CS 407 Theory of Computation
Spring 2022

Main Text *:*

Elements of Theory of Computation, Papadimitriou & Lewis, Prentice *Hall 1998*

Auxiliary Texts *:*

1- Introduction to the Theory of Computation, Sipser, 1997 PWS

2- Computers and Intractability, Garey& Johnson, Freeman 2000

What is this course about ?

It is about problems and their solutions

How do human species express themselves formally ?
(In particular problems and solutions)

As a sequence of symbols from an alphabet written from left to right
(or right to left or top to bottom etc.). Any such set of sequences is called a (written) language.

PART 1 (Problems of solvability or decidability)

Can we quantify the total number of possible problems (languages)
and the total number of candidate solutions (languages) ?
What if the number of problems by far outnumber the number of solutions ?

PART 2 (Problems of computational complexity and intractability)

How do we measure the complexity of a problem instance and its
solution in terms of the resources (time and space) it uses as a function of
the problem instance size ?

What if the solution resource size explodes beyond imagination as the problem size grows ?

Can we write a ***universal debugger (UD) ?***

UD *is a computer program that takes* ***any*** *program* ***P*** *as an input and decides whether* ***P*** *gets stuck (halts in an undesirable state) .* ***Answer :*** *Impossible !!!*

It is stipulated (with little justification) that the memory consists of ***images :*** *pictures, sounds, smells, touches and all possible patterns of sense ; that are stored in terms of a subgraphs of nodes (neurons) that are interconnected in a specific way by edges in the brain, which itself consists of a much larger graph containing all the past knowledge of the individual.*

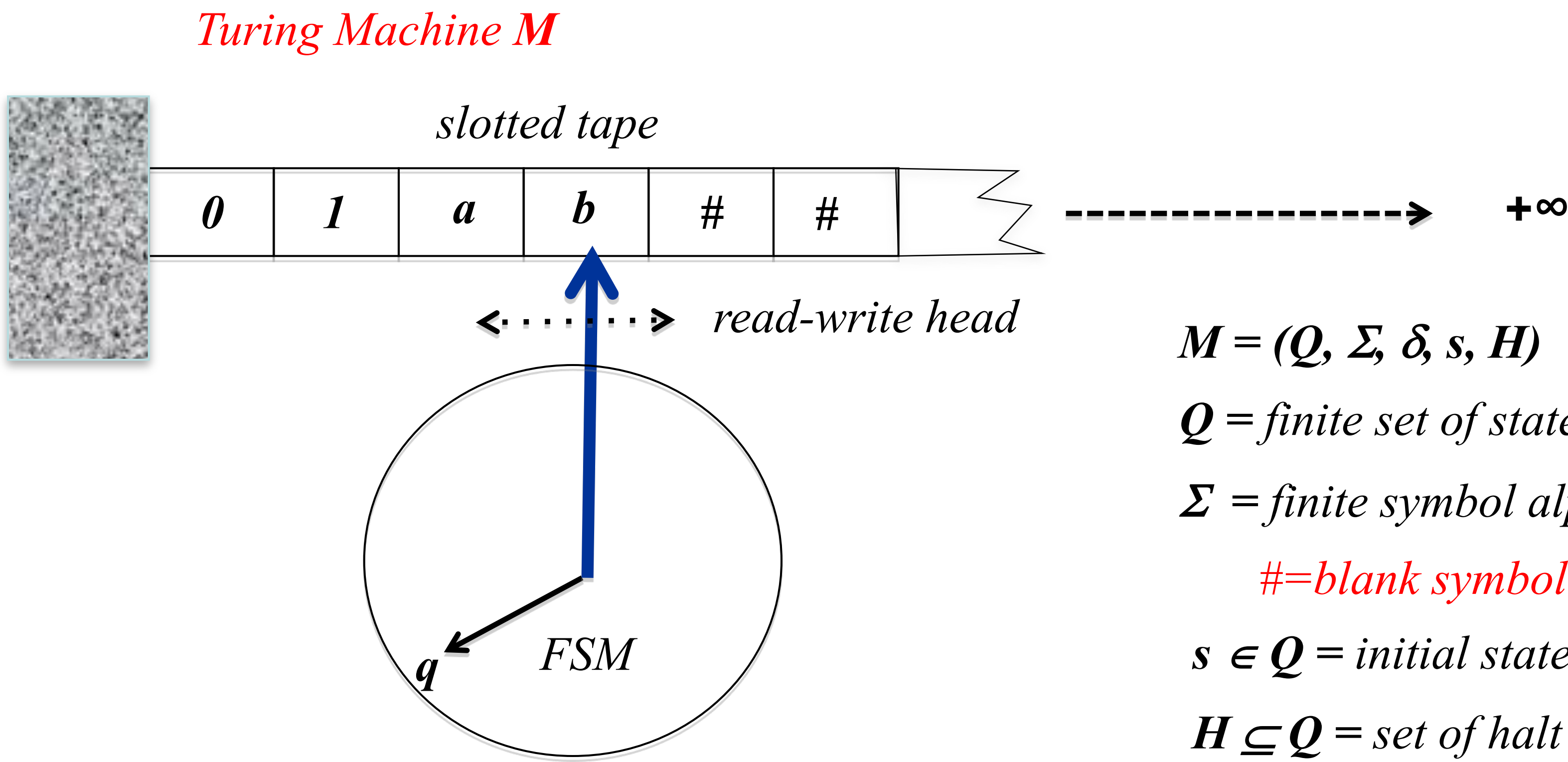
Given a possible subgraph with ***m*** *nodes and a total graph with* ***n*** *nodes* ***(m =< n)*** *what is the computational effort of determining whether the total graph contains the subgraph ?*

Answer *: possibly* $2^{K.n}$ = ***practically infinite !!!***

Three key concepts of the course :

DECIDABILITY , RECURSION *and* ***NP-COMPLETENESS***

Introduction to Turing Machines



$M = (Q, \Sigma, \delta, s, H)$

Q = *finite set of states*

Σ = *finite symbol alphabet*

#=blank symbol

$s \in Q$ = *initial state*

$H \subseteq Q$ = *set of halt states*

usually $H = \{h_{YES}, h_{NO}\}$

δ: $Q\text{-}H \times \Sigma \rightarrow Q \times \{\rightarrow(\text{right move}), \leftarrow(\text{left move}), \Sigma(\text{write})\}$

δ = *transition function*

Instantaneous Description (ID) of a TM

(q , u $\underline{a}$ v) : q *=current state of the FSM,*

a = *the symbol under the head ;* ***u*** $\in \Sigma^*$ *= the string to the* ***left*** *of the head*

v $\in \Sigma^*$ *= the string to the* ***right*** *of the head*

(q, u $\underline{a}$ v) $\in Q \times (\Sigma^* \times \Sigma \times (\Sigma^*.(\Sigma - \{\#\}) \cup e))$

In short :
(s, $\underline{\#}$ w)

Start convention : (s, e $\underline{\#}$ w) ; *where* ***w*** $\in \Sigma_0^*$; $\Sigma_0 \subseteq \Sigma - \{\#\}$ *the input alphabet*

Computational notation : (q , u $\underline{a}$ v) $|\text{--}_M^*$ (p , x $\underline{b}$ y) , *a finite step (*) computation*

Tabular Representation of the Transition Function

current state	*symbol under head*	*next state*	*action*

no. of rows $= |\boldsymbol{Q-H}| . |\boldsymbol{\Sigma}|$

Example : Clean-up TM : (s, $\underline{\#}$ w) |-- (h, $\underline{\#}$)*

Current state	Symbol under head	Next state	Action
s	#	q_f	$\rightarrow$
s	0	x	x
s	1	x	x
q_f	#	q_{b1}	$\leftarrow$
q_f	0	q_f	$\rightarrow$
q_f	1	q_f	$\rightarrow$
q_{b1}	#	h	#
q_{b1}	0	q_{b2}	#
q_{b1}	1	q_{b2}	#
q_{b2}	#	q_{b1}	$\leftarrow$
q_{b2}	0	x	x
q_{b2}	1	x	x

