9/17/2003 NOTES:
Sotirios Kotsopoulos

4.217 PROSEMINAR IN DESIGN & COMPUTATION

TURING MACHINES

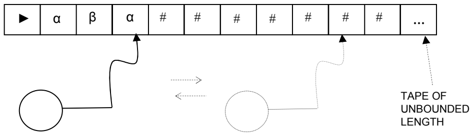
OBJECTIVE: DEFINE THE LIMIT OF “COMPUTABLE” & “UNCOMPUTABLE”

A GENERAL- PURPOSE COMPUTATIONAL MODEL EQUIVALENT IN POWER TO PROGRAMMING LANGUAGES, THAT IS SIMPLE ENOUGH FORMALLY SO THAT WE CAN PROVE WHAT CANNOT BE COMPUTED

REFERENCE PAPERS:

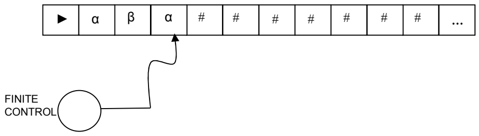
- **A. M. TURING, “ON COMPUTABLE NUMBERS WITH AN APPLICATION TO THE ENTSCHELDUNGSPROBLEM” (1936)**
- **D. HILBERT, “MATHEMATICAL PROBLEMS”, (1900)**
- **K. GODEL, “ON FORMALLY UNDECIDABLE PROPOSITIONS” (1931)**

THE BASIC TURING MACHINE



- **TAPE OF UNBOUNDED LENGTH**
- **HEAD CAN READ, WRITE, MOVE LEFT, MOVE RIGHT**
- **# IS THE BLANK SYMBOL**
- **ALL BUT A FINITE NUMBER OF SQUARES ARE BLANK**

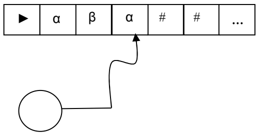
FORMAL DEFINITIONS FOR TMs



A DETERMINISTIC TM IS A 5-TUPLE (K, Σ, δ, s, H)
WHERE

- K IS A FINITE SET OF STATES
- Σ IS AN ALPHABET
  - CONTAINING THE BLANK SYMBOL #
  - NOT CONTAINING THE LEFT END SYMBOL ►
  - NOT CONTAINING THE MOVE SYMBOLS → AND ← (MOVE RIGHT AND LEFT)
- s Є K IS THE START STATE
- H ⊆ K IS THE SET OF HALTING STATES
- δ IS THE TRANSITION FUNCTION, A FUNCTION FROM (K – H) x (Σ U {►}) TO K x (ΣU {←, →}) SUCH THAT δ(q, ►) Є K x {→} FOR ALL q Є K-H

FORMAL DEFINITIONS FOR TMs



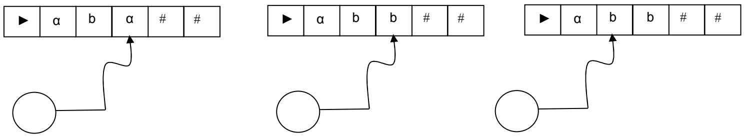
- TM TRANSITION FUNCTION (K – H) x (Σ U {►}) TO K x (ΣU {←, →})
  → MEANS "MOVE RIGHT"
  ← MEANS "MOVE LEFT"
  BUT → AND ← ARE NOT SYMBOLS OF THE ALPHABET

- THE BLANK SYMBOL # IS PART OF EVERY ALPHABET

- THE LEFT ENDMARKER ► INDICATES THE LEFT END OF THE TAPE
  (IT CANNOT BE WRITTEN TO ANY OTHER SQUARE OF THE TAPE)

- FOR ANY STATE q Є K, THERE IS A q' SUCH THAT δ(q, ►) = (q', →)
  THAT IS THE HEAD ALWAYS MOVES BACK ON THE TAPE IF IT FALLS OFF

