Sapienza Università di Roma

Matteo D'Agostino
Mirko Magistrado Dacara

Sapienza Università di Roma
a.a. 2022/23

INDECIDIBILITÀ DELL'IMPLICAZIONE LOGICA IN LOGICA DEL PRIMO ORDINE

Indice

Parte I

- Excursus storico
- L' *Entscheidungsproblem*
- Macchine di Turing (MdT)
- Formalizzazione di una generica MdT in FOL: dal modello computazionale alla codifica in formule

Parte II

- Soddisfacibilità in logica proposizionale e FOL
- Dal *theorem proving* alla soddisfacibilità
- *Halting problem*
- Dimostrazione dell'indecidibilità dell'implicazione logica al primo ordine tramite riduzione

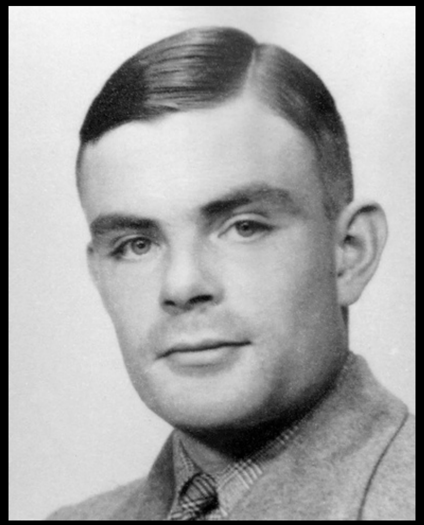
CONTESTO STORICO

LEIBNITZ (1670)

- "Calculemus!" leibniziano.
- La necessità di una "caratteristica universale"

HILBERT (1900)

- I "problemi di Hilbert"
- Entscheidungsproblem: il principale problema della logica matematica



TURING, CHURCH(1936)

- Macchine di Turing e λ-calcolo
- Un sogno infranto con relative limitazioni

IL PROBLEMA...

Sia S un sistema logico e sia ϕ una sua formula. Esiste un algoritmo in grado di determinare se ϕ è un teorema di S?

- **Teoremi e dimostrazioni in Formal Proof Systems**
- **Inferenza (sintassi) vs Implicazione logica (semantica)**
- **Soundness e Completeness per "Sistemi di dimostrazione"**
- **Teorema di completezza di Godel (1929)**

Una formula FOL è logicamente valida se e solo se è conclusione di una deduzione formale
(provability = validity)

... RIFORMULATO

DOMANDA

Esiste un algoritmo in grado di determinare se una formula FOL è o meno valida (o anche insoddisfacibile per Refutation Principle)?

... RIFORMULATO

DOMANDA

Esiste un algoritmo in grado di determinare se una formula FOL è o meno valida (o anche insoddisfacibile per Refutation Principle)?

↓

RISPOSTA

No. Sarà pertanto scopo delle successive slides dimostrare che l’insieme delle FOL sentences soddisfacibili non è decidibile, dunque la sua funzione caratteristica non calcolabile

MACCHINA DI TURING

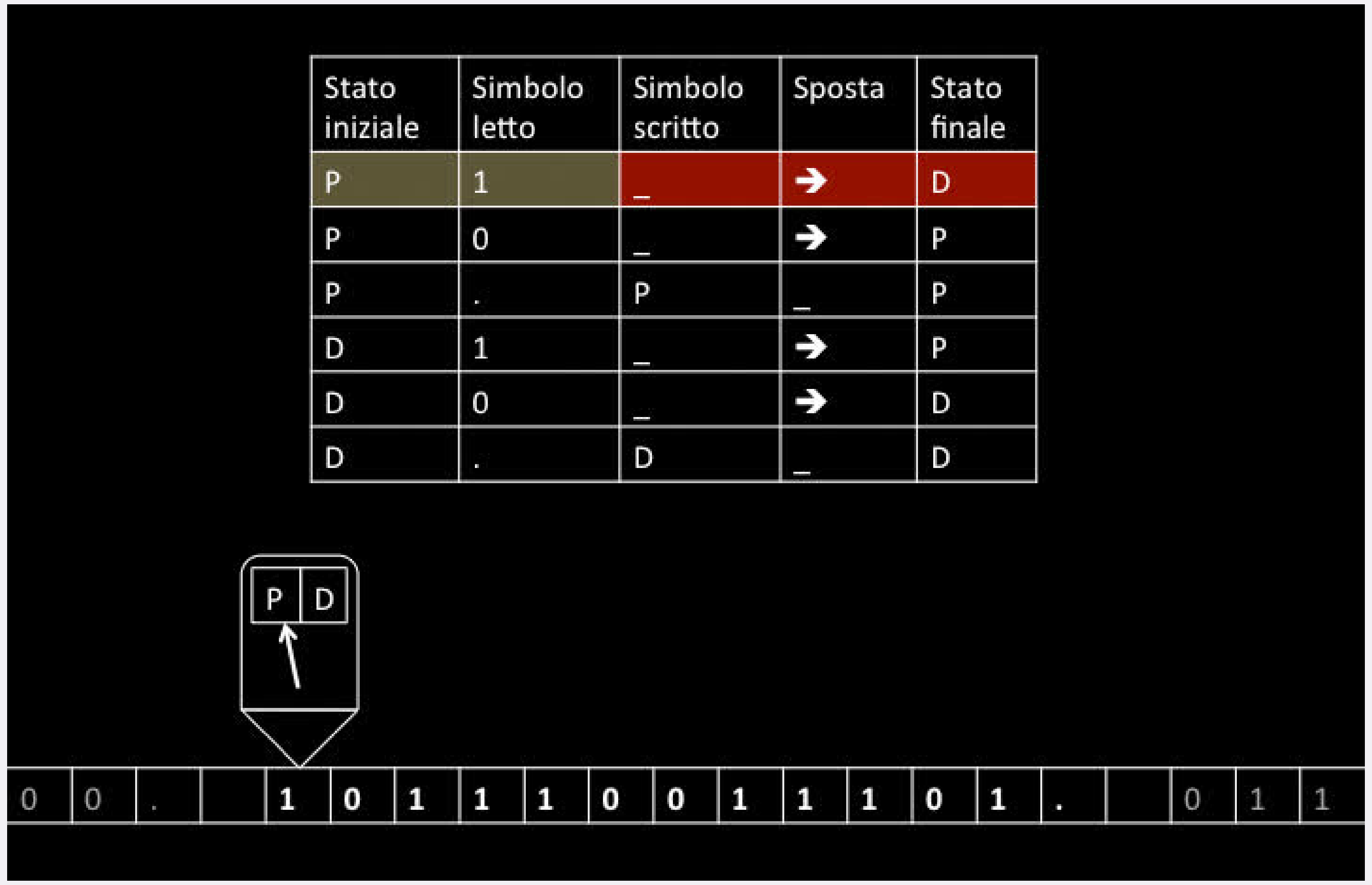
$$(\mathbf{Q}, \boldsymbol{\Sigma}, \delta, \mathbf{q_0}, \mathbf{q_1})$$

