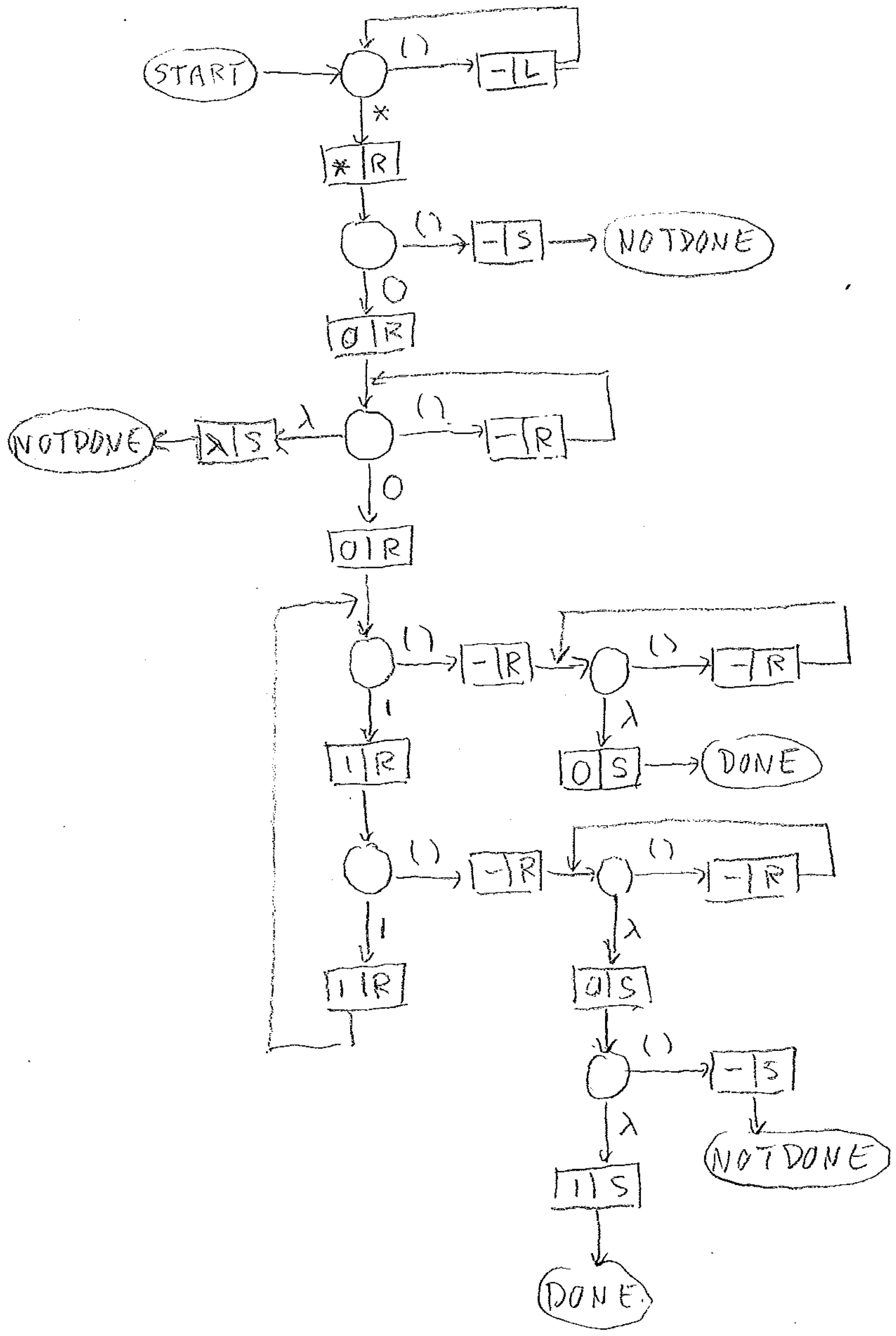
6. Assuming Turing machine B is started with an input that is a sequence of length 3, where is the head when the machine halts?

   What does Turing machine B do if started with an input that is not a sequence of length 3?

7. Design a Turing machine DELETE that, when started with input $(a_1, \ldots, a_{n-1}, a_n)$ on the tape, halts in state DONE with output $(a_1, \ldots, a_{n-1})$.

   Here n can be any positive natural number. That is, your machine must work properly regardless of the length of the input sequence. It doesn't matter what your machine does if the initial tape contents do not have the correct form to input a sequence.

   You may represent your Turing machine using a list of instructions, a table, or a flowchart representation, whichever you prefer.


START
()
-|L
*
*|R
()
-|S
NOTDONE
0
0|R
λ
λ|S
NOTDONE
()
-|R
0
0|R
()
-|R
()
-|R
1
1|R
λ
0|S
DONE
()
-|R
()
-|R
1
1|R
λ
0|S
()
-|S
NOTDONE
λ
1|S
DONE