

## TEOREMA DI RICE

Sia  $P$  una qualsiasi PROPRIETÀ non banale di un linguaggio riconosciuto da una TM (è r.e.). Allora, il problema di determinare se un linguaggio di una dete TM ha la proprietà  $P$  è NON DECIDIBILE.

IDEA:  $A_{TM} \leq P$ .

$P$  è NON BANALE se  $P \neq \emptyset \wedge P^c \neq \emptyset$ .

Per assurdo, sia  $P$  decidibile  $\Rightarrow \exists$  decisore  $R$  per  $P$ .

Sia  $T_0$  una TM che rifiuta sempre:  $L(T_0) = \emptyset$ . Supponiamo che  $T_0 \notin P$ ; posso farlo, perché altrimenti considero  $P^c$  al posto di  $P$ , anch'esso decidibile.

Sia  $T_1$  una TM tale che  $T_1 \in P$ ; esiste, poiché  $P$  è non banale.

Costruiamo un decisore  $S$  per  $A_{TM}$ :

$S =$  "On input  $\langle M, w \rangle$ :

1. Build from  $\langle M, w \rangle$  the encoding of the following TM  $M_w$ :

$M_w =$  "On input  $x$ :

- 1. Simulate  $M$  on  $w$ ; if  $M(w)$  rejects, then reject;
- 2. Simulate  $T_1$  on  $x$ ; if  $T_1(x)$  accepts, then accept."

2. Run  $R$  on input  $\langle M_w \rangle$ ;

3. If  $R(\langle M_w \rangle)$  accepts, then accept; otherwise reject."

\*  $M_w$  non viene eseguita! S termina sempre, poiché  $R$  termina, in quanto decisore.

$\Rightarrow S$  è un DECISORE.

- se  $w \in L(M) \Rightarrow M(w)$  accetta  $\Rightarrow M_w(x)$  va al passo 2  $\Rightarrow L(M_w) = L(T_1)$
  - se  $w \notin L(M) \Rightarrow M(w)$  non accetta  $\Rightarrow L(M_w) = \emptyset = L(T_0)$
- $M \in T_1 \in P \Rightarrow R(\langle M_w \rangle) = R(\langle T_1 \rangle) = \text{ACCEPT} \Rightarrow S(\langle M, w \rangle)$  accetta
- $M \in T_0 \notin P \Rightarrow R(\langle M_w \rangle) = R(\langle T_0 \rangle) = \text{REJECT} \Rightarrow S(\langle M, w \rangle)$  rifiuta

Dunque:

$$S(\langle M, w \rangle) = \begin{cases} \text{accetta, se } M(w) \text{ accetta} \\ \text{rifiuta, se } M(w) \text{ non accetta} \end{cases}$$

$\Rightarrow S$  decide  $A_{TM}$

$\nexists \quad \underline{\text{ASSURDO}}$

ok!!

$\Rightarrow P$  è NON DECIDIBILE

### Esempio di applicazioni:

- (1)  $REGULAR_{TM}$  è non decidibile, poiché le "regolarità" è una proprietà non basata sui linguaggi.
- (2) Analogamente,  $[CFL_{TM} \text{ è } \underline{\text{non decidibile}}]$

### TECNICHE PER DIMOSTRARE LA NON DECIDIBILITÀ

- (1) Metodo di Diagnalizzazione di Cantor;
- (2) "Riduzione" ad un problema già risolto ( $A_{TM}, E_{TM}, \dots$ );
- (3) Computation History.

### COMPUTATION HISTORY:

Una STORIA DI COMPUTAZIONE è definita solo per computazioni che terminano. Date  $M(w)$ , una computation history accettante è una sequenza finita di configurazioni  $C_1, \dots, C_k$  con:

- $C_1 = q_0^w$ ;
- $C_i \rightarrow C_{i+1}$  in  $M$ ,  $\forall i = 1, \dots, k-1$ ;
- $q_k \in C_k$  (oppure  $q_k \in C_k$  se rifiutante).

Se  $M(w)$  non termina  $\Rightarrow \nexists$  computation history!

\* Le DTM's hanno al più 1 C.H.

\* Le NTM's possono avere più di una C.H.

## LBA - Linear Bounded Automaton

Una LBA è una TM con 1 solo NASTRO e 1 testina, che è limitata a muoversi nella porzione di nastro che contiene l'input all'inizio.

\* Se input  $w = \epsilon \Rightarrow$  nastro = 1 blank = "L".

La quantità di nastro utilizzabile è lineare alla dimensione dell'input.

• **TEOREMA:**  $A_{LBA} = \{ \langle M, w \rangle \mid M \text{ is an LBA accepting } w \}$  è DECIDIBILE.

Consideriamo la CH  $M(w) : \underbrace{C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \dots}_{q,w}$

Sia  $q = |Q|$ ;  $g = |\Gamma|$ ;  $m = |w| = \text{dim. nastro}$ . Dunque abbiamo che:

# possibili configurazioni di un LBA =  $q \cdot m \cdot g^m$

Se il numero di configurazioni supera  $(q \cdot m \cdot g^m)$ , ci sarà una configurazione Ci che si ripete! Essendo l'LBA deterministico, vuol dire che non termina mai  $\rightarrow$  riesco a copiare quando NON TERMINA!

Consideriamo una UTM  $\lambda$ :

$\lambda =$  "On input  $\langle M, w \rangle$ , where  $M$  is an LBA:

1. Simulate  $M$  on input  $w$  for  $(q \cdot m \cdot g^m)$  steps or until  $M(w)$  halts;

2. If  $M$  has halted, then accept or reject accordingly to  $M(w)$ .

$\lambda$  accepts  $\Leftrightarrow M(w)$  accepts  $\Rightarrow \lambda$  decide  $A_{LBA} \Rightarrow A_{LBA}$  È DECIDIBILE

• **TEOREMA:**  $E_{LBA} = \{ \langle M \rangle \mid M \text{ is an LBA s.t. } \lambda(M) = \emptyset \}$  è NON DECIDIBILE.

IDEA:  $A_{TM}$  "reduces via computation history" to  $E_{LBA}$ .

•  $\langle M, w \rangle \in A_{TM}$ ?

Costruiamo un linguaggio  $\lambda(B) = \{ x \mid x \text{ è una CH accettante per } M(w) \}$

Facciamo vedere che una TM  $B$  che accetta  $x \Leftrightarrow x \in \lambda(B)$ , quindi  $B$  tale che  $\lambda(B) = \lambda(B)$  sopra descritto, è un LBA.

Una computation history  $x$  accettante è del tipo:

$$x = "C_0 \# C_1 \# C_2 \# \dots \# C_k"$$

$B =$  "On input  $x$ :

1. Check  $x$  is a series of strings divided by "#";
2. Check that  $C_0$  is the start configuration of  $M(w)$ ;
3. Check that  $C_k$  includes the state  $q_A$  of  $\text{TM } M$ ;
4. For any  $i$  from 0 to  $k-1$ : check that  $C_i \rightarrow C_{i+1}$ ;
5. If all checks have been passed, then accept; otherwise, reject."

(1)  $B$  è un decisore

(2)  $B$  fa check in-place; non ha bisogno di un mastro più lungo di  $1 \times 1$

$\Rightarrow B$  è un LBA !

Per assurdo, sia  $E_{\text{LBA}}$  decidibile  $\Rightarrow \exists$  decisore  $R$  per  $E_{\text{LBA}}$ .

Costruiamo un decisore  $S$  per  $A_{\text{TM}}$ :

$S =$  "On input  $\langle M, w \rangle$ :

1. Build from  $\langle M, w \rangle$  the encoding of the LBA  $B$ , that checks if an accepting computation history is valid;
2. Run  $R$  on input  $\langle B \rangle$ ;
3. If  $R(\langle B \rangle)$  accepts, then reject; otherwise accept."

