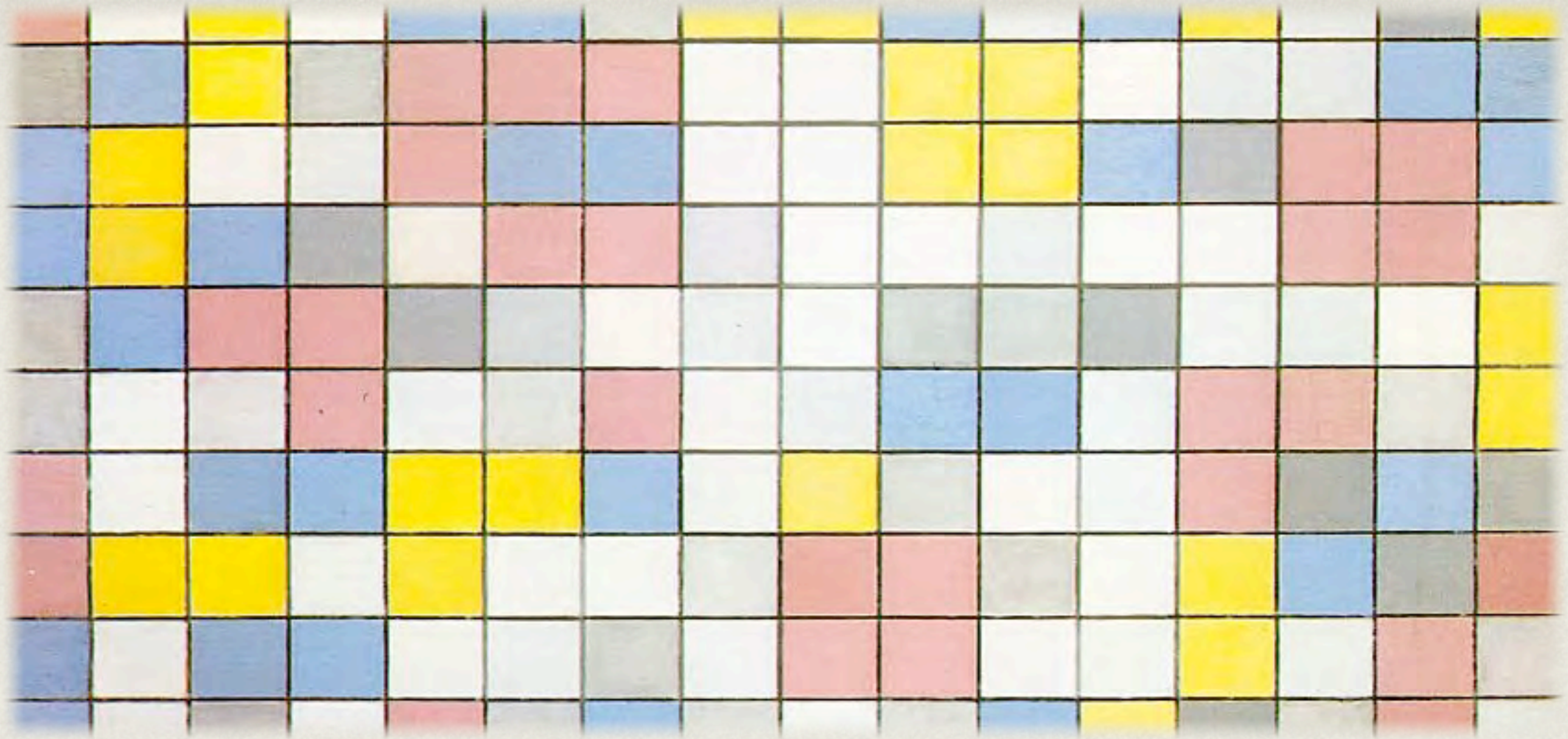


Marco Elio Tabacchi

Complessità Astratta

Corso di Informatica Teorica

Studiare la complessità

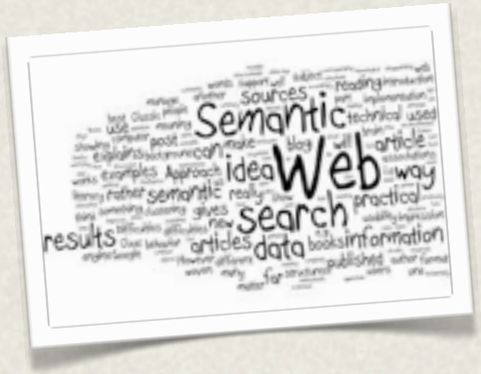
1. **Una complessità in direzione meno costruttiva**, classificando i problemi insolubili algoritmicamente:
    - MdT con Oracolo, S con opportune istruzioni V ➛ V'.
2. **Una complessità concreta**, che mira a classificare i problemi in base alla loro trattabilità:
    - calcolabilità in termini di input, classificazione dei problemi ed inclusione delle classi NL, P, NP, PSPACE, EXPTIME, EXSPACE.
3. **Una complessità strutturale**, con la teoria degli algoritmi, una complessità che non è solo complicazione:
    - Definizione di casualità/informazione/probabilità, Kolmogorov-Solomonov-Chaitin, Ipotesi di von Neumann.
4. **La Complessità Astratta**:
    - ...che vedremo in un attimo.

Misurare la Complessità

- Un compito niente affatto facile o banale!
- La complessità è un fenomeno prettamente multidominio – non esiste un concetto standar di complessità che si possa applicare con successo ad ogni misurazione
- Quando ci chiediamo “Quanto è complesso?” facciamo in realtà domande diverse a seconda del dominio e diverse domande per lo stesso dominio

Cosa rappresenta?
Quanto è difficile riprodurlo?
In che stile è dipinto?

Da cosa è prodotto?
Come posso accordarmi?



Quale informazione da?
Qual è il suo significato?
Che regolarità presenta?

- Il meglio che possiamo fare è trovare un (o sperabilmente più di un) metodo per ogni dominio, e dei buoni e coerenti metodi di fusione delle informazioni acquisite

Misurare la Complessità

nel Dominio dei Programmi

- Le cose migliorano, ma non di molto, se ci concentriamo sul dominio dei programmi

```
class Question(models.Model):
    subject = models.ForeignKey(Subject)
    stem = models.TextField()
    user = models.ForeignKey(User)
    question_type = models.ForeignKey(QuestionType)

    def __unicode__(self):
        return self.stem[:50]
```

Quanto posso comprimerlo?
Che spazio occupa in memoria?
Quanto tempo impiega a dare un risultato?

- Esiste però, almeno nel caso dei programmi, la possibilità di dare una definizione di altissimo livello – quindi scevra da difficoltà dovute alle implementazioni – che però cattura molte interessanti proprietà della complessità, è applicabile in maniera generale ed è sorprendentemente semplice.

Manuel Blum (1938 -)
Turing Award, 1995

La Complessità Astratta

gli Assiomi di Blum

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni “buona” funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni “buona” funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni "buona" funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:

  1. **C(x,i)** converge SSE **P_i(x)** converge

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni "buona" funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:

  1. **C(x,i)** converge SSE **P_i(x)** converge
  2. Il predicato **C(x,i) ≤ y** è ricorsivo

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni "buona" funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:
  1. $\mathbf{C(x,i)}$ converge SSE $\mathbf{P_i(x)}$ converge
  2. Il predicato $\mathbf{C(x,i) \leq y}$ è ricorsivo
- $\mathbf{C_i(x)}$ (per comodità di notazione in analogia con $\mathbf{P_i(x)}$) è quindi definita come la **complessità computazionale** del programma con numero di Gödel **i** quando viene eseguito con input **x**.

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni "buona" funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:
    1. $\mathbf{C(x,i)}$ converge SSE $\mathbf{P_i(x)}$ converge
    2. Il predicato $\mathbf{C(x,i) \leq y}$ è ricorsivo
- $\mathbf{C_i(x)}$ (per comodità di notazione in analogia con $\mathbf{P_i(x)}$) è quindi definita come la **complessità computazionale** del programma con numero di Gödel **i** quando viene eseguito con input **x**.
- Un altro modo per dirlo:

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni "buona" funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:
    1. $\mathbf{C(x,i)}$ converge SSE $\mathbf{P_i(x)}$ converge
    2. Il predicato $\mathbf{C(x,i) \leq y}$ è ricorsivo
- $\mathbf{C_i(x)}$ (per comodità di notazione in analogia con $\mathbf{P_i(x)}$) è quindi definita come la **complessità computazionale** del programma con numero di Gödel **i** quando viene eseguito con input **x**.
- Un altro modo per dirlo:
    1. The resource is defined at those inputs—and at those inputs only—at which the computation being measured is defined.