Don't care combinations

The Composite Turing Machine

M . N = *If and when the TM **M** halts then control is passed to TM **N** sharing the same tape.*

$$\begin{tikzcd} & P \\ M \arrow[ur, "a"] \arrow[dr, "b"'] & \\ & R \end{tikzcd}$$

= *If and when the TM **M** halts then control is passed to TM **P** or **R** if current tape slot under the head has the symbol **a** or **b** respectively.*

Basic Turing Machines

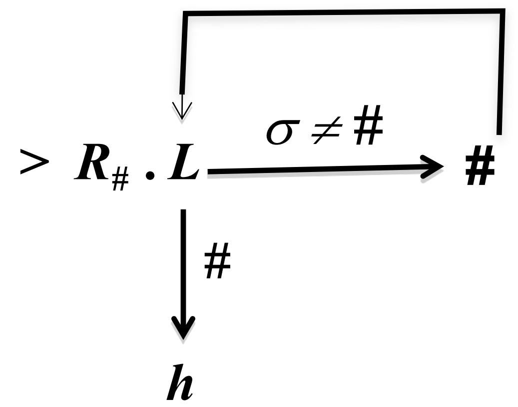
R(L) = *TM that moves one slot right(left) and halts.*

$\boldsymbol{\sigma}$ = *TM that writes on the current tape slot the symbol* $\boldsymbol{\sigma}$ *and halts.*

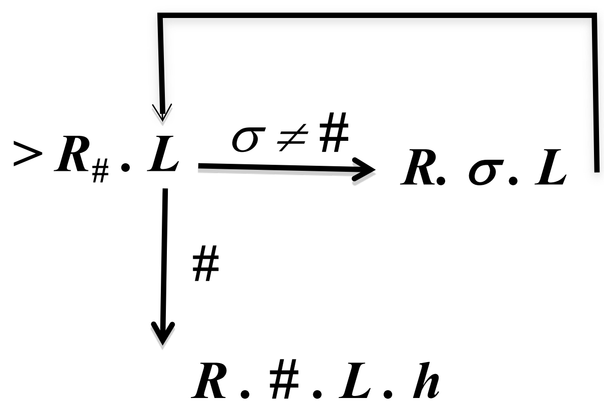
$\boldsymbol{R_A}$ $\boldsymbol{(L_A)}$ **=** *TM that keeps on moving the head **right (left)** as long as the symbol under the head is NOT in* $\boldsymbol{A} \subseteq \boldsymbol{\Sigma}$ (*a short hand notation is used as* # instead of ***{#}*** as an instance of the set *A*)

$\boldsymbol{h}$, $\boldsymbol{h_{YES}}$, $\boldsymbol{h_{NO}}$ = *TM that is in halted state : neutral, YES or NO!*

Clean-up TM revisited



Example : The right shift machine RS : (s , $\underline{\#}$ w) $|\!\!-\!\!-_{RS}{}^{*}$ (h, $\underline{\#}$ # w)



verification example : take ***w = 010***

$(\underline{\#}010)\ R_{\#} \rightarrow\ (\#010\underline{\#})\ L \rightarrow\ (\#01\underline{0})\ R \rightarrow$ *(0 remembered)*

$(\#010\underline{\#})\ 0 \rightarrow\ (\#010\underline{0})\ L \rightarrow\ (\#01\underline{0}0)\ L \rightarrow (\#0\underline{1}00)\ R \rightarrow$ *(1 remembered)*

$(\#01\underline{0}0)\ 1 \rightarrow\ (\#01\underline{1}0)\ L \rightarrow\ (\#0\underline{1}10)\ L \rightarrow (\#\underline{0}110)\ R \rightarrow$ *(0 remembered)*

$(\#0\underline{1}10)\ 0 \rightarrow\ (\#0\underline{0}10)\ L \rightarrow\ (\#\underline{0}010)\ L \rightarrow (\underline{\#}0010)\ R \rightarrow$ *(# detected)*

$(\#\underline{0}010)\ \# \rightarrow\ (\#\underline{\#}010)\ L \rightarrow\ (\underline{\#}\#010)\ h$

Tabular Representations

Clean-up machine C : $(s, \underline{\#}\ w) \vdash_{LS}^{*} (h, \underline{\#})$

Label	Condition	Next TM
>	-	$R_{\#}\ L\ B$
B	$\sigma \neq \#$	$\#\ L\ B$
	$\sigma = \#$	h

Right Shift Machine RS : $(s, \underline{\#}\ w) \vdash_{RS}^{*} (h, \underline{\#}\ \#\ w)$

TM	Condition	Next TM
>	-	$R_{\#}\ L\ B$
B	$\sigma \neq \#$	$R\ \sigma\ L\ L\ B$
	$\sigma = \#$	$R\ \#\ L\ \ h$

Tabular Representations (Cont')

Left Shift Machine LS : $(s\ ,\ \#\ w\ \underline{\#}\ )\ |\text{--}_{RS}\ ^*\ (h,\ w\ \underline{\#}\ )$

TM	Condition	Next TM
>		$L_{\#}\ R\ B$
B	$\sigma \neq \#$	$L\ \sigma\ R\ R\ B$
	$\sigma = \#$	$L\ \#\ h$

Basic Definitions on Turing machines

A TM $\boldsymbol{M}$ *with* $\boldsymbol{H = \{ h_{YES} , h_{NO} \}}$ *is said to* ***DECIDE*** *a language* $\boldsymbol{L \subseteq \Sigma_0{}^*}$ *if :*

$(s, \underline{\#}\ w) |\text{--}_M{}^* (h_{YES}, u\ \underline{a}\ v)$, *if* $w \in L$

$(s, \underline{\#}\ w) |\text{--}_M{}^* (h_{NO}, u\ \underline{a}\ v)$, *if* $w \notin L$

A TM $\boldsymbol{M}$ *with* $\boldsymbol{H = \{ h \}}$ *is said to* ***compute*** *a function* $\boldsymbol{f : \Sigma_0{}^* \rightarrow \Sigma_0{}^*}$ *if :*

$(s, \underline{\#}\ w) |\text{--}_M{}^* (h, u\ \underline{a}\ v)$, *iff* $u = e$; $a = \#$ *and* $v = f(w)$

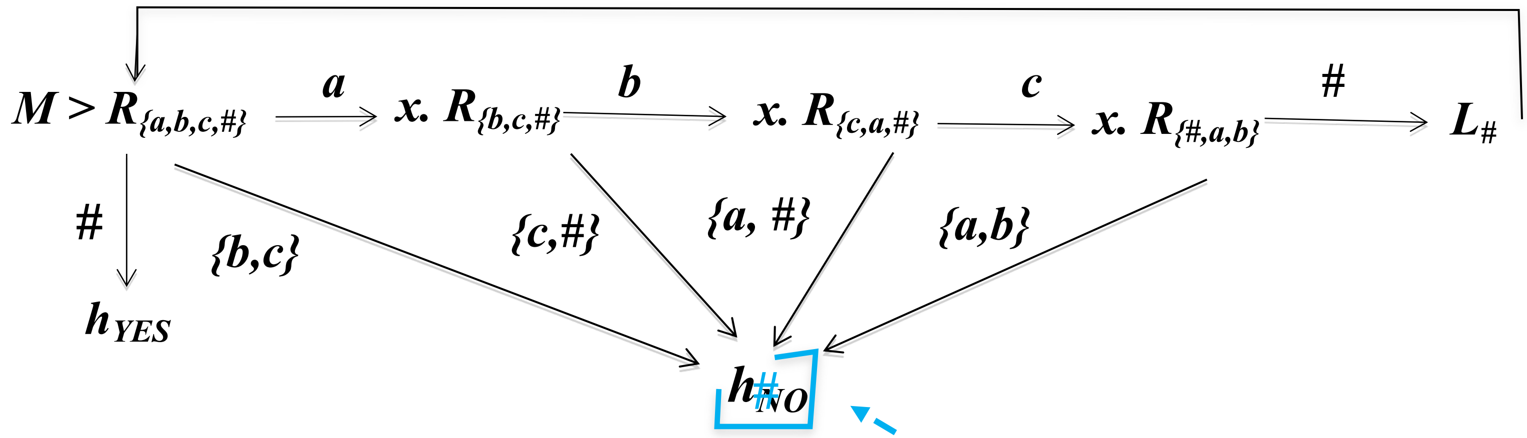
A TM $\boldsymbol{M}$ *with* $\boldsymbol{H = \{ h \}}$ *is said to* ***SEMIDECIDE (ACCEPT)*** *a language* $\boldsymbol{L \subseteq \Sigma_0{}^*}$ *if :*