- $\mathbf{Q}$ é l'insieme finito degli stati, $\mathbf{q_0} \in Q$ stato finale e $\mathbf{q_1} \in Q$ stato iniziale.
- $\boldsymbol{\Sigma}$ é un insieme finito di simboli denominato *alfabeto*.
- δ é la cosiddetta ***funzione** di transizione* definita nel modo seguente:
  $\delta \; : \; \mathbf{Q} \setminus \{\mathbf{q_0}\} \times \boldsymbol{\Sigma} \to \mathbf{Q} \times \boldsymbol{\Sigma} \times \{\mathbf{L}, \mathbf{C}, \mathbf{R}\}$

MACCHINA DI TURING

$$(\mathbf{Q}, \boldsymbol{\Sigma}, \delta, \mathbf{q_0}, \mathbf{q_1})$$

- $\mathbf{Q}$ é l'insieme finito degli stati, $\mathbf{q_0} \in Q$ stato finale e $\mathbf{q_1} \in Q$ stato iniziale.
- $\boldsymbol{\Sigma}$ é un insieme finito di simboli denominato *alfabeto*.
- δ é la cosiddetta ***funzione** di transizione* definita nel modo seguente:
  $\delta \;:\; \mathbf{Q} \setminus \{\mathbf{q_0}\} \times \boldsymbol{\Sigma} \rightarrow \mathbf{Q} \times \boldsymbol{\Sigma} \times \{\mathbf{L}, \mathbf{C}, \mathbf{R}\}$



Stato iniziale	Simbolo letto	Simbolo scritto	Sposta	Stato finale
P	1	_	➔	D
P	0	_	➔	P
P	.	P	_	P
D	1	_	➔	P
D	0	_	➔	D
D	.	D	_	D

CONFIGURAZIONE

Una configurazione (*complete configuration*) di una MdT é una tripla ordinata $(\mathbf{x}, \mathbf{q}, \mathbf{k}) \in \mathbf{\Sigma}^* \times \mathbf{Q} \times \mathbf{N}$. Ad ogni tempo $\mathbf{t} \in \mathbf{N}$ la macchina di Turing risulterá essere in una ed una sola configurazione.

CONFIGURAZIONE

Una configurazione (*complete configuration*) di una MdT é una tripla ordinata $(\mathbf{x}, \mathbf{q}, \mathbf{k}) \in \mathbf{\Sigma}^* \times \mathbf{Q} \times \mathbf{N}$. Ad ogni tempo $\mathbf{t} \in \mathbf{N}$ la macchina di Turing risulterá essere in una ed una sola configurazione.

Come è possibile, dunque, andare a formalizzare tale concetto nella Logica di primo ordine dato che quest'ultima non modellizza by design il tempo discreto (come accade invece nella Logica Temporale Lineare) ?

CONFIGURAZIONE

Una configurazione (*complete configuration*) di una MdT é una tripla ordinata $(\mathbf{x}, \mathbf{q}, \mathbf{k}) \in \mathbf{\Sigma}^* \times \mathbf{Q} \times \mathbf{N}$. Ad ogni tempo $\mathbf{t} \in \mathbf{N}$ la macchina di Turing risulterá essere in una ed una sola configurazione.

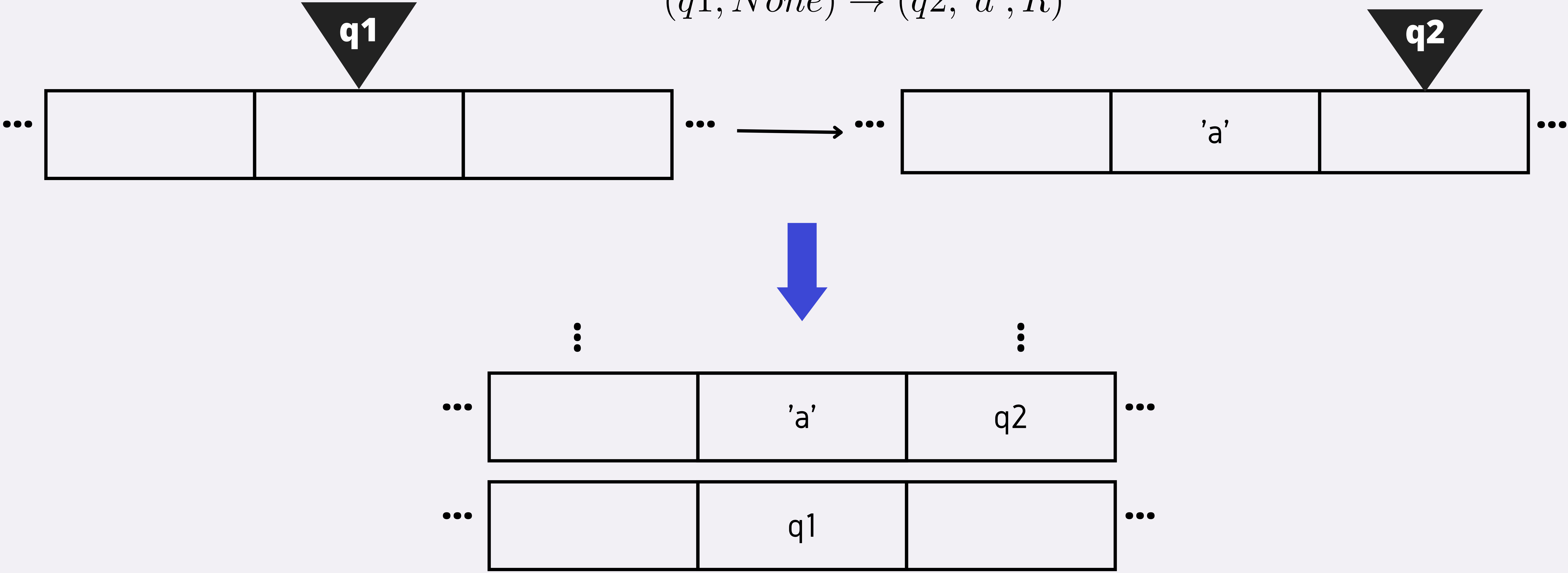
Come è possibile, dunque, andare a formalizzare tale concetto nella Logica di primo ordine dato che quest'ultima non modellizza by design il tempo discreto (come accade invece nella Logica Temporale Lineare) ?