$\bullet \langle M, w \rangle \in A_{\text{TM}} \Leftrightarrow R(\langle B \rangle)$  rejects  $\Leftrightarrow L(B) \neq \emptyset \Leftrightarrow \exists$  accepting C.H. for  $M(w)$   
 $\Leftrightarrow M(w)$  accepts  $\Leftrightarrow \underline{S(\langle M, w \rangle)}$  accepts

$\Rightarrow S$  decide  $A_{\text{TM}}$   $\Rightarrow \exists$  ASSURDO  $\rightarrow E_{\text{LBA}}$  è non decidibile!

TEOREMA:

$\text{All}_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG with } \Sigma \text{ such that } L(G) = \Sigma^* \}$   
è non decidibile.

IDEA:  $A_{\text{TM}}$  "reduces via computation history" to  $\text{All}_{\text{CFG}}$ .

Consideriamo una CFG  $G$  che genera tutte le STRINGHE che NON codificano una COMPUTATION HISTORY ACCETTANTE per  $M(w)$ .

Per assurdo, sia  $\text{All}_{\text{CFG}}$  decidibile  $\Rightarrow \exists$  decisore  $R$  per  $\text{All}_{\text{CFG}}$

Le CFG  $G$  generano:

- 1) ogni stringa composta da configurazioni non separate da  $\#$ ;
- 2) ogni stringa che non inizia con  $C_0 = q_0 w$ ;
- 3) ogni stringa che non termina con una configurazione accettante;
- 4) ogni stringa che contiene 2 configurazioni consecutive per le quali non esiste una TRANSIZIONE LEGALE in  $\Pi$ .

Poiché  $\text{CFG} \equiv \text{PDA}$ , costruiamo un PDA  $P$  che riconosce le stringhe di sopra.

L'unica condizione difficile da realizzare è la (4).

\* non posso spostare le testine all'indietro, quindi devo memorizzare le  $C_i$  sullo STACK, ma le ottengo alla rovescia:

- 1° TRUCCO: " $\# C_1 \# C_2^R \# C_3 \# C_4^R \dots$ "  
Se  $C_1 = q_0 w \rightarrow \# q_1 w' \Rightarrow C_2^R = w'^R q_1 \#$

Ma, confrontate  $C_1$  e  $C_2^R$ , se sono uguali, devo endere eventi; e come salvare  $C_2^R$  sullo STACK? → • 2° TRUCCO: NONDETERMINISMO:

inserisco all'inizio un numero arbitrario di " $\#$ " e ne consumo 1 ogni volta che faccio un confronto, fino a consumerli tutti.

L'automa ACCETTA se elenca 1 raso di computazione rileva una transizione ILLEGALE!

$S =$  "On input  $\langle \Pi, w \rangle$ :

1. Build the encoding  $\langle P \rangle$  from  $\langle \Pi, w \rangle$  of a PDA  $P$  that recognizes all strings that are not accepting computation history for  $\Pi(w)$ ;
2. Build from  $\langle P \rangle$  the encoding  $\langle G \rangle$  of an equivalent CFG  $G$ ;
3. Run  $R$  on  $\langle G \rangle$ ;
4. If  $R(\langle G \rangle)$  accepts, then reject; oth., accept."

$\langle \Pi, w \rangle \in A_{\text{TM}} \Leftrightarrow \Pi(w) \text{ accepts} \Leftrightarrow \exists \text{ accepting C.H. for } \Pi(w) \Leftrightarrow P(x) \text{ rejects} \Leftrightarrow x \notin L(G)$   
(detto "x")

$\Leftrightarrow L(G) \neq \Sigma^* \Leftrightarrow \langle G \rangle \notin \text{All}_{\text{CFG}}$   $\Leftrightarrow R(\langle G \rangle) \text{ rejects} \Leftrightarrow S(\langle \Pi, w \rangle) \text{ accepts}$

Quindi  $S$  DECIDE  $A_{\text{TM}} \rightarrow \boxed{\text{ASSURDO!}} \Rightarrow \text{All}_{\text{CFG}} \text{ è } \underline{\text{NON DECIDIBILE}}$ !

OLO!!!

• TEOREMA:  $\text{EQ}_{\text{CFG}} = \{\langle G, H \rangle \mid G \text{ and } H \text{ are CFG's s.t. } L(G) = L(H)\}$  è non decidibile.

$\text{All}_{\text{CFG}} \leq \text{EQ}_{\text{CFG}}$

Per essurdo, esiste un deciso R per  $\text{EQ}_{\text{CFG}}$ .

S = "On input  $\langle G \rangle$ :

1. Build  $\langle H \rangle$ , where H is a CFG such that  $L(H) = \Sigma^*$ ;
2. Run R on input  $\langle G, H \rangle$ ;
3. If R( $\langle G, H \rangle$ ) accepts, then accept; oth. reject."

S decide  $\text{All}_{\text{CFG}} \Rightarrow \exists \text{ ASSURDO!} \Rightarrow \text{EQ}_{\text{CFG}} \text{ è non decidibile!}$

• Homework:  $\text{EQ}_{\text{CFG}}^c$  è r.e.?

### POST CORRESPONDENCE PROBLEM

Le istanze del problema sono un insieme di "tessere del domino":

|               |
|---------------|
| top string    |
| bottom string |

PROBLEMA: Esiste una sequenza di tessere, eventualmente ripetute, tali che la concatenazione delle top strings coincida con quelle delle bottom strings?

\* PCP = Post Correspondence Problem

Consideriamo:

$\text{MPCP} = \{\langle P, d \rangle \mid P \text{ è un'istanza di PCP che ha un match, ma tale che}\}$   
 parte sempre delle tessere selezionate d

Costruiamo una RIDUZIONE:  $\text{MPCP} \leq \text{PCP}$ , tale per cui:

$$\langle P, d \rangle \in \text{MPCP} \Leftrightarrow \langle P' \rangle \in \text{PCP}.$$

Dunque cerchiamo di DIMOSTRARE CHE ESISTE UNA RIDUZIONE.

Se  $\Sigma^*$  un insieme di stringhe e " $\cdot$ "  $\notin \Sigma^*$ ; allora, se  $s = \text{"hello" } \in \Sigma^*$ :

- $s = \cdot h \cdot e \cdot l \cdot l \cdot o \cdot \cdot$
- $s_0 = \cdot h \cdot e \cdot l \cdot l \cdot o \cdot \cdot$
- $s_0 = \cdot h \cdot e \cdot l \cdot l \cdot e \cdot o \cdot \cdot$

Consideriamo  $\langle P, d \rangle$ :

$$\langle P, d \rangle = \left\langle \begin{array}{c} t_1 \\ b_1 \end{array}, \begin{array}{c} t_2 \\ b_2 \end{array}, \dots, \begin{array}{c} t_n \\ b_n \end{array}, \begin{array}{c} t_s \\ b_s \end{array} \right\rangle$$

Dunque,  $\langle P' \rangle$  sarà così:

$$\langle P' \rangle = \left\langle \begin{array}{c} \bullet t_1 \\ \bullet b_1 \end{array}, \begin{array}{c} \bullet t_2 \\ \bullet b_2 \end{array}, \dots, \begin{array}{c} \bullet t_n \\ \bullet b_n \end{array}, \begin{array}{c} \bullet \$ \\ \$ \end{array} \right\rangle \rightarrow$$

la tessera  
+ iniziale  
+ finale

- $\langle P, d \rangle \in \text{MPCP} \Rightarrow \langle P' \rangle \in \text{PCP}$ :

se  $\langle P, d \rangle \in \text{MPCP}$ ,  $\exists$  configurazione:

$$\frac{t_1 \dots t_j \dots t_m}{b_1 \dots b_j \dots b_m}$$

Ma allora:  $\frac{\bullet t_1 \dots \bullet t_j \dots \bullet t_m \bullet \$}{\bullet b_1 \dots \bullet b_j \dots \bullet b_m \bullet \$}$  è uguale sopra e sotto  $\Rightarrow \langle P' \rangle \in \text{PCP}$

- $\langle P' \rangle \in \text{PCP} \Rightarrow \langle P, d \rangle \in \text{MPCP}$ :

per mettere, devo iniziare per forza con

$$\begin{array}{c} \bullet t_1 \\ \bullet b_1 \end{array}$$

e terminare con

$$\begin{array}{c} \$ \\ \$ \end{array}$$

Se  $\langle P' \rangle \in \text{PCP} \Rightarrow$  esiste una corrispondenza  $\Rightarrow$  togliendo tessera iniziale e finale ed i pallini, si ottiene un match per MPCP  $\Rightarrow \langle P, d \rangle \in \text{MPCP}$ .