$(s, \underline{\#}\ w) |\text{--}_M{}^* (h, u\ \underline{a}\ v)$, *iff* $w \in L$

*Example : A TM **M** that **decides** the language $L = \{a^n\, b^n\, c^n\ ;\ n \geq 0\}$*

(semidecides)

Let $\Sigma = \{a,b,c,x,\#\}$



(semidecide case)

*Verification **w= aabbcc***

$(\underline{\#}aabbcc)\ R_{\{a,b,c,\#\}} \rightarrow (\#\underline{a}abbcc)\ x \rightarrow (\#\underline{x}abbcc)\ R_{\{b,c,\#\}} \rightarrow (\#xa\underline{b}bcc)\ x \rightarrow (\#xa\underline{x}bcc)\ R_{\{c,a,\#\}} \rightarrow (\#xaxb\underline{c}c)\ x \rightarrow (\#xaxb\underline{x}c)\ R_{\{\#,a,b)} \rightarrow (\#xaxbxc\underline{\#})\ L_\# \rightarrow$

$(\underline{\#}xaxbxc)\ R_{\{a,b,c,\#\}} \rightarrow (\#x\underline{a}xbxc)\ x \rightarrow (\#x\underline{x}xbxc)\ R_{\{b,c,\#\}} \rightarrow (\#xxx\underline{b}xc)\ x \rightarrow (\#xxx\underline{x}xc)\ R_{\{c,a,\#\}} \rightarrow (\#xxxxx\underline{c})\ x \rightarrow (\#xxxxx\underline{x})\ R_{\{\#,a,b)} \rightarrow (\#xxxxxx\underline{\#})\ L_\# \rightarrow$

$(\underline{\#}xxxxxx)\ R_{\{a,b,c,\#\}} \rightarrow (\#xxxxxx\underline{\#})$ (# detected) h_{YES}

*Verification **w= aabcc***

$(\underline{\#}aabcc)\ R_{\{a,b,c,\#\}} \rightarrow (\#\underline{a}abcc)\ x \rightarrow (\#\underline{x}abcc)\ R_{\{b,c,\#\}} \rightarrow (\#xa\underline{b}cc)\ x \rightarrow (\#xa\underline{x}cc)\ R_{\{c,a,\#\}} \rightarrow (\#xax\underline{c}c)\ x \rightarrow (\#xax\underline{x}c)\ R_{\{\#,a,b)} \rightarrow (\#xaxxc\underline{\#})\ L_\# \rightarrow$

$(\underline{\#}xaxxc)\ R_{\{a,b,c,\#\}} \rightarrow (\#x\underline{a}xxc)\ x \rightarrow (\#x\underline{x}xxc)\ R_{\{b,c,\#\}} \rightarrow (\#xxxx\underline{c})$ (c detected) $\rightarrow h_{NO}$

Example : *a TM **M** that computes the function*

$f(w) = w.w^R$; $(s, \underline{\#}\, w) \vdash_M^* (h, \underline{\#}\, w.w^R)$

$$\begin{tikzcd} M > R_\# \,.\, L \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"] & x\,.R_\#\,.\,\sigma.L_x\,.\sigma \arrow[l, bend right] \\ h \end{tikzcd}$$

Label	Condition	Next TM
$>$	-	$R_\#\, L\, B$
B	$\sigma \neq \#$	$x\, R_\#\, \sigma\, L_x\, \sigma\, L\, B$
	$\sigma = \#$	h

*where **w** $\in \Sigma_0^*$, **x** $\notin \Sigma_0$ and $\Sigma = \Sigma_0 \cup \{x,\#\}$*

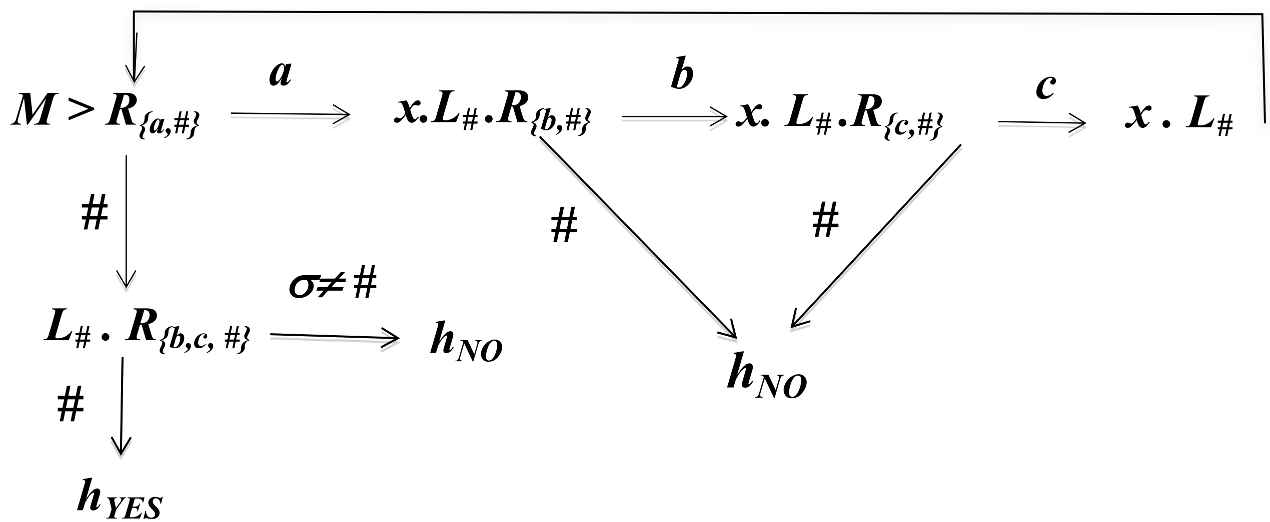
*Verification **w= ab ; f(w)= abba***

($\underline{\#}$ab) $R_\#$ → (#ab$\underline{\#}$) L → (#a$\underline{b}$) x → (#a$\underline{x}$) (b remembered)$R_\#$ → (#ax$\underline{\#}$) b → (#ax$\underline{b}$) L_x → (#a$\underline{x}$b) b → (#a$\underline{b}$b) L →

(#$\underline{a}$bb) x→ (#$\underline{x}$bb) (a remembered) $R_\#$→ (#xbb$\underline{\#}$) a → (#xbb$\underline{a}$) L_x→ (#$\underline{x}$bba) a → (#$\underline{a}$bba) L → ($\underline{\#}$abba) (# detected) h

*Example : A TM **M** that **decides** the language **L = {ω ϵ {a,b,c}* | #as = #bs = #cs}***

