***CS 407 Theory of Computation***
***Spring 2018***

***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***

# 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*

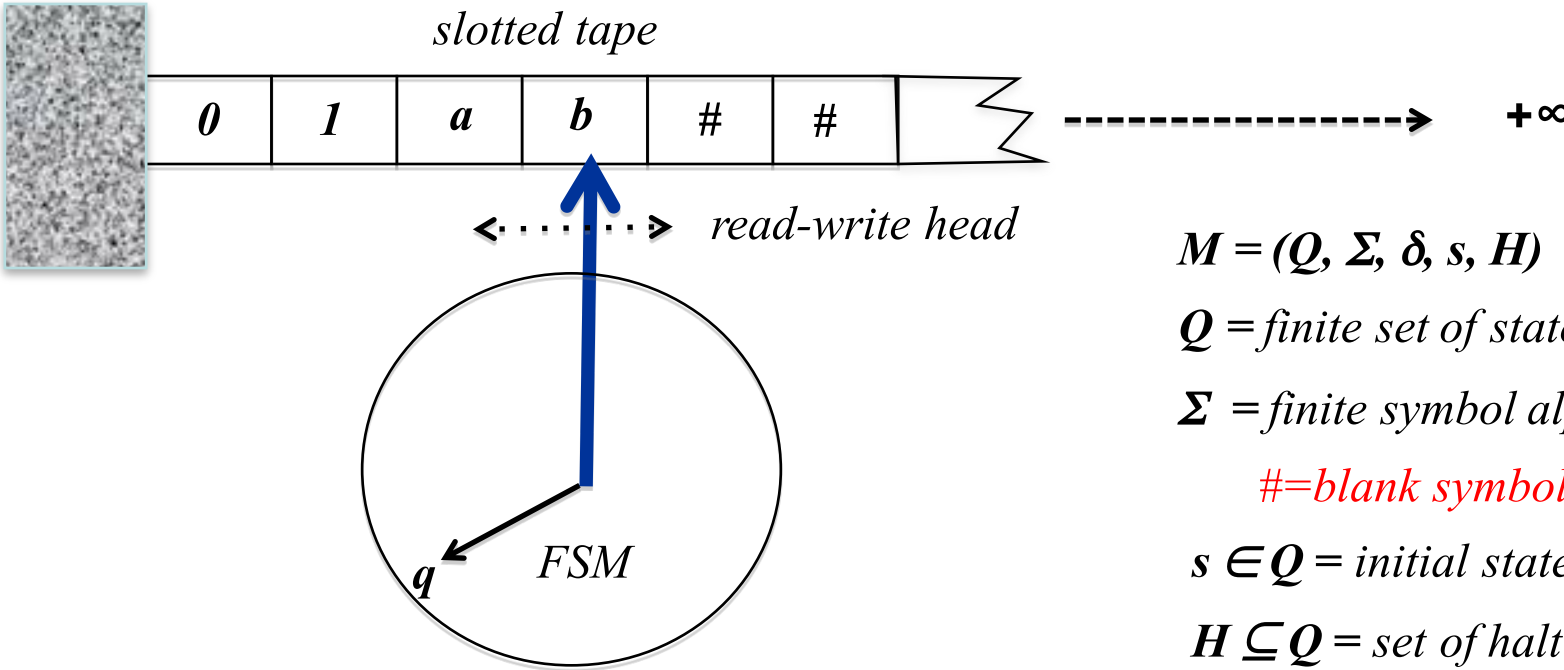
$(q, u\ \underline{a}\ v) \in Q \times (\Sigma^* \times \Sigma \times (\Sigma^*.(\Sigma - \{\#\}) \cup e))$

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

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

# Introduction to Turing Machines

*Turing Machine **M***




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

$Q$ = *finite set of states*

$\Sigma$ = *finite symbol alphabet*

*#=blank symbol*

$s \in Q$ = *initial state*

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

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

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

$\delta$ = *transition function*

# *Tabular Representation of the Transition Function*

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

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

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

| 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 | $q_f$ | $\rightarrow$ |
| $q_{b1}$ | # | $h$ | # |
| $q_{b1}$ | 0 | $q_{b2}$ | # |
| $q_{b1}$ | 1 | $q_{b2}$ | # |
| $q_{b2}$ | # | $q_{b1}$ | $\leftarrow$ |
| $q_{b2}$ | 0 | $q_{b1}$ | $\leftarrow$ |
| $q_{b2}$ | 1 | $q_{b1}$ | $\leftarrow$ |
| | | | |