Considerando l'esecuzione di una MdT come una sequenza di configurazioni in pila tale che per ogni coppia di configurazioni adiacenti esiste la corrispondente transizione nella definizione della MdT.

CONFIGURAZIONE

Se ad esempio avessimo la seguente regola di transizione in δ:

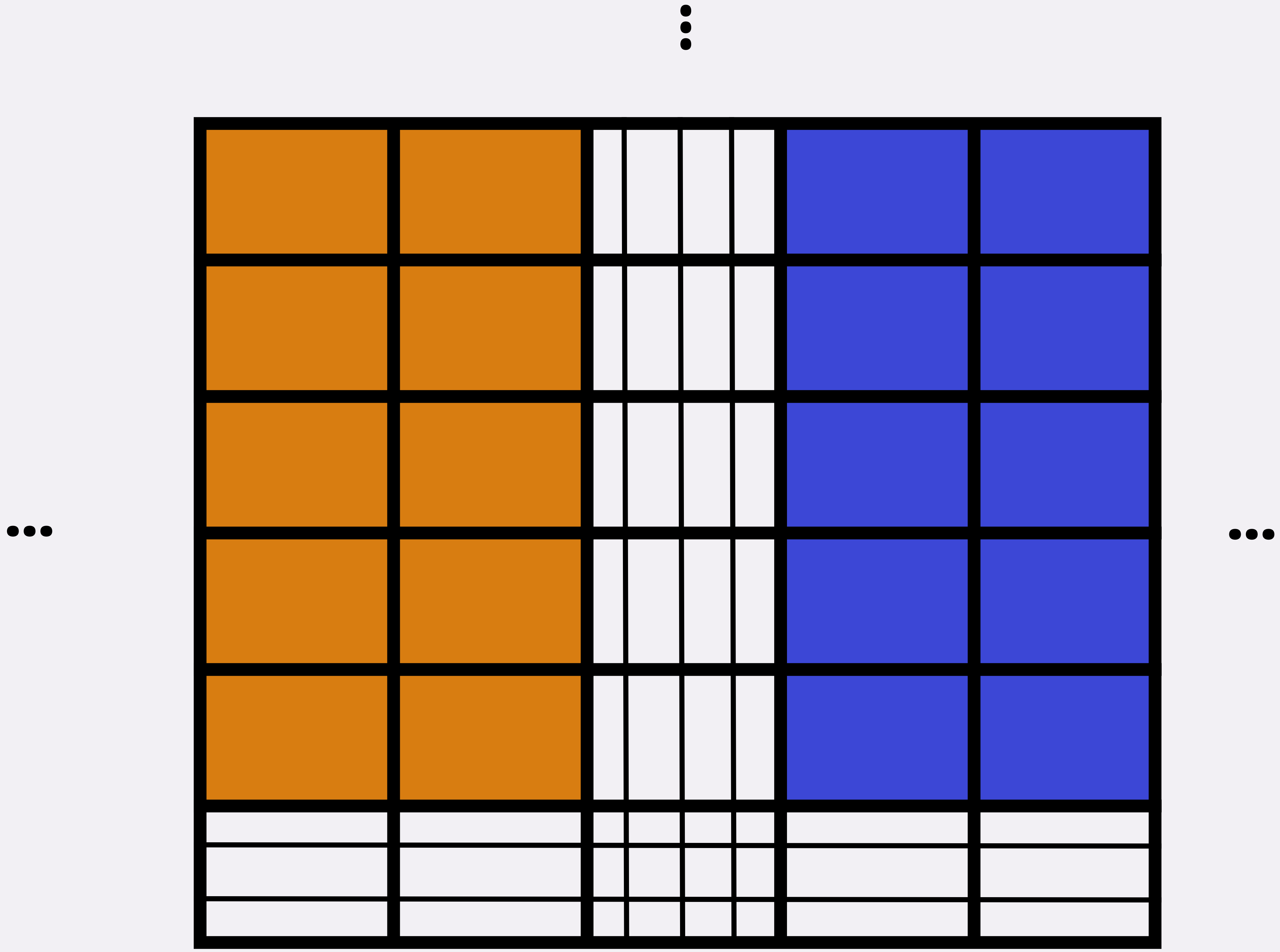
$$(q1, None) \rightarrow (q2, 'a', R)$$



DALLA MDT AL LINGUAGGIO FOL

Costanti	Funzioni/arità	Predicati/arità
origin	north/1 east/1 west/1	bottom/1 y-axis/1 left-side/1 right-side/1 head/1 S0, … , Sj / 1 Q0, … , Qk / 1

