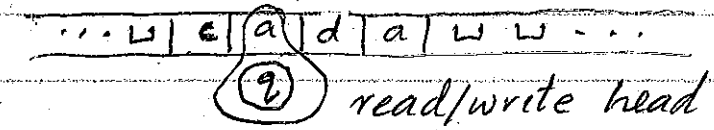
Ch. 5 – TURING MACHINES & COMPUTATIONS

§1 Turing Machines & their basic operations

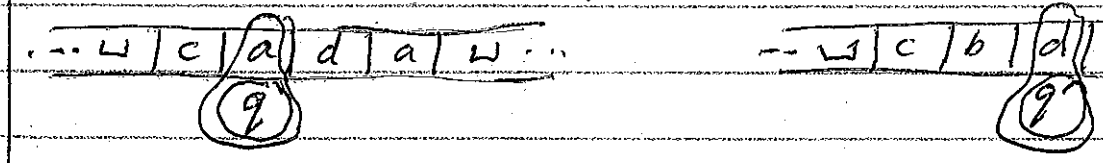


two-way infinite tape

Def. A Turing Machine (TM) is a 7-tuple $M = \langle Q, T, U, \Delta, q_0, \sqcup, \mathcal{A} \rangle$ where
Q is a finite set of control states.
T is an alphabet called the input alphabet
$U \supseteq T$ is an alphabet called the tape alphabet
$q_0 \in Q$ is a specially designated initial state
$\sqcup \in U - T$ is a special character called the blank symbol
$\mathcal{A} \subseteq Q$ is a set of designated accepting states
and Δ is a relation from $Q \times U$ to $Q \times U \times \{L, R\}$ called the transition relation

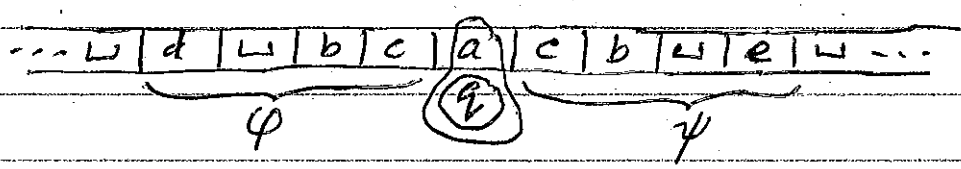
If $\langle \langle q, a \rangle, \langle q', b, R \rangle \rangle \in \Delta$, we will write
$\langle q, a \rangle \longrightarrow \langle q', b, R \rangle$ or $q \xrightarrow{a/b/R} q'$.
This means that if the head is in state q and reads and "a", then it will replace the "a" by a "b", move one square to the right, and then change the control state of the head to q'. The transition $q \xrightarrow{a/b/L} q'$ means the same thing except that we move one square to the left.

$$q \xrightarrow{a/b/R} q'$$



Def A <u>configuration</u> in a TM M is an ordered pair $\langle q, \varphi\underline{\psi} \rangle$ where q is the control state of the head, φ = non-blank portion of the tape to the left of the head, and ψ = the non-blank portion of the tape below the head and to the right of it. The underlined ψ means that the position of the head is above the first character in ψ.

So a configuration $\langle q, \varphi\underline{\psi} \rangle$ is just an instantaneous description of the tape of the TM.

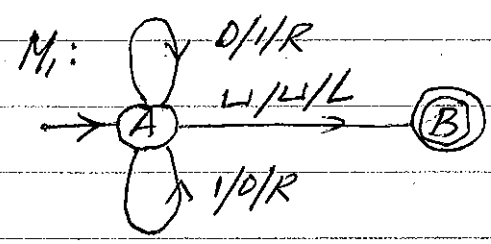


Configuration is $\langle q, d\sqcup bc\underline{a}cb\sqcup e \rangle$
The $\ldots\sqcup$ and $\sqcup\ldots$ means that these portion of the tapes are all blanks. Sometime we include some of these blanks (from the blank portions) to show what the head will see if it moves to the right or to the left.

Ex. 1 Let M_1 be the TM that is defined as follows.
$Q = \{A, B\}$, $T = \{0, 1\}$, $U = \{0, 1, \sqcup\}$, $q_0 = A$, $\mathcal{A} = \{B\}$
and Δ consists of the transitions

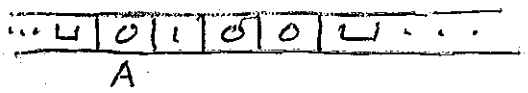
$$\langle A, 0\rangle \to \langle A, 1, R\rangle, \quad \langle A, 1\rangle \to \langle A, 0, R\rangle, \quad \langle A, \sqcup\rangle \to \langle B, \sqcup, L\rangle$$

We can represent M_1 by a <u>transition graph</u> as shown to the right.