# The Composite Turing Machine

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

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

= *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* $\boldsymbol{A \subseteq \Sigma}$ (*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

$$\begin{tikzcd} > R_{\#} \,.\, L \arrow[r, "\Sigma-\{\#\}"] \arrow[d, "\#"] & \# \arrow[l, bend right] \\ h \end{tikzcd}$$

*remove to simplify*

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

$$(s\,,\, \underline{\#}\;\; w) \mid\!\!-\!-_{RS}{}^{*}\, (h,\, \underline{\#}\;\; \#\, w)$$

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

# *Tabular Representations*

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

| *TM* | *Condition* | *Next TM* |
|---|---|---|
| $R_\#$ | - | *B* |
| *B* = *L* | $\sigma \neq \#$ | #. *B* |
| | $\sigma = \#$ | *h* |

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

| *TM* | *Condition* | *Next TM* |
|---|---|---|
| $R_\#$ | - | *B* |
| *B* = *L* | $\sigma \neq \#$ | *R*. $\sigma$. *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* |
|---|---|---|
| $L_{\#}$ | - | *B* |
| *B* = *R* | $\sigma \neq \#$ | $L \,.\, \sigma \,.\, R \,.\, B$ |
| | $\sigma = \#$ | $L \,.\, \# \,.\, h$ |

## 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 :*

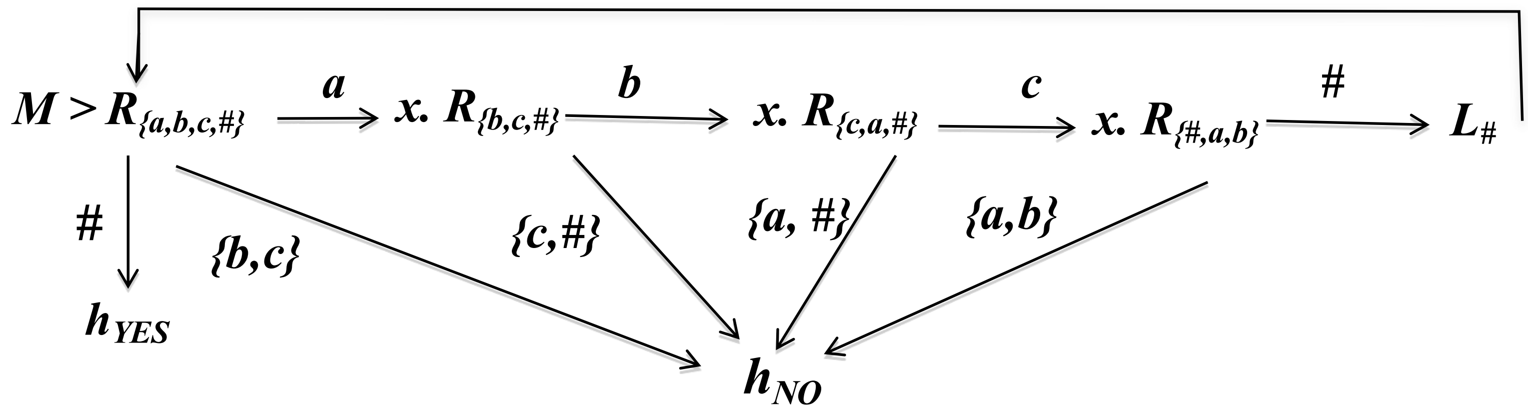
$(s, \underline{\#}\ w) |\text{--}_M^* (h, u\ \underline{a}\ v)$, iff $u = e$ ; $a = \#$ and $v = f(w)$