INTERPRETAZIONE DEL MODELLO



$\{x : \text{bottom}(x)\}$

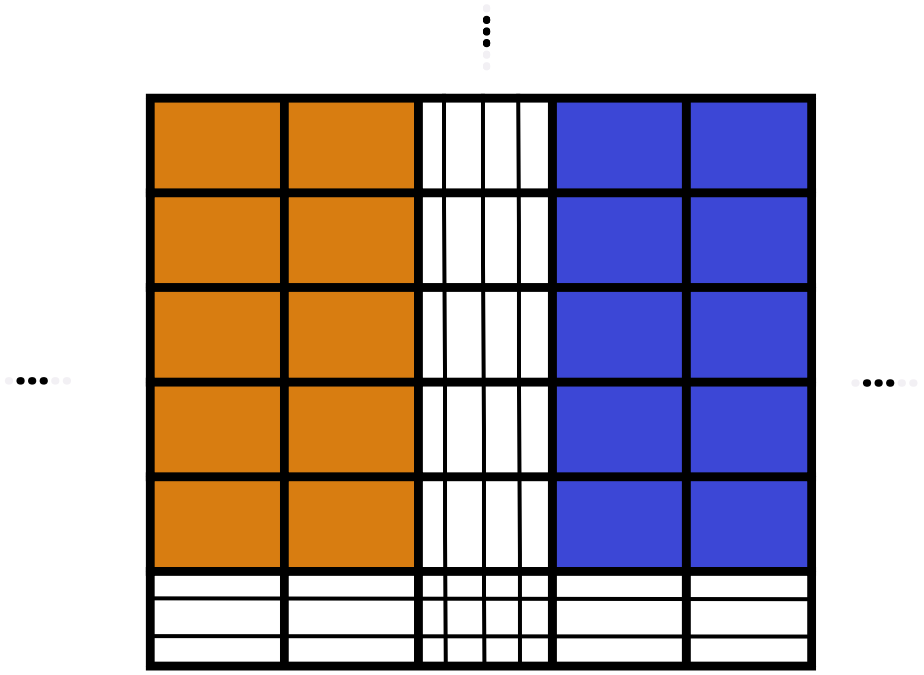
$\{x : \text{y-axis}(x)\}$

$\{x : \text{left-side}(x)\}$

$\{x : \text{right-side}(x)\}$

origin

FORMULE PER IL NASTRO



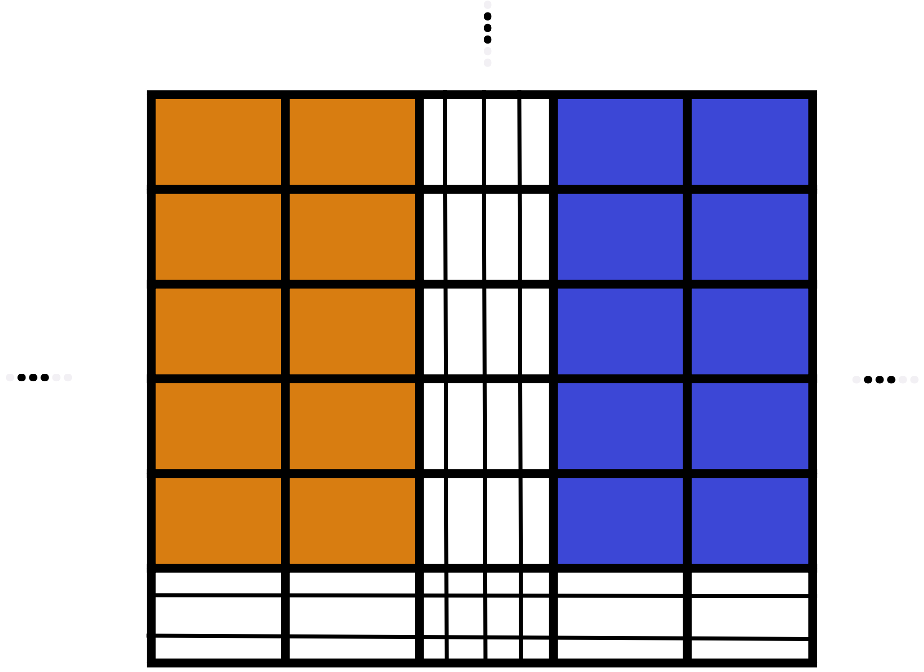
$$\mathsf{bottom}(\mathsf{origin})$$

$$(\forall x)(\mathsf{bottom}(x) \leftrightarrow ((\mathsf{origin} = x) \vee (\exists y)(\mathsf{bottom}(y) \wedge \mathsf{east}(y) = x))$$

$$(\forall x)(\mathsf{bottom}(x) \leftrightarrow ((\mathsf{origin} = x) \vee (\exists y)(\mathsf{bottom}(y) \wedge \mathsf{west}(y) = x))$$

$$(\forall x, y)(\mathsf{north}(x) = \mathsf{north}(y) \rightarrow x = y)$$

FORMULE PER IL NASTRO



$(\forall x)\neg\mathsf{bottom}(\mathsf{north}(x))$

$(\forall x)(\mathsf{north}(\mathsf{east}(x)) = \mathsf{east}(\mathsf{north}(x)))$

$(\forall x)((\mathsf{west}(\mathsf{east}(x)) = x) \wedge (x = \mathsf{east}(\mathsf{west}(x))))$

$(\forall x)(\mathsf{left\text{-}side}(x) \vee \mathsf{right\text{-}side}(x) \vee \mathsf{y\text{-}axis}(x))$

$\mathsf{right\text{-}side}(\mathsf{east}(\mathsf{origin})) \wedge \mathsf{left\text{-}side}(\mathsf{west}(\mathsf{origin})) \wedge \neg\mathsf{right\text{-}side}(\mathsf{origin}) \wedge \neg\mathsf{left\text{-}side}(\mathsf{origin})$

$(\forall x)(\mathsf{y\text{-}axis}(x) \rightarrow \neg(\mathsf{left\text{-}side}(x) \vee \mathsf{right\text{-}side}(x)))$

$(\forall x)(\mathsf{left\text{-}side}(x) \rightarrow \mathsf{left\text{-}side}(\mathsf{west}(x)))$

$(\forall x)(\mathsf{right\text{-}side}(x) \rightarrow \mathsf{right\text{-}side}(\mathsf{east}(x)))$

FORMULE PER UNA MDT QUALUNQUE

$$(\forall x) \bigvee_{i=0}^{j} \left(S_i(x) \wedge \bigwedge_{k \neq i} \neg S_k(x) \right)$$

$$(\forall x)(\mathsf{bottom}(x) \rightarrow (S_0(x) \wedge (x = \mathsf{origin} \leftrightarrow \mathsf{head}(x)))) \wedge Q_1(\mathsf{origin}) \wedge \bigwedge_{k \neq 1} \neg Q_k(\mathsf{origin})$$

$$(\forall x)\ \neg \mathbf{head}(x) \rightarrow (\bigvee_{i=0}^{j} (S_i(x) \leftrightarrow S_i(\mathbf{north}(x))))$$

FORMULE PER UNA MDT SPECIFICA

$$\boxed{(s_j, q_i) \Rightarrow (s_b, L, q_d)}$$

$$(\forall x)((\mathsf{head}(x) \land Q_i(x) \land S_j(x)) \rightarrow$$
$$(\mathsf{head}(\mathsf{north}(\mathsf{west}(x))) \land Q_d(\mathsf{north}(\mathsf{west}(x))) \land S_b(\mathsf{north}(x))))$$

$$\boxed{(s_j, q_i) \Rightarrow (s_b, C, q_d)}$$

$$(\forall x)((\mathsf{head}(x) \land Q_i(x) \land S_j(x)) \rightarrow (\mathsf{head}(\mathsf{north}(x)) \land Q_d(\mathsf{north}(x)) \land S_b(\mathsf{north}(x))))$$

$$\boxed{(s_j, q_i) \Rightarrow (s_b, R, q_d)}$$

$$(\forall x)((\mathsf{head}(x) \land Q_i(x) \land S_j(x)) \rightarrow$$
$$(\mathsf{head}(\mathsf{north}(\mathsf{east}(x))) \land Q_d(\mathsf{north}(\mathsf{east}(x))) \land S_b(\mathsf{north}(x))))$$

FORMULE PER UNA MDT SPECIFICA

$$
\begin{aligned}
&(\forall x)(\mathsf{head}(x) \land Q_i(x) \land S_j(x)) \rightarrow \\
&\Big[(x = \mathsf{origin}) \lor \\
&(\exists y)\Big((\mathsf{north}(\mathsf{east}(y)) = x) \land \bigvee_{(Q_a,S_b)\Rightarrow(Q_i,R,S_j)} ((\mathsf{head}(y) \land Q_a(y) \land S_b(y))) \\
&\quad \lor ((\mathsf{north}(y) = x) \land \bigvee_{(Q_a,S_b)\Rightarrow(Q_i,C,S_j)} (\mathsf{head}(y) \land Q_a(y) \land S_b(y))) \\
&\quad \lor ((\mathsf{north}(\mathsf{west}(y)) = x) \land \bigvee_{(Q_a,S_b)\Rightarrow(Q_i,L,S_j)} (\mathsf{head}(y) \land Q_a(y) \land S_b(y))) \Big)\Big]
\end{aligned}
$$