La Complessità Astratta

gli Assiomi di Blum

- Blum introduce, nella sua tesi di Dottorato, due semplici assiomi che ogni "buona" funzione di complessità deve rispettare. Questi assiomi sono alla base del concetto di Complessità Astratta:
- Una funzione in due parametri **C** definita sui naturali **N** è una funzione di complessità per il programma **P** di codice **i** eseguito con input **x**, se soddisfa gli assiomi di Blum:
    1. $\mathbf{C(x,i)}$ converge SSE $\mathbf{P_i(x)}$ converge
    2. Il predicato $\mathbf{C(x,i) \leq y}$ è ricorsivo
- $\mathbf{C_i(x)}$ (per comodità di notazione in analogia con $\mathbf{P_i(x)}$) è quindi definita come la **complessità computazionale** del programma con numero di Gödel **i** quando viene eseguito con input **x**.
- Un altro modo per dirlo:
    1. The resource is defined at those inputs—and at those inputs only—at which the computation being measured is defined.
    2. It is a decidable problem to determine whether a given value is indeed the amount of resource corresponding to a given input.

Misure che verificano gli assiomi

Il Contapassi

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

Misure che verificano gli assiomi

Il Contapassi

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x)$ = numero di istruzioni eseguite dal programma i con input x.**

Verifica degli assiomi di Blum:

Misure che verificano gli assiomi

Il Contapassi

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x)$ = numero di istruzioni eseguite dal programma i con input x.**

Verifica degli assiomi di Blum:

1. Ovvio (Se **$C_i(x)$** non converge, l'esecuzione non termina).

Misure che verificano gli assiomi

Il Contapassi

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x)$ = numero di istruzioni eseguite dal programma i con input x.**

Verifica degli assiomi di Blum:

1. Ovvio (Se **$C_i(x)$** non converge, l'esecuzione non termina).

2. Abbiamo dimostrato la calcolabilità del predicato STP, associando ad ogni istruzione del programma un'istruzione di incremento di apposito contatore.

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$M_i(x)$ = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero
     2.b.Una variabile supera il valore y

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero
     2.b.Una variabile supera il valore y
     ➛ Falso

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero
     2.b.Una variabile supera il valore y
     ➛ Falso
     2.c. Si incontra due volte lo stesso snapshot: è un loop

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero
     2.b.Una variabile supera il valore y
     ➛ Falso
     2.c. Si incontra due volte lo stesso snapshot: è un loop
     ➛ Falso

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero
     2.b.Una variabile supera il valore y
     ➛ Falso
     2.c. Si incontra due volte lo stesso snapshot: è un loop
     ➛ Falso
- Ed il teorema della fermata?

Misure che verificano gli assiomi

Il valor massimo delle variabili

1. **C(x,i)** converge SSE **P_i(x)** converge
2. Il predicato **C(x,i) ≤ y** è ricorsivo

- **M_i(x) = il valore più alto assunto da una variabile nel programma i con input x se esso converge.**
  Verifica degli assiomi di Blum:
  1. Dalla definizione.
  2. L'idea è di verificare se **M(x,i) ≤ y** una volta fissati **i**, **x** ed **y**. Utilizzando il concetto di snapshot (ovvero una "foto" dell'insieme di tutti i valori di variabili e puntatori), e ricordando che fissato y esistono solo un numero finito di stati in cui tutte le variabili hanno valori minori di y, possiamo distinguere 3 casi:
     2.a. Il programma si ferma e nessuna variabile supera y
     ➛ Vero
     2.b.Una variabile supera il valore y
     ➛ Falso
     2.c. Si incontra due volte lo stesso snapshot: è un loop
     ➛ Falso
- Ed il teorema della fermata?
- I casi indeterminati si trovano in 2.b.!

Misure che verificano gli assiomi

La funzione autocomplessa

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

Misure che verificano gli assiomi

La funzione autocomplessa

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x) = P_i(x)$, ovvero la complessità è uguale all'output**

Verifica degli assiomi di Blum:

Misure che verificano gli assiomi

La funzione autocomplessa

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x) = P_i(x)$, ovvero la complessità è uguale all'output**

  Verifica degli assiomi di Blum:

  1. Ovvio.

Misure che verificano gli assiomi

La funzione autocomplessa

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x) = P_i(x)$, ovvero la complessità è uguale all'output**

Verifica degli assiomi di Blum:

1. Ovvio.

2. Sorpresa – il secondo assioma **non è verificato!** Basta infatti scegliere un programma che calcoli la funzione

Misure che verificano gli assiomi

La funzione autocomplessa

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x) = P_i(x)$, ovvero la complessità è uguale all'output**

Verifica degli assiomi di Blum:

1. Ovvio.
2. Sorpresa – il secondo assioma **non è verificato!** Basta infatti scegliere un programma che calcoli la funzione

   $P_i(x) = 0$ se x appartiene ad un insieme r.e. ma non ricorsivo, non converge altrimenti

Misure che verificano gli assiomi

La funzione autocomplessa

1. **$C(x,i)$** converge SSE **$P_i(x)$** converge
2. Il predicato **$C(x,i) \leq y$** è ricorsivo

- **$C_i(x) = P_i(x)$, ovvero la complessità è uguale all'output**

Verifica degli assiomi di Blum:

1. Ovvio.

2. Sorpresa – il secondo assioma **non è verificato!** Basta infatti scegliere un programma che calcoli la funzione

   $P_i(x) = 0$ se x appartiene ad un insieme r.e. ma non ricorsivo, non converge altrimenti

   In questo caso **$C(x,i) \leq 0$** corrisponde a determinare se x appartiene all'insieme non ricorsivo - per definizione un problema non ricorsivo!

Le interessanti proprietà di Blum

Lo Scaling

Le interessanti proprietà di Blum

Lo Scaling

- **$C_i(x)$** è una **misura di complessità**, **$r(x)$** è un **fattore di scala**

Le interessanti proprietà di Blum

Lo Scaling

- **$C_i(x)$** è una **misura di complessità**, **$r(x)$** è un **fattore di scala**
    - Definiamo fattore di scala una funzione che ha le seguenti proprietà:

Le interessanti proprietà di Blum

Lo Scaling

- **$C_i(x)$** è una **misura di complessità**, **$r(x)$** è un **fattore di scala**
    - Definiamo fattore di scala una funzione che ha le seguenti proprietà:
        1. **r** è non decrescente

Le interessanti proprietà di Blum

Lo Scaling

- $\mathbf{C_i(x)}$ è una **misura di complessità**, $\mathbf{r(x)}$ è un **fattore di scala**
    - Definiamo fattore di scala una funzione che ha le seguenti proprietà:
        1. $\mathbf{r}$ è non decrescente
        2. $\mathbf{\lim_{(x \to \infty)} r(x) = \infty}$ (può assumere valori grandi a piacere)

Le interessanti proprietà di Blum

Lo Scaling

- $\mathbf{C_i(x)}$ è una **misura di complessità**, $\mathbf{r(x)}$ è un **fattore di scala**
    - Definiamo fattore di scala una funzione che ha le seguenti proprietà:
        1. $\mathbf{r}$ è non decrescente
        2. $\mathbf{\lim_{(x \to \infty)} r(x) = \infty}$ (può assumere valori grandi a piacere)
- Se $\mathbf{D_i(x) = r(C_i(x))}$ allora $\mathbf{D_i(x)}$ è una misura di complessità

Le interessanti proprietà di Blum

Lo Scaling

- **$C_i(x)$** è una **misura di complessità**, **$r(x)$** è un **fattore di scala**
    - Definiamo fattore di scala una funzione che ha le seguenti proprietà:
        1. **r** è non decrescente
        2. **$\lim_{(x \to \infty)} r(x) = \infty$** (può assumere valori grandi a piacere)
- Se **$D_i(x) = r(C_i(x))$** allora **$D_i(x)$** è una misura di complessità
- Questa proprietà (che non dimostriamo) è intuitiva – d'altro canto è facile assumere che se a è una misura di complessità lo sia anche A*2, 2^A, etc.,

Le interessanti proprietà di Blum

Lo Scaling

- $\mathbf{C_i(x)}$ è una **misura di complessità**, $\mathbf{r(x)}$ è un **fattore di scala**
    - Definiamo fattore di scala una funzione che ha le seguenti proprietà:
        1. $\mathbf{r}$ è non decrescente
        2. $\mathbf{\lim_{(x \to \infty)} r(x) = \infty}$ (può assumere valori grandi a piacere)
- Se $\mathbf{D_i(x) = r(C_i(x))}$ allora $\mathbf{D_i(x)}$ è una misura di complessità
- Questa proprietà (che non dimostriamo) è intuitiva – d'altro canto è facile assumere che se a è una misura di complessità lo sia anche A*2, 2^A, etc.,
- ma sottende una nozione importante: qualsiasi coppia di misure di complessità, per quanto diversa, è sempre legata da un fattore di scala.