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

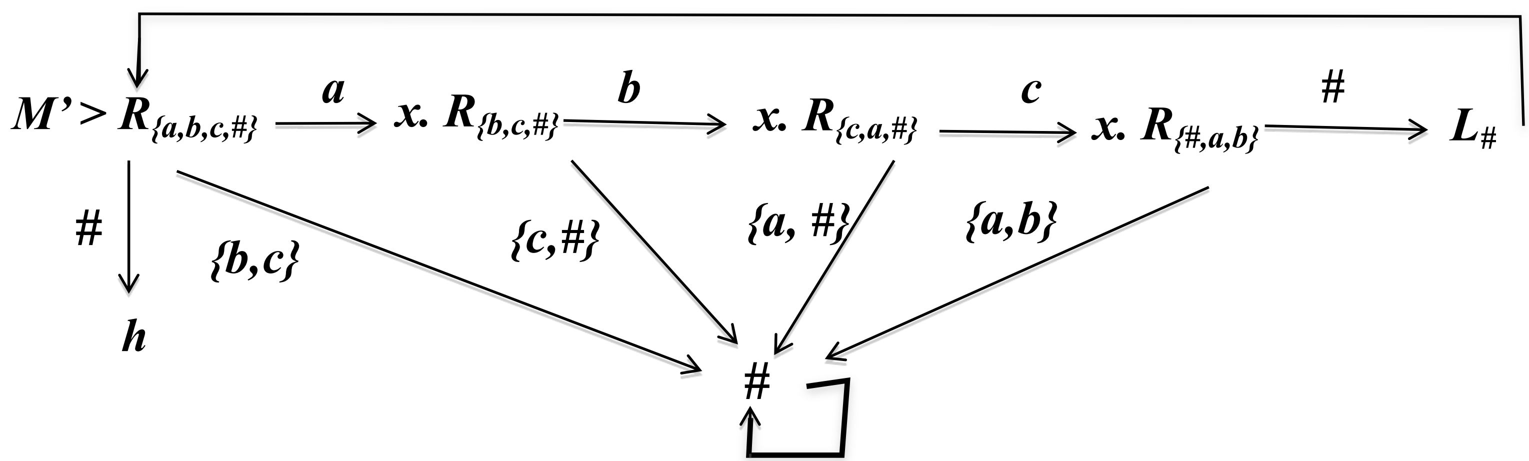
$(s, \underline{\#}\ w) |\text{--}_M^* (h, u\ \underline{a}\ v)$, iff $w \in L$

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

*Let **Σ= {a,b,c,x,#}***




*Example : A TM **M'** that **semidecides** the language **L = {$a^n$ $b^n$ $c^n$ ; n≥ 0}***



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

***f(w) = w.w^R ; (s, e <u>#</u> w) |--_M* (h, e <u>#</u> w.w^R )***

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

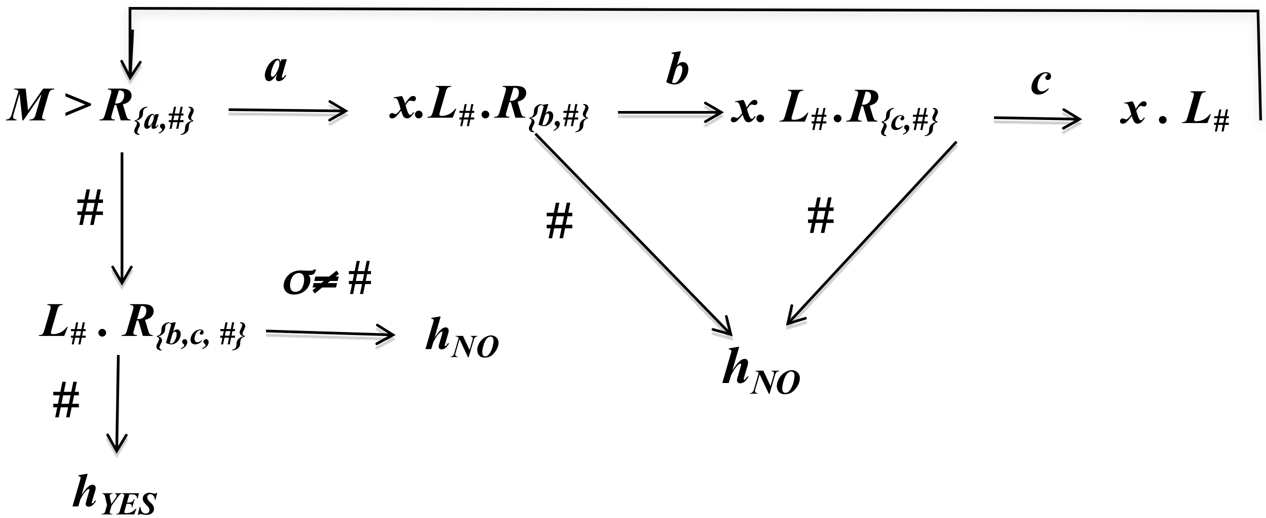
$$\begin{tikzcd}
M > R_\# \,.\, L \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"] & x\,.R_\#\,.\,\sigma.L_x\,.\sigma \arrow[l, bend right] \\
h &
\end{tikzcd}$$