Let $\Sigma = \{a,b,c,x,\#\}$



*Verification **w= acabbc** (**w= acabbcb**)*

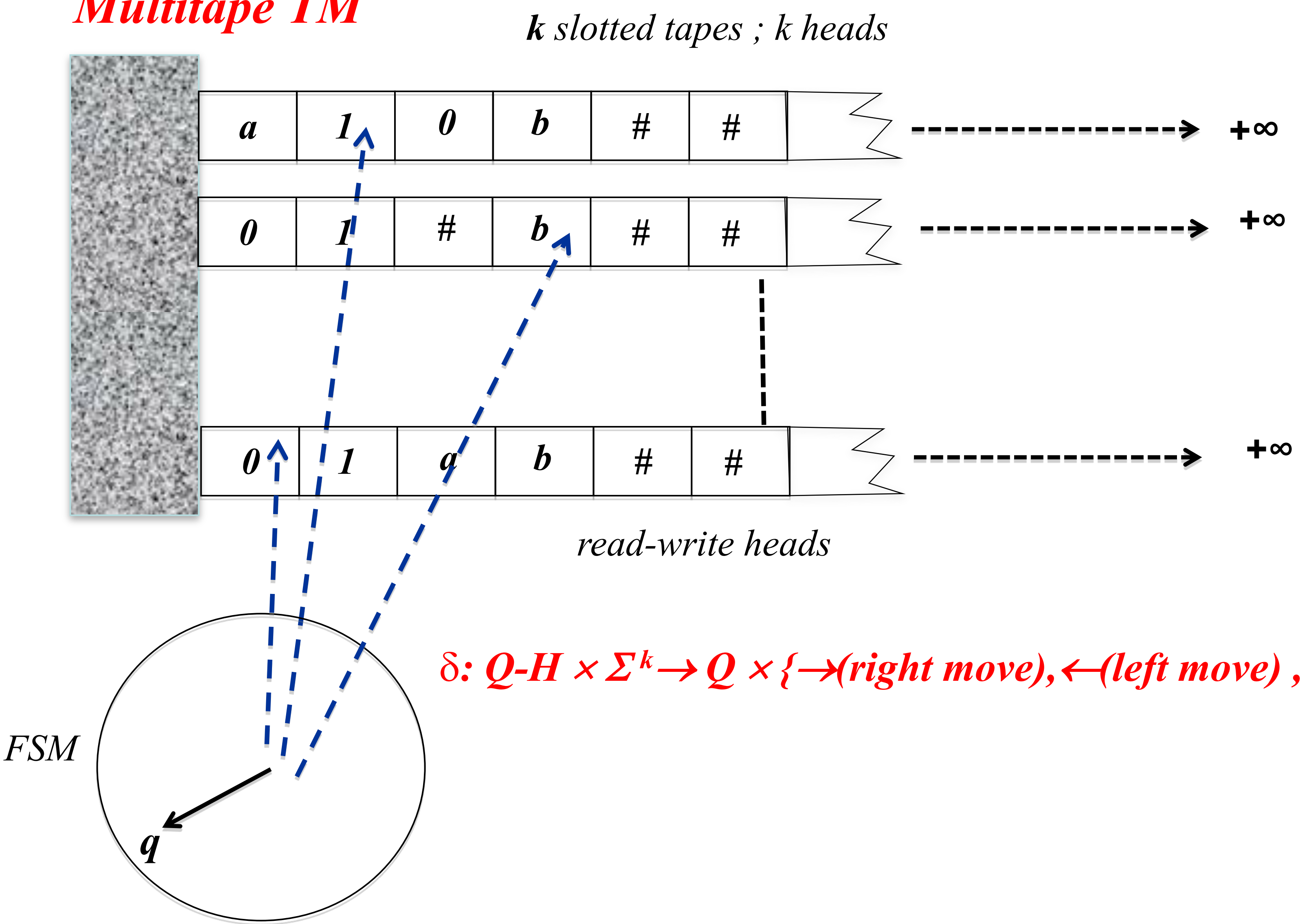
$(\underline{\#}acabbcb)\ R_{\{a,\#\}} \rightarrow (\#\underline{a}cabbcb)\ x \rightarrow (\#\underline{x}cabbcb)\ L_\# \rightarrow (\underline{\#}xcabbcb)\ R_{\{b,\#\}} \rightarrow (\#xca\underline{b}bcb)\ x \rightarrow (\#xca\underline{x}bcb)\ L_\# \rightarrow (\underline{\#}xcaxbcb)\ R_{\{c,\#\}} \rightarrow (\#x\underline{c}axbcb)\ x \rightarrow$

$(\#x\underline{x}axbcb)\ L_\# \rightarrow (\underline{\#}xxaxbcb)\ R_{\{a,\#\}} \rightarrow (\#xx\underline{a}xbcb)\ x \rightarrow (\#xx\underline{x}xbcb)\ L_\# \rightarrow (\underline{\#}xxxxbcb)\ R_{\{b,\#\}} \rightarrow (\#xxxx\underline{b}cb)\ x \rightarrow (\#xxxx\underline{x}cb)\ L_\# \rightarrow (\underline{\#}xxxxxcb)\ R_{\{c,\#\}} \rightarrow$

$(\#xxxxx\underline{c}b)\ x \rightarrow (\#xxxxxx\underline{x}b)\ L_\# \rightarrow (\underline{\#}xxxxxxxb)\ R_{\{a,\#\}}$ (#detected) $\rightarrow (\#xxxxxxxb\underline{\#})\ L_\# \rightarrow (\underline{\#}xxxxxxxb)\ R_{\{b,c,\#\}} \rightarrow (\#xxxxxxxb\underline{\#})$ (# detected) h_{YES}

$(\#xxxxxxx\underline{b})$ (b detected) h_{NO}

Multitape TM



δ: $Q\text{-}H \times \Sigma^k \rightarrow Q \times \{\rightarrow(\text{right move}), \leftarrow(\text{left move}), \Sigma(\text{write})\}^k$

Instantaneous Description (ID) of a Multitape TM

$(q\ ;\ u_1\ \underline{a}_1\ v_1\ , \ldots ,\ u_k\ \underline{a}_k\ v_k)$: q =current state of the FSM,

a_j = the symbol under head j; $u_j \in \Sigma^*$ = the string to the **left** of head j

$v_j \in \Sigma^*$ = the string to the **right** of head j

$(q\ ;\ u_1\ \underline{a}_1\ v_1\ , \ldots ,\ u_k\ \underline{a}_k\ v_k) \in Q \times (\Sigma^* \times \Sigma \times (\Sigma^*.(\Sigma - \{\#\}) \cup e))^k$

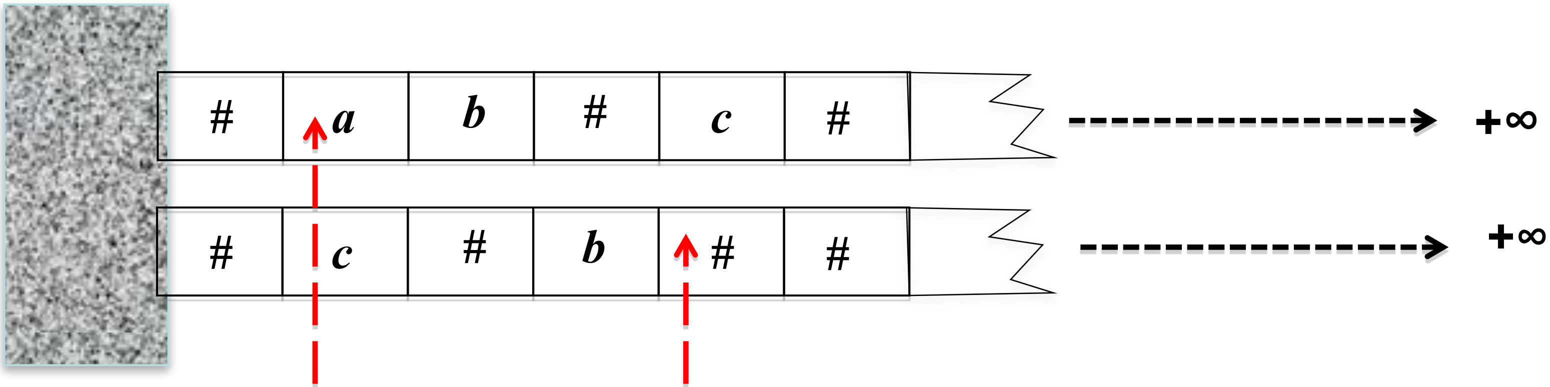
Start convention : $(s, \underline{\#}\ w, \ldots, \underline{\#})$; where $w \in \Sigma_0^*$; $\Sigma_0 \subseteq \Sigma$ is the input alphabet