FORMAL DEFINITIONS FOR TMs



MOVES OF A TM

- **A TM CANNOT MOVE THE HEAD LEFT AND REWRITE A TAPE-SQUARE IN ONE STEP**

EXAMPLES:

δ(q, α) = (p, b) WHERE b Є Σ MEANS

“REWRITE α AS b IN THE CURRENT SQUARE AND LEAVE THE HEAD AT THE SAME PLACE”

δ(q, b) = (p, ←)
MEANS “MOVE THE HEAD LEFT WITHOUT WRITING ANYTHING ON THE TAPE ”

(WITHOUT LOSS OF POWER AN TM CAN WRITE AND MOVE IN TWO STEPS)

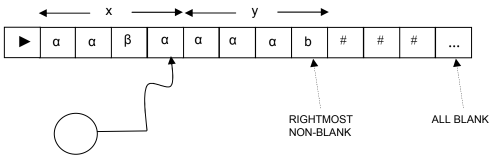
EXAMPLE OF A TM

q	α	δ(q, σ)
$\mathbf{q_0}$	α	(q_0, ←)
$\mathbf{q_0}$	#	(h, #)
$\mathbf{q_0}$	▶	(q_0, →)

- **THE TABLE DESCRIBES A TM THAT SCANS TO THE LEFT UNTIL IT FINDS A BLANK AND THEN HALTS**

(THE MACHINE GOES INTO A LOOP IF A BLANK SQUARE CANNOT BE FOUND)

FORMAL DEFINITIONS FOR TMs

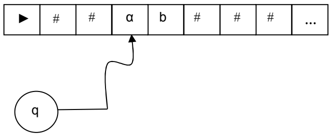


A CONFIGURATION OF A TM (K, Σ, δ, s, H)
LOOKS LIKE (q, ►x, y)
WHERE q IS THE CURRENT STATE AND THE TAPE LOOKS LIKE THE ONE ABOVE

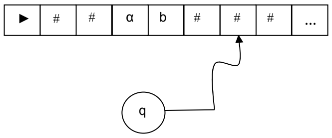
- **THE HEAD IS OVER THE RIGHTMOST SYMBOL OF ►x**
  IF x = e (THE EMPTY STRING) THEN THE HEAD IS OVER THE LEFT END OF THE TAPE
- **y EXTENDS TO THE RIGHTMOST NON-BLANK**
  IF y=e THEN THE HEAD IS TO THE RIGHT OF THE RIGHT MOST NON-BLANK OF x

---

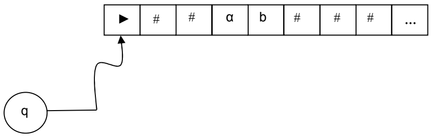
EXAMPLES OF CONFIGURATIONS OF A TM



$(q, \blacktriangleright\#\#\alpha, b)$ **OR** $(q, \blacktriangleright\#\#\underline{\alpha}b)$



$(q, \blacktriangleright\#\#\alpha b\#\#, e)$ **OR** $(q, \blacktriangleright\#\#\alpha b\#\underline{\#})$

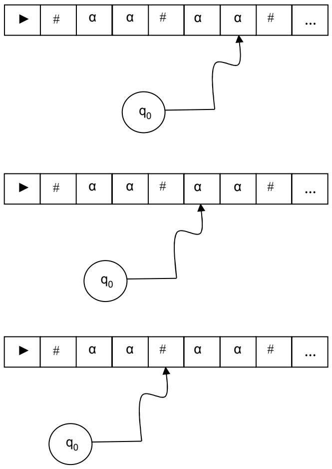


$(q, \blacktriangleright, \#\#\alpha b)$ **OR** $(q, \underline{\blacktriangleright}\#\#\alpha b)$

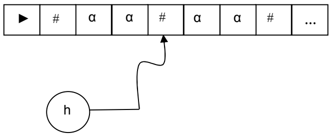
(THE HEAD IS OVER THE LEFT END)