$\Rightarrow$  È possibile ridurre MPCP " $\leq$ " PCP

OK!!!

- TEOREMA:** MPCP è non decidibile

$\text{ATM} \leq \text{MPCP}$ .

Consideriamo  $\langle M, w \rangle$ , con  $M$  una TM semi-infinite tape e L-R moves.

Costruiamo un'istanza  $\langle P, d \rangle$  per MPCP:

- TESSERA  $d$ :

$$\text{Se } w = w_1 \dots w_m \Rightarrow d = \frac{\#}{\# w_1 \dots w_m}$$

- RIGHT-Moves:

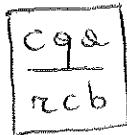
$\forall a, b \in \Gamma, \forall q, r \in Q, q \neq q_R$ ; se esiste  $\delta(q, a) = (r, b, R)$ :

allora  $\begin{array}{c} q_R \\ b_R \end{array} \in P$ ;

$$\begin{array}{c} q_R \\ b_R \end{array}$$

• MOVE LEFT - MOVE:

$\forall a, b, c \in \Gamma, \forall q_1, q_2 \in Q, q \neq q_R ;$  se esiste  $\delta(q, a) = (q_1, b, L)$ , allora:



$\in P$

• NULL-MOVE:  $\forall q \in \Gamma : \boxed{\frac{q}{q}} \in P$

• ACCEPTANCE:

$\forall q \in \Gamma : \boxed{\frac{aq_1}{q_A}} \in P$  and  $\boxed{\frac{q_A a}{q_A}} \in P$

• SKIP SYMBOL:

se  $\# \notin \Gamma \Rightarrow \boxed{\frac{\#}{\#}} \in P$

• TAPE EXPANSION TO RIGHT:  $\boxed{\frac{\#}{\# \#}} \in P$

• NULL-LEFT-MOVE:

$\forall a, b, c \in \Gamma, \forall q_1, q_2 \in Q, q \neq q_R ;$  se esiste  $\delta(q, a) = (q_1, b, R)$ , allora:

$\boxed{\frac{\# q_2}{\# q_1}} \in P$  (per evitare problemi logici ai nastri)  
semplicemente verso sx

Per assurdo, sia MPCP decidibile  $\Rightarrow \exists$  decisore R per MPCP.

S = "On input  $\langle M, w \rangle$ :

1. Build the instance  $\langle P, d \rangle$  corresponding to  $\langle M, w \rangle$ ;

2. Run R on input  $\langle P, d \rangle$ ;

3. If  $R(\langle P, d \rangle)$  accepts, then accept; otherwise reject."

Ma S decide  $A_{TM} \rightarrow \exists$  ASSURDO !

$\Rightarrow [MPCP \text{ è } \underline{\text{NON}} \text{ decidibile!}]$



del!!

• TEOREMA: PCP è non decidibile.

MPCP " $\leq$ " PCP. Abbiamo visto che tale riduzione è possibile, facendo corrispondere  $\langle P, d \rangle \in \text{MPCP} \Leftrightarrow \langle P' \rangle \in \text{PCP}$ .

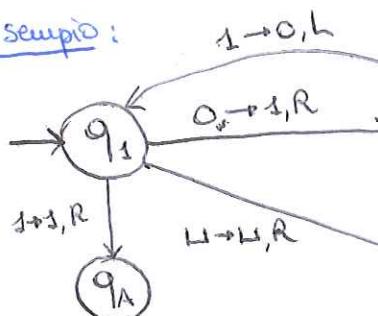
Per esserlo, esiste un deciso R per PCP.

$T =$  "On input  $\langle P, d \rangle$ :

1. Build  $\langle P' \rangle$  instance for PCP, from  $\langle P, d \rangle$ ;
2. Run R on input  $\langle P' \rangle$ ;
3. If  $R(\langle P' \rangle)$  accepts, then accept; oth. reject."

Ma, T decide MPCP  $\rightarrow \exists$  ASSURDO  $\Rightarrow$  PCP è non decidibile!

• Esempio:



Sia e' input  $w = "01"$

M:

Costruiamo l'istanza  $\langle P, d \rangle$  da M e w:

$$d = \frac{\#}{\# q_1 0 \$ \#}$$

RIGHT MOVES:

$$\frac{q_1 0}{-1 q_2}, \quad \frac{q_1 1}{-1 q_A}$$

LEFT MOVES:

$$\frac{0 q_2 1}{q_1 0 0}, \quad \frac{1 q_2 1}{q_1 1 0}, \quad \frac{L q_2 1}{q_1 L 0}, \quad \frac{\# q_2 1}{\# q_1 0}$$

SKIP MOVES:

$$\frac{L}{L}, \quad \frac{0}{0}, \quad \frac{1}{1}, \quad \frac{\#}{\#}$$

EXPANSION:

$$\frac{\#}{L \#}$$

ALL YOU CAN EAT:

$$\frac{q_1 0}{q_A}, \quad \frac{q_A 1}{q_A}, \quad \frac{q_A L}{q_A}, \quad \frac{0 q_A}{q_A}, \quad \frac{1 q_A}{q_A}, \quad \frac{L q_A}{q_A}$$

ACCEPTANCE:

$$\frac{q_A \# \#}{\#}$$

Simuliamo  $M(w) = M("01")$ :

$\# q_1 0 \# 1 q_2 1 \# q_1 1 0 \# 1 q_A 0 \# q_A 0 \# q_A \#\#$  → MATCH!

$\# q_1 0 \# 1 q_2 1 \# q_1 1 0 \# 1 q_A 0 \# q_A 0 \# q_A \#\#$

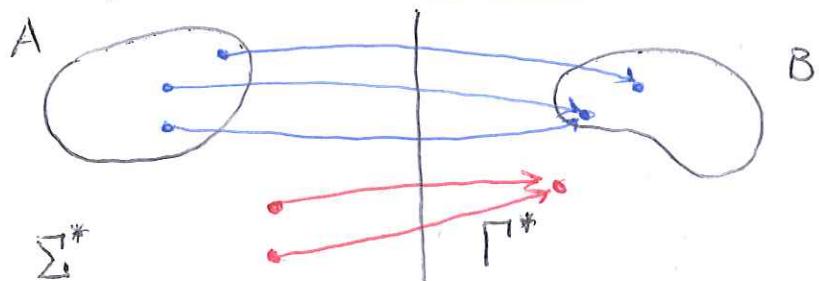
### • FUNZIONE CALCOLABILE:

Una funzione  $f: \Sigma^* \rightarrow \Gamma^*$  è una FUNZIONE CALCOLABILE se esiste qualche TM  $M$ ,  $\forall w \in \Sigma^*$ , termina con  $f(w)$  nel mestiere.

### • RIDUZIONE:

Un linguaggio  $A \subseteq \Sigma^*$  si dice "MAPPING REDUCIBLE" o "MANY-ONE REDUCIBLE" ad un linguaggio  $B \subseteq \Gamma^*$ , e si indica " $A \leq_m B$ ", se  $\exists f: \Sigma^* \rightarrow \Gamma^*$  tale che  $f$  sia una FUNZIONE CALCOLABILE e,  $\forall w \in \Sigma^*, [w \in A \Leftrightarrow f(w) \in B]$

La funzione  $f$  è chiamata "RIDUZIONE" da  $A$  a  $B$ .



### • TEOREMA 1: Se $A \leq_m B$ e $B$ è decidibile $\Rightarrow A$ è decidibile

Sia  $M_B$  il decisore di  $B$  e sia  $M_f$  la TM che calcola la riduzione  $f: \Sigma^* \rightarrow \Gamma^*$ .