Le interessanti proprietà di Blum

Il Collegamento Ricorsivo

Le interessanti proprietà di Blum

Il Collegamento Ricorsivo

- **$C_i(x)$ e $D_i(x)$** sono **misure di complessità arbitrarie.**
  Per ogni coppia **esiste sempre** una funzione ricorsiva

  $$r(x,y)$$

  tale che quasi ovunque (in particolare per x>i)

  $$C_i(x) \leq r(x, D_i(x))$$
  e
  $$D_i(x) \leq r(x, C_i(x))$$

Le interessanti proprietà di Blum

Il Collegamento Ricorsivo

- **$C_i(x)$ e $D_i(x)$** sono **misure di complessità arbitrarie.**
  Per ogni coppia **esiste sempre** una funzione ricorsiva

  $$r(x,y)$$

  tale che quasi ovunque (in particolare per x>i)

  $$C_i(x) \leq r(x, D_i(x))$$
  e
  $$D_i(x) \leq r(x, C_i(x))$$

- Questa proprietà (che ancora una volta non dimostriamo) è certamente molto meno intuitiva, ma il suo significato è sorprendente: dimostrato un teorema per una certa misura di complessità, è spesso possibile estendere il risultato a tutte le altre misure di complessità.

Le interessanti proprietà di Blum

Il Collegamento Ricorsivo

- **$C_i(x)$ e $D_i(x)$** sono **misure di complessità arbitrarie.**
  Per ogni coppia **esiste sempre** una funzione ricorsiva

  $$r(x,y)$$

  tale che quasi ovunque (in particolare per x>i)

  $$C_i(x) \leq r(x, D_i(x))$$
  e
  $$D_i(x) \leq r(x, C_i(x))$$

- Questa proprietà (che ancora una volta non dimostriamo) è certamente molto meno intuitiva, ma il suo significato è sorprendente: dimostrato un teorema per una certa misura di complessità, è spesso possibile estendere il risultato a tutte le altre misure di complessità.

- L'apparente semplicità degli assiomi di Blum fa però sì che le misure di complessità che li rispettano siano fortemente interrelate.

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia **$g(x)$** una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva **$t(x)$**
  tale che, per valori di **x** che superano **i** (quindi quasi ovunque),
  se **$C_i(x) \leq g(t(x))$** allora **$C_i(x) \leq t(x)$**

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia **$g(x)$** una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva **$t(x)$**
  tale che, per valori di **x** che superano **i** (quindi quasi ovunque),
  se **$C_i(x) \leq g(t(x))$** allora **$C_i(x) \leq t(x)$**
- E quindi "[the] functions computable of complexity measure **$t(x)$** are the same as functions computable of complexity measure **$g(t(x))$**"

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia $g(x)$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $t(x)$
  tale che, per valori di x che superano i (quindi quasi ovunque),
  se $C_i(x) \leq g(t(x))$ allora $C_i(x) \leq t(x)$
- E quindi "[the] functions computable of complexity measure $t(x)$ are the same as functions computable of complexity measure $g(t(x))$"
- Per dimostrarlo costruiamo un esempio di $t(x)$:
  $t(x) = \forall i<x\ (C_i(x) \leq y \vee C_i(x) > g(x))$

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia $\mathbf{g(x)}$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $\mathbf{t(x)}$
  tale che, per valori di $\mathbf{x}$ che superano $\mathbf{i}$ (quindi quasi ovunque),
  se $\mathbf{C_i(x) \leq g(t(x))}$ allora $\mathbf{C_i(x) \leq t(x)}$
- E quindi "[the] functions computable of complexity measure $\mathbf{t(x)}$ are the same as functions computable of complexity measure $\mathbf{g(t(x))}$"
- Per dimostrarlo costruiamo un esempio di $\mathbf{t(x)}$:
  $\mathbf{t(x) = \forall i{<}x\ (C_i(x) \leq y \vee C_i(x) > g(x))}$

1. Per ogni $\mathbf{x}$ esiste un $\mathbf{y}$, visto che:

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia $g(x)$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $t(x)$
  tale che, per valori di x che superano i (quindi quasi ovunque),
  se $C_i(x) \leq g(t(x))$ allora $C_i(x) \leq t(x)$
- E quindi "[the] functions computable of complexity measure $t(x)$ are the same as functions computable of complexity measure $g(t(x))$"
- Per dimostrarlo costruiamo un esempio di $t(x)$:
  $t(x) = \forall i<x\ (C_i(x) \leq y \vee C_i(x) > g(x))$

1. Per ogni x esiste un y, visto che:

   1.a. Se $C_i(x)$ non è definita, allora per ogni y, $C_i(x) > g(x)$ *

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia $\mathbf{g(x)}$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $\mathbf{t(x)}$
  tale che, per valori di $\mathbf{x}$ che superano $\mathbf{i}$ (quindi quasi ovunque),
  se $\mathbf{C_i(x) \leq g(t(x))}$ allora $\mathbf{C_i(x) \leq t(x)}$
- E quindi "[the] functions computable of complexity measure $\mathbf{t(x)}$ are the same as functions computable of complexity measure $\mathbf{g(t(x))}$"
- Per dimostrarlo costruiamo un esempio di $\mathbf{t(x)}$:
  $\mathbf{t(x) = \forall i{<}x\ (C_i(x) \leq y \vee C_i(x) > g(x))}$

1. Per ogni $\mathbf{x}$ esiste un $\mathbf{y}$, visto che:

   1.a. Se $\mathbf{C_i(x)}$ non è definita, allora per ogni $\mathbf{y}$, $\mathbf{C_i(x) > g(x)}$ *

   1.b. Se $\mathbf{C_i(x)}$ è definita, allora esiste $\mathbf{y}$ tale che $\mathbf{C_i(x) < y}$ (Perchè?)

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia $\mathbf{g(x)}$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $\mathbf{t(x)}$
  tale che, per valori di $\mathbf{x}$ che superano $\mathbf{i}$ (quindi quasi ovunque),
  se $\mathbf{C_i(x) \leq g(t(x))}$ allora $\mathbf{C_i(x) \leq t(x)}$
- E quindi "[the] functions computable of complexity measure $\mathbf{t(x)}$ are the same as functions computable of complexity measure $\mathbf{g(t(x))}$"
- Per dimostrarlo costruiamo un esempio di $\mathbf{t(x)}$:
  $\mathbf{t(x) = \forall i{<}x\ (C_i(x) \leq y \vee C_i(x) > g(x))}$

1. Per ogni $\mathbf{x}$ esiste un $\mathbf{y}$, visto che:

   1.a. Se $\mathbf{C_i(x)}$ non è definita, allora per ogni $\mathbf{y}$, $\mathbf{C_i(x) > g(x)}$ *

   1.b. Se $\mathbf{C_i(x)}$ è definita, allora esiste $\mathbf{y}$ tale che $\mathbf{C_i(x) < y}$ (Perchè?)

2. $\mathbf{y}$ può essere calcolato ricorsivamente. Perché?

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

- Sia $g(x)$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $t(x)$
  tale che, per valori di x che superano i (quindi quasi ovunque),
  se $C_i(x) \leq g(t(x))$ allora $C_i(x) \leq t(x)$
- E quindi "[the] functions computable of complexity measure $t(x)$ are the same as functions computable of complexity measure $g(t(x))$"
- Per dimostrarlo costruiamo un esempio di $t(x)$:
  $t(x) = \forall i<x\ (C_i(x) \leq y \vee C_i(x) > g(x))$

1. Per ogni x esiste un y, visto che:
   1.a. Se $C_i(x)$ non è definita, allora per ogni y, $C_i(x) > g(x)$ *
   1.b. Se $C_i(x)$ è definita, allora esiste y tale che $C_i(x) < y$ (Perchè?)
2. y può essere calcolato ricorsivamente. Perché?
   2.a. $C_i(x) \leq y$ e $C_i(x) > g(x)$ sono entrambi predicati ricorsivi

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

Il teorema della Lacuna (Trakhtenbrot/Borodin)