EXAMPLE OF CONFIGURATIONS OF A TM

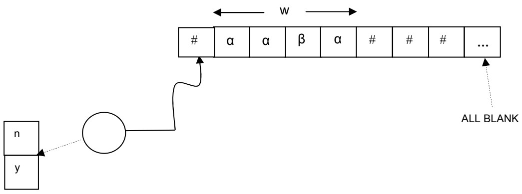
LET M BE THE MACHINE THAT SEARCHES FOR A BLANK TO THE LEFT OF THE CURRENT SQUARE



$$\begin{aligned} (q_0, \blacktriangleright\#\alpha\alpha\#\alpha\underline{\alpha}) &\vdash_M (q_0, \blacktriangleright\#\alpha\alpha\#\underline{\alpha}\alpha) \\ &\vdash_M (q_0, \blacktriangleright\#\alpha\alpha\underline{\#}\alpha\alpha) \\ &\vdash_M (h, \blacktriangleright\#\alpha\alpha\underline{\#}\alpha\alpha) \end{aligned}$$

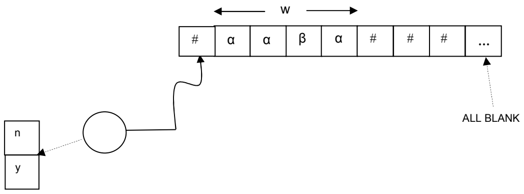


HOW TMS ARE USED?



- **AS LANGUAGE RECOGNIZERS, OR DECISION PROCEDURES**
  i.e. TO ANSWER QUESTIONS OF THE FORM:
  $w \in L$?
- **FOR THIS PURPOSE A TM HAS TWO HALTING STATES: *y* AND *n***
- **THE TM IS STARTED FOR INPUT THE STRING w (STRING w MUST NOT CONTAIN BLANKS)**
  AND HALTS IN STATE *y* OR *n*
  DEPENDING ON WHETHER $w \in L$ OR $w \notin L$

DECIDING A LANGUAGE



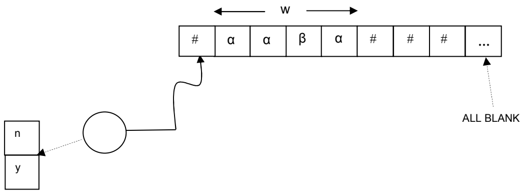
LET A DETERMINISTIC TM M= (K, Σ, δ, s, H)
WHERE H = { *y*, *n* }

- A CONFIGURATION WITH STATE *y* IS ACCEPTING
- A CONFIGURATION WITH STATE *n* IS REJECTING

- LET $\Sigma_0 \subseteq \Sigma$ BE AN ALPHABET NOT CONTAINING #

- M ACCEPTS w Є Σ_0^* IFF (s, ►# w) YIELDS AN ACCEPTING CONFIGURATION
- M REJECTS w IFF (s, ►# w) YIELDS A REJECTING CONFIGURATION

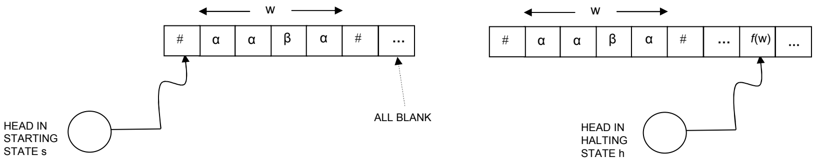
THE RECURSIVE LANGUAGES



THE MACHINE M DECIDES THE LANGAUGE $L \subseteq \Sigma_0^*$ IFF FOR ANY STRING w Є Σ_0^*
w ∉ L IFF M ACCEPTS THE STRING w
w Є L IFF M REJECTS THE STRING w

- A LANGUAGE L IS RECURSIVE IFF THERE IS A TM THAT DECIDES IT
  (NO GUARANTEES ABOUT WHAT THE MACHINE MAY DO IF THE INPUT IS INSERTED IMPROPERLY)
- QUESTION: HOW MANY RECURSIVE SETS ARE THERE?
  ANSWER: COUNTABLY MANY BECAUSE THERE ARE DECIDED BY TMS WHICH ARE FINITE