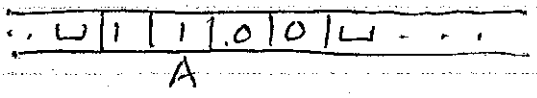


Let us show what happens if we start M_1 with the string 0100 as input.

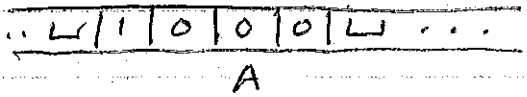
initial tape



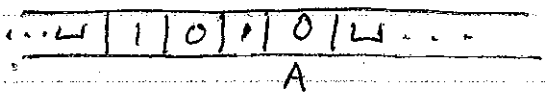
$\langle A, \underline{0}100 \rangle$ (initial config.)



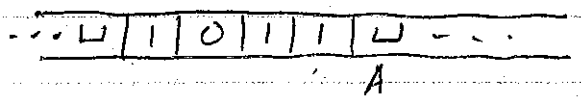
$\vdash \langle A, 1\underline{1}00 \rangle$



$\vdash \langle A, 10\underline{0}0 \rangle$

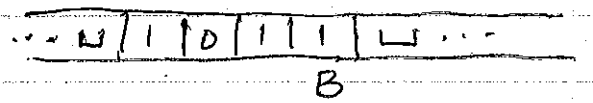


$\vdash \langle A, 101\underline{0} \rangle$



$\vdash \langle A, 1011\underline{\sqcup} \rangle$

final tape



$\vdash \langle B, 101\underline{1}\sqcup \rangle$ (halting config.)

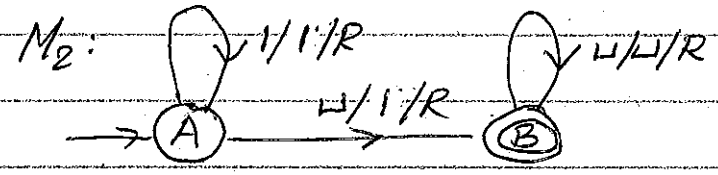
At this point, no more transitions can be executed and we say that the TM <u>halts</u>

<u>Def.</u> A <u>halting configuration</u> in a TM is a configuration from which no transition can be executed. A TM is said to <u>halt</u> if it reaches a halting configuration. An <u>initial</u> configuration is one in which the control state is the <u>initial state</u>.

<u>Def.</u> A <u>computation</u> by a TM is a sequence of transition that takes M from an initial configuration to a halting configuration. We denote it by $\langle q_0, \omega \rangle \vdash^* \langle q_F, \varphi\underline{\psi} \rangle$

<u>Ex. 2</u> $\langle A, \underline{0}11 \rangle \vdash \langle A, 1\underline{1}1 \rangle \vdash \langle A, 10\underline{1} \rangle \vdash \langle A, 100\underline{\sqcup} \rangle \vdash \langle B, 10\underline{0} \rangle$ is a computation by the TM M_1 from Ex. 1 and we write $\langle A, \underline{0}11 \rangle \vdash^* \langle B, 10\underline{0} \rangle$

Note: A TM will not always halt on a given input. For example, the TM M_2 below with input alphabet $\{1\}$, will never halt on any input.



With input $\varphi = 11$, this TM will move as follows.

$\langle A, \underline{1}1\rangle \vdash \langle A, 1\underline{1}\rangle \vdash \langle A, 11\underline{\sqcup}\rangle \vdash \langle B, 111\underline{\sqcup}\rangle$
$\vdash \langle B, 1111\underline{\sqcup}\rangle \vdash \langle B, 11111\underline{\sqcup}\rangle \vdash \cdots$

Def. A TM M is said to be a <u>Deterministic</u> TM if from each configuration (that is reachable from the initial state) at most one transition is executable.

Both of the examples M_1 & M_2 are deterministic TMs. In fact most of the TMs we give will be deterministic ones — because they are a lot easier to understand. We have not said anything yet about the accepting states of a TM — but we will do so in the next section. Also, whenever we give a TM as a transition graph, we have to make sure that we specify the input alphabet.