Costruiamo il decisore  $M_A$  per  $A$ :

$M_A$  = "On input  $w$ :

1. Compute  $f(w)$  via  $M_f$ ;
2. Run  $M_B$  on input  $f(w)$ ;
3. If  $M_B(f(w))$  accepts, then accept; otherwise, reject."

Per il mapping dovuto alla riduzione,  $M_A$  decide a seconda di  $M_B$ ;  
quindi  $M_A$  decide  $A \Rightarrow A$  è decidibile!

OK!!

- COROLARIO: Se  $A \leq_m B$  e  $A$  è non decidibile  $\Rightarrow B$  è non decidibile

Se  $B$  fosse decidibile, si avrebbero le ipotesi del Teorema, per cui  $A$  dovrebbe essere decidibile. Ma  $A$  è non decidibile.  $\rightarrow \exists$  ASSURDO

$\Rightarrow B$  è non decidibile.

OK!!!

- ESEMPIO:  $A_{TM} \leq_m \text{Halt}_{TM}$

Esibiamo una TM  $F$  che calcola la RIDUZIONE:

$F =$  "On input  $\langle M, w \rangle$ :

1. If  $\langle M, w \rangle$  is not an encoding of a TM and a string, then output  $\langle E, w \rangle$ , where  $E$  is a TM that never halts, and halt;
2. Build the encoding  $\langle M', w \rangle$ , where  $M'$  is:

$M' =$  "On input  $w$ :

1. Run  $M$  on input  $w$ ;
2. If  $M(w)$  accepts, then halt;
3. If  $M(w)$  rejects, then enter in an endless loop."

3. Output  $\langle M', w \rangle$  and halt."

$$M'_w \text{ halts} \Leftrightarrow M(w) \text{ accepts} \rightarrow \langle M' \rangle \in \text{Halt}_{TM} \Leftrightarrow \langle M, w \rangle \in A_{TM}$$

Ma  $A_{TM}$  è non decidibile e  $A_{TM} \leq_m \text{Halt}_{TM} \Rightarrow \text{Halt}_{TM}$  è non decidibile!

$$(2) A_{TM} \leq_m MPCP \leq_m PCP$$

$$(3) E_{TR} \leq_m EQ_{TR}$$

- Homework: La funzione di riducibilità " $\leq_m$ " è TRANSITIVA.

Se  $A \leq_m B$  e  $B \leq_m C \Rightarrow$  Esistono riduzioni  $f$  e  $g$  tali che:

$$f: A \xrightarrow[w]{\quad} B \quad ; \quad g: B \xrightarrow{x} C \quad ; \quad \text{Costruiamo } h: A \xrightarrow[w]{\quad} C$$

$$\Rightarrow A \leq_m C. \quad \blacksquare \quad \text{OK!!!}$$

- \* NOTA: Adesso, per le (2), si ha:  $A_{TM} \leq_m PCP$  !

• **TEOREMA:** Se  $A \leq_w B$  e  $B$  è r.e.  $\Rightarrow A$  è r.e.

Analogo alle decidibilità, ma con TM's che non sono decisivi.

• **COROLLARIO:** Se  $A \leq_w B$  e  $A$  NON è r.e.  $\Rightarrow B$  NON è r.e.

Se  $B$  fosse r.e., per il Teorema  $A$  sarebbe r.e.  $\Rightarrow$  ASSURDO  $\rightarrow B$  NON è r.e. !

• **LEMMA:**  $A \leq_w B \iff A^c \leq_w B^c$

Ovvio! Dalle def. di RIDUZIONE.

• **Conseguenze:**

$A_{\text{TH}} \leq_w E_{\text{TH}}$ . Sia, per assurdo,  $E_{\text{TH}}$  deciso da una TM  $R$ .

$N =$  "On input  $\langle M, w \rangle$ :

1. Build a TM  $M'$  such that  $M'$  rejects any input  $x \neq w$ , and simulates  $M(w)$  if  $x = w$ ;
2. Run  $R$  on  $\langle M' \rangle$ ;
3. If  $R(\langle M' \rangle)$  accepts, then reject; oth. accept."

$N$  decide  $E_{\text{TH}} \rightarrow$  ASSURDO  $\Rightarrow A_{\text{TH}}$  è non decidibile.

\* **NOTA:** in realtà la riduzione è  $A_{\text{TH}} \leq_w E_{\text{TH}}^c$

(2)  $\left[ E_{\text{TH}} \text{ mon è r.e.}, E_{\text{TH}}^c \text{ mon è r.e.} \right]$

$A_{\text{TH}}^c \leq_w E_{\text{TH}}^c$  e  $A_{\text{TH}}^c \leq_w E_{\text{TH}}$ . Costruiamo TM  $F$  che calcola la riduzione:

$F =$  "On input  $\langle M, w \rangle$ :

1. Build a TM  $M_1$ :  $M_1 =$  "On any input: REJECT." (ACCEPT for  $E_{\text{TH}}^c$ );

2. Build a TM  $M_2$ :

$M_2 =$  "On any input: run  $M(w)$  and accept if  $M(w)$  accepts;

3. Output  $\langle M_1, M_2 \rangle$

$$A_{\text{TH}}^c \leq_w E_{\text{TH}} \quad \text{e} \quad A_{\text{TH}}^c \leq_w E_{\text{TH}}^c$$

↑                      ↑  
mot.r.e.            mot.r.e.  
                         ↑                      ↑  
                         mot.r.e.            mot.r.e.

## THE SELF TM

SELF è una TM che lascia sul nastro la descrizione di se stessa (<SELF>).

Supponiamo di avere il seguente algoritmo:

Print two copies of the following, but the second one in quotes  
 "Print two copies of the following, but the second one in quotes"

Questo è ciò che fa SELF: è la concatenazione di 2 TM A e B:

$$\text{SELF} = A \circ B$$

- A produce le copie tra virgolette;
- B fa stampare prime le copie senza virgolette e poi elmette.

• LEMMA:

Esiste una FUNZIONE CALCOLABILE  $q: \Sigma^* \rightarrow \Sigma^*$  tale che, se  $w \in \Sigma^*$ , allora  $q(w) = < P_w >$ , dove:

$P_w$  = "On any input:

1. Erase the tape;
2. Write  $w$  on tape;
3. Halt."

• COROLLARIO:

Esiste una TM Q tale che:

$Q =$  "On input  $w$ :

1. On the tape, replace  $w$  with  $< P_w >$ ;
2. Halt."

E' una conseguenza del fatto che la funzione  $q$  del Lemma sia una FUNZIONE CALCOLABILE, e quindi esiste una TM che la calcola.

•  $\text{SELF} = A \circ B$ , se  $A = < P_B >$  → stampa la codifica di B sul nastro

$B =$  "On input  $< M >$ , where M is a part of a TM:

1. Make a copy of  $< M >$ ;
2. Run Q on the leftmost copy of  $< M >$ ;
3. Link  $< P_{< M >} >$  and  $< M >$ , to get  $< P_{< M >} \circ M >$ ;
4. Halt."

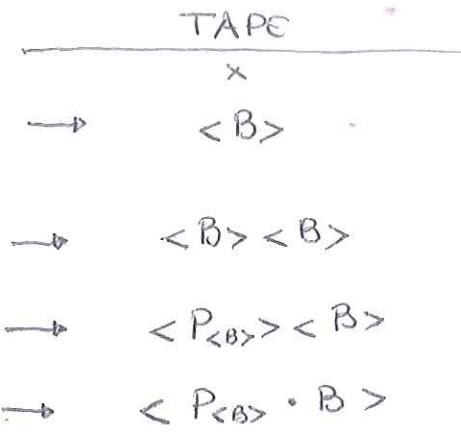
Eseguiamo  $\text{SELF}(x) = A \cdot B(x)$  :

$\rightarrow A = \langle P_B \rangle(x) \rightarrow$  cancella il master e scrive  $\langle B \rangle$

$\rightarrow B(\langle B \rangle) \rightarrow$  copie dell'Input

Run Q : replaces  $\langle B \rangle$  with  $\langle P_{\langle B \rangle} \rangle$

Link



Ma,  $\langle P_{\langle B \rangle} \rangle \cdot B = \langle A \cdot B \rangle = \langle \text{SELF} \rangle \rightsquigarrow \underline{\text{oh!!}}$

### TEOREMA DI RICORSIONE : RT

Sia T una TM che calcola una funzione  $t: (\sum^* \times \sum^*) \mapsto \sum^*$ .