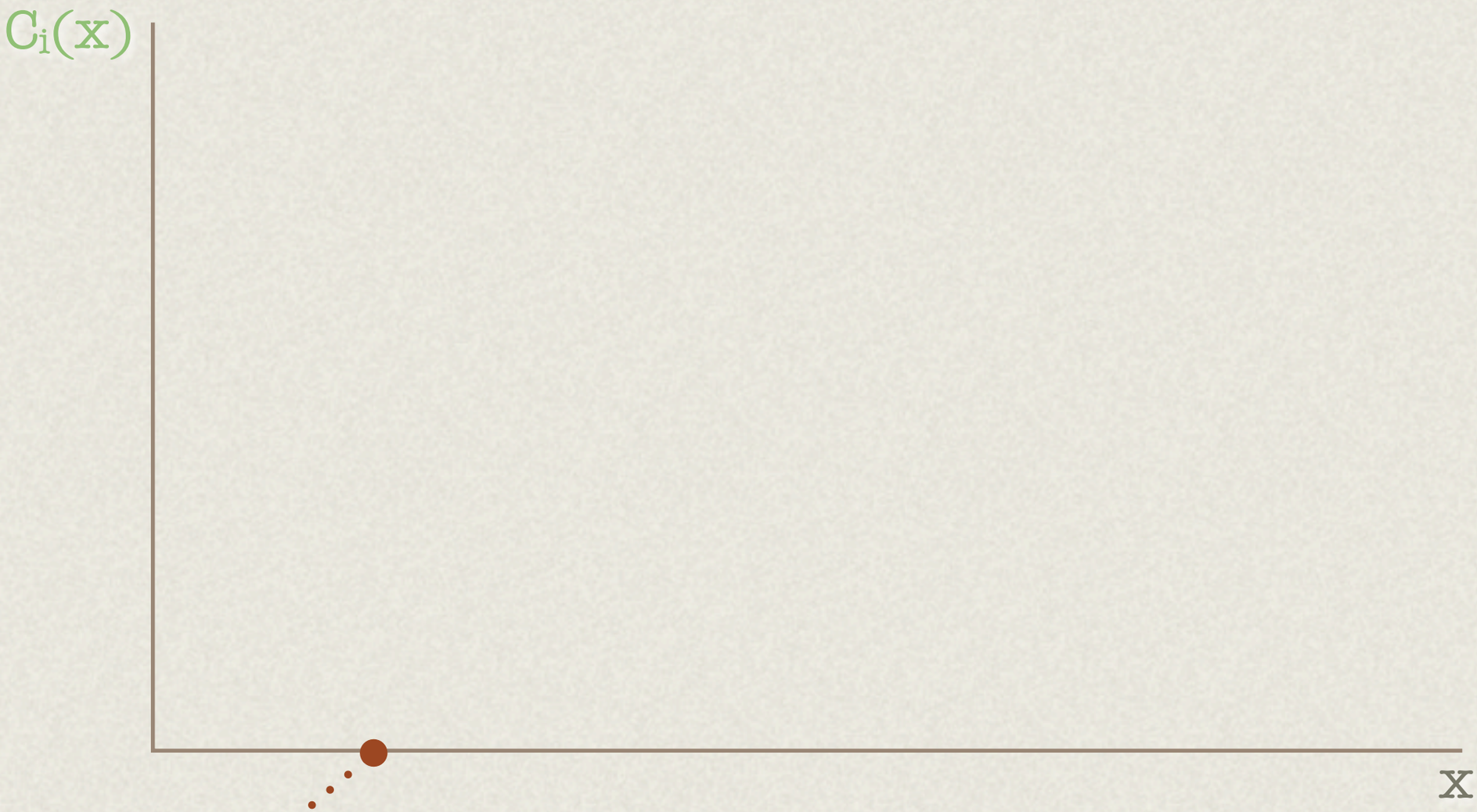
- Sia $g(x)$ una funzione ricorsiva.
  Esiste sempre una funzione ricorsiva $t(x)$
  tale che, per valori di x che superano i (quindi quasi ovunque),
  se $C_i(x) \leq g(t(x))$ allora $C_i(x) \leq t(x)$
- E quindi "[the] functions computable of complexity measure $t(x)$ are the same as functions computable of complexity measure $g(t(x))$"
- Per dimostrarlo costruiamo un esempio di $t(x)$:
  $t(x) = \forall i<x\ (C_i(x) \leq y \vee C_i(x) > g(x))$

1. Per ogni x esiste un y, visto che:
   1.a. Se $C_i(x)$ non è definita, allora per ogni y, $C_i(x) > g(x)$ *
   1.b. Se $C_i(x)$ è definita, allora esiste y tale che $C_i(x) < y$ (Perchè?)
2. y può essere calcolato ricorsivamente. Perché?
   2.a. $C_i(x) \leq y$ e $C_i(x) > g(x)$ sono entrambi predicati ricorsivi
3. Quindi t soddisfa il teorema della lacuna!

* Per convenzione se g(x) non converge, g(x)= ∞

Le interessanti proprietà di Blum

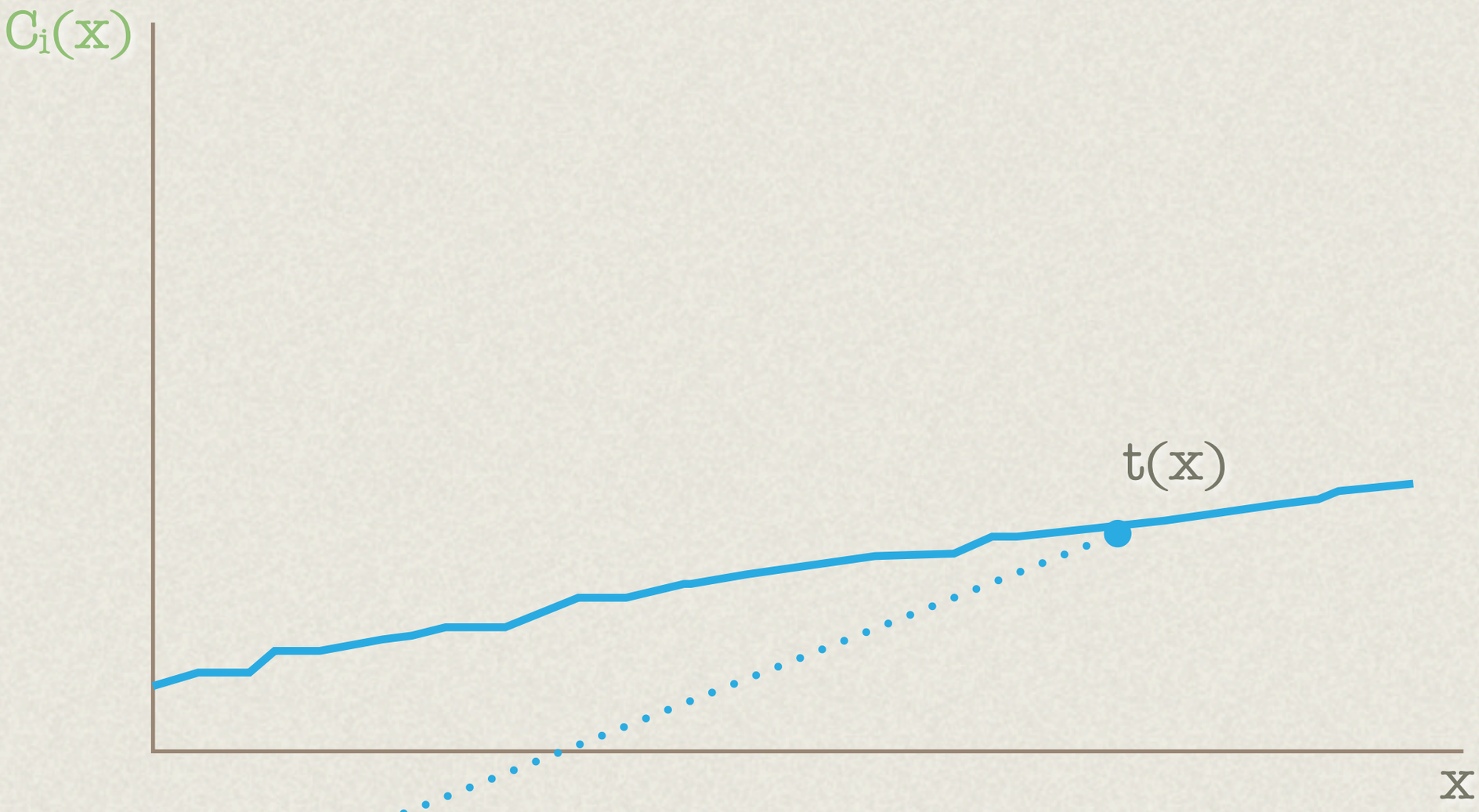
Il teorema della Lacuna



1 In questo grafico plottiamo il valore di complessità del programma i (fissato) con input x (che varia sull'asse delle ascisse)