§2. Turing Machines as language recognizers

Def. Let M be a TM with input alphabet T. The language recognized by M is defined by

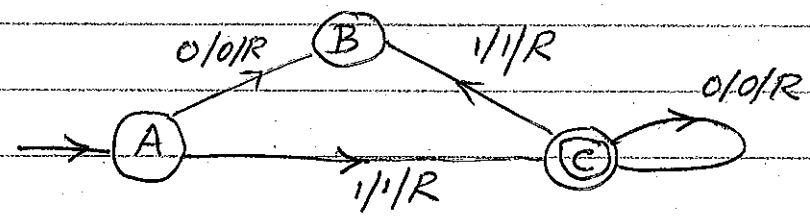
$$L(M) = \{w \in T^* : \langle q_0, \underline{w}\rangle \vdash^* \langle q_F, \varphi\underline{\psi}\rangle \text{ is a halted computation with } q_F \in A(M)\}$$

If M halts in an accepting state with $w \in T^*$ as input, we say that M <u>accepts</u> the string w. So $L(M)$ = set of all strings in T^* that are accepted by M.

<u>Note</u>: For a given string $w \in T^*$, M may <u>not</u> halt with w as input, or M may halt in a non-accepting state with w as input. In both of these cases, we say that M is <u>not</u> accepted (or rejected) by the TM M.

<u>Ex. 1</u> Let $T = \{0,1\}$. Find a TM M_1 which recognizes the language $L_1 = \underline{1}0^*$.

M_1:



Lets us look at how M_1 behaves with the inputs (a) 100 (b) 101 (c) 011 (d) λ

(a) $\langle A, \underline{1}00\rangle \vdash \langle C, 1\underline{0}0\rangle$
$\vdash \langle C, 10\underline{0}\rangle$
$\vdash \langle C, 100\underline{\sqcup}\rangle$ halts
accepted

(b) $\langle A, \underline{1}01\rangle \vdash \langle C, 1\underline{0}1\rangle$
$\vdash \langle C, 10\underline{1}\rangle$
$\vdash \langle B, 101\underline{\sqcup}\rangle$ halts
rejected.

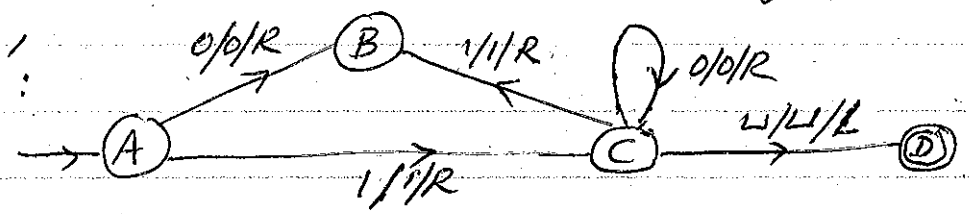
(c) $\langle A, \underline{0}11\rangle \vdash \langle B, 0\underline{1}1\rangle$ halts
rejected

(d) $\langle A, \underline{\sqcup}\rangle \vdash$ halts
rejected

Notice that M_1 did not have to process the whole string in (c) before it rejected 011. Notice also that λ is represented by the completely blank tape.

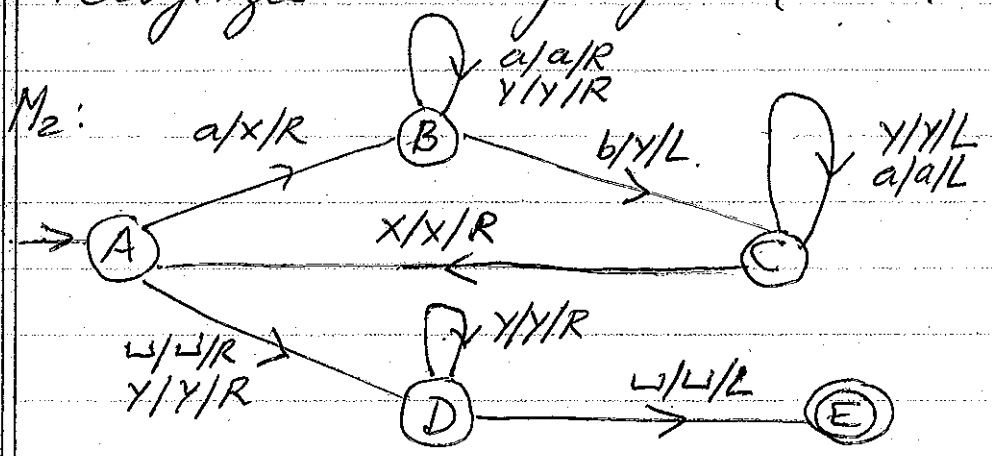
There are other TMs which recognizes the language $L_1 = 1\underline{0}^*$ and usually we try to have only one accepting state with no transitions exiting that state. Here is another TM, M_1', which recognizes $1\underline{0}^*$