Allora esiste una TM R, che calcola una funzione  $r: \sum^* \mapsto \sum^*$  tale che:  
 $\forall w \in \sum^*: r(w) = t(\langle R \rangle, w)$ .

R è la concatenazione di 3 TM's :  $R = A \cdot B \cdot T$ , dove A e B, come in SELF, servono per ottenere  $\langle R \rangle$  insieme a T.

Modifichiamo  $P_x$  vista per SELF, poiché non deve cancellare l'input:

$P'_x =$  "On input w:  
1. Move to the left of w;  
2. Print x to the left of w and halt."  $\Rightarrow Q'(x) = \langle P'_x \rangle$

•  $A = P'_{\langle B \cdot T \rangle}$

•  $B =$  "On input  $\langle M, w \rangle$ , where M is a fragment of a TM:

1. Copy  $\langle M \rangle$  to the left of the input;  
2. Run Q' on the leftmost copy of  $\langle M \rangle$ ;  
3. Link  $\langle P'_{\langle M \rangle} \rangle$  and  $\langle M \rangle$  to get  $\langle P'_{\langle M \rangle}, M \rangle$ ;  
4. Output  $\langle P'_{\langle M \rangle} \cdot M, w \rangle$  and halt."

Consideriamo  $R(w) = (A \cdot B \cdot T)(w)$ :

TAPE

w

$A = P'_{\langle B \cdot T \rangle} \rightarrow$  dalle codifiche  $\langle B \cdot T \rangle$   
senza cancellare w  $\rightarrow \langle B \cdot T \rangle, w$

$(B \cdot T)(\langle B \cdot T \rangle, w) \rightarrow$  copy  $\langle B \cdot T \rangle$   $\rightarrow \langle B \cdot T \rangle, \langle B \cdot T \rangle, w$   
Run Q  $\rightarrow \langle P'_{\langle B \cdot T \rangle} \rangle, \langle B \cdot T \rangle, w$   
Link  $\rightarrow \langle P'_{\langle B \cdot T \rangle} \cdot B \cdot T \rangle, w$

Ma,  $\langle P'_{\langle B \cdot T \rangle} \cdot B \cdot T \rangle, w = \langle A \cdot B \cdot T \rangle, w = \langle R \rangle, w = T(\langle R \rangle, w)$ .

$$\Rightarrow R(w) = T(\langle R \rangle, w)$$

oh!!

- ESEMPIO: Come ottenere SELF tramite il Teorema di Ricorsione:

$T =$  "On input  $\langle R, w \rangle$ :

1. Erase w;

2. Halt with output  $\langle R \rangle$ ;"

Per il Teorema di Ricorsione,  $\exists$  un  $R$  tale che:  $R(w) = T(\langle R \rangle, w)$

Ma,  $T(\langle R \rangle, w) = \langle R \rangle \Rightarrow R(w) = \langle R \rangle, \forall w \in \Sigma^* \Rightarrow R = \text{SELF}$  !

Quindi:  $\text{SELF} =$  "On any input:

1. Obtain, via RT, own description  $\langle \text{SELF} \rangle$  on tape;  
2. Halt."

- TEOREMA:  $[A_{TM}$  è non decidibile]

ALTERNATIVE PROOF: Per assurdo, esiste un decisore  $H$  per  $A_{TM}$ .

$B =$  "On input w:

1. Obtain, via RT, own description  $\langle B \rangle$ ;  
2. Run  $H$  on input  $\langle B, w \rangle$ ;  
3. If  $H(\langle B, w \rangle)$  accepts, then reject; oth. accept."

$B(w)$  accetta  $\Leftrightarrow H(\langle B, w \rangle)$  rifiuta  $\Leftrightarrow \langle B, w \rangle \notin A_{TM} \Leftrightarrow B(w)$  non accetta

$B(w)$  rifiuta  $\Leftrightarrow H(\langle B, w \rangle)$  accetta  $\Leftrightarrow \langle B, w \rangle \in A_{TM} \Leftrightarrow B(w)$  accetta

$\rightarrow$  ASSURDO  $\Rightarrow A_{TM}$  è non decidibile!

oh!!

- $M$  è una "MINIMAL TM" se  $\exists_{TM} M'$  tale che  $M$  è equivalente ad  $M'$  e  $|M'| < |M|$ .

• TEOREMA:  $\text{Min}_{TM} = \{ \langle M \rangle \mid M \text{ is a minimal TM} \}$  NON è r.e.

Per assurdo, sia  $\text{Min}_{TM}$  r.e.  $\Rightarrow \exists$  enumeratore  $E$  per  $\text{Min}_{TM}$

$C =$  "On input  $w$ :

1. Obtain, via RT, own description  $\langle C \rangle$ ;
2. Run  $E$  until a machine  $D$  appears with  $|\langle D \rangle| > |\langle C \rangle|$ ;
3. Simulate  $D$  on input  $w$ .

$C(w) = D(w) \Rightarrow C$  è equivalente a  $D$ , ma  $|\langle D \rangle| > |\langle C \rangle| \Rightarrow D$  non è minimale  $\Rightarrow D \notin \text{Min}_{TM} \rightarrow \exists$  ASSURDO  $\Rightarrow \text{Min}_{TM}$  non è r.e.!

• TEOREMA del PUNTO FISSO per funzioni calcolabili:

Sia  $t: \Sigma^* \mapsto \Sigma^*$  una funzione calcolabile. Allora, esiste una TM  $F$ , per la quale  $t(\langle F \rangle) = \langle G \rangle$ , dove  $G$  è una TM equivalente ad  $F$ .

• Assunzione: se una stringa non codifica correttamente una TM, allora si genera una codifica di una TM che rifiuta sempre.

$F =$  "On input  $w$ :

1. Obtain, via RT, own description  $\langle F \rangle$ ;
2. Compute  $t(\langle F \rangle) = \langle G \rangle$ ;
3. Run  $G$  on input  $w$  and output the result."

$F(w) = G(w) \Rightarrow F$  è equivalente a  $G$ !

$\Rightarrow \exists_{TM} G, \exists_{TM} F$  tali che  $t(\langle F \rangle) = \langle G \rangle$  e  $L(F) = L(G)$

$\rightarrow$  Il Teorema è DIMOSTRATO!



OK!!!

## ORACLE TM

Un ORACOLO per un linguaggio  $B$  è un dispositivo capace di determinare se una qualsiasi stringa  $w$  appartiene al linguaggio  $B$ . Se  $B$  è DECIDIBILE, l'oracolo può essere una TM.

Una ORACLE TM è una TM  $M^B$  con le capacità addizionali di poter interroghere un oracolo per un linguaggio  $B$ :

$$M^B := M \text{ is a TM with an ORACLE for language } B.$$

### • Esempio:

Una TM  $T$  con ORACOLO per  $A_{TM}$  ( $T^{A_{TM}}$ ) può DECIDERE  $E_{TM}$ !

$T^{A_{TM}} :=$  "On input  $\langle M \rangle$ :

1. Build the following TM  $N$ :

$N =$  "On any input:

1. Run  $M$  in parallel on all strings in  $\Sigma^*$ ;

2. If  $M$  accepts any of these strings, then accept."

2. Query the oracle to determine if  $\langle N, \emptyset \rangle \in A_{TM}$ ;

3. If the oracle answers NO, then accept;

if the oracle answers YES, then reject."