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

In **logica proposizionale** sì:

- **TABELLE DI VERITÀ**
- **TABLEAUX**
- **DPLL**

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

In **logica proposizionale** sì:

- **TABELLE DI VERITÀ**

  Naive, brute force

- **TABLEAUX**

  Brute force, migliore delle tabelle di verità

- **DPLL**

  Il migliore nella pratica

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

In **logica proposizionale** sì:

- **TABELLE DI VERITÀ**

  Naive, brute force

- **TABLEAUX**

  Brute force, migliore delle tabelle di verità

$$2^n$$

- **DPLL**

  Il migliore nella pratica

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

In **logica del primo ordine**?

- 1890/1900 formalizzazione della teoria dei numeri naturali:

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

In **logica del primo ordine**?

- 1890/1900 formalizzazione della teoria dei numeri naturali:

1) È possibile usare la FOL per formalizzare la matematica?

SODDISFACIBILITÀ IN LOGICA PROPOSIZIONALE E FOL

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno?

In **logica del primo ordine**?

- 1890/1900 formalizzazione della teoria dei numeri naturali:

1) È possibile usare la FOL per formalizzare la matematica?

2) $\exists$ algoritmo per fare *theorem proving*? Ovvero,
   Data una teoria **N** e un teorema **T**, è possibile verificare:

$$N \models T \quad ?$$

DAL THEOREM PROVING ALLA SODDISFACIBILITÀ

Dalle proprietà dell'implicazione logica del primo ordine:

Sia Γ un insieme di formule chiuse, e A una formula chiusa,

vale la <u>REFUTATION PRINCIPLE</u> (o <u>PRINCIPLE OF PROOF BY CONTRADICTION</u>)

$\Gamma \models A$ se e solo se $\Gamma \cup \{\neg A\}$ insoddisfacibile

DAL THEOREM PROVING ALLA SODDISFACIBILITÀ

Dalle proprietà dell'implicazione logica del primo ordine:

Sia Γ un insieme di formule chiuse, e A una formula chiusa,

vale la REFUTATION PRINCIPLE (o PRINCIPLE OF PROOF BY CONTRADICTION)

$\Gamma \models A$ se e solo se $\Gamma \cup \{\neg A\}$ insoddisfacibile

Ma quindi, in **logica del primo ordine**,

$\exists$ algoritmo che, data una formula, verifica se è soddisfacibile o meno,

o equivalentemente, il problema dell'implicazione logica è decidibile?

DAL THEOREM PROVING ALLA SODDISFACIBILITÀ

NO

NON esiste alcun algoritmo tale che, PER OGNI formula ϕ in logica del primo ordine, è in grado di verificare se essa è soddisfacibile o meno

DAL THEOREM PROVING ALLA SODDISFACIBILITÀ

NO

<u>NON esiste</u> alcun algoritmo tale che, <u>PER OGNI formula ϕ</u> in logica del primo ordine, è in grado di verificare se essa è soddisfacibile o meno

TEOREMA

Il problema di verificare se una formula in FOL è soddisfacibile è INDECIDIBILE (in particolare SEMI-DECIDIBILE)

DAL THEOREM PROVING ALLA SODDISFACIBILITÀ

NO

NON esiste alcun algoritmo tale che, **PER OGNI formula ϕ** in logica del primo ordine, è in grado di verificare se essa è soddisfacibile o meno

TEOREMA

Il problema di verificare se una formula in FOL è soddisfacibile è INDECIDIBILE (in particolare SEMI-DECIDIBILE)

La dimostrazione procede tramite riduzione: si riduce il problema dell'halting problem all'indecidibilità della FOL

Cos'è una RIDUZIONE?

- Algoritmo per trasformare un problema **A** ad un problema **B**
- Usate per: prove di upper/lower bound, **indecidibilità**
- Intuitivamente, il problema A (noto) è riducibile ad un problema B (nuovo problema), se un algoritmo per risolvere B (se esiste) può essere usato per risolvere A, mappando le istanze sì/no di A nelle istanze sì/no di B

Notazione:

$$A \leq B$$

Cos'è una RIDUZIONE?

Notazione:

$$A \leq B$$

- Algoritmo per trasformare un problema **A** ad un problema **B**
- Usate per: prove di upper/lower bound, **indecidibilità**
- Intuitivamente, il problema A (noto) è riducibile ad un problema B (nuovo problema), se un algoritmo per risolvere B (se esiste) può essere usato per risolvere A, mappando le istanze sì/no di A nelle istanze sì/no di B

- Se A ≤ B, risolvere A non può essere più difficile* della risoluzione di B, perché una soluzione di B porta ad una soluzione di A.

- In termini della teoria della calcolabilità, con A ≤ B:
  - se B è decidibile, allora anche A è decidibile
  - se A è indecidibile, allora anche B è indecidibile.

Esempio di riduzione:

- moltiplicazione ≤ elevamento al quadrato

$$a \times b = \frac{\left((a+b)^2 - a^2 - b^2\right)}{2}$$

Supponendo di poter solamente sommare, sottrarre, elevare al quadrato e dividere per due, possiamo calcolare il prodotto tra due numeri con questa formula

