

# Indecidibilità della logica predicativa diadica

Eugenio G. Omodeo



UNIVERSITÀ  
DEGLI STUDI DI TRIESTE

Dip. Matematica e Geoscienze — DMI



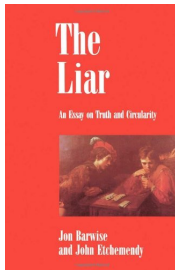
Trieste, 26.05.2021



# Il paradosso del mentitore ( Epimenide di Creta, 600 a.C. )

Lettera di San Paolo a Tito I, 12-13:

*"[11]A questi tali bisogna chiudere la bocca, perché mettono in scompiglio intere famiglie, insegnando per amore di un guadagno disonesto cose che non si devono insegnare. [12]Uno dei loro, proprio un loro profeta, già aveva detto: «I Cretesi son sempre bugiardi, male bestie, ventri pigri». [13]Questa testimonianza è vera. Perciò correggili con fermezza,"*



Questa frse contiene tre erori.

Questa frse contiene tre errori.

$$\{ x \mid x \notin x \}$$

Questa frase contiene tre errori.

$$\{x \mid x \notin x\} \overset{?}{\in} \{x \mid x \notin x\}$$

*“Il fenomeno di un programma che agisce sulla propria descrizione viene talvolta chiamato auto-referenza ed è fonte di molti risultati fondamentali in teoria della computabilità. L'intero punto della diagonalizzazione, [...], è in effetti quello di ottenere un'auto-referenza contraddittoria.”*

*[DSW94, pag. 98]*

*“Il fenomeno di un programma che agisce sulla propria descrizione viene talvolta chiamato auto-referenza ed è fonte di molti risultati fondamentali in teoria della computabilità. L'intero punto della diagonalizzazione, [...], è in effetti quello di ottenere un'auto-referenza contraddittoria.”*

*[DSW94, pag. 98]*





Visit the main page



WIKIPEDIA  
The Free Encyclopedia

- [Main page](#)
- [Contents](#)
- [Featured content](#)
- [Current events](#)
- [Random article](#)
- [Donate to Wikipedia](#)
- [Wikipedia store](#)

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

Article

[Talk](#)

Read

[Edit](#)

[View history](#)

Search Wikipedia

# *Entscheidungsproblem*


From Wikipedia, the free encyclopedia

In [mathematics](#) and [computer science](#), the ***Entscheidungsproblem*** (pronounced [ɛntˈʃaɪdʊŋsʁoːblɛːm], [German](#) for "decision problem") is a challenge posed by [David Hilbert](#) and [Wilhelm Ackermann](#) in 1928.<sup>[1]</sup> The problem asks for an [algorithm](#) that considers, as input, a statement and answers "Yes" or "No" according to whether the statement is *universally valid*, i.e., valid in every [structure](#) satisfying the axioms.

# Ipotesi di partenza

Supponiamo vi sia un algoritmo in grado di rispondere *sempre*, in modo *corretto*, al quesito

$$\mathcal{K} \stackrel{?}{\models} \exists y q(y, \vec{n}).$$


 finito

Dapprima

# Ipotesi di partenza

Supponiamo vi sia un algoritmo in grado di rispondere *sempre*, in modo *corretto*, al quesito

$$\mathcal{K} \stackrel{?}{\models} \exists y q(y, \vec{n}).$$

 finito

Dapprima

- 1) 'trasferiremo' questo assunto dalle clausole Horn al dominio  $\mathbb{N}$ ;

poi

# Ipotesi di partenza

Supponiamo vi sia un algoritmo in grado di rispondere *sempre*, in modo *corretto*, al quesito

$$\text{finito} \quad \mathcal{K} \stackrel{?}{\models} \exists y q(y, \vec{n}).$$

Dapprima

1) 'trasferiremo' questo assunto dalle clausole Horn al dominio  $\mathbb{N}$ ;

poi

2) ne trarremo una contraddizione, donde l'importante *risultato limitativo*:





## Teorema di Church, 1936

Non c'è algoritmo in grado di stabilire se valga o meno

$$\vdash \alpha$$

per ogni dato enunciato  $\alpha$  nel calcolo predicativo del 1<sup>o</sup> ordine.

# Teorema di Church e sua dimostrazione



## Teorema di Church, 1936

Non c'è algoritmo in grado di stabilire se valga o meno

$$\vdash \alpha$$

per ogni dato enunciato  $\alpha$  nel calcolo predicativo del 1<sup>o</sup> ordine.