L'oracolo risponde YES  $\Leftrightarrow N \text{ accepts} \Leftrightarrow M(w) \text{ accepts on any } w \in \Sigma^* \Leftrightarrow L(M) \neq \emptyset$

$\Leftrightarrow M \notin E_{TM}$

$\Rightarrow T^{A_{TM}}$  decide  $E_{TM} \Rightarrow E_{TM}$  è "DECIDIBILE RELATIVAMENTE" ad  $A_{TM}$ .

### • TURING-REDUCIBLE LANGUAGE:

Un linguaggio  $A$  è "TURING-REDUCIBLE" ad un linguaggio  $B$ , se  $A$  è DECIDIBILE RELATIVAMENTE a  $B$ .

In tal caso, si indica con:

$$A \leq_T B$$

• TEOREMA: Se  $A \leq_T B$  e  $B$  è decidibile  $\Rightarrow A$  è decidibile.

Se  $A \leq_T B \Rightarrow \exists$  oracolo TM  $T^B$ ;

Se  $B$  è decidibile, un oracolo per  $B$  può essere un decisore  $T'$ .

Interrogando l'oracolo  $T'$ , che decide sempre facilmente, decide  $A$ .

$\Rightarrow A$  è decidibile!

\*  $A_{\text{TM}} \leq_T A'_{\text{TM}}$ !

• Homeworks:

1) Se  $A \leq_m B \Rightarrow A \leq_T B$

2) Se  $A \leq_T B, B \leq_T C \Rightarrow A \leq_T C$  (TRANSITIVITÀ)

3)  $A'_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is an oracle TM and } M^{A_{\text{TM}}} \text{ accepts } w \}$

$A'_{\text{TM}}$  non è TURING-REDUCIBLE to  $A_{\text{TM}}$ .

## MISURA QUANTITATIVA DELL' INFORMAZIONE

La quantità di informazione di una stringa è PROPORTIONALE alla LUNGHEZZA delle stringhe stesse, o meglio, della sua descrizione minimale.

Approccio basato sulla Teoria delle Computazioni → Complessità di Kolmogorov

Dato una stringa  $x \in \{0,1\}^*$ :

la complessità è  $|< M >|$ , dove  $M(x) = x$ .

Ma, questa descrizione non è conveniente, poiché ci vorrebbe  $\frac{1}{2}$  stato in  $M$  quasi per ogni bit di  $x$ .

• DESCRIZIONE di  $x \in \{0,1\}^*$ :

Una DESCRIZIONE è una codifica (ENCODING)  $< M, w >$ , dove  $M$  è una TM e  $w \in \{0,1\}^*$  è una stringa di input, tale che  $M(w) = x$ .

Stavolta, la maggior efficienza è dovuta alla presenza di  $w$ , che, per esempio, può essere il pattern "01" di  $x = (01)^n$ , oppure  $w = x$  se  $x$  è randomica.

Ma, come definiamo l'ENCODING?

Sia  $x \in \{0,1\}^* \Rightarrow <x> = x \rightarrow$  la codifica di  $x$  è  $x$  stessa

Siamo  $x, y \in \{0,1\}^* \Rightarrow <x,y> = \tilde{x}01y$ , con  $\tilde{x} = x$  raddoppiando tutti i bit,  
"01" è il MARCATORE DI FINE STRINGA

Dunque:  $|<x,y>| = 2|x| + |y| + 2 \rightarrow$  importante!

\* Una codifica più efficiente è la seguente:

$$<x,y> = |x|, x, y$$

Essendo la lunghezza di una stringa  $|x|$  rappresentabile con  $2\log_2 |x|$  e avendo bisogno comunque del terminatore di fine stringa tra  $|x|$  e  $x$ , si ha:

$$|<x,y>| = 2\log_2 |x| + |x| + |y| + 2$$

## • DESCRIZIONE MINIMALE!

$d(x) :=$  è la PIÙ PICCOLA stringa  $\langle M, w \rangle$  che sia una descrizione per  $x$ .  
 Se ci sono più stringhe delle stesse lunghezze,  $d(x)$  è la 1<sup>a</sup> in ORDINE LESSICOGRAFICO.

### COMPLESSITÀ DI KOLMOGOROV

Si definisce KOLMOGOROV COMPLEXITY di una stringa  $x$ , la lunghezza della sua descrizione minima  $d(x)$ :

$$K(x) := |d(x)|$$

• TEOREMA: Esiste una costante  $c$ , tale che,  $\forall x \in \{0,1\}^*$  si ha:

$$K(x) \leq |x| + c$$

Sia  $M$  la TM che termina con l'input sul nastro senza fare nullate.

Quindi:  $M(x) = x$ . Per la descrizione vista in precedenza:

$$K(x) = |d(x)| \leq |\langle M, x \rangle| \leq \underbrace{2|\langle M \rangle| + 2}_{c} + |x| = |x| + c$$

$$\Rightarrow K(x) \leq |x| + c$$

■

Q.D.!!!

\*NOTA: La Complessità di Kolmogorov non può essere maggiore delle lunghezze delle stringhe stesse, e meno di 1 costante  $c$ , che dipende dalla TM  $M$ !

• TEOREMA: Esiste una costante  $c$ , tale che  $\forall x \in \{0,1\}^*$ :

$$K(xx) \leq K(x) + c$$

$M =$  "On input  $\langle N, w \rangle$ :

1. Run  $N$  on input  $w$ , until it halts with output  $s$ ;
2. Make a copy of  $s$ ;
3. Halt."

$$K(xx) = |d(xx)| \leq |\langle M, d(x) \rangle| = |\langle M, \langle N, w \rangle \rangle| \leq 2 \underbrace{|\langle M \rangle| + 2}_{c} + |d(x)| = K(x) + c$$

$$\Rightarrow \underline{K(xx) \leq K(x) + c} \quad \text{ok!!!}$$

\* Nota: Ripetere 2 volte le stesse stringhe, non aumenta significativamente le quantità di informazioni ( $K(xx) \approx K(x)$ )!

- $K(x^m) \leq K(x) + c \cdot m$

- $K(x^m) \leq K(x) + \log_2(m^2) + c$

• TEOREMA: Esiste una costante  $c$ , tale che  $\forall x, y \in \{0,1\}^*$ :

$$K(x,y) \leq 2K(x) + K(y) + c$$

$M = \text{On input } \langle \langle N_1, w_1 \rangle, \langle N_2, w_2 \rangle \rangle:$

1. Run  $N_1$  on input  $w_1$ , and leave its output  $x$  on tape;
2. Run  $N_2$  on input  $w_2$ , and leave its output  $y$  to the right of  $x$ ;
3. Halt.

$$M(\langle d(x), d(y) \rangle) = xy$$

$$\Rightarrow K(xy) = |d(xy)| \leq |\langle M, \langle d(x), d(y) \rangle \rangle| \leq 2|\langle M \rangle| + 2 + 2|d(x)| + 2 + |d(y)| =$$

$$= 2K(x) + K(y) + c \quad \Rightarrow \underline{K(xy) \leq 2K(x) + K(y) + c}$$

ok!!!

• TEOREMA: Esiste una costante  $c$ , tale che  $\forall x, y \in \{0,1\}^*$ :

$$K(xy) \leq 2\log_2(K(x)) + K(x) + K(y) + c$$

Il risultato è basato su una codifica più efficiente delle stringhe.

$$K(xy) = |d(xy)| \leq |\langle M, \langle d(x), d(y) \rangle \rangle| \leq 2\log_2(k_M) + |\langle M \rangle| + 2 + |\langle d(x), d(y) \rangle| =$$

$$= c + 2\log_2(|d(x)| + |d(y)| + 2) = 2\log_2(K(x) + K(x) + K(y) + c)$$

ok!!!

ok!!!

### • TEOREMA:

Non esiste alcuna costante  $c$  tale che  $\forall x, y \in \{0,1\}^*$ :

$$K(xy) \leq K(x) + K(y) + c$$

Rappresenta un LIMITE INFERIORE per  $K(xy)$ !