HALTING PROBLEM - TEOREMA

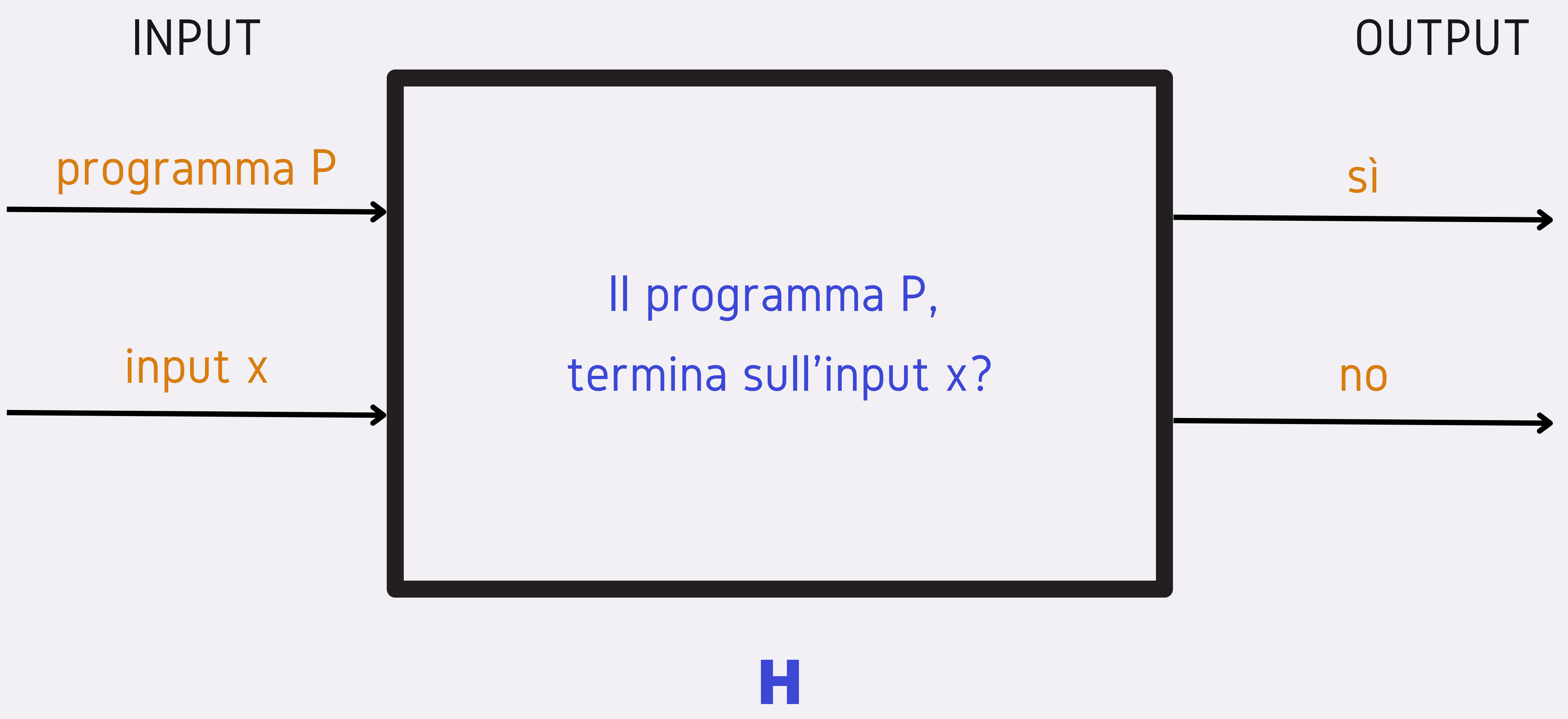
Siano dati un alfabeto $\mathbf{\Sigma}$ e una codifica che associa ad ogni macchina di Turing $\mathbf{M}$ una sua descrizione $c_M \in \Sigma^*$.
La funzione

$$h(c_M, x) = \begin{cases} 1 & \text{se } M \text{ termina su input } x \\ 0 & \text{se } M \text{ non termina su input } x \end{cases}$$

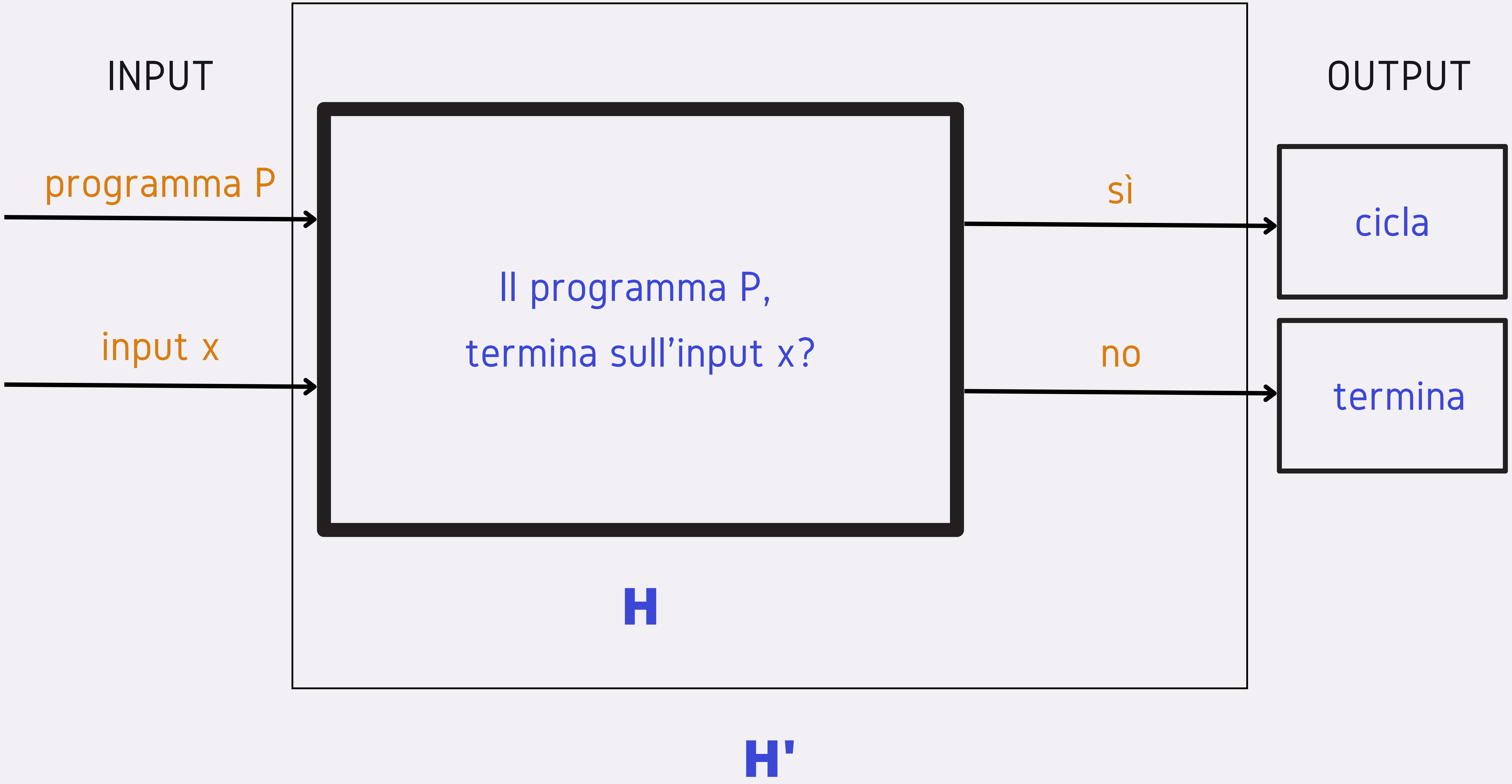
non é costruibile tramite una MdT

Ovvero, ∄ una macchina di Turing che, ∀ macchina di Turing M, ∀ input x,
può dire se M con input x, si ferma o meno