### Dim.:

Una volta ottenuta la contraddiz. cercata, otterremo per il teor. di completezza e in base al principio di deduzione, l'indcidibilità del problema

$$\vdash \underbrace{(\wedge \mathcal{K}) \rightarrow \exists y q(y, \vec{n})}_{\alpha}.$$

(Per il teor. di Church, occorre vi siano relatori di arità almeno 2).

Chiariamo l'assunto iniziale:

“ *un algoritmo è in grado di dirci se  $\mathcal{K} \models \exists y q(y, \vec{n})$  o no* ”

Stiamo supponendo che:

- $\mathcal{K}$  stia per una base ( finita ) di clausole di Horn;

Chiariamo l'assunto iniziale:

“ *un algoritmo è in grado di dirci se  $\mathcal{K} \models \exists y q(y, \vec{n})$  o no* ”

Stiamo supponendo che:

- $\mathcal{K}$  stia per una base ( finita ) di clausole di Horn;
- $\neg q(Y, \vec{n})$  per una *meta* ( o 'goal' ) mono-letterale, ove



Chiariamo l'assunto iniziale:

“ *un algoritmo è in grado di dirci se  $\mathcal{K} \models \exists y q(y, \vec{n})$  o no* ”

Stiamo supponendo che:

- $\mathcal{K}$  stia per una base ( finita ) di clausole di Horn;
- $\neg q(Y, \vec{n})$  per una *meta* ( o 'goal' ) mono-letterale, ove
- $\vec{n}$  sta per una sequenza

$$\underline{n_1}, \dots, \underline{n_a},$$

con  $a \geq 1$ ,  $a+1$  arità di  $q$ , di NUMERALI

Chiariamo l'assunto iniziale:

“ un algoritmo è in grado di dirci se  $\mathcal{K} \models \exists y q(y, \vec{n})$  o no ”

Stiamo supponendo che:

- $\mathcal{K}$  stia per una base ( finita ) di clausole di Horn;
- $\neg q(Y, \vec{n})$  per una *meta* ( o 'goal' ) mono-letterale, ove
- $\vec{n}$  sta per una sequenza

$$\underline{n_1}, \dots, \underline{n_a},$$

con  $a \geq 1$ ,  $a+1$  arità di  $q$ , di NUMERALI intesi così:

$$\begin{aligned} \underline{0} &=_{\text{Def}} 0, \\ \underline{m+1} &=_{\text{Def}} s(\underline{m}) \end{aligned}$$

(per  $m$  che varia in  $\mathbb{N}$ );

Chiariamo l'assunto iniziale:

“ un algoritmo è in grado di dirci se  $\mathcal{K} \models \exists y q(y, \vec{n})$  o no ”

Stiamo supponendo che:

- $\mathcal{K}$  stia per una base ( finita ) di clausole di Horn;
- $\neg q(Y, \vec{n})$  per una *meta* ( o 'goal' ) mono-letterale, ove
- $\vec{n}$  sta per una sequenza

$$\underline{n_1}, \dots, \underline{n_a},$$

con  $a \geq 1$ ,  $a+1$  arità di  $q$ , di NUMERALI intesi così:

$$\begin{aligned} \underline{0} &=_{\text{Def}} 0, \\ \underline{m+1} &=_{\text{Def}} s(\underline{m}) \end{aligned}$$

(per  $m$  che varia in  $\mathbb{N}$ );

- $0_{/0}$  ed  $s_{/1}$  siano i soli costrutti di Herbrand.

# Osservazioni preliminari

Il fatto che  $\mathcal{K}$  sia di Horn ci garantisce l'esistenza del *modello minimo*; grazie a ciò,

Il fatto che  $\mathcal{K}$  sia di Horn ci garantisce l'esistenza del *modello minimo*; grazie a ciò,

$$\mathcal{K} \models \exists y q(y, \vec{n})$$

equivale all'esistenza di un  $y$  in  $\mathbb{N}$  tale che

$$\mathcal{K} \models q(\underline{y}, \underline{n}_1, \dots, \underline{n}_a).$$

Il fatto che  $\mathcal{K}$  sia di Horn ci garantisce l'esistenza del *modello minimo*; grazie a ciò,

$$\mathcal{K} \models \exists y q(y, \vec{n})$$

equivale all'esistenza di un  $y$  in  $\mathbb{N}$  tale che

$$\mathcal{K} \models q(\underline{y}, \underline{n}_1, \dots, \underline{n}_a).$$

Ai nostri fini basterà come

$\vec{n}$  una coppia di forma  $\underline{m}, \underline{m}$ ;

stiamo, in effetti, per ricorrere alla cosiddetta '**diagonalizzazione**'.