Le interessanti proprietà di Blum

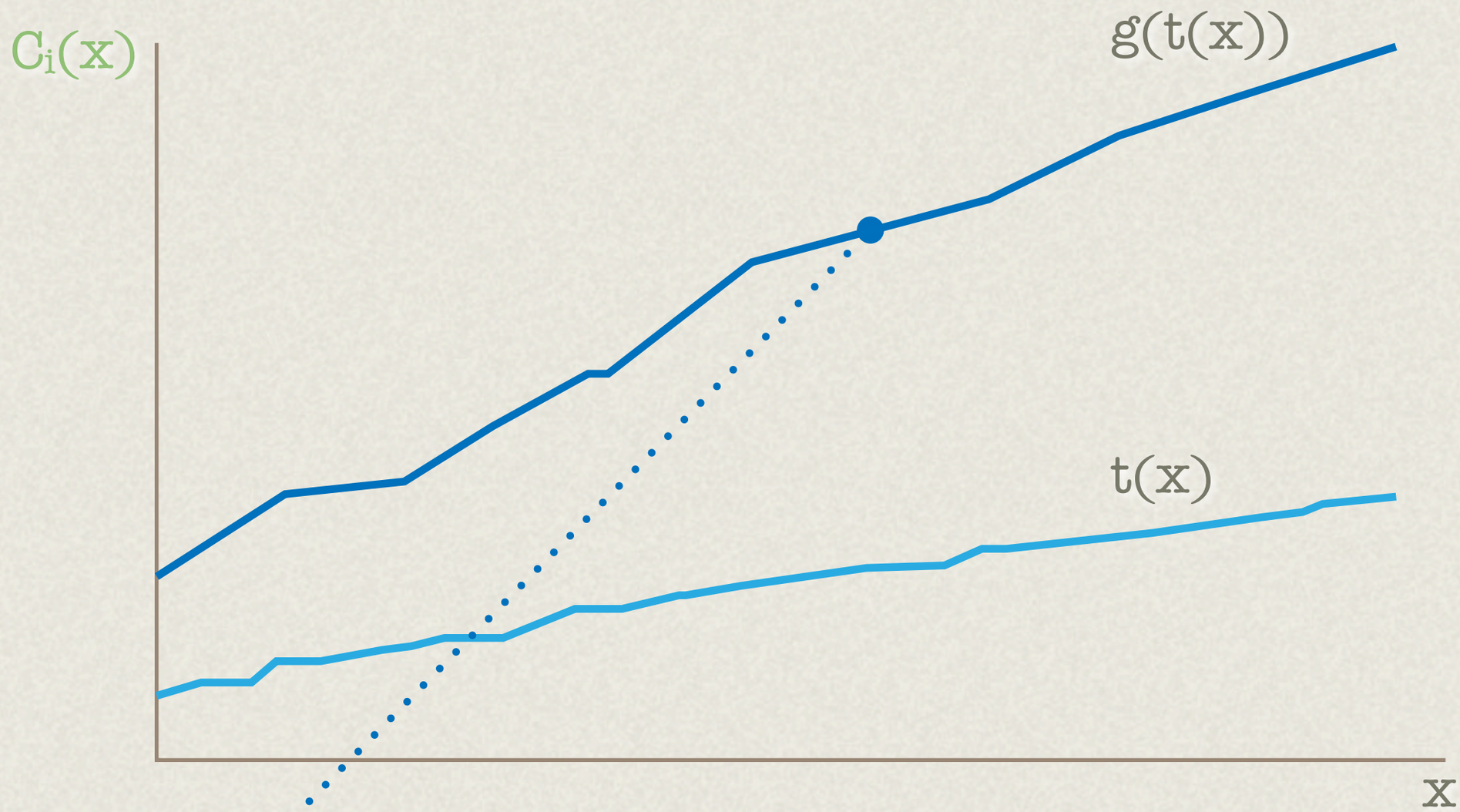
Il teorema della Lacuna



2 Questa linea rappresenta t(x), ovvero il nostro collo di bottiglia computazionale. All'inizio la nostra complessità può anche superare t(x), ma da un certo punto in poi, passato il valore x=i, dovrà necessariamente sottostarvi.

Le interessanti proprietà di Blum

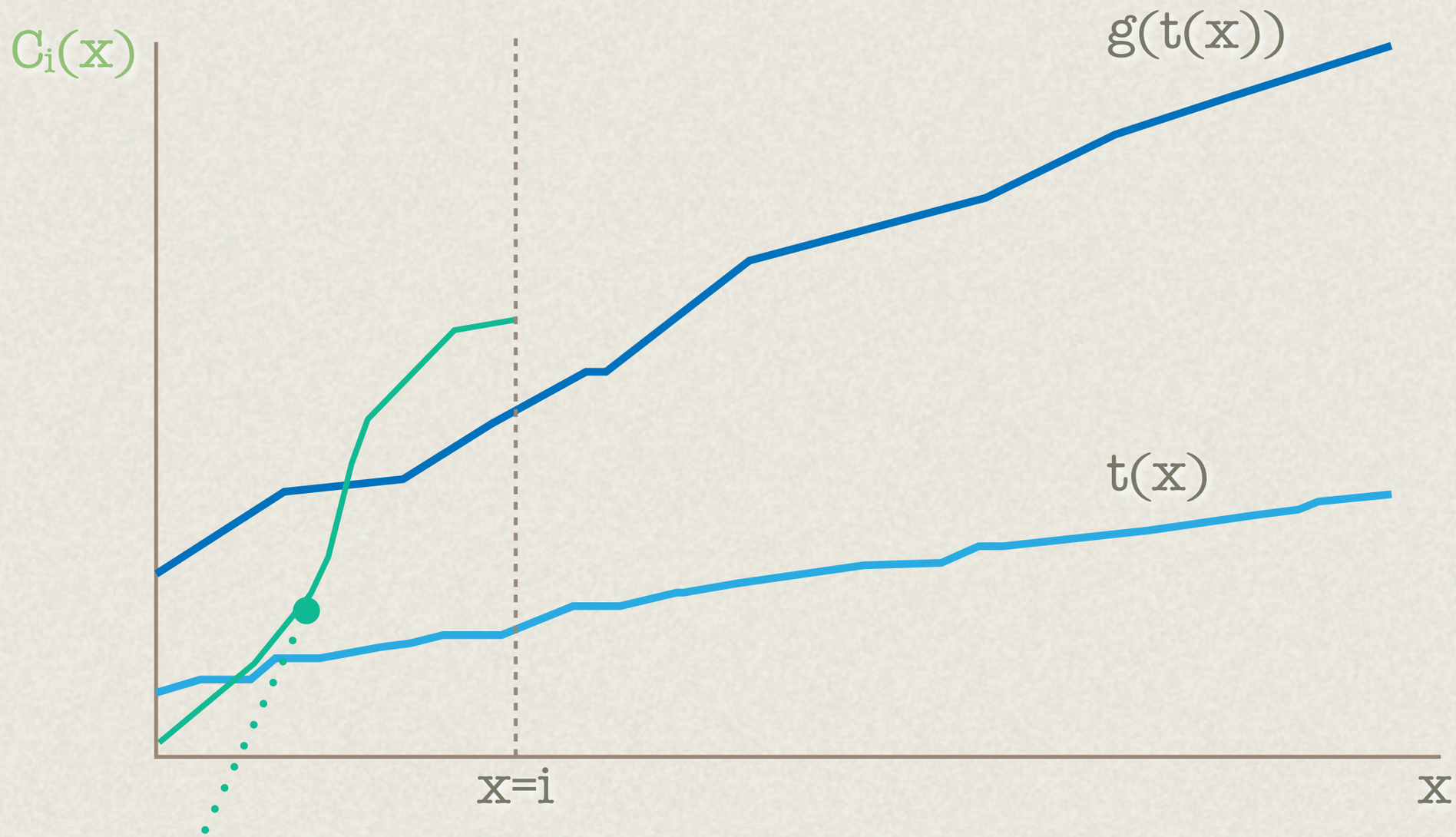
Il teorema della Lacuna



3 Abbiamo vinto al totip! Ora possiamo permetterci un computer molto più veloce, il cui limite computazionale è una funzione di t(x) che cresce molto velocemente: g(x,t(x))