M_1':



<u>Ex. 2</u> Let $T = \{a, b\}$. Find a TM M_2 which recognizes the language $\{a^k b^k : k \geq 0\}$

M_2:



Let us look at how M_2 behaves with the inputs
(a) ab (b) aab (c) λ

(a) $\langle A, \underline{a}b \rangle \vdash \langle B, X\underline{b} \rangle$
$\vdash \langle C, \underline{X}Y \rangle$
$\vdash \langle A, X\underline{Y} \rangle$
$\vdash \langle D, XY\underline{\sqcup} \rangle$
$\vdash \langle E, X\underline{Y}\sqcup \rangle$
halts, accepted

(b) $\langle A, \underline{a}ab \rangle \vdash \langle B, X\underline{a}b \rangle$
$\vdash \langle B, Xa\underline{b} \rangle$
$\vdash \langle C, X\underline{a}Y \rangle$
$\vdash \langle C, \underline{X}aY \rangle$
$\vdash \langle A, X\underline{a}Y \rangle$
$\vdash \langle B, XX\underline{Y} \rangle$
$\vdash \langle B, XXY\underline{\sqcup} \rangle$
halts, rejected

(c) $\langle A, \underline{\sqcup} \rangle \vdash \langle D, \sqcup\underline{\sqcup} \rangle$
$\vdash \langle E\ \underline{\sqcup}\sqcup \rangle$
halts, accepted.

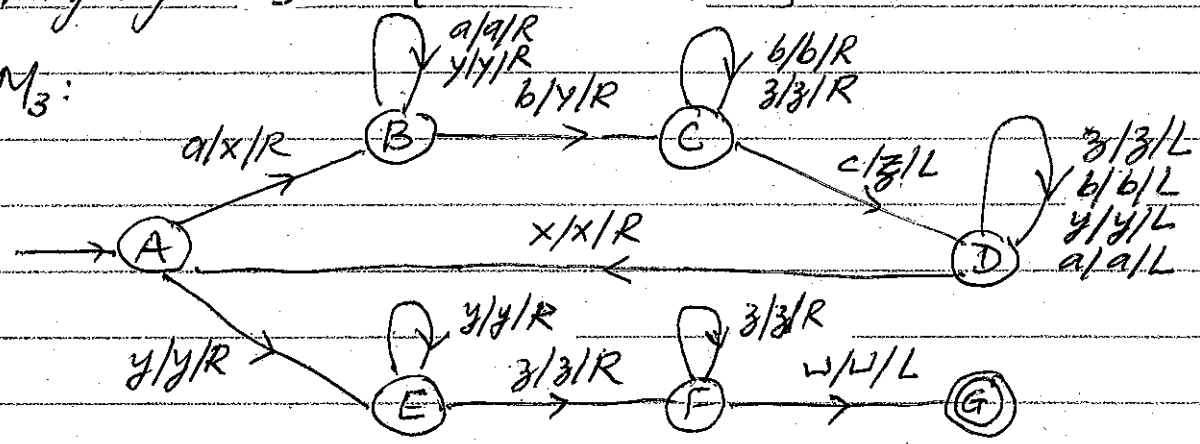
(+)

<u>Extra H.W. Problems</u>: Show how the TM M_2 behaves with the inputs

(a) abb (b) aba (c) aabb (d) ba.