# 1) Funzioni computabili viste come numeri



Do per scontato che possiamo rappresentare, tramite un numero naturale, qualsiasi sequenza

$$g_0, \dots, g_M$$

certificante che  $g_M$  è funzione *ricorsiva generale*<sup>1</sup> ai sensi della definizione che ne abbiamo data in precedenza. In altre parole:

---

<sup>1</sup>∴ computabile parziale



Do per scontato che possiamo rappresentare, tramite un numero naturale, qualsiasi sequenza

$$g_0, \dots, g_M$$

certificante che  $g_M$  è funzione *ricorsiva generale*<sup>1</sup> ai sensi della definizione che ne abbiamo data in precedenza. In altre parole:

**1a)** Data una tal seq. di  $g_i$ , possiamo calcolare un numero che la rappresenti.

---

<sup>1</sup>∴ computabile parziale

Do per scontato che possiamo rappresentare, tramite un numero naturale, qualsiasi sequenza

$$g_0, \dots, g_M$$

certificante che  $g_M$  è funzione *ricorsiva generale*<sup>1</sup> ai sensi della definizione che ne abbiamo data in precedenza. In altre parole:

- 1a) Data una tal seq. di  $g_i$ , possiamo calcolare un numero che la rappresenti.
- 1b) Di converso, dato un  $m$  in  $\mathbb{N}$ , possiamo concretam. stabilire se  $m$  codifichi o meno una tal seq., della quale. . .

---

<sup>1</sup>∴ computabile parziale

Do per scontato che possiamo rappresentare, tramite un numero naturale, qualsiasi sequenza

$$g_0, \dots, g_M$$

certificante che  $g_M$  è funzione *ricorsiva generale*<sup>1</sup> ai sensi della definizione che ne abbiamo data in precedenza. In altre parole:

- 1a) Data una tal seq. di  $g_i$ , possiamo calcolare un numero che la rappresenti.
- 1b) Di converso, dato un  $m$  in  $\mathbb{N}$ , possiamo concretam. stabilire se  $m$  codifichi o meno una tal seq., della quale...
- 1c) ...in caso affermativo, possiamo individuare:
  - ▶ la lunghezza  $M + 1$ ,
  - ▶ ciascuna componente  $g_i$ ;

---

<sup>1</sup>∴ computabile parziale

1d) ... in caso affermativo ...

- ▶ siamo anche in grado di ricavare da  $m$  una base  $\mathcal{B}$  di clausole di Horn che *rappresenti* la funzione parziale<sup>2</sup>

$$x \mapsto g_M(x, 0, \dots, 0).$$

tramite un predicato diadico

$$p(Y, X),$$

---

<sup>2</sup>Ai nostri fini correnti, non serve considerare funzioni  $g_M$  di arità  $> 1$ : quando  $g_M$  è a più argomenti, la rimpiazziamo con la funzione qui indicata, ottenibile da essa per composiz. con funzioni iniziali.

1d) ... in caso affermativo ...

- ▶ siamo anche in grado di ricavare da  $m$  una base  $\mathcal{B}$  di clausole di Horn che *rappresenti* la funzione parziale<sup>2</sup>

$$\mathbf{x} \mapsto \mathbf{g}_M(\mathbf{x}, 0, \dots, 0).$$

tramite un predicato diadico

$$p(Y, X),$$

*nel senso che* per ogni  $\mathbf{x}$  in  $\mathbb{N}$  e ogni  $\mathbf{y}$  in  $\mathbb{N}$ :

$$\begin{array}{ll} \mathcal{B} \models \exists y p(y, \underline{\mathbf{x}}) & \text{sse } \mathbf{g}_M(\mathbf{x}, 0, \dots, 0) \text{ ha un valore,} \\ \mathcal{B} \models p(\underline{\mathbf{y}}, \underline{\mathbf{x}}) & \text{sse } \mathbf{g}_M(\mathbf{x}, 0, \dots, 0) = \mathbf{y}. \end{array}$$

---

<sup>2</sup>Ai nostri fini correnti, non serve considerare funzioni  $\mathbf{g}_M$  di arità  $> 1$ : quando  $\mathbf{g}_M$  è a piú argomenti, la rimpiazziamo con la funzione qui indicata, ottenibile da essa per composiz. con funzioni iniziali.



## 2) Contraddizione tramite il metodo diagonale

Il celebre **teorema di universalità** di Turing (1936), riferito a

- la nozione di ricorsione generale da noi qui adottata e
- al caso delle funzioni ricorsive generali *monadiche*,

ci dice che

Esistono una base  $\mathcal{U}$  di clausole di Horn e un relatore triadico  $u$  tali che per ogni coppia  $m, x$  di naturali tale che  $m$  codifichi una seq.

$$g_0, \dots, g_M$$

come sopra, le affermazioni

$$“ \mathcal{U} \models \exists y u(y, \underline{m}, \underline{x}) ”$$

e

$$“ g_M(x, 0, \dots, 0) \text{ ha un valore } ”$$

siano equivalenti una all'altra.