Le interessanti proprietà di Blum

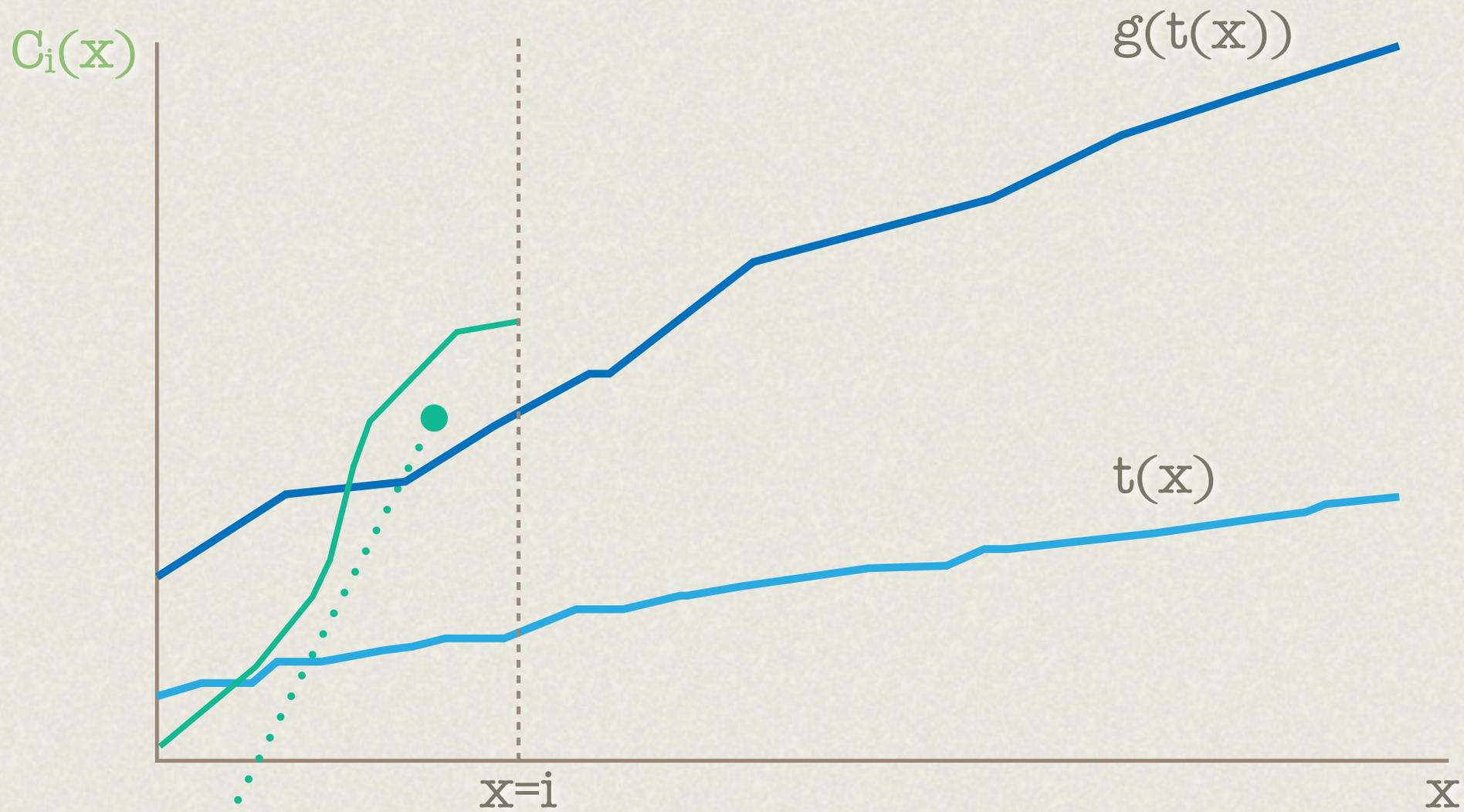
Il teorema della Lacuna



4 Cosa succede alla nostra $C_i(x)$? Fino ad x=i tutto va bene: la funzione può superare t(x) entrando nello spazio tra le due funzioni (la c.d. “lacuna”)...

Le interessanti proprietà di Blum

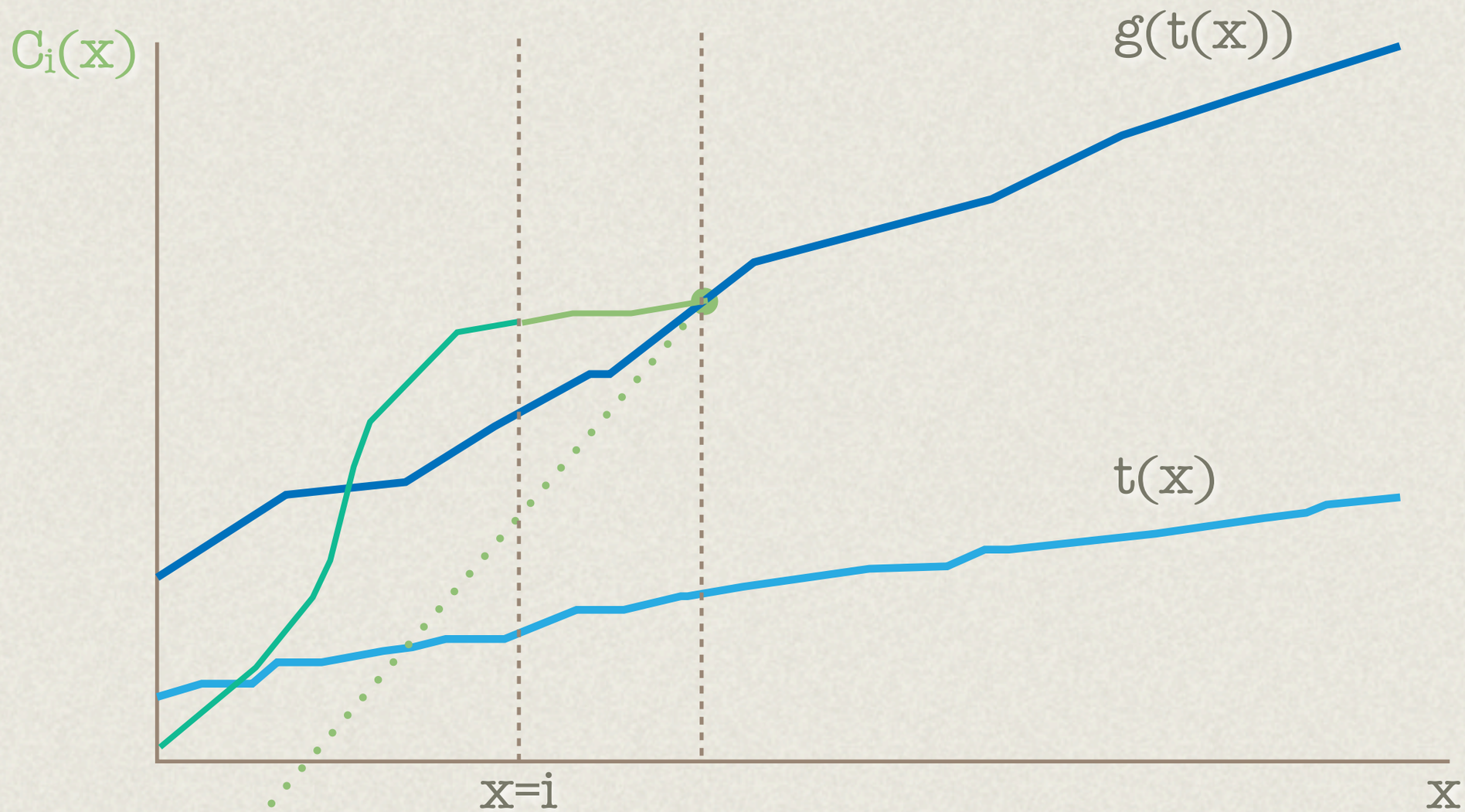
Il teorema della Lacuna



5 Cosa succede alla nostra $C_i(x)$? Fino ad x=i tutto va bene: la funzione può superare t(x) entrando nello spazio tra le due funzioni (la c.d. “lacuna”), e superare anche g(t(x)).

Le interessanti proprietà di Blum

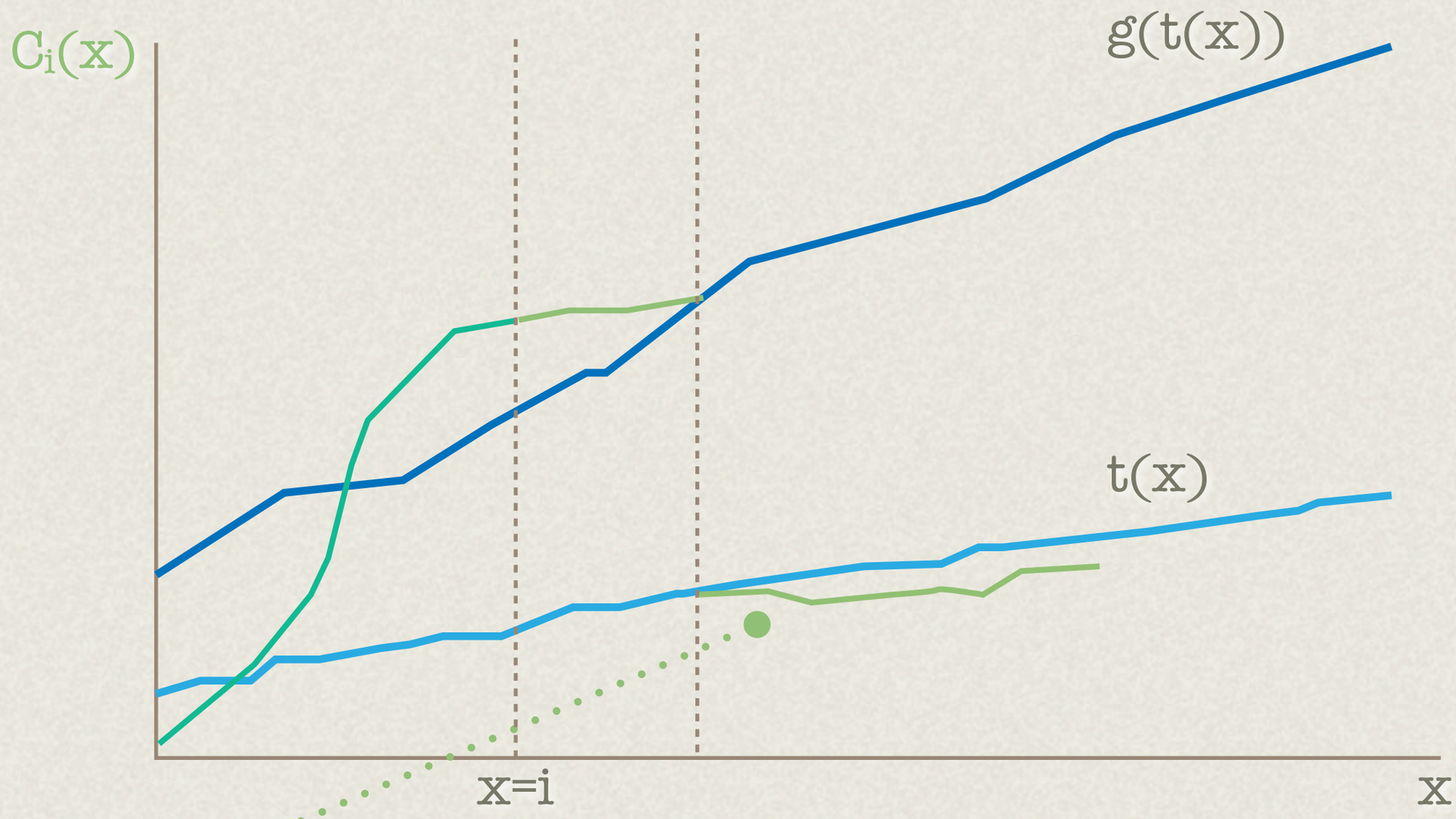
Il teorema della Lacuna



6 Ma da un certo valore di x in poi (superiore ad i) il teorema della lacuna ci garantisce che, per specifici t(x), nel momento in cui $C_i(x)$ diventerà inferiore a g(t(x))...