USING TMS TO COMPUTE FUNCTIONS



TMS ARE OFTEN USED TO COMPUTE FUNCTIONS

LET *f* BE A FUNCTION $\Sigma_0^* \to \Sigma_1^*$

A TM M COMPUTES *f* IFF FOR ANY w Є Σ_0^*

$$(s, \blacktriangleright \underline{\#} w) \vdash^*_M (h, \blacktriangleright \# \# \# \ldots \# f (w))$$

THAT IS, IF

- **THE ARGUMENT IS WRITTEN ON THE LEFT END OF A BLANK TAPE, PRECEDED BY ONE BLANK**
- **THE HEAD IS PLACED ON THE BLANK (JUST LEFT TO THE ARGUMENT)**
- **AND THE MACHIME IS STARTED IN ITS START STATE s, THEN THE MACHINE EVENTUALLY HALTS WITH THE VALUE ON OTHERWISE BLANK TAPE**

THE CLASS OF RECURSIVE FUNCTIONS

A FUNCTION *f* : $\Sigma_0^* \to \Sigma_1^*$ IS RECURSIVE IF THERE IS AN TM THAT COMPUTES IT

- **A FUNCTION FROM NUMBERS TO NUMBERS IS RECURSIVE IF THE STRING FUNCTION ON THEIR BINARY NOTATIONS IS RECURSIVE**

LET NUM = 1 { 0, 1 }* U { 0 } BE THE SET OF BINARY NUMERALS

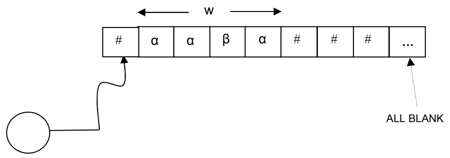
AND num : N → NUM BE THE FUNCTION THAT CONVERTS NUMBERS TO NUMERALS

(e.g. num (7) = '1 1 1')

THEN

- **A FUNCTION *f* : N → N IS RECURSIVE IFF THE FUNCTION *f* ' IS RECURSIVE, WHERE *f* ' (num(n)) = num ((*f* (n)) FOR EVERY n Є N**

RECURSIVELY ENUMERABLE SETS (RE) – SEMIDECIDING A LANGUAGE



LET M= (K, Σ, δ, s, H) BE A TM
AND LET $\Sigma_0 \subseteq \Sigma$ BE AN ALPHABET NOT CONTAINING #
ALSO, LET $L \subseteq \Sigma_0^*$

THEN, M SEMIDECIDES L IFF FOR ANY $w \in \Sigma_0^*$,

- **$w \in L$ IFF M HALTS ON INPUT w**
  (AND SO IF $w \notin L$, M DOES NOT HALT WHEN STARTED ON THIS INPUT)
- **THE LANGUAGE L IS RECURSIVELY ENUMERABLE (RE) IFF THERE IS A TM THAT SEMIDECIDES THE SET L**

SOME USES OF TMS

TO DECIDE A RECURSIVE SET **TO COMPUTE A RECURSIVE FUNCTION**	ALWAYS HALTS
TO SEMIDECIDE AN RE SET **TO COMPUTE A PARTIAL RECURSIVE FUNCTION**	MAY NOT HALT

- **NOT EVERY TM DECIDES A LANGUAGE OR COMPUTES A RECURSIVE FUNCTION**
  (EXAMPLE: A TM THAT NEVER HALTS)
- **AS LONG AS $\Sigma_0 \subseteq \Sigma - \{\#\}$, (WHERE Σ_0 IS AN ALPHABET NOT CONTAINING #)**
  THERE IS ASSOCIATED WITH M AN RE SET $\subseteq \Sigma_0^*$ IT SEMIDECIDES

A PROGRAMMING NOTATION FOR TMS

TURING MACHINES CAN BE COMBINED.

INDIVIDUAL MACHINES BECOME STATES CONNECTED TO EACH OTHER.