### • DESCRIPTION LANGUAGE:

E' una FUNZIONE CALCOLABILE  $p: \Sigma^* \mapsto \Sigma^*$ , ma a cui associano un significato:  
se  $p \equiv \text{Python} \Rightarrow p(\text{"Python program"}) = \text{"program result"}$ .

Si definisce:  

- $d_p(x) :=$  la MINIMAL DESCRIPTION di  $x$  rispetto a  $p$ ; cioè le 1° stringhe  $s$  in ordine di stringhe tale che  $p(s) = x$ ;
- $K_p(x) := |d_p(x)|$ .

### • TEOREMA DI INVARIANZA:

Per ogni DESCRIPTION LANGUAGE  $p$ ,  $\exists$  costante  $c_p$  tale che,  $\forall x \in \{0,1\}^*$ :

$$K(x) \leq K_p(x) + c_p$$

$p$  è una funzione calcolabile  $\Rightarrow \exists \text{ TM } P$  che calcola  $p$

$M =$  "On input  $w$ :

- 1. Run  $P$  on input  $w$  and output  $p(w)$ ;
- 2. Halt."

$$K(x) = |d(x)| \leq |\langle M, d_p(x) \rangle| \leq 2|\langle M \rangle| + 2 + |d_p(x)| = K_p(x) + c_p$$

$$\Rightarrow K(x) \leq K_p(x) + c_p$$

OK!!!

\* NOTA: A seconda del linguaggio, delle codifiche che usiamo, non portiamo alcune variazioni alle quantità di informazioni; quindi, a meno di una costante, la Complessità di Kolmogorov è la stessa:

$$K(x) \approx K_p(x)$$

## • INCOMPRESSIBLE STRING :

- Una stringa si dice "c-compressibile" se  $\exists c > 0$  tale che:  $K(x) \leq |x| - c$
- Una stringa è "incompressible by c" se non è c-compressibile.
- Una stringa è "INCOMPRESSIBLE" se non è compressibile per nessun  $c$ , numero  $C=1$ .

## • TEOREMA:

$\forall m \in \mathbb{N}^+$ , esiste almeno 1 stringa INCOMPRESSIBILE di lunghezza  $m$ .

Sia  $m > 0$ ; Su  $\Sigma^* = \{0,1\}^*$  il #stringhe di lunghezza  $m = 2^m$ .

$$\text{Il numero di descrizioni di lunghezza minore di } m \text{ è } 1 + 2 + 2^2 + \dots + 2^{m-1} = \\ = \sum_{i=0}^{m-1} 2^i = \frac{1-2^m}{1-2} = 2^m - 1 < 2^m \Rightarrow \exists \text{ almeno una stringa di lunghezza } m \\ \text{che è INCOMPRESSIBILE!} \quad \blacksquare$$

• COROLARIO: Almeno  $(2^m - 2^{m-c+1} + 1)$  stringhe di lunghezza  $m$  sono incompressible by  $c$ .

• TEOREMA: La Complessità di Kolmogorov  $K: \{0,1\}^* \rightarrow \mathbb{N}$  è NON calcolabile.

Per assurdo, sia  $K$  calcolabile  $\Rightarrow \exists \text{ TM } KM$  che calcola  $K(x)$

Costruiamo un'altra TM  $M$ :

$M$  = "On input  $w$  (on any input):

1. Obtain, via RT, own description  $\langle M \rangle$ ;
2. Compute  $|\langle M \rangle|$ ;
3. Enumerate all strings  $x \in \{0,1\}^*$  in string order, and:
  4. Run  $KM$  on input  $x$ , to get  $K(x)$ ;
  5. If  $K(x) > |\langle M \rangle|$ , then output  $x$  and halt."

$$K(x) \leq |\langle M, \varepsilon \rangle| \leq |\langle M \rangle| ; \text{ Ma } x \text{ viene stampata solo se } K(x) > |\langle M \rangle|.$$

$\Rightarrow$  ASSURDO!  $\Rightarrow K$  non è calcolabile!!!



du!!

• TEOREMA:

Se  $\text{Halt}_{\text{TM}}$  fosse decidibile, allora  $K$  sarebbe calcolabile.

Inversamente, se  $K$  fosse calcolabile, allora  $\text{Halt}_{\text{TM}}$  sarebbe decidibile.

Si dimostra facendo riferimento alla MACHINE Busy-BEAVER (Tibor Radó, 1962).

• TEOREMA:

$I = \{x \in \{0,1\}^* \mid x \text{ is incompressible}\}$  è non decidibile.

Se  $I$  fosse decidibile  $\Rightarrow$  "Generate the lexicographically first incompressible string of length greater than this algorithm."

Se fosse vero, allora:  $|< M >| < |x|$ , dove  $x$  è la stringa incompressibile.

$\Rightarrow K(x) < |x| \leq |x| - c$ , con  $c \geq 1 \rightarrow \exists \underline{\text{ASSURDO}}!$ ,  $x$  è incompressibile

$\Rightarrow I$  è non decidibile!

■ OK!!!

## LOGICA DEL 1° ORDINE

È un INSIEME DI REGOLE per costruire affermazioni che sono SEMPRE VERE, e prescindere del contesto ("TAUTOLOGICAMENTE VERE").

- Tautologie proposizionali:

- "Principio del 3° escluso": o è vera A oppure non A  $\Rightarrow (A \vee \bar{A})$  sempre vera
- "Principio di consistenza":  $\text{not}(A \text{ and } \text{not}A) \Rightarrow \overline{(A \wedge \bar{A})}$  è sempre vera.

- Modus ponens: "Se A è valida e  $(A \text{ implica } B)$  è valida, allora B è valida":

$$(A \wedge (A \Rightarrow B)) \Rightarrow B$$

- Cambio di variabili

- Aggiunte di quantificatori

- Eliminazione di quantificatori: Per tutti gli x,  $A(x)$  è valido  $\Rightarrow A(\forall)$  è valido

- Manipolazione di quantificatori: se "not (for all x,  $A(x)$ )" è valida, allora "there exists x such that not  $A(x)$ " è valida.

- ASSIOMI DI PEANO per  $\mathbb{N}$ :

(1) ZERO EXISTS: esiste un elemento 0 tale che, per tutti gli x,  $S(x) \neq 0$ ;  
 $S(x)$  è la funzione SUCCESSORE.

(2) OGNI INTERO HA AL PIÙ 1 PREDECESSORE: for all x, if  $S(x) = S(y)$ ;  
then  $x = y$ .

$\Rightarrow S(x)$  è INIETTIVA!

- FORMULA Logica:

Una "FORMULA" è una buona formula stringa sull' alfabeto:

$$\{\wedge, \vee, \neg, (,), [, ], \forall, \exists, x, \underbrace{R_1, \dots, R_k}_{\text{relation symbols}}\}$$

\* "VARIABILI":  $x_1 = x$ ;  $x_2 = xx = y$ ;  $x_3 = xxx = z$ ; ...

### • FORMULA ATOMICA:

E'  $R_j(x_1 \dots x_k)$  ed è di ARITÀ  $k$ .

Una strinque  $\Phi$  è una FORMULA se:

- )  $\Phi$  è una formula atomica;
- )  $\Phi = \overline{\Phi}$
- )  $\phi = \phi_1 \wedge \phi_2$
- )  $\phi = \phi_1 \vee \phi_2$
- )  $\phi = \exists x [\Phi']$
- )  $\phi = \forall x [\Phi']$

\* Una VARIABILE senza quantificatori  $\exists, \forall$  è una VARIABILE LIBERA.

### • SENTENZA:

E' una FORMULA SENZA VARIABILI LIBERE !

$\forall x_1 [R_1(x_1, x_2) \wedge R_2(x_3)]$  non è una sentenza;  $x_2$  e  $x_3$  sono variabili libere

$\forall x_1 \exists x_2 \forall x_3 [R_1(x_1, x_2) \wedge R_2(x_3)]$  è una SENTENZA!