<u>Ex. 3</u> Let $T = \{a, b, c\}$. Find a TM M_3 which recognizes the language $L_3 \doteq \{a^k b^k c^k : k \geq 1\}$

M_3:



Let us see how M behaves with the inputs

(a) abc (b) aabc (c) λ

(a)	(b)	(c)
$\langle A, \underline{a}bc\rangle$	$\langle A, \underline{a}abc\rangle$	$\langle A, \underline{\sqcup}\rangle$
$\vdash \langle B, x\underline{b}c\rangle$	$\vdash \langle B, x\underline{a}bc\rangle$	halts,
$\vdash \langle C, xy\underline{c}\rangle$	$\vdash \langle B, xa\underline{b}c\rangle$	rejected.
$\vdash \langle D, x\underline{y}z\rangle$	$\vdash \langle C, xay\underline{c}\rangle$	
$\vdash \langle D, \underline{x}yz\rangle$	$\vdash \langle D, xa\underline{y}z\rangle$	
$\vdash \langle A, x\underline{y}z\rangle$	$\vdash \langle D, x\underline{a}yz\rangle$	
$\vdash \langle E, xy\underline{z}\rangle$	$\vdash \langle D, \underline{x}ayz\rangle$	
$\vdash \langle F, xyz\underline{\sqcup}\rangle$	$\vdash \langle A, x\underline{a}yz\rangle$	
$\vdash \langle G, xyz\underline{\sqcup}\rangle$	$\vdash \langle B, x\underline{x}yz\rangle$	
accepted	$\vdash \langle B, xx\underline{y}z\rangle$	
	rejected	

<u>Def.</u> A language $L \subseteq T^*$ is said to be <u>Turing semi-decidable</u> if we can find a TM M with input alphabet T such that $L(M) = L$.

Def. A language $L \subseteq T^*$ is said to be <u>Turing-decidable</u> if we can find a TM M with input alphabet T such that M halts in an accepting state for each $w \in L$, and M halts in a non-accepting state for each $w \in T^* - L$.

Note: There is a difference between the concepts of Turing-decidable and Turing semi-decidable. To be Turing decidable, we have to find a TM M which always halts on each input from T — and if $w \in L$, we will have to halt in an accepting state, but if $w \notin L$ we will have to halt in a non-accepting state.

To be Turing semi-decidable, the TM M we find does not always have to halt. It only has to halt in an accepting state if $w \in L$. But if $w \notin L$, then M halt in a non-accepting state or M may fail to halt. This difference makes all Turing-decidable languages Turing semi-decidable, but the converse is not true.

<u>Extra H.W. Problems</u>: Show how the TM M_3 behaves on the following inputs

(a) aabbcc (b) aabbc (c) abca
(d) abbcc (e) abcc (f) bac

Ex.4 The languages L_1, L_2 & L_3 from Ex. 1–3 are all Turing-decidable languages.

§3. Turing Machines as computational devices.

Def. Let T be an alphabet. A <u>partial function</u> $f: T^* \to T^*$ is a function $f: D \to T^*$ where D is a subset of T^*. If $D = T^*$, then we say that f is a <u>total function</u>.

Def. Let $f: T^* \to T^*$ be a partial function. We say that f is <u>Turing-computable</u> if we can find a TM M with input alphabet T such that we have $w \in Dom(f) \Leftrightarrow \langle q_0, \underline{w} \rangle \vdash^* \langle q_F, \varphi \underline{f(w)} \rangle$ is a halted computation with $q_F \in \mathcal{A}(M)$. Here the head is over the first character of $f(w)$ and φ is consider the "rough work".

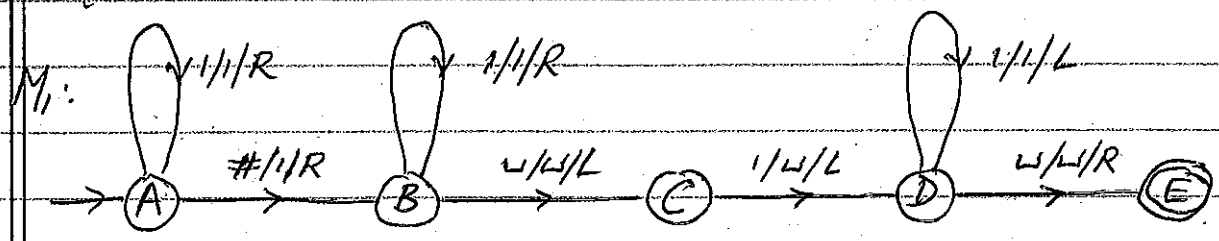
<u>Representations of numbers</u>. To keep matters simple, we will restrict ourselves to partial functions from $\mathbb{N}^k$ to $\mathbb{N}$. Now there are many ways of representing a number as a string.