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

| *TM* | *Condition* | *Next TM* |
|---|---|---|
| $R_\#$ | - | $B$ |
| $B = L$ | $\sigma \neq \#$ | $x\,.R_\#.\sigma.L_x\,.\sigma.B$ |
| | $\sigma = \#$ | $h$ |

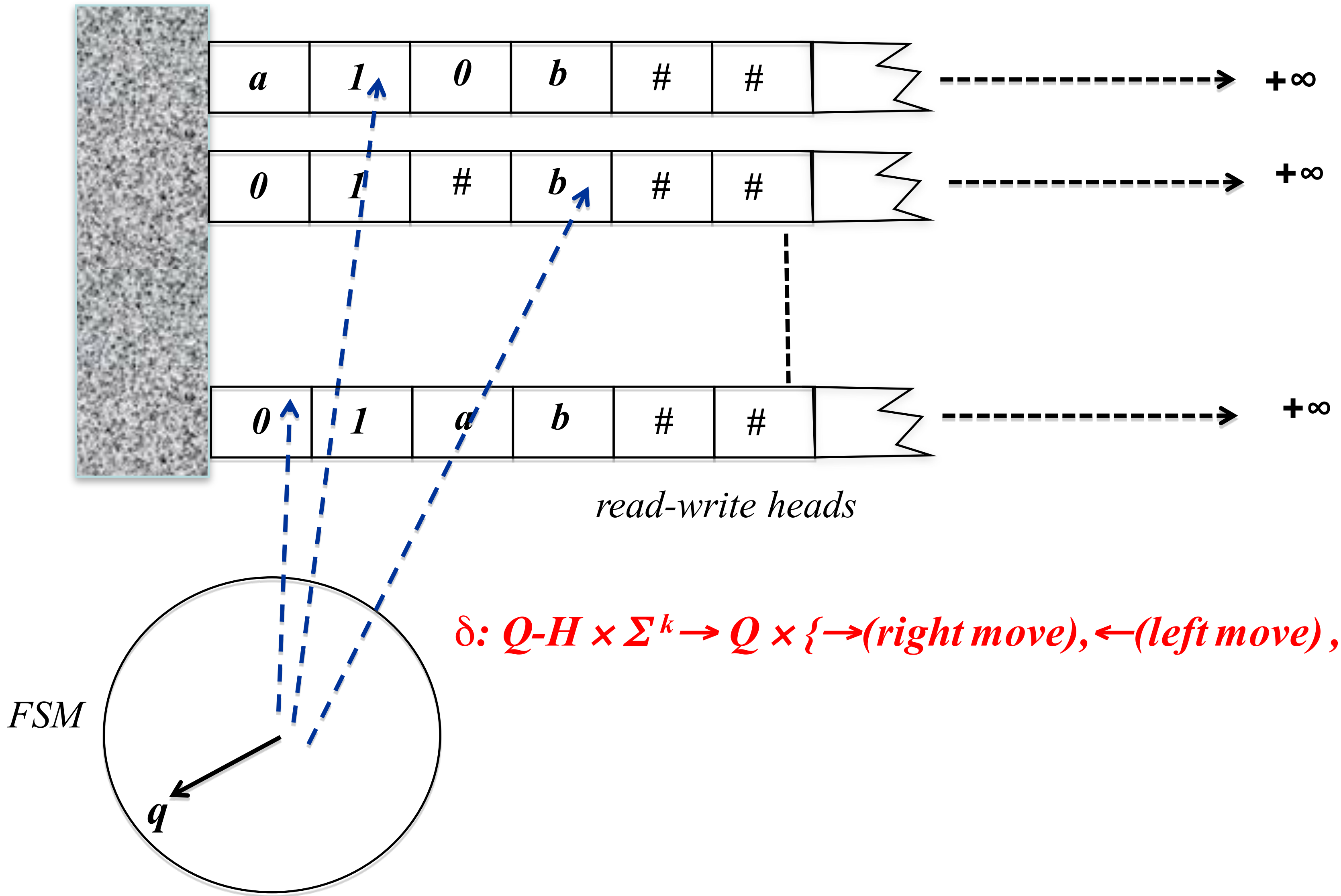
*Example : A TM* ***M*** *that* ***decides*** *the language* $\mathbf{L = \{\omega \in \{a,b,c\}^* \mid \#a\text{s} = \#b\text{s} = \#c\text{s}\}}$