A MACHINE MAY START WHEN A PREVIOUS ONE HALTS.

THE MACHIMNE STARTS FROM ITS INITIAL STATE WITH THE TAPE AND HEAD POSITION AS THEY WERE LEFT BY THE FIRST MACHINE

$>M_1 \rightarrow M_2$

- “ EXECUTE M_1 UNTIL IT WOULD HALT;
  THEN BEGIN M_2 ”

$>M_1 \xrightarrow{\alpha} M_2$

- “ EXECUTE M_1 UNTIL IT WOULD HALT;
  IF THE HEAD IS OVER AN α,
  THEN BEGIN M_2 ”

$$\begin{tikzcd}
>M_1 \arrow[r, "\alpha"] \arrow[d, "b"'] & M_2 \\
M_3 &
\end{tikzcd}$$

- “ EXECUTE M_1 UNTIL IT WOULD HALT;
  IF THE HEAD IS OVER AN α, BEGIN M_2
  ELSE IF THE HEAD IS OVER A b INITIATE M_3 ”

A PROGRAMMING NOTATION FOR TMS

EXAMPLES CONTINUE:

$\overline{\alpha}$	“…ANYTHING BUT α…”
R	“MOVE RIGHT ONE SQUARE AND HALT”
L	“MOVE LEFT ONE SQUARE AND HALT”
α	“WRITE THE SYMBOL α”

$>M_1 \xrightarrow{\overline{\alpha}} M_2$

- “ EXECUTE M_1 UNTIL IT WOULD HALT;
  IF THE HEAD IS OVER TO ANYTHING BUT α, BEGIN M_2 ”

$> R \circlearrowleft^{\overline{\alpha}}$

- “ SEARCH TO THE RIGHT FOR AN α” R_α

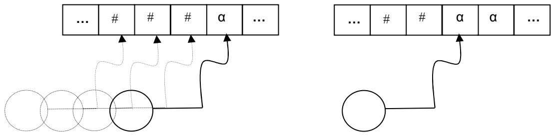
$> L \circlearrowleft_{\overline{\#}}$

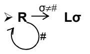
- “ SEARCH TO THE LEFT FOR A NON-BLANK SQUARE $L_\#$

$> R_\alpha \rightarrow R_\alpha$

- “ SEARCH TO THE RIGHT FOR THE SECOND α” R_α^2

A PROGRAMMING NOTATION FOR TMS

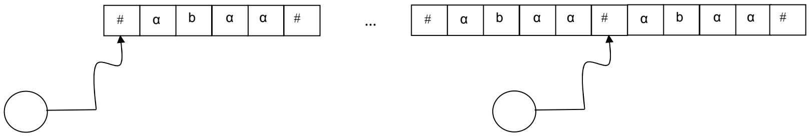


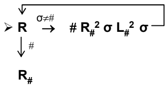


➢ $R \xrightarrow{\sigma \neq \#} L\sigma$

- “ SEARCH TO THE RIGHT FOR A NONBLANK SQUARE,
  THEN WRITE THAT SYMBOL IN THE SQUARE JUST TO THE LEFT OF WHER IT WAS FOUND ”

THE COPYING MACHINE C



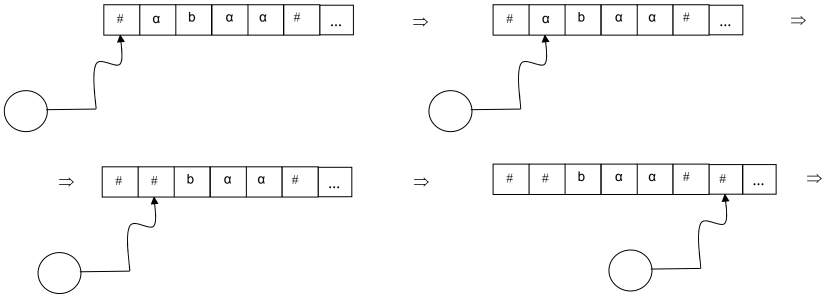


➢ $R \xrightarrow{\sigma \neq \#} \# R_{\#}^{2} \sigma L_{\#}^{2} \sigma$

$R_{\#}$

- THE MACHINE STARTS WITH SOME INPUT w ON AN OTHERWISE BLANK TAPE.
  EVENTUALLY THE MACHINE STOPS WITH # w # w # ON ITS OTHERWISE BLANK TAPE.
  (IT TRANSFORMS # w # INTO # w # w #)