Computational notation :

$(q\ ;\ u_{11}\ \underline{a}_{11}\ v_{11}\ , \ldots ,\ u_{k1}\ \underline{a}_{k1}\ v_{k1})\ |\!\!-\!\!-_M^*\ (p\ ;\ u_{1m}\ \underline{a}_{1m}\ v_{1m}\ , \ldots ,\ u_{km}\ \underline{a}_{km}\ v_{km})$,

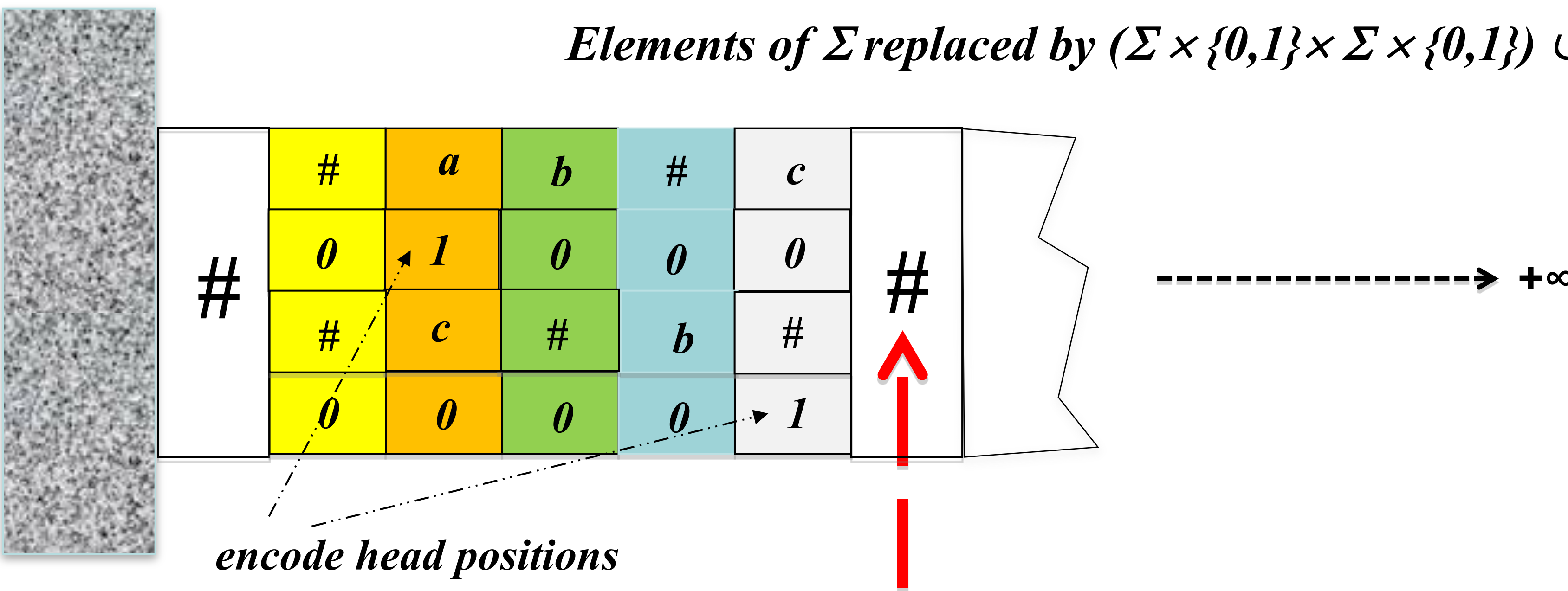
*An (**m**-step) finite step (*) computation*

Simulating A Multitape TM on a single tape one



Each step of the computation of the 2 tape TM is accomplished by finite-steps (scans) of the single tape TM

Elements of Σ replaced by $(\Sigma \times \{0,1\} \times \Sigma \times \{0,1\}) \cup \#$



Fact

Every multitape TM can be simulated by a standard TM

*For a given **k**-tape TM* $\boldsymbol{M_k}$ *there is a corresponding standard TM **M** such that :*

- *If* $\boldsymbol{M_k}$ ***decides*** *a language **L** then **M** **decides** the language **L***

- *If* $\boldsymbol{M_k}$ ***semidecides*** *a language **L** then **M** **semidecides** the language **L***

- *If* $\boldsymbol{M_k}$ ***computes*** *a function* $\boldsymbol{f}$ **:** $(s, \underline{\#}\, w, \ldots, \underline{\#}) \mid\!\!-\!\!-_{Mk}^{*} (h, \underline{\#}\; f(w), \ldots, \underline{\#})$

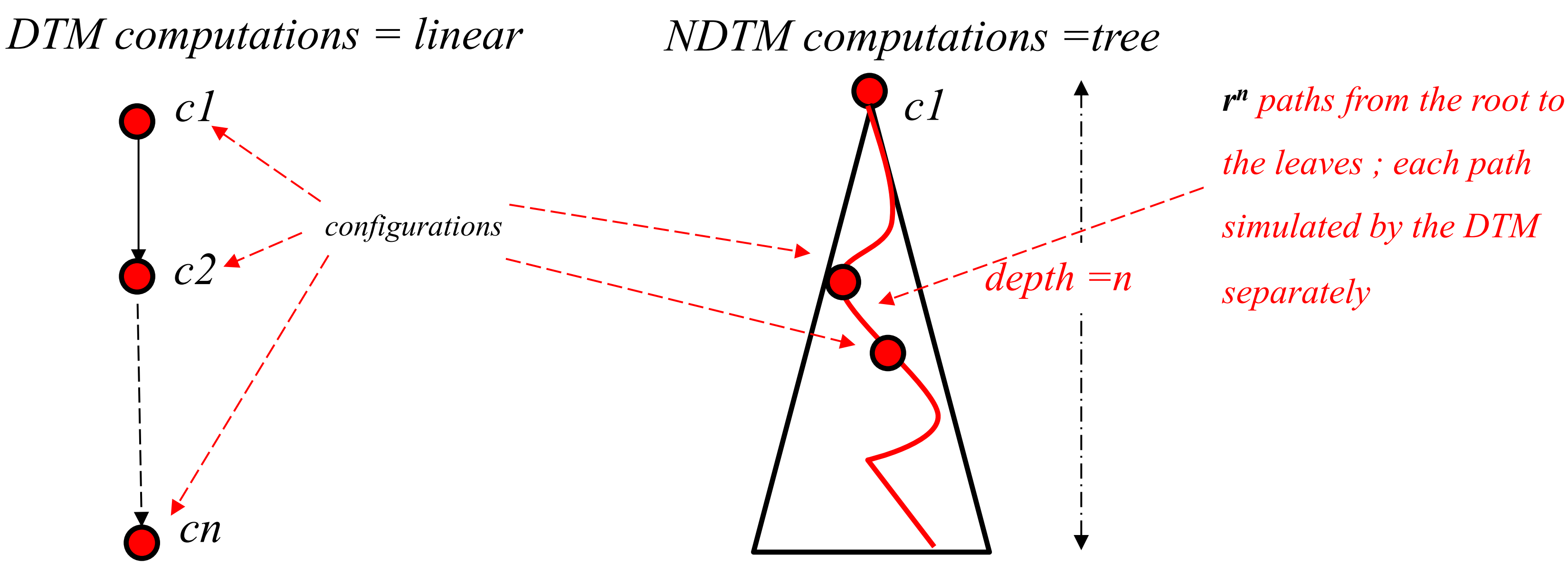
*Then **M** **computes** the function* $\boldsymbol{f}$, $(s, \underline{\#}\, w) \mid\!\!-\!\!-_{M}^{*} (h, \underline{\#}\; f(w))$

Nondeterministic TM (NDTM)

The only difference is in the transition function :

$$\delta : Q\text{-}H \times \Sigma \rightarrow 2^{Q \times \{\rightarrow(\text{right move}), \leftarrow(\text{left move}),\ \Sigma(\text{write})\}}$$

At each configuration a NDTM can have at most $r = |Q|.(2+|\Sigma|)$ *next configurations.*



Nondeterministic TM (NDTM)

A NDTM halts because of 2 reasons
1- $(s,\underline{\#}w) \vdash^* (h,u\underline{a}v)$; $h \in H$
2 - $(s,\underline{\#}w) \vdash^* (q,u\underline{a}v)$; $q \in Q\text{-}H$; $\delta(q,a)$ = null set

$$\delta : Q\text{-}H \times \Sigma \rightarrow 2^{\,Q \times \{\rightarrow(\text{right move}),\leftarrow(\text{left move}),\ \Sigma(\text{write})\}}$$

Every nondeterministic TM can be simulated by a standard TM !