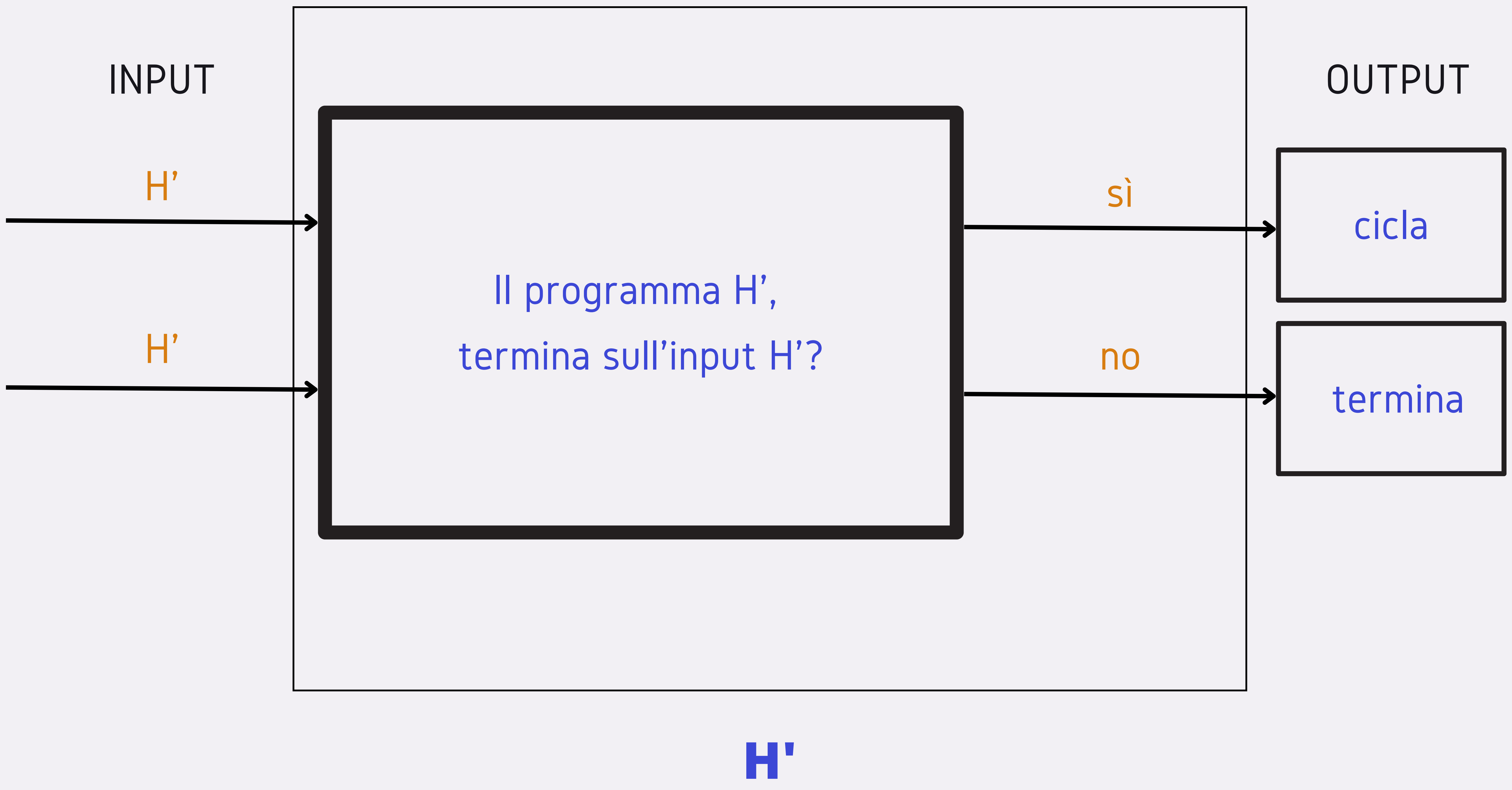
HALTING PROBLEM - DIMOSTRAZIONE



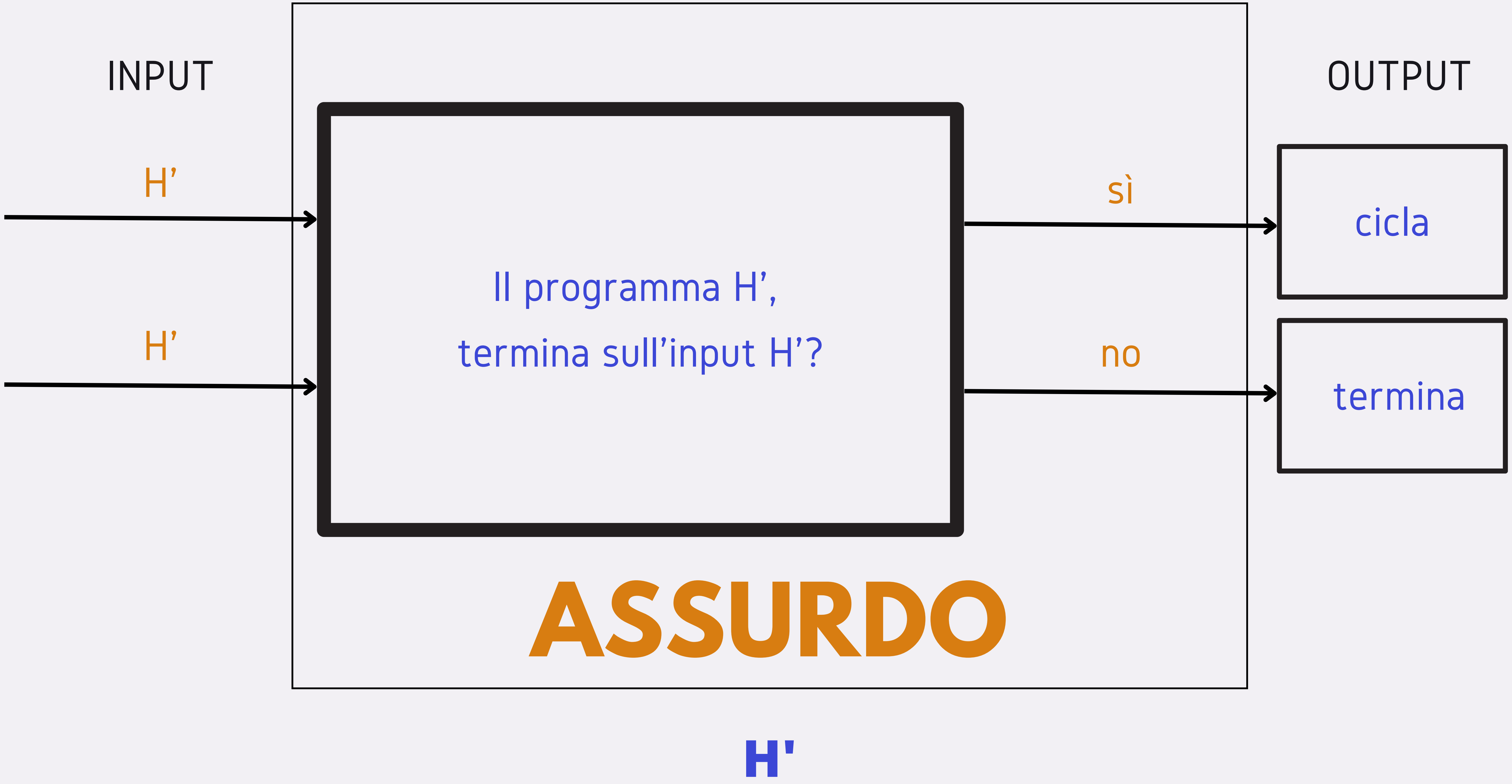
HALTING PROBLEM - DIMOSTRAZIONE



HALTING PROBLEM - DIMOSTRAZIONE



HALTING PROBLEM - DIMOSTRAZIONE



DAL THEOREM PROVING ALLA SODDISFACIBILITÀ

NO

NON esiste alcun algoritmo tale che, PER OGNI formula ϕ in logica del primo ordine, è in grado di verificare se essa è soddisfacibile o meno

TEOREMA

Il problema di verificare se una formula in FOL è soddisfacibile è INDECIDIBILE (in particolare SEMI-DECIDIBILE)

La dimostrazione procede tramite riduzione: si riduce il problema dell'halting problem all'indecidibilità della FOL

INDECIDIBILITÀ DELL'IMPLICAZIONE LOGICA AL PRIMO ORDINE - DIMOSTRAZIONE

Sfrutteremo la seguente riduzione:

Halting problem ≤ Indecidibilità dell'implicazione logica in FOL

Denotiamo con ϕ la congiunzione delle formule per la formalizzazione di una macchina di Turing M, che ha un numero finito di transizioni, dunque è finita anche ϕ, quindi una formula in FOL, e con w il suo input.

INDECIDIBILITÀ DELL'IMPLICAZIONE LOGICA AL PRIMO ORDINE - DIMOSTRAZIONE

Sfrutteremo la seguente riduzione:

Halting problem ≤ Indecidibilità dell'implicazione logica in FOL

Denotiamo con ϕ la congiunzione delle formule per la formalizzazione di una macchina di Turing M, che ha un numero finito di transizioni, dunque è finita anche ϕ, quindi una formula in FOL, e con w il suo input.

La formula ϕ esprime l'esistenza di una sequenza di configurazioni di M che segue le sue regole di transizione, (secondo la sua funzione di transizione δ), corrisponde quindi a una computazione di M su w.

INDECIDIBILITÀ DELL'IMPLICAZIONE LOGICA AL PRIMO ORDINE - DIMOSTRAZIONE

Sfrutteremo la seguente riduzione:

Halting problem ≤ Indecidibilità dell'implicazione logica in FOL

Denotiamo con ϕ la congiunzione delle formule per la formalizzazione di una macchina di Turing M, che ha un numero finito di transizioni, dunque è finita anche ϕ, quindi una formula in FOL, e con w il suo input.

La formula ϕ esprime l'esistenza di una sequenza di configurazioni di M che segue le sue regole di transizione, (secondo la sua funzione di transizione δ), corrisponde quindi a una computazione di M su w.

Se M si ferma su w, allora la formula ϕ sarà verificata (soddisfacibile), poiché una computazione valida
Se M non si ferma su w, allora la formula ϕ sarà falsa (insoddisfacibile), per l'assenza di una computazione valida

INDECIDIBILITÀ DELL'IMPLICAZIONE LOGICA AL PRIMO ORDINE - DIMOSTRAZIONE