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



# Multitape TM

*k slotted tapes ; k heads*




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

# 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 :**

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

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

## *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)

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

**Every nondeterministic TM can be simulated by a standard TM ?**

**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*

$(s, \#\, w) \mid\!\!-\!\!-_M^{K}\ C$ *(i.e. All computations* **halt** *with YES or NO before* **K** *steps !)*

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

*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 :* $(s, \#\, w) \mid\!\!-\!\!-_M^{*}\ (h_{YES}, u\, \underline{a}\, v)$

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

*1- Condition 1- of decidability above holds*

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

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

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

| Description | TM | Condition | Next TM |
|---|---|---|---|
| Immediately accept if $\omega = e$ | $R^1$ | $\sigma^1 = \#$ | $h_{YES}$ |
| Move head to a midpoint nondeterministically ; copy first entry of 2nd half to 2nd tape 1st entry and delete it with **d** ; if # is reached then reject ! | $A = R^1$ | $\sigma^1 = x \neq \#$ | $A$ |
| | | $\sigma^1 = x \neq \#$ | $R^2 . x^2 . d^1 . B$ |
| | | $\sigma^1 = \#$ | $h_{NO}$ |
| Copy entire 2nd half of 1st tape to 2nd tape meanwhile replacing copied entries with **d** | $B = R^1R^2$ | $\sigma^1 = x \neq \#$ | $x^2 . d^1 . B$ |
| | | $\sigma^1 = \#$ | $C$ |
| Replace all **d**s with # in tape 1 and after that move heads 1 and 2 to leftmost # to make them ready for comparison | $C = L^1$ | $\sigma^1 = d$ | $\#^1 . C$ |
| | | $\sigma^1 \neq d$ | $L_{\#}^1 . L_{\#}^2 . D$ |
| Compare contents of 1st tape and 2nd tape ; if equal accept with $h_{YES}$ ; if different reject with $h_{NO}$ ! | $D = R^1R^2$ | $\sigma^1 = \sigma^2 \neq \#$ | $D$ |
| | | $\sigma^1 = \sigma^2 = \#$ | $h_{YES}$ |
| | | else | $h_{NO}$ |

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

$(s\ ;\ \underline{\#}\ \omega_1 \underline{\#}\ \ \omega_2\ ,\ \underline{\#}\ )\ |\text{--}\ (h,\ \underline{\#}\ \ \text{“}\omega_1 + \omega_2\text{”},\ \underline{\#}\ \ )$

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

*Copy **$\omega_1$** 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*

# 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 → # $(q_1, a_1, q'_1, action_1) \ldots (q_n, a_n, q'_n, action_n)$ #***

***Tape 3 is current state → #xx#***

# *The Halting Problem*

***A simple-minded CS formulation of the 'Halting Problem'***

*Let **X,Y** be codes that may be interpreted both as **data** and an **executable program***

***halts (X, Y)** is a predicate function with arguments as codes **X** and **Y** and is true iff*

***X** as a program halts on the data **Y.***

*Now consider the program **diagonal** with input data **X** as below :*

***diagonal (X)***

***a :** if **halts(X,X)** then go to **a** **;** else **halt***

***What does diagonal (diagonal) do ?***

***diagonal(diagonal)** halts if and only if : **halts( diagonal, diagonal)** is false*

*if and only if **diagonal(diagonal)** does not halt ! A logical contradiction !*

## *The Halting Problem*

*Given a UTM* $\boldsymbol{U}$ *with alphabet* $\boldsymbol{\Sigma_U}$ *every TM and its input can be encoded by symbols in* $\boldsymbol{\Sigma_U}$

*If* $\boldsymbol{M}$ *is the TM and* $\boldsymbol{\omega}$ *is its input then let* $< \boldsymbol{M} >$ *and* $< \boldsymbol{\omega} >$ *denote their corresponding encodings for* $\boldsymbol{U}$ *, which are strings in* $\boldsymbol{\Sigma_U}^*$.

*Let* $\boldsymbol{H_0} \subseteq \boldsymbol{\Sigma_U}^*$ *be defined as below :*

$\boldsymbol{H_0} := \{ \boldsymbol{u} \in \boldsymbol{\Sigma_U}^* \mid \boldsymbol{u} = < \boldsymbol{M,\omega} > =$ *legitimate joint encoding of a TM and its input ;* $\boldsymbol{M}$ *halts on* $\boldsymbol{\omega} \}$

## *Theorem*

$\boldsymbol{H_0}$ *is a* ***semidecidable*** *but not a* ***decidable*** *language*