Alternatively : computation tree is ***finite***

Definitions

*A nondeterministic TM **M** is said to **decide** a language **L** if*

*1- There is an integer **K** such that there is no configuration **C** such that*

$(s, \underline{\#}\, w) \vdash_M^K C$ *(i.e. all computations **halt** (or get stuck) before **K** steps !)*

2- $w \in L$ ***iff*** *there is at least one computation :* $(s, \underline{\#}\, w) \vdash_M^* (h_{YES}, u\, \underline{a}\, v)$

*A nondeterministic TM **M** is said to **semidecide** a language **L** if*

*1- There is no integer **K** that satisfies 1- above*

2- $w \in L$ ***iff*** *there is a computation :* $(s, \underline{\#}\, w) \vdash_M^* (h_{YES}, u\, \underline{a}\, v)$

*A nondeterministic TM **M** is said to **compute** a function **f** if :*

1- Condition 1- of decidability above holds

2- $(s, \underline{\#}\, w) \vdash_M^* (h_{YES}, u\, \underline{a}\, v)$ ***iff*** $u=e$; $a=\#$; $v = f(w)$

Example *A two-tape NDTM that decides the language*

$$L = (\omega \in \Sigma_0{}^* \mid \omega = u.u \; ; u \in \Sigma_0{}^*) \; ; \text{start at } (s \; ; \underline{\#}\, \omega \, , \underline{\#}) \; ; d \notin \Sigma_0$$

head is on 2nd element of $\boldsymbol{\omega}$ *in tape 1*

Immediately accept if $\boldsymbol{\omega = e}$ ***; else move to*** *A*

Move head to a midpoint nondeterministically ; copy ***first entry of 2nd half*** *to 2nd tape 1st entry and replace it with* ***d*** *; if # is reached then reject !*

Copy entire 2nd half of 1st tape to 2nd tape meanwhile replacing copied entries with ***d***

Replace all ***d****s with # in tape 1 and after that move heads 1 and 2 to leftmost # to make them ready for comparison*

Compare contents of 1st tape and 2nd tape ; if equal accept with $\boldsymbol{h_{YES}}$ *; if different reject with* $\boldsymbol{h_{NO}}$ *!*

TM	Condition	Next TM
$> R^1$	$\sigma^1 = \#$	h_{YES}
	$\sigma^1 \neq \#$	$R^1.A$
A	$\sigma^1 = x \neq \#$	$R^1.A$
	$\sigma^1 = x \neq \#$	$R^2.\,x^2.\,d^1.R^1.R^2.B$
	$\sigma^1 = \#$	h_{NO}
B	$\sigma^1 = x \neq \#$	$x^2.\,d^1.R^1.R^2.\,B$
	$\sigma^1 = \#$	$L^1.C$
C	$\sigma^1 = d$	$\#^1.L^1.C$
	$\sigma^1 \neq d$	$L_{\#}{}^1.L_{\#}{}^2.R^1.R^2.\,D$
D	$\sigma^1 = \sigma^2 \neq \#$	$R^1.R^2.D$
	$\sigma^1 = \sigma^2 = \#$	h_{YES}
	else	h_{NO}

$L = (\omega \in \Sigma_0{}^* \mid \omega = u.u \,;\, u \in \Sigma_0{}^*)$; start at $(s \,;\, \underline{\#}\,\omega, \underline{\#})$; $d \notin \Sigma_0$

$\omega = abab$ start at $(\underline{\#}\, a\, b\, a\, b\, ,\, \underline{\#})$

$(\underline{\#}\, a\, b\, a\, b\, ,\, \underline{\#})\ R^1 \rightarrow\ (\#\, \underline{a}\, b\, a\, b\, ,\, \underline{\#})\ R^1A.R^1A \rightarrow (\#\, a\, b\, \underline{a}\, b\, ,\, \underline{\#})\ R^2 \rightarrow$

$(\#\, a\, b\, \underline{a}\, b\, ,\, \#\, \underline{\#})\ a^2 d^1 \rightarrow\quad (\#\, a\, b\, \underline{d}\, b\, ,\, \#\, \underline{a})\ R^1R^2 \rightarrow$

$(\#\, a\, b\, d\, \underline{b}\, ,\, \#\, a\, \underline{\#})\ b^2\, d^1 \rightarrow\quad (\#\, a\, b\, d\, \underline{d}\, ,\, \#\, a\, \underline{b})\ R^1R^2 \rightarrow$

$(\#\, a\, b\, d\, d\, \underline{\#}\, ,\, \#\, a\, b\, \underline{\#})\ L^1 \rightarrow\quad (\#\, a\, b\, d\, \underline{d}\, ,\, \#\, a\, b\, \underline{\#})\ \#^1 \rightarrow$

$(\#\, a\, b\, d\, \underline{\#}\, ,\, \#\, a\, b\, \underline{\#})\ L^1 \rightarrow\quad (\#\, a\, b\, \underline{d}\, ,\, \#\, a\, b\, \underline{\#})\ \#^1 \rightarrow$

$(\#\, a\, b\, \underline{\#}\, ,\, \#\, a\, b\, \underline{\#})\ L^1 \rightarrow\quad (\#\, a\, \underline{b}\, ,\, \#\, a\, b\, \underline{\#})\ L_\#{}^1.L_\#{}^2 \rightarrow$

$(\underline{\#}\, a\, b\, ,\, \underline{\#}\, a\, b) \rightarrow$ compare h_{YES}

TM	Condition	Next TM
$> R^1$	$\sigma^1 = \#$	h_{YES}
	$\sigma^1 \neq \#$	$R^1.\ A$
A	$\sigma^1 = x \neq \#$	$R^1.\ A$
	$\sigma^1 = x \neq \#$	$R^2.\ x^2.\ d^1.R^1.R^2\ \ B$
	$\sigma^1 = \#$	h_{NO}
B	$\sigma^1 = x \neq \#$	$x^2.\ d^1.\ R^1.R^2\ \ B$
	$\sigma^1 = \#$	$L^1.C$
C	$\sigma^1 = d$	$\#^1.L^1.C$
	$\sigma^1 \neq d$	$L_\#{}^1.L_\#{}^2.R^1.R^2.\ D$
D	$\sigma^1 = \sigma^2 \neq \#$	$R^1.R^2.D$
	$\sigma^1 = \sigma^2 = \#$	h_{YES}
	else	h_{NO}

Example — *A two-tape TM that adds two "binary coded" integers*

$(s\ ;\ \underline{\#}\ \omega_1 \underline{\#}\ \ \omega_2 , \underline{\#}\ ) \mid\!\!--^* (h, \underline{\#}\ \ \text{"}\omega_1 + \omega_2\text{"}, \underline{\#}\ \ )$

*Copy ω_1 into the 2nd tape and replace copied entries and # with **0**s ; move heads 1 and 2 to rightmost least significant digits to start addition*

*Start addition of digits with the result replacing the content of 1st tape digit ; if there is a **carry** digit move to **C** ; else continue with **B ;** stop if # is reached in tape 2 with head 1 in leftmost #*