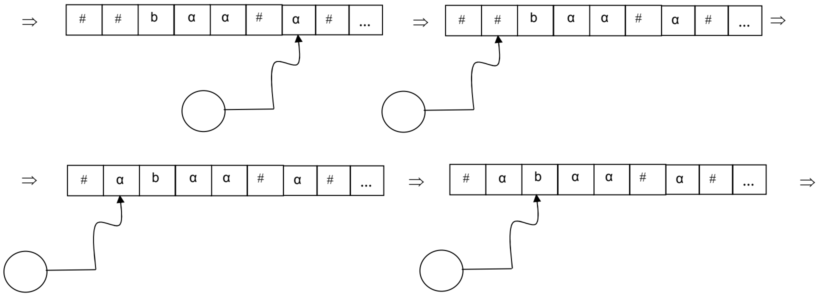
THE COPYING MACHINE C



➢ R $\xrightarrow{\sigma\neq\#}$ # $R_{\#}^{2}$

- THE MACHINE STARTS AT THE LEFTMOST # AND SCANS TO THE RIGHT FOR A NON-BLANK SQUARE. IT FINDS AN α (THAT MEANS σ = α) AND ERASES IT, WRITTING AN # ON ITS PLACE. THEN IT SCANS TO THE RIGHT FOR THE SECOND BLANK SQUARE.

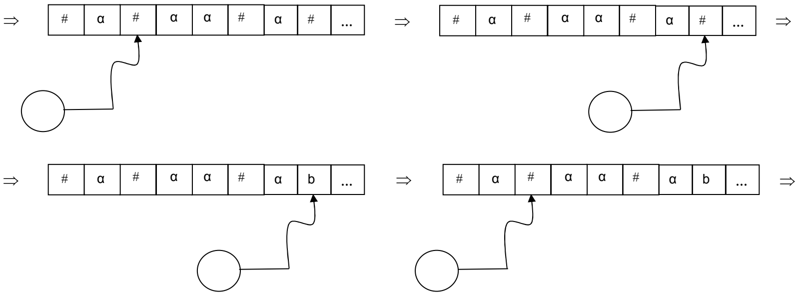
THE COPYING MACHINE C



➢ R $\xrightarrow{\sigma\neq\#}$ # $R_{\#}^{2}$ σ $L_{\#}^{2}$

- WHEN IT FINDS IT, IT WRITES AN α ON IT.
  THEN, IT MOVES TO THE LEFT UNTIL IT SCANS THE SECOND BLANK SQUARE, AND WRITES AGAIN α 0N IT.
  IT THEN REPEATS THE STEPS, SCANING TO THE RIGHT FOR THE FIRST NON- BLANK SQUARE. IT FINDS A b (THAT MEANS σ = b)

THE COPYING MACHINE C



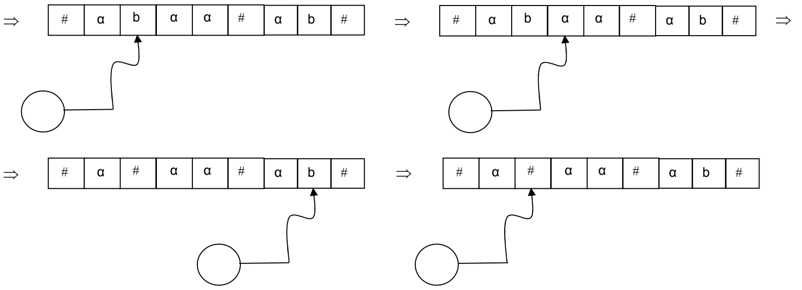
$\succ R \xrightarrow{\sigma \neq \#} \# R_{\#}^{2} \sigma L_{\#}^{2} \sigma$

- IT ERASES IT, WRITTING AN # ON ITS PLACE.