1. <u>Basic ten representation</u>: Alphabet $T = \{0, 1, 2, \ldots, 9\}$
   $a_n a_{n-1} \cdots a_0$ represents $a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \cdots + a_1 \cdot 10^1 + a_0 \cdot 10^0$
   So $(247)_{10} = 2 \cdot (10)^2 + 4 \cdot (10)^1 + 7 \cdot 10^0$

2. <u>Binary (Base 2) representation</u>: Alphabet $T = \{0, 1\}$
   $a_n a_{n-1} \cdots a_1 a_0$ represents $a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \cdots + a_1 \cdot 2^1 + a_0$
   So $(1101)_2 = 1 \cdot (2)^3 + 1 \cdot (2)^2 + 0 \cdot (2)^1 + 1 \cdot (2)^0 = (13)_{10}$

3. <u>Monadic Representation</u>: Alphabet $T = \{1\}$
   $111 \cdots 1$ (n times) repr. n. So $(4)_{10} = 1111$ & $(0)_{10} = \lambda$

Ex. 1 Find a TM M_1 with input alphabet $T = \{1, \#\}$ which computes the total function $f: \mathbb{N}^2 \to \mathbb{N}$, $f(k,m) = k+m$ in Monadic notation.



Here the "#" is used like a comma to separate the two inputs. Let us see how M_1 behaves when trying to compute $f(1,2) = 1+2$. The input will be $1\#11$.

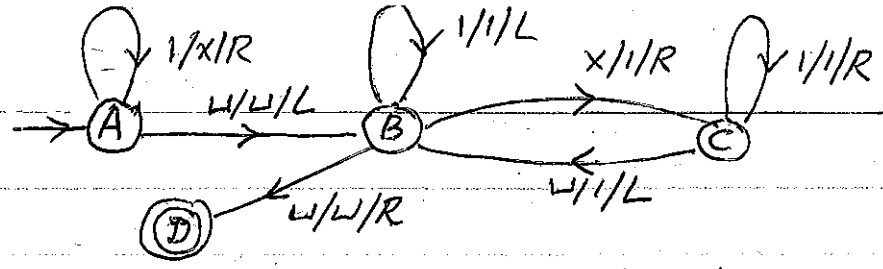
$\langle A, \underline{1}\#11\rangle \vdash \langle A, 1\underline{\#}11\rangle \vdash \langle B, 11\underline{1}1\rangle \vdash \langle B, 111\underline{1}\rangle$
$\vdash \langle B, 1111\underline{\sqcup}\rangle \vdash \langle C, 111\underline{1}\sqcup\rangle \vdash \langle D, 11\underline{1}\sqcup\sqcup\rangle$
$\vdash \langle D, \underline{1}11\rangle \vdash \langle D, \underline{1}11\rangle \vdash \langle D, \underline{\sqcup}111\rangle \vdash \langle E, \underline{1}11\rangle$

Let's see what happens if we try to compute $f(0,0) = 0+0$. The input will be $\underline{\#}$
$\langle A, \underline{\#}\rangle \vdash \langle B, 1\underline{\sqcup}\rangle \vdash \langle C, \underline{1}\sqcup\rangle \vdash \langle D, \underline{\sqcup}\sqcup\rangle \vdash \langle E, \sqcup\underline{\sqcup}\rangle$.
So our answer is λ which in monadic notation means 0. Hence $f(0,0) = 0+0 = 0$.

<u>Extra H.W Problems</u>: Show how the TM M_1 computes
(a) $1+1$ (b) $0+2$ (c) $2+0$ (d) $2+1$.

Ex. 2 Find a TM M_2 with input alphabet $T = \{1\}$ which computes the total function $f: \mathbb{N} \to \mathbb{N}$, $f(n) = 2n$ in monadic notation.

Ex.2 M_2:



The idea behind the construction of M_2 is below.

1. Replace each "1" in the input by an x. Then
2. Find the right most x & replace it by a "1". Then
3. Go to the beginning of the blank region on the right and create a "1" there.
4. Repeat steps 2 & 3 until there are no more x's.
5. Finally move the head over the leftmost 1.