***Carry** account continues*

***Carry** account disappears*

***Carry** account is terminated ; result in tape 1*

***Carry** account propogates to left digits*

TM	Condition	Next TM
$A = R^1R^2$	$\sigma^1 = x \neq \#$	$0^1\ x^2\ A$
	$\sigma^1 = \#$	$0^1\ R^1_{\#}\ L^1L^2\ B$
B	$\sigma^1\sigma^2 = 01 \vee 10$	$1^1\ \#^2\ \ L^1\ L^2\ B$
	$\sigma^1\sigma^2 = 00$	$0^1\ \#^2\ L^1\ L^2\ B$
	$\sigma^1\sigma^2 = 11$	$0^1\ \#^2\ L^1\ L^2\ C$
	$\sigma^2 = \#$	$L^1_{\#}\ h$
C	$\sigma^1\sigma^2 = 01 \vee 10$	$0^1\ \#^2\ L^1\ L^2\ C$
	$\sigma^1\sigma^2 = 00$	$1^1\ \#^2\ \ L^1\ L^2\ B$
	$\sigma^1\sigma^2 = 11$	$1^1\ \#^2\ L^1\ L^2\ C$
	$\sigma^2 = \#$	D
D	$\sigma^1 = 0$	$1^1\ L^1_{\#}\ h$
	$\sigma^1 = 1$	$0^1\ L^1 D$

The Universal Turing Machine

***Coding Alphabet* = { (,) , $, ‘,’ , 0 , 1, # }**

 # = blank character

Binary Encoding Convention :

States : 0 → $HALT_{Yes}$; 1 → $HALT_{No}$; . . .

Input/Action : 0 → *Right Move* ; 1 → *Left Move* ; . . .

Tape representation (xx denotes encoded character)

Tape 1 is input tape → # xx,xx, … xx $ head position xx , … xx #

Tape 2 is transition table → # $(q_1, a_1, q'_1, action_1)$… $(q_n, a_n, q'_n, action_n)$ #

Tape 3 is current state → #xx#

How does the Universal Turing Machine work ?

*Suppose that the TM **M** simulated by the UTM **U** makes the transition : **$(q\ ,\ \sigma) \rightarrow (q'\ ,\ b)$** where :*

*;- **q** is the current state encoded as **E(q)** in **tape 3***

*;- **σ** is the current symbol under the head encoded as **E(σ)** just before the **$** symbol in **tape 1***

*;- **q'** is the next state dictated by the transition fn. (3^{rd} element of the row)*

*;- **b** is the encoded next action again dictated by the transition fn. (4^{th} element of the row) which is '→' or '←' (move the **$** mark in **tape 1**) , or some **σ'** to be printed as **E(σ')** before the **$** marked slot in **tape 1**.*

The UTM simulation takes place in terms of two phases :

The ***search phase*** *and the* ***action implementation phase***

Search Phase : *Search among the encoded rows of the transition function until a row is found such that there is an exact match between the first two entries of this row and the encoded current state in* ***tape 3*** *and the encoded input before the* ***$*** *symbol in* ***tape 1.***

Action Implement Phase : *Replace the current state in* ***tape 3*** *with the next state (3rd element of the matching row); move the* ***$*** *mark in* ***tape 1*** *right or left ; or print* ***E(σ')*** *before the* ***$*** *marked slot in* ***tape 1*** *in accordance with the 4^{th} entry of the matching row.*

The Halting Problem

Consider the set of ***all*** *Deterministic Turing Machines (DTMs) with an input alphabet* $\boldsymbol{\Sigma = \{\#,0,1\}}$ *and a two halt states ,that is* $\boldsymbol{H = \{h_{YES}, h_{NO}\}}$ **.**

The transition function $\boldsymbol{\delta : \{Q\text{-}H\} \times \Sigma \to Q \times (\{``\to", ``\leftarrow"\} \cup \Sigma)}$ *of any such DTM is a finite table with* $\boldsymbol{(|Q|-2) \cdot |\Sigma|}$ *rows and* **4** *columns where* $|\Sigma| = 3$ *!*

Every row of the entire transition function can be encoded by ***positive integers*** *as follows : integers* $\boldsymbol{j > 0}$ *and* $\boldsymbol{k > 0}$ *corresponds to elements of* $\boldsymbol{Q}$ *and* $\boldsymbol{\Sigma}$ *where* $\boldsymbol{j=1,2}$ *corresponds to the halt states* $\boldsymbol{h_{YES}}$ *and* $\boldsymbol{h_{NO}}$ *and* $\boldsymbol{k = 3, \ldots, |\Sigma|+2}$ *correspond to the inputs in* $\boldsymbol{\Sigma}$**.** *The special integers* **1** *and* **2** *are reserved for encoding the actions for* ***right*** *and* ***left*** *motions of the head of the DTM respectively. Hence every row of the transition function* $\boldsymbol{\delta}$ *is represented by* **4** *positive integers that are separated by some symbol, such as a comma, acting a separator :* $\boldsymbol{i, j, k, m,}$

A binary encoding for a single row of the transition function is as follows :

$\boldsymbol{r = 0^{i}\,1\,0^{j}\,1\,0^{k}\,1\,0^{m}\,1}$ *And the entire transition function* $\boldsymbol{\delta}$ *be encoded as* $\boldsymbol{E(\delta)}$ *which is a concatenation of* $\boldsymbol{R = (|Q|-1).|\Sigma|}$ *binary rows as :*

$\boldsymbol{E(\delta) = r_1 . r_2 \ldots r_R}$ *where* $\boldsymbol{r_j}$ *denotes the binary code for the* $\boldsymbol{j^{th}}$ *row of the transition function. Note that distinct encodings corresponding to different orders in which the rows are sequenced and different integer encodings for states all correspond to the same DTM.*

Thus ***every*** *DTM corresponds to* ***possibly different binary strings*** *each represented by some* $\boldsymbol{E(\delta)}$*, depending on the permutation of the rows and the integer encodings of the states of the transition function. Finally as a convention we choose the first block of zeros in the first chosen row as the encoding of the* ***initial state*** *.*

If the binary sequence is simply ***0*** *or* ***00*** *then the DTM is a* $\boldsymbol{halt_{YES}}$ *or* $\boldsymbol{halt_{NO}}$ *machine*

Example : Clean-up TM : $(s, \underline{\#}\ w) \vdash^* (h, \underline{\#})$

Current state	Symbol under head	Next state	Action
s	#	q_f	$\rightarrow$
s	0	q_f	$\rightarrow$
s	1	q_f	$\rightarrow$
q_f	#	q_{b1}	$\leftarrow$
q_f	0	q_f	$\rightarrow$
q_f	1	q_f	$\rightarrow$
q_{b1}	#	h	$\rightarrow$
q_{b1}	0	q_{b2}	#
q_{b1}	1	q_{b2}	#
q_{b2}	#	q_{b1}	$\leftarrow$
q_{b2}	0	q_{b1}	$\leftarrow$
q_{b2}	1	q_{b1}	$\leftarrow$

$\underline{3 , 3 , 4 , 1}$

3 , 4 , 4, 1

3 , 5 , 4 , 1

4 , 3 , 5 , 2

4 , 4 , 4 , 1

4 , 5 , 4 , 1

. . . .

6, 5 , 5 , 2

000100010000101

$\rightarrow$ → 1

$\leftarrow$ → 2

→ 3

0 → 4

1 → 5

Let $\mathbf{L_{DTM}} \subseteq \{0,1\}^*$ *be the language corresponding to any legitimate encoding of the transition function of DTM with binary inputs with the conventions as described in the previous slides. Also note that for* ***distinct*** *binary strings* $u \in \{0,1\}^*$ *the strings* $\mathbf{1u}$ *correspond to* ***distinct*** *positive numbers covering all the integers* **1,2,…** *as follows :*

$1e \rightarrow 1;\ 10 \rightarrow 2\ ;\ 11 \rightarrow 3\ ;\ 100 \rightarrow 4\ ;\ 101 \rightarrow 5 \ldots$ ***etc.***

In view of the definition above the infinite number of positive integers :

$$1L_{DTM} \rightarrow 0 < x_1 < x_2 < \ldots, < x_m < \ldots$$

cover all the DTMs ,although a single DTM may correspond to more than one – but a finite number - of the integers above.