Sfrutteremo la seguente riduzione:

Halting problem ≤ Indecidibilità dell'implicazione logica in FOL

Denotiamo con ϕ la congiunzione delle formule per la formalizzazione di una macchina di Turing M, che ha un numero finito di transizioni, dunque è finita anche ϕ, quindi una formula in FOL, e con w il suo input.

La formula ϕ esprime l'esistenza di una sequenza di configurazioni di M che segue le sue regole di transizione, (secondo la sua funzione di transizione δ), corrisponde quindi a una computazione di M su w.

Se M si ferma su w, allora la formula ϕ sarà verificata (soddisfacibile), poiché una computazione valida
Se M non si ferma su w, allora la formula ϕ sarà falsa (insoddisfacibile), per l'assenza di una computazione valida

Dunque, utilizzando questo algoritmo per la soddisfacibilità di ϕ potremmo risolvere il problema della fermata, che è assurdo in quanto sappiamo essere indecidibile

BIBLIOGRAFIA

1. Catalani, L. (2018, April 23). Calculemus: Il sogno di Leibniz. Medium. Retrieved July 12, 2023, from https://medium.com/@luigicatalani/calculemus-il-sogno-di-leibniz-196b11a55766

2. Wikipedia contributors. (2023, July 11). Entscheidungsproblem. In Wikipedia, The Free Encyclopedia. Retrieved July 12, 2023, from https://en.wikipedia.org/wiki/Entscheidungsproblem

3. StackExchange User (2017, July 31). The Entscheidungsproblem and the notion of decidability in first-order logic. Mathematics Stack Exchange. Retrieved July 12, 2023, from https://math.stackexchange.com/questions/2346305/the-entscheidungsproblem-and-the-notion-of-decidability-in-first-order-logic

4. StackExchange User (2018, March 26). Semi-decidability of first-order logic. Mathematics Stack Exchange. Retrieved July 12, 2023, from https://math.stackexchange.com/questions/2680611/semi-decidability-of-first-order-logic

BIBLIOGRAFIA

5. StackExchange User (2015, June 8). How is first-order logic complete but not decidable? Philosophy Stack Exchange. Retrieved July 12, 2023, from https://philosophy.stackexchange.com/questions/15525/how-is-first-order-logic-complete-but-not-decidable

6. New Mexico State University. (n.d.). First Order Logic Undecidability. Retrieved July 12, 2023, from https://www.cs.nmsu.edu/historical-projects/Projects/FoLundecidability.pdf

7. Gries, D., & Schneider, F. B. (2002). First Order Logic. Stanford University. Retrieved July 12, 2023, from http://kilby.stanford.edu/~rvg/154/handouts/fol.html

8. F. D’Amore, G. Ausiello e G. Gambosi (2002). Linguaggi, modelli, complessità