Let us see how M_2 computes $f(2) = 2(2)$. Input = 11

$\langle A, \underline{1}1\rangle \vdash \langle A, x\underline{1}\rangle \vdash \langle A, xx\underline{\sqcup}\rangle \vdash \langle B, x\underline{x}\rangle \vdash \langle C, x1\underline{\sqcup}\rangle$
$\vdash \langle B, x\underline{1}1\rangle \vdash \langle B, \underline{x}11\rangle \vdash \langle C, 1\underline{1}1\rangle \vdash \langle C, 11\underline{1}\rangle$
$\vdash \langle C, 111\underline{\sqcup}\rangle \vdash \langle B, 11\underline{1}1\rangle \vdash \langle B, \underline{1}111\rangle \vdash \langle B, \underline{1}111\rangle$
$\vdash \langle B, \underline{\sqcup}1111\rangle \vdash \langle D, \sqcup\underline{1}111\rangle$. So $f(2) = 4$

Let us also see how M_2 computes $f(0)$. Input = λ

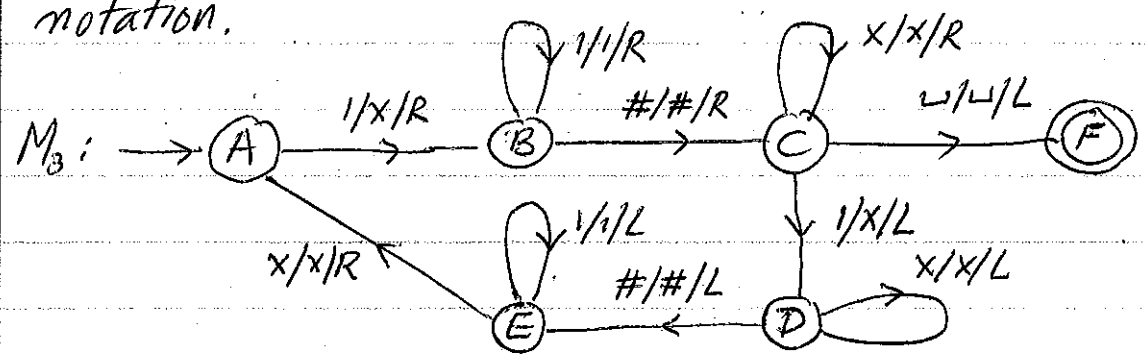
$\langle A, \underline{\sqcup}\rangle \vdash \langle B, \underline{\sqcup}\sqcup\rangle \vdash \langle D, \sqcup\underline{\sqcup}\rangle$. So $f(0) = 0$.

<u>Extra H.W. Problems</u>: Show how M_2 computes
(a) $f(1)$ (b) $f(3)$

<u>Def.</u> Let R be a binary relation on $\mathbb{N}$. We say that R is <u>Turing semi-decidable</u> if we can find a TM M such that for the input $\langle m,n\rangle$, M halts in an accepting state if $\langle m,n\rangle \in R$; and M halts in a non-accepting state or M fails to halt if $\langle m,n\rangle \notin R$. (Note: $\langle m,n\rangle \in R$ is the same as saying mRn.)

Def. We say that R is Turing-decidable if we can find a TM M such that for the input $\langle m,n\rangle$,
M halts in an accepting state if $\langle m,n\rangle \in R$; and
M halts in a non-accepting state if $\langle m,n\rangle \notin R$.

Ex.3 Show that the relation R defined by mRn if $m>n$, is Turing decidable using monadic notation.



Here, again in M_3, as in M_1 — we use "#" to separate the input. The TM M_3 shows that R is Turing-decidable. Let us see how M behaves when $\langle 1,2\rangle$ is the input. In monadic notation, this input is $1\#11$.

$\langle A, \underline{1}\#11\rangle \vdash \langle B, X\underline{\#}11\rangle \vdash \langle C, X\#\underline{1}1\rangle \vdash \langle D, X\underline{\#}X1\rangle$
$\vdash \langle E, \underline{X}\#X1\rangle \vdash \langle A, X\underline{\#}X1\rangle$ halts in non-accepting state. Hence $1 \not> 2$.

Let's see what happens when $\langle 2,0\rangle$ is the input.
$\vdash \langle A, \underline{1}1\#\sqcup\rangle \vdash \langle B, X\underline{1}\#\sqcup\rangle \vdash \langle B, X1\underline{\#}\sqcup\rangle \vdash \langle C, X1\#\underline{\sqcup}\rangle$
$\vdash \langle F, X1\underline{\#}\sqcup\rangle$ halts in an accepting state. $\therefore 2>0$.