Le interessanti proprietà di Blum

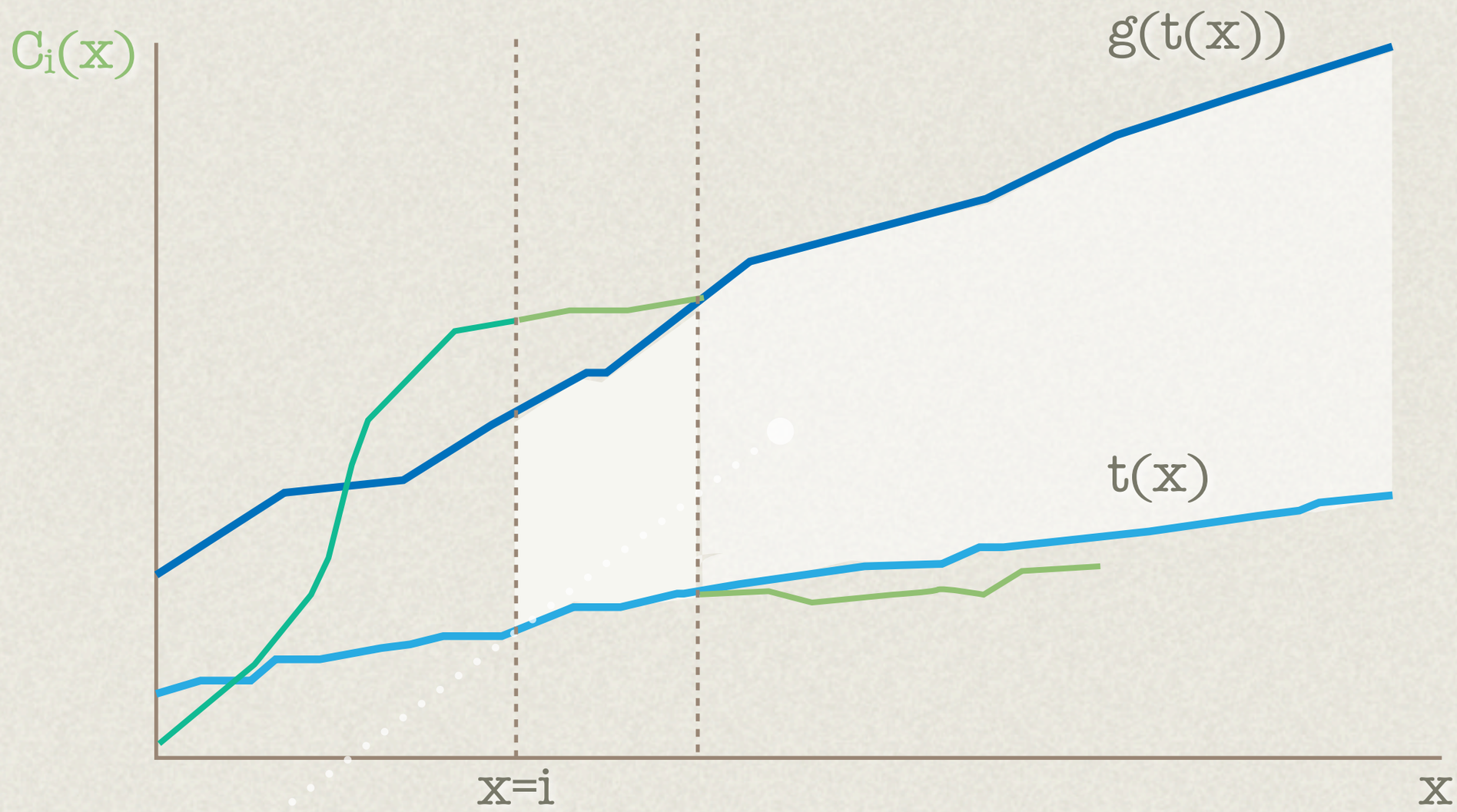
Il teorema della Lacuna



7 Ma da un certo valore di x in poi (superiore ad i) il teorema della lacuna ci garantisce che, per specifici t(x), nel momento in cui $C_i(x)$ diventerà inferiore a g(t(x)) diventerà inferiore anche a t(x)

Le interessanti proprietà di Blum

Il teorema della Lacuna



8 La lacuna non può essere invasa!

Le interessanti proprietà di Blum

Il teorema della Lacuna

Le interessanti proprietà di Blum

Il teorema della Lacuna

Detto in altri termini:

Le interessanti proprietà di Blum

Il teorema della Lacuna

Detto in altri termini:

generalmente

Le interessanti proprietà di Blum

Il teorema della Lacuna

Detto in altri termini:

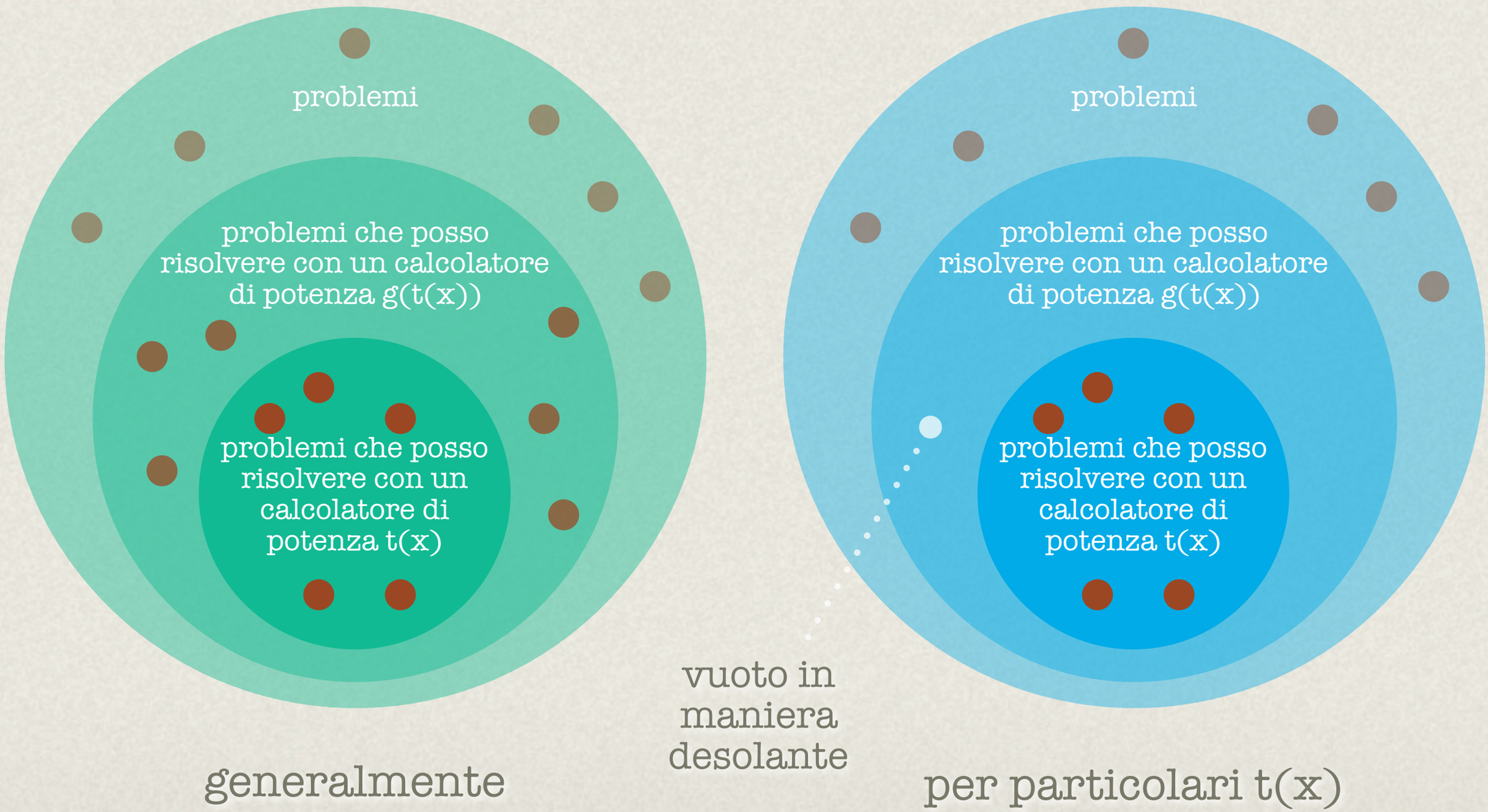
generalmente

per particolari t(x)

Le interessanti proprietà di Blum

Il teorema della Lacuna

Detto in altri termini:



Le interessanti proprietà di Blum

Il teorema della Lacuna

Le interessanti proprietà di Blum

Il teorema della Lacuna

- Il teorema della Lacuna ci condanna veramente ad accontentarci di un VIC 20 per i nostri calcoli?