Esistono una base  $\mathcal{U}$  di clausole di Horn e un relatore triadico  $u$  tali che per ogni coppia  $m, x$  di naturali tale che  $m$  codifichi una seq.

$$g_0, \dots, g_M$$

come sopra, le affermazioni

$$“ \mathcal{U} \models \exists y u(y, \underline{m}, \underline{x}) ”$$


e


$$“ g_M(x, 0, \dots, 0) \text{ ha un valore } ”$$

siano equivalenti una all'altra.


COME?



 Osserviamo che il seguente procedimento è algoritmico e in corrispondenza a qualsiasi coppia  $m$ ,  $x$  fornitagli in ingresso o emette un responso (da intendersi come affermativo) oppure non dà risultato:

 Osserviamo che il seguente procedimento è algoritmico e in corrispondenza a qualsiasi coppia  $m$ ,  $x$  fornitagli in ingresso o emette un responso (da intendersi come affermativo) oppure non dà risultato:

0. estrai  $B$  e  $p/2$  dal dato  $m$  ( se l'estraz. è impossibile, cicla pure all'infinito ), come descritto sopra al punto **1d**);

 Osserviamo che il seguente procedimento è algoritmico e in corrispondenza a qualsiasi coppia  $m$ ,  $x$  fornitagli in ingresso o emette un responso (da intendersi come affermativo) oppure non dà risultato:

0. estrai  $B$  e  $p/2$  dal dato  $m$  ( se l'estraz. è impossibile, cicla pure all'infinito ), come descritto sopra al punto **1d**);
1. elenca le dimostrazioni che utilizzano  $B$  come insieme (finito) di premesse;

☞ Osserviamo che il seguente procedimento è algoritmico e in corrispondenza a qualsiasi coppia  $m$ ,  $x$  fornitagli in ingresso o emette un responso (da intendersi come affermativo) oppure non dà risultato:

0. estrai  $B$  e  $p/2$  dal dato  $m$  ( se l'estraz. è impossibile, cicla pure all'infinito ), come descritto sopra al punto **1d**);
1. elenca le dimostrazioni che utilizzano  $B$  come insieme (finito) di premesse;
2. per ciascuna delle dimostrazioni, via via che vengono elencate, controlla se l'ultimo passo della dimostrazione sia un enunciato della forma

$$p(\underline{y}, \underline{x});$$

☞ Osserviamo che il seguente procedimento è algoritmico e in corrispondenza a qualsiasi coppia  $m$ ,  $x$  fornitagli in ingresso o emette un responso (da intendersi come affermativo) oppure non dà risultato:

0. estrai  $B$  e  $p/2$  dal dato  $m$  ( se l'estraz. è impossibile, cicla pure all'infinito ), come descritto sopra al punto **1d**);
1. elenca le dimostrazioni che utilizzano  $B$  come insieme (finito) di premesse;
2. per ciascuna delle dimostrazioni, via via che vengono elencate, controlla se l'ultimo passo della dimostrazione sia un enunciato della forma

$$p(\underline{y}, \underline{x});$$

3. ed emetti il responso—fermandoti—se questo è il caso;

☞ Osserviamo che il seguente procedimento è algoritmico e in corrispondenza a qualsiasi coppia  $m$ ,  $x$  fornitagli in ingresso o emette un responso (da intendersi come affermativo) oppure non dà risultato:

0. estrai  $\mathcal{B}$  e  $p/2$  dal dato  $m$  ( se l'estraz. è impossibile, cicla pure all'infinito ), come descritto sopra al punto **1d**);
1. elenca le dimostrazioni che utilizzano  $\mathcal{B}$  come insieme (finito) di premesse;
2. per ciascuna delle dimostrazioni, via via che vengono elencate, controlla se l'ultimo passo della dimostrazione sia un enunciato della forma

$$p(\underline{y}, \underline{x});$$

3. ed emetti il responso—fermandoti—se questo è il caso;
4. altrimenti protra l'attività **1.**

☞ Visto che di un procedim. algoritmico qui si tratta, la funzione parziale  $u(m, x)$  ch'esso computa può essere certificata computabile tramite un'opportuna seq. di funzioni ricorsive generali,

└



☞ Visto che di un procedim. algoritmico qui si tratta, la funzione parziale  $u(m, x)$  ch'esso computa può essere certificata computabile tramite un'opportuna seq. di funzioni ricorsive generali, dalla quale sappiamo come ottenere la base voluta,  $\mathcal{U}$ , di clausole di Horn:

└

👉 Visto che di un procedim. algoritmico qui si tratta, la funzione parziale  $u(m, x)$  ch'esso computa può essere certificata computabile tramite un'opportuna seq. di funzioni ricorsive generali, dalla quale sappiamo come ottenere la base voluta,  $\mathcal{U}$ , di clausole di Horn: volendo evitare ogni ricorso alla tesi di Church-Turing, possiamo effettivam.—se sfidati—*implementare*  $\mathcal{U}$ .