<u>Extra H.W. Problems</u>. Show how M_3 behaves with the inputs
(a) $\langle 2,1\rangle$ (b) $\langle 1,1\rangle$ (c) $\langle 1,0\rangle$ (d) $\langle 0,2\rangle$ (e) $\langle 3,2\rangle$

§4. Universal TMs, Halting Problem & Busy-beaver function

Even though, TMs are extremely slow & cumbersome, it can be shown that for any task for which there is an algorithm, a TM can be constructed to do the same task. TMs can be constructed to compute all of the familiar functions such as

(a) $f_1(m,n) = m.n$ (b) $f_2(n) = 2^n$ (c) $f_3(n) = n!$

(d) $f_4(m,n) = m^n$ (e) $f_5(n) = \lfloor \sqrt{n} \rfloor$ (g) $f_6(n) = \lceil \sqrt{n} \rceil$

Here $\lfloor x \rfloor$ = largest integer $\leq x$ (floor function) and $\lceil x \rceil$ = smallest integer $\geq x$ (ceiling function).

<u>Universal Turing Machines</u>: There are even Turing machines, M_u such that if we are given an arb. TM, M, and an arbitrary input, w for M, we can simulate the action of M on w and obtain the correct output by using M_u. Such Turing Machines, M_u, are called <u>Universal Turing Machines</u> (UTMs). The trick is to code everything in M & w as strings from the input alphabet of M_u and then show the configurations that we obtain once M starts operating on w.

M_u: $\ldots \sqcup \mid \# \mid c(M) \mid \# \mid c(w) \mid \# \mid c(\text{init config. of } M) \mid \sqcup \ldots$

code of M ; code of w

$\mid \# \mid c(M) \mid \# \mid c(w) \mid \# \mid c(\text{final config. of } M) \mid \sqcup \ldots$

The TM, M_u, will know what M will do at each step with the input w — because it can run back and check in $c(M)$ & $c(w)$ whenever it needs to.

Def The <u>Halting Problem</u> is the following question.

Is there a TM, H, such that for any arb. TM, M, and any arb. input, w for M, if we use $c(M)\#c(w)$ as input,

H will halt in an accepting state, if M halts on w;

& H will halt in a non-accepting state, if M doesn't halt on w

[Here $c(M)$ & $c(w)$ are codings of M & w into the alphabet of H and "#" is used to separate inputs.]

The answer to the Halting Problem is NO. Even though the proof is important, it is not simple enough to present in class. So we will refer the reader to any of the standard textbooks. One might think that a UTM, M_u, can serve as H – but this is not so.

M_u will halt in an accepting state if M halts on w but M_u will <u>not</u> halt if M doesn't halt on w since M_u just simulates the action of M on w.

<u>Theorem 1</u>: Let R be the binary relation on $\mathbb{N}$ which is defined as follows.

$\langle c(M), c(w)\rangle \in R$ iff M halts with w as input. Then R is <u>not</u> a Turing-decidable relation. Here $c(M)$ & $c(w)$ are the codings of M & w as positive integers.

The proof Theorem 1 just boils down to the fact that the answer to the Halting problem is NO. Indeed, if R was Turing-decidable, then we would be able to find a TM H with exactly the properties in the Halting Problem

So we have found a binary relation R on $\mathbb{N}$ which is <u>not</u> Turing-decidable. A universal TM will readily show that R is Turing semi-decidable, but R^c is not even Turing semi-decidable. It is natural to ask if there is a function $f: \mathbb{N} \to \mathbb{N}$ which is not Turing-computable — and, indeed, there is.

<u>Def.</u> Let $\mathcal{T}_n$ = the set of all TMs with n states and with tape alphabet $\{1, \sqcup\}$. Since there are only a finite number of transitions between the n states, the number of different TMs in $\mathcal{T}$ will be finite.

<u>Def.</u> Let $\mathcal{H}_n$ = set of all TMs with n states and with tape alphabet $\{1, \sqcup\}$ which halts with the blank tape as input. We define the Busy-beaver function $\beta: \mathbb{N} \to \mathbb{N}$ as follows.

$\beta(n)$ = maximum number of 1's that a TM in $\mathcal{H}_n$ produces when it halts, after it is started on the blank tape.

<u>Theorem 2</u>: The function $\beta: \mathbb{N} \to \mathbb{N}$ is not Turing-computable.

In otherwords, there is no TM, M_B, such that for each $n \in \mathbb{N}$, $\langle q_0, n \rangle \vdash^* \langle q_F, \varphi \underline{\beta(n)} \rangle$ is a halted computation in M_B.

END OF Ch. 5.