Le interessanti proprietà di Blum

Il teorema della Lacuna

- Il teorema della Lacuna ci condanna veramente ad accontentarci di un VIC 20 per i nostri calcoli?
- Ovviamente no, e per (almeno) due motivi – uno intuitivo (viste le nozioni apprese), uno legato ad un concetto cardine della complessità:

Le interessanti proprietà di Blum

Il teorema della Lacuna

- Il teorema della Lacuna ci condanna veramente ad accontentarci di un VIC 20 per i nostri calcoli?
- Ovviamente no, e per (almeno) due motivi – uno intuitivo (viste le nozioni apprese), uno legato ad un concetto cardine della complessità:
    1. I valori di **x** per i quali il Teorema della Lacuna "entra in funzione" sono quelli maggiori di **i**. Per ogni applicazione pratica **i** è in generale un valore molto grande (e questo dovrebbe essere noto), e tipicamente molto più grande degli **x** che trattiamo;

Le interessanti proprietà di Blum

Il teorema della Lacuna

- Il teorema della Lacuna ci condanna veramente ad accontentarci di un VIC 20 per i nostri calcoli?
- Ovviamente no, e per (almeno) due motivi – uno intuitivo (viste le nozioni apprese), uno legato ad un concetto cardine della complessità:
    1. I valori di **x** per i quali il Teorema della Lacuna "entra in funzione" sono quelli maggiori di **i**. Per ogni applicazione pratica **i** è in generale un valore molto grande (e questo dovrebbe essere noto), e tipicamente molto più grande degli **x** che trattiamo;
    2. La funzione **t(x)** che abbiamo visto nella dimostrazione è una funzione costruita ad hoc, proprio per dimostrare che questa situazione può verificarsi, ma in realtà sarebbe una poverissima scelta per una misura di complessità: le misure che permettono ai concetti di gerarchie di complessità di funzionare sono sempre **constructable functions***, a tal punto che qualcuno ha postulato la necessità di inserire una 'constructability assumption' nelle gerarchie complesse. **t(x)** non è una funzione time-constructable.

Le interessanti proprietà di Blum

Il teorema della Lacuna

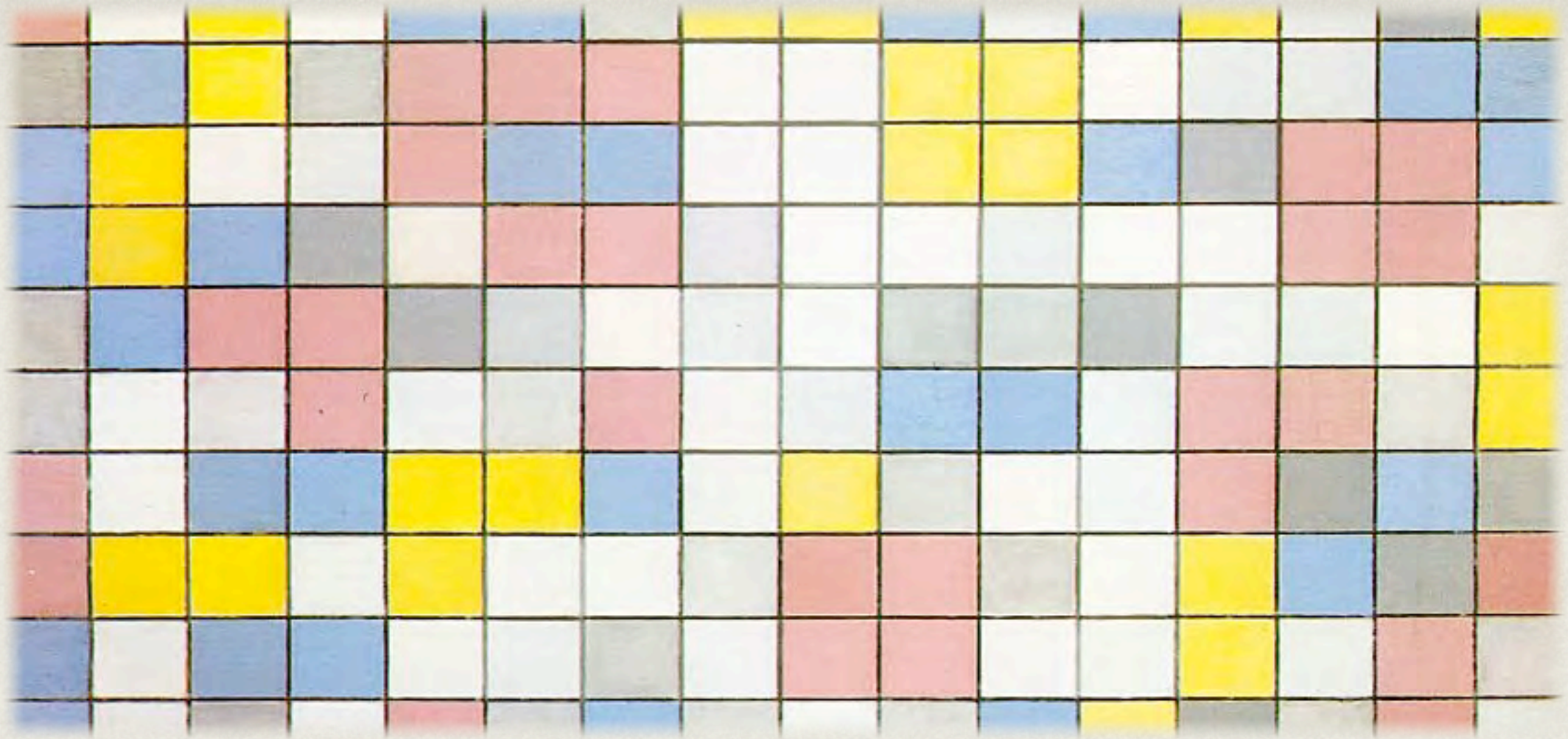
- Il teorema della Lacuna ci condanna veramente ad accontentarci di un VIC 20 per i nostri calcoli?
- Ovviamente no, e per (almeno) due motivi – uno intuitivo (viste le nozioni apprese), uno legato ad un concetto cardine della complessità:
    1. I valori di **x** per i quali il Teorema della Lacuna "entra in funzione" sono quelli maggiori di **i**. Per ogni applicazione pratica **i** è in generale un valore molto grande (e questo dovrebbe essere noto), e tipicamente molto più grande degli **x** che trattiamo;
    2. La funzione **t(x)** che abbiamo visto nella dimostrazione è una funzione costruita ad hoc, proprio per dimostrare che questa situazione può verificarsi, ma in realtà sarebbe una poverissima scelta per una misura di complessità: le misure che permettono ai concetti di gerarchie di complessità di funzionare sono sempre **constructable functions***, a tal punto che qualcuno ha postulato la necessità di inserire una 'constructability assumption' nelle gerarchie complesse. **t(x)** non è una funzione time-constructable.
- Ma voi queste cose non le studierete mai qui, perché qualcuno ha deciso che argomenti come la complessità non sono professionalizzanti!

Le interessanti proprietà di Blum

Il teorema della Lacuna

- Il teorema della Lacuna ci condanna veramente ad accontentarci di un VIC 20 per i nostri calcoli?
- Ovviamente no, e per (almeno) due motivi – uno intuitivo (viste le nozioni apprese), uno legato ad un concetto cardine della complessità:
    1. I valori di **x** per i quali il Teorema della Lacuna "entra in funzione" sono quelli maggiori di **i**. Per ogni applicazione pratica **i** è in generale un valore molto grande (e questo dovrebbe essere noto), e tipicamente molto più grande degli **x** che trattiamo;
    2. La funzione **t(x)** che abbiamo visto nella dimostrazione è una funzione costruita ad hoc, proprio per dimostrare che questa situazione può verificarsi, ma in realtà sarebbe una poverissima scelta per una misura di complessità: le misure che permettono ai concetti di gerarchie di complessità di funzionare sono sempre **constructable functions** *, a tal punto che qualcuno ha postulato la necessità di inserire una 'constructability assumption' nelle gerarchie complesse. **t(x)** non è una funzione time-constructable.
- Ma voi queste cose non le studierete mai qui, perché qualcuno ha deciso che argomenti come la complessità non sono professionalizzanti!

* A function **f** is time-constructible when there is a TM that computes **f** in **f(n)** time.



Fin

©2013 CC-BY-NC-SA