### • MODELLO:

Un UNIVERSO è l'insieme di tutti i valori assegnabili alle variabili.

Un MODELLO := un Universo unito ad un'assegnazione di specifiche relazioni valide sull'Universo ed un simbolo dell'Universo.

Per esempio:  $U = \mathbb{N}$ ;  $R_1 =$  "less than or equal to"  $\rightarrow \leq$

$\Rightarrow R_1 = \{(2,2); (2,3); (4,5); \dots\} ; (5,4) \notin R_1$ .

### • LINGUAGGIO DI UN MODELLO:

[E' l'insieme delle FORMULE che usano solo simboli di relazione assegnati del modello e le usano con le giuste ARITÀ (numero di variabili).]

\* Ogni SENTENZA di un modello può essere VERA o FALSA in quel modello.

Per esempio:  $\forall x, y [R_1(x, y) \vee R_1(y, x)]$

- se  $U = \mathbb{N}$  ed  $R_1 = \leq \rightarrow$  la sentenza è VERA
- se  $U = \mathbb{N}$  ed  $R_1 = < \rightarrow$  la sentenza è FALSA

## QUIZ:

Sie  $U = \mathbb{N}$ . Dine se le sentenze sono vere o false.

$$(1) \forall q \exists p \forall x,y [p > q \wedge (x,y > 1 \Rightarrow xy \neq p)] \rightarrow \text{VERA: INFINITÀ DEI NUMERI PRIMI} \\ (\text{Euclide, 300 a.C.})$$

$$(2) \forall a,b,c,m [(a,b,c > 0 \wedge m > 2) \Rightarrow a^m + b^m \neq c^m] \rightarrow \text{VERA: ULTIMO TEOREMA DI FERMAT}$$

$$(3) \forall q \exists p \forall x,y [p > q \wedge (x,y > 1 \Rightarrow xy \neq p \wedge xy \neq p+2)] \rightarrow ??? \text{ NON SI SA}$$

è la CONGETTURA DEI NUMERI PRIMI GENELLI

## • TEORIA DI UN MODELLO:

Dato un modello  $M$ , la TEORIA di  $M$  ( $\text{Th}(M)$ ) è l'insieme di tutte le SENTENZE nel linguaggio del modello che sono VERE in quel modello.

## • TEOREMA: $\boxed{\text{Th}(\mathbb{N}, +)}$ è DECIDIBILE

$\text{Th}(\mathbb{N}, +) \leq_m A_{\text{DFA}}$  (decidibile). Date una formula  $\phi$ , si costruisce un DFA che accetta  $E \Leftrightarrow \phi$  è una SENTENZA VERA.

## • TEOREMA: $\boxed{\text{Th}(\mathbb{N}, +, \times)}$ è NON DECIDIBILE

$A_{\text{TM}} \leq_m \text{Th}(\mathbb{N}, +, \times)$  e  $A_{\text{TM}}$  è non decidibile!

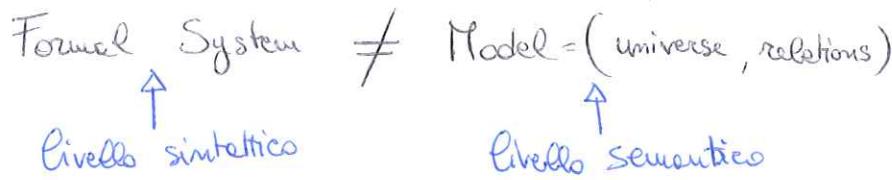
## • "DIMOSTRAZIONE FORMALE":

Una "DIMOSTRAZIONE FORMALE" di una sentenza  $\Psi$  è una sequenza di istruzioni  $s_1, \dots, s_n$ , dove  $s_k \equiv \Psi$  e ogni  $s_i$  è un ASSIOMA oppure può essere derivata dalle sentenze che le precedono nella sequenza, utilizzando le regole delle Logiche del Primo Ordine.

\* La correttezza di una dimostrazione formale è DECIDIBILE

## SISTEMA FORMALE

E' composto da ASSIOMI + regole LOGICA del 1° ORDINE



Un sistema formale puo' essere:

- (1) CONSISTENTE: se non c'è nessuna sentenza  $p$  tale che sia  $p$  che  $\neg p$  possono essere derivate dagli assiomi;
  - (2) COMPLETO: per ogni sentenza  $p$ ,  $p$  oppure  $\neg p$  può essere derivata;
  - (3) SOUND (ROBUSTO): Se ogni sentenza  $p$  derivabile dagli assiomi è VERA in QUALUNQUE modello del sistema formale.

\* NOTA : SOUNDNESS  $\Rightarrow$  CONSISTENCY  
CONSISTENCY  $\not\Rightarrow$  SOUNDNESS

## • TEOREMA DI CORPIEZZA DI GÖDEL:

Ogni sistema formale consistente (insieme di assiomi) ammette almeno il modello.

Per esempio, il sistema formale degli ASSIOMI di ZERMELO - FRANKLIN ammette il MODELLO INSIEMISTICO.

- **TEOREMA:** Nel sistema formale  $(\mathbb{N}, +, \times)$  l'insieme delle sentenze dimostrabili è z.e.

$P = "On \ input <\Psi>, \ where \ \Psi \ is \ a \ sentence:$

1. Generate all strings in string order;
  2. Discard any strings that doesn't encode a formal proof;
  3. If a formal proof for  $\Psi$  is found, then accept.

Principe  $\Psi \Rightarrow \{ \langle \Psi \rangle \mid \Psi \text{ is a demonstrable sentence} \}$  è r.e.

ok!!!

• TEOREMA: Alcune sentence in  $\text{Th}(\mathbb{N}, +, \times)$  sono non derivabili.

Per assurdo, siano tutte derivabili.

$D =$  "On input  $\Psi$ , where  $\Psi$  is a sentence:

1. Run in parallel algorithm  $P$  on  $\Psi$  and  $\bar{\Psi}$ ;
2. If  $P$  accepts  $\Psi$ , then accept;
3. If  $P$  accepts  $\bar{\Psi}$ , then reject."

$\Rightarrow D$  decide  $\text{Th}(\mathbb{N}, +, \times)$ ; ma  $\text{Th}(\mathbb{N}, +, \times)$  è non decabile  $\rightarrow$  ASSURDO!

$\Rightarrow$  alcune sentence sono non derivabili! □

• TEOREMA: In  $\text{Th}(\mathbb{N}, +, \times)$  non è possibile dimostrare una simile sentenza:  
 "Questa sentenza non può essere dimostrata"; tuttavia, è vera.

$A_{\text{TH}} \leq_m \text{Th}(\mathbb{N}, +, \times);$

$S =$  "On any input:

1. Obtain, via RT, own description  $\langle S \rangle$ ;
2. Build the sentence  $\exists x [\phi_{S,0}] = \Psi_0$ ;
3. Run algorithm  $P$  on  $\Psi_0$ ;
4. If  $P$  accepts  $\Psi_0$ , then accept."

$\Psi_0$  è vera  $\Leftrightarrow S$  non accetta 0;

• Se  $P$  trova una dimostrazione per  $\Psi_0 \Rightarrow S(0)$  accetta  $\Rightarrow \Psi_0$  è falsa  $\Rightarrow$   
 $\Rightarrow (\mathbb{N}, +, \times)$  è INCONSISTENTE.

• Se  $P$  non trova una dimostrazione per  $\Psi_0 \Rightarrow S(0)$  non accetta  $\Rightarrow \Psi_0$  è vera  
 $\rightarrow \Psi_0$  è vera, MA non esiste una dimostrazione! □

OK!!!

## • 1° TEOREMA DI INCOMPLETEZZA DI GöDEL

No sufficiently-powerful formal system can be both complete and sound.

E' del 1931. Successivamente, Rosser dimostra che addirittura un sistema formale non puo' essere sia completo e CONSISTENTE.

## • 2° TEOREMA DI INCOMPLETEZZA DI GöDEL

No sufficiently-powerful formal system that is CONSISTENT can prove its own consistency.