In view of the definition $\boldsymbol{L_{DTM}}$ *for each* $\boldsymbol{w \in L_{DTM}}$ *the unique DTM* ***M*** *can be written as a* ***function*** *of the string* ***w*** *as* $\boldsymbol{M = M(w)}$

We extend the encoding of a DTM ***M*** *in the following manner :*

By the term ***accept*** *for an input* $\boldsymbol{u \in \{0,1\}^*}$ *for a DTM* ***M*** *, it is meant that* ***M*** *eventually* ***halts*** *at its legitimate halt state* $\boldsymbol{h_{YES}}$ *starting from the initial string* ***u*** *on its tape. If* $\boldsymbol{w \in \{0,1\}^* \sim L_{DTM}}$ *-* ***w*** *does* ***not*** *correspond to a binary encoding of a DTM as explained above - we let* $\boldsymbol{M(w) := R}$ *, that is a DTM which moves head in the right direction all the time. Finally 2 special DTMs* $\boldsymbol{halt_{YES}}$ *and* $\boldsymbol{halt_{NO}}$ *machines are encoded by* ***0*** *and* ***00***

We first define the following diagonal language $\boldsymbol{D \subseteq \{0,1\}^*}$

$\boldsymbol{D := (w \in \{0,1\}^* \mid M(w)}$ ***accepts*** $\boldsymbol{w\ )}$

The complement diagonal language is :

$\boldsymbol{D^c := (\ w \in \{0,1\}^* \mid M(w)}$ ***does not accept*** $\boldsymbol{w)}$

Key question *: Is there a DTM that* ***semidecides the language*** $\boldsymbol{D^c}$ ***?***

Answer : NO ! WHY NOT ?

Suppose there is a DTM $\boldsymbol{M^*}$ *that semidecides* $\boldsymbol{D^c}$

Let $\boldsymbol{u^*}$ *be a binary encoding of* $\boldsymbol{M^*}$ *in* $\boldsymbol{L_{DTM}}$ *so that* $\boldsymbol{M(u^*) = M^*}$

How does $\boldsymbol{M(u^*)}$ *behave on* u^* *as its input ?*

First note that there are only **2** *possibilities : either* $\boldsymbol{u^* \in D^c}$ *or* $\boldsymbol{u^* \in D}$

CASE (1) $\boldsymbol{u^* \in D^c}$

M* ACCEPTS u* ***because M**** **semidecides** $\boldsymbol{D^c}$ **and** $\boldsymbol{u^* \in D^c}$

M* DOES NOT ACCEPT u* ***follows*** *from the* **definition** *of* $\boldsymbol{D^c}$ *and* $\boldsymbol{u^* \in D^c}$

Logical contradiction : not possible

$\boldsymbol{D^c := (w \in \{0,1\}^* \mid M(w)}$ ***does not accept*** $\boldsymbol{w)}$

CASE (2) $\boldsymbol{u^* \in D}$ *(or* $\boldsymbol{u^* \notin D^c}$ *)*

M* DOES NOT ACCEPT u* ***because M**** **semidecides** $\boldsymbol{D^c}$ *and* $\boldsymbol{u^* \notin D^c}$

M* ACCEPTS u* ***follows*** *from the* **definition** *of* $\boldsymbol{D^c}$ *and* $\boldsymbol{u^* \notin D^c}$

Logical contradiction : not possible

A Logical Contradiction is the end of rational thought ;

Hence no such DTM ***M**** *encoded by* ***u**** *exists !*

*Is there a DTM **M** that decides **D** ?* ***NO ! Why***

*If a DTM **M** decides **D** then :*

*for **M'** same as **M** except h_{YES} and h_{NO} interchanged then **M'** **decides D^c***

*But this contradicts the previous result since D^c is not even **semidecidable***

*hence certainly not **decidable** !*

*But if **D** is NOT **decidable** then H_0 is not **decidable** where*

$$H_0 = (u \in \{0,1\}^* ; \{0,1\}^* \mid u = (u_1 ; u_2) ; M = M(u_1) \text{ and } M \text{ halts on } u_2)$$

Why ? *Because **D** is the special case of H_0 with $u_2 = u_1$*

*H_0 is a **semidecidable language** semidecided by a universal TM.*

The Anatomy of Problem Solvability

$1L_{DTM} \rightarrow 0 < x_1 < x_2 < \ldots, < x_m < \ldots \rightarrow +\infty$

Since every encoded DTM is represented by a positive integer we can list ALL DTMs as below **:**

$$T_1, T_2, \ldots, T_m, \ldots \rightarrow +\infty$$

For every DTM T_m there is a unique language L_m ***semidecided*** *by it* ($u \in L_m \Leftrightarrow T_m$ ***halts*** *on* u)

$$L_1, L_2, \ldots, L_m, \ldots \rightarrow +\infty$$

$\rightarrow$ ***ALL semidecidable*** *languages in* $\{0,1\}^*$

Some of the L_j above have the desirable property that for some $k > 0$ $L_j^c = L_k$

A special DTM composed using T_j and T_k, decides L_j by halting on ***ALL*** u *in* $\{0,1\}^*$ **:**

At h_{YES} *if* $u \in L_j$ *and at* h_{NO} *if* $u \in L_j^c = L_k$

These composed DTMs with 2 halt states and the languages they decide are listed below:

$$M_1, M_2, \ldots, M_m, \ldots \rightarrow +\infty$$

$$N_1, N_2, \ldots, N_m, \ldots \rightarrow +\infty$$

$\Longrightarrow$ *Each M_p is called an* ***ALGORITHM*** *that solves the* ***problem*** *encoded by the decidable language N_p*

Integer Interpretation of Problems and Solutions

$1L_{DTM} \rightarrow 0 < x_1 < x_2 < \ldots, < x_m < \ldots \rightarrow +\infty$

$T_1, T_2, \ldots, T_m, \ldots \rightarrow +\infty$ *every DTM (semi-solution candidate) is identified with an integer*

$L_1, L_2, \ldots, L_m, \ldots \rightarrow +\infty$

*Hence every **semidecidable** language (semi-solvable encoded problem) is also identified with an integer : namely the integer corresponding to the T_m that semidecides the language L_m*

Hence the set of DTMs and associated semidecidable languages are both COUNTABLE

*BUT every language **L** (encoded problem) is in **1-to-1 correspondence** with a **SUBSET of integers** using the construct : **1.L***

*Fact :the **SET OF ALL SUBSETS** of integers is **NOT COUNTABLE ! (see next slide)***

*Hence :the **SET OF ALL LANGUAGES (ALL ENCODED PROBLEMS)** is **NOT COUNTABLE !***

CONCLUSION : ***DECIDABLE PROBLEMS < SEMIDECIDABLE PROBLEMS < ALL PROBLEMS***

COUNTABLE ***UNCOUNTABLE***

Proof *(by contradiction)* ***of***

“ the ***SET OF ALL SUBSETS*** *of integers is* ***NOT COUNTABLE !*** *”*

Suppose it is ***countable*** *so that* ***all*** *subsets* **N** *are as counted as below :*

$S_1 , S_2 , \ldots , S_m , \ldots$

Define $D := (m \in N \mid m \in S_m) \subseteq N$, *so that the complement* $D^c = (m \in N \mid m \notin S_m) \subseteq N$

Since the above count covers ***all subsets of*** *N we must have for some* k *:*

$D^c = S_k$. *We ask the question whether* $k \in D^c$. *There are* **2** *cases :*

CASE1 : $k \in D^c$ *Then by definition* $k \notin S_k$ *so that* $D^c \neq S_k$

CASE2 : $k \notin D^c$ *(or* $k \in D$ *)* *Then again by definition* $k \in S_k$ *so that again* $D^c \neq S_k$

Therefore the countability assumption above is false and the result follows.