⊣

☞ Visto che di un procedim. algoritmico qui si tratta, la funzione parziale  $u(m, x)$  ch'esso computa può essere certificata computabile tramite un'opportuna seq. di funzioni ricorsive generali, dalla quale sappiamo come ottenere la base voluta,  $\mathcal{U}$ , di clausole di Horn: volendo evitare ogni ricorso alla tesi di Church-Turing, possiamo effettivam.—se sfidati—*implementare*  $\mathcal{U}$ .  
( Ed, in effetti, il 'programma universale' è stato da innumerevoli autori specificato in completo dettaglio. ) →

Disponendo di un metodo effettivo in grado di dirci, per ogni coppia  $\mathbf{m}$ ,  $\mathbf{x}$ , se

$$\mathcal{U} \models \exists y u(y, \underline{\mathbf{m}}, \underline{\mathbf{x}})$$

oppure no—*questa la nostra ipotesi di partenza*—, possiamo servircene in particolare per stabilire se

$$\mathcal{U} \models \exists y u(y, \underline{\mathbf{m}}, \underline{\mathbf{m}})$$

o no;

Disponendo di un metodo effettivo in grado di dirci, per ogni coppia  $\underline{m}$ ,  $\underline{x}$ , se

$$\mathcal{U} \models \exists y u(y, \underline{m}, \underline{x})$$

oppure no—*questa la nostra ipotesi di partenza*—, possiamo servircene in particolare per stabilire se

$$\mathcal{U} \models \exists y u(y, \underline{m}, \underline{m})$$

o no; dunque la funzione

$$t(\underline{m}) = \begin{cases} 1 & \text{se } \mathcal{U} \models \exists y u(y, \underline{m}, \underline{m}) , \\ 0 & \text{altrimenti,} \end{cases}$$

è computabile totale;

Disponendo di un metodo effettivo in grado di dirci, per ogni coppia  $\mathbf{m}$ ,  $\mathbf{x}$ , se

$$\mathcal{U} \models \exists y u(y, \underline{\mathbf{m}}, \underline{\mathbf{x}})$$

oppure no—*questa la nostra ipotesi di partenza*—, possiamo servircene in particolare per stabilire se

$$\mathcal{U} \models \exists y u(y, \underline{\mathbf{m}}, \underline{\mathbf{m}})$$

o no; dunque la funzione

$$t(\mathbf{m}) = \begin{cases} 1 & \text{se } \mathcal{U} \models \exists y u(y, \underline{\mathbf{m}}, \underline{\mathbf{m}}) , \\ 0 & \text{altrimenti,} \end{cases}$$

è computabile totale; partendo da essa otteniamo la

$$d(\mathbf{m}) = \begin{cases} 0 & \text{se } t(\mathbf{m}) = 0, \\ \text{non ha val.} & \text{altrimenti,} \end{cases}$$

che sarà ricorsiva generale, in quanto

$$d(\mathbf{m}) = \min_{x \geq 0} \left( t(\text{proj}_1(\mathbf{m}, x)) = 0 \right) .$$

Sia ora  $k$  il numero che codifica una seq.

$$d_0, \dots, d_D$$

definiente quest'ultima  $d(-)$ . Ecco la contraddizione che cercavamo:

$$\begin{array}{lll}
 d(k) & \text{ha un valore} & \text{sse} \\
 t(k) & = 0 & \text{sse} \\
 \mathcal{U} & \neq \exists y u(y, \underline{k}, \underline{k}) & \text{sse} \\
 d_D(k) & \text{non ha un valore} & \text{sse} \\
 d(k) & \text{non ha un valore.} & \neg
 \end{array}$$

Alonzo Church ( 1903–1995 ),  
Alan Mathison Turing ( 1912–1954 )







Martin D. Davis, Ron Sigal, and Elaine J. Weyuker.  
*Computability, complexity, and languages - Fundamentals of theoretical computer science.*  
Computer Science ad scientific computing. Academic Press,  
1994.