  IT THEN MOVES TO THE RIGHT LOOKING FOR THE SECOND NON-BLANK.

  IT WRITES A b ON IT, AND THEN SCANS TO THE LEFT LOOKINF FOR THE SECOND BLANK.

THE COPYING MACHINE C



$\succ R \xrightarrow{\sigma \neq \#} \# R_{\#}^{2} \sigma L_{\#}^{2} \sigma$

$\downarrow \#$

$R_{\#}$

- IT WRITES A b ON IT AND REPEATS OPERATION, SCANNING TO THE RIGHT FOR THE FIRST NON-BLANK… IT FINDS AN α (THAT IS σ = α)… ETC.

  FOLLOWING THE ALGORITHM EVENTUALLY THE MACHINE TERMINATES WITH # w # w # ON ITS TAPE.

THE LEFT-SHIFTING MACHINE S_L

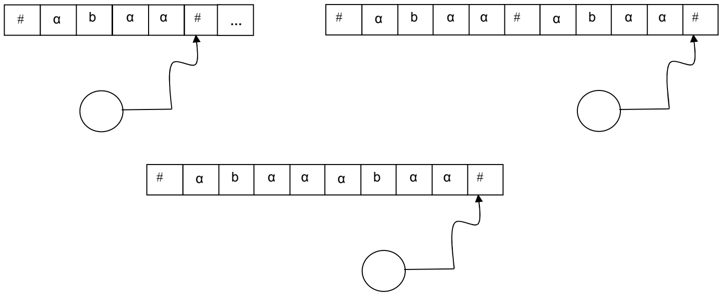
$$\begin{tikzcd} L_{\#} \arrow[r] & R \arrow[r, "\sigma\neq\#"] \arrow[d, "\#"] & L\sigma R \arrow[l, bend right] \\ & L\# & \end{tikzcd}$$

- THE MACHINE STARTS WITH SOME INPUT # w # ON THE TAPE.
  EVENTUALLY STOPS WITH w # MOVING THE STRING w ONE BLANK TO THE LEFT.
  NOTE THAT THE S_L MACHINE DOES NOT CARE IF THE BLANK SQUARE TO THE LEFT IS THE LEFTMOST SQUARE OF THE TAPE.

ACCORINGLY A COMBINATION OF MACHINES CAN COMPUTE THE FUNCTION f (w) = w w

- C S_L
- C $R_{\#}$ $S_L L_{\#}$

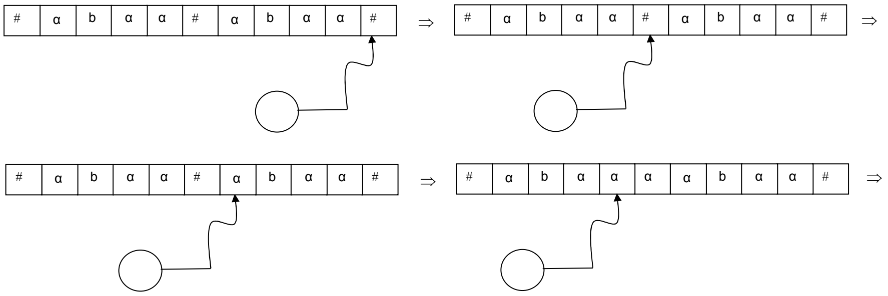
COMPUTING THE FUNCTION f (w) = w w

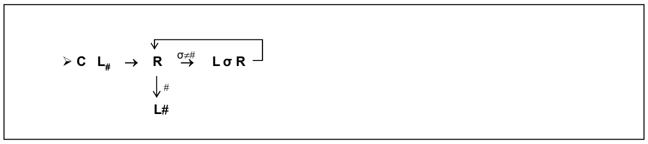


THE COMBINATION OF THE MACHINES C AND S_L COMPUTES THE FUNCTION f (w) = w w

- C S_L

f (w) = w w



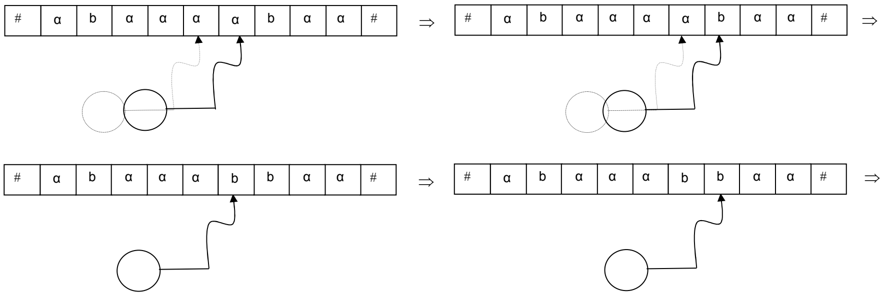


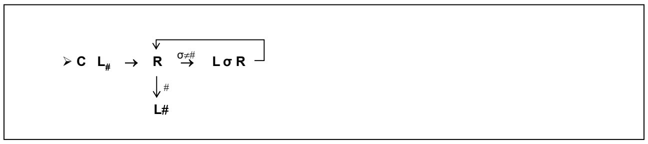
➢ C $L_{\#}$ → R $\xrightarrow{\sigma \neq \#}$ L σ R

$\downarrow \#$

L#

f (w) = w w



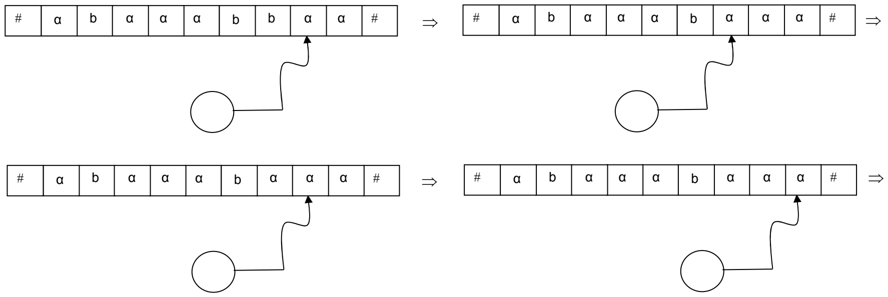


➢ C $L_{\#}$ → R $\xrightarrow{\sigma \neq \#}$ L σ R

$\downarrow \#$

L#

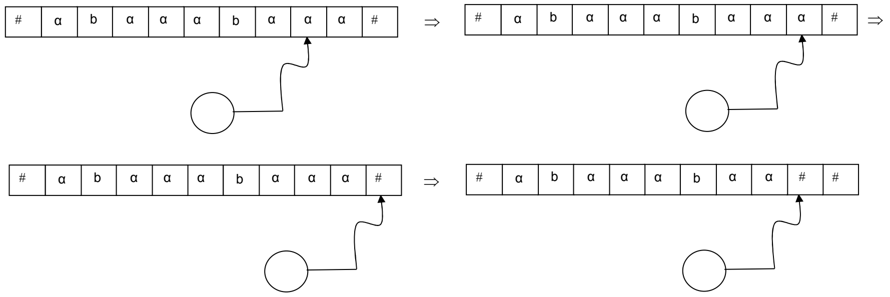
f (w) = w w



➢ $$\begin{tikzcd} C\ L_{\#} \arrow[r] & R \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"] & L \sigma R \arrow[l, bend right] \\ & L\# & \end{tikzcd}$$

---

f (w) = w w



➢ $$\begin{tikzcd} C\ L_{\#} \arrow[r] & R \arrow[r, "\sigma \neq \#"] \arrow[d, "\#"] & L \sigma R \arrow[l, bend right] \\ & L\# & \end{tikzcd}$$