Sommario

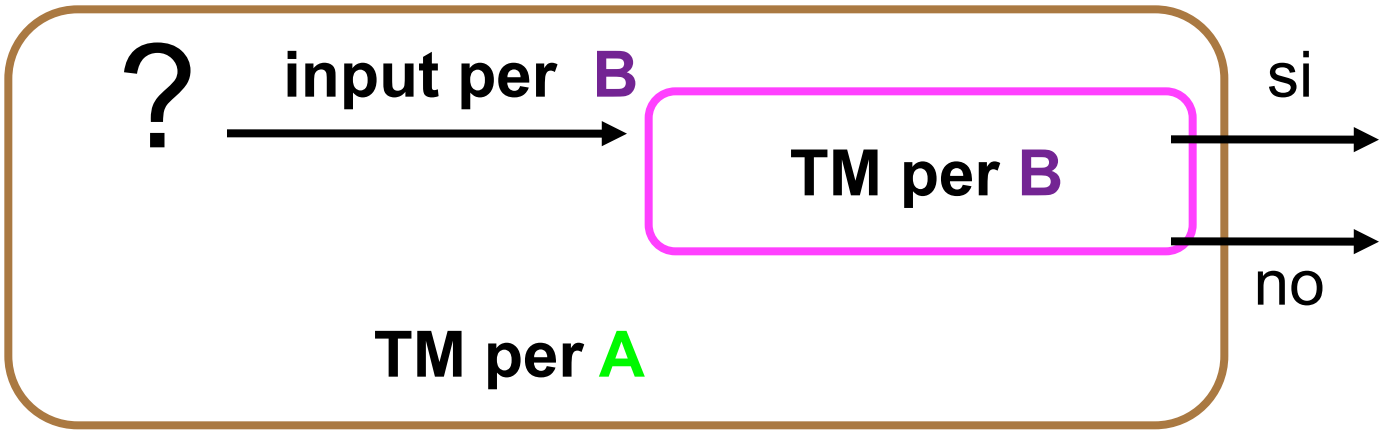
- **Riduzione di un problema ad un altro: introduzione e definizione**
- **proprietà delle riduzioni ed esempi.**

Riduzione: l'utilità

Nelle precedenti dimostrazioni di indecidibilità abbiamo supposto per assurdo l'esistenza di un algoritmo (una TM) per il nostro problema per costruirne uno per un altro problema già noto per la sua indecidibilità. (esempio: la fermata)

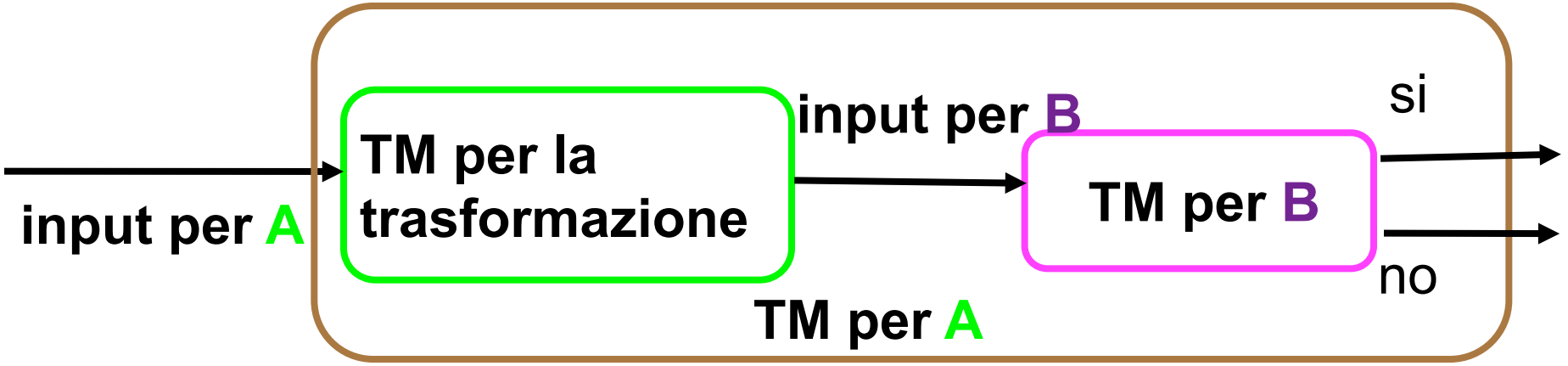
In quelle di decidibilità abbiamo utilizzato l'esistenza di un algoritmo (una TM) per un problema noto per costruirne uno per il nostro problema.

Se riuscissimo a stabilire una regola generale per fare questo, disporremmo di un potente metodo per dimostrare sia la decidibilità che l'indecidibilità di problemi

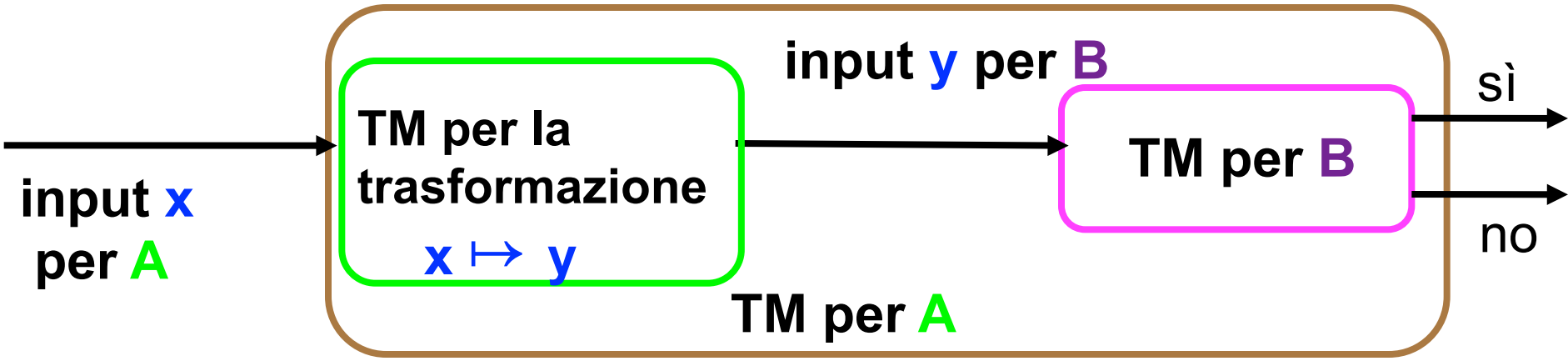


Riduzione: l'idea

Per usare l'algoritmo per B per risolvere A potremmo trasformare un input per A in un input per B in modo calcolabile e tale che combinando le due TM otteniamo una TM per A.



Cosa dobbiamo pretendere dalla trasformazione perchè la TM per A ottenuta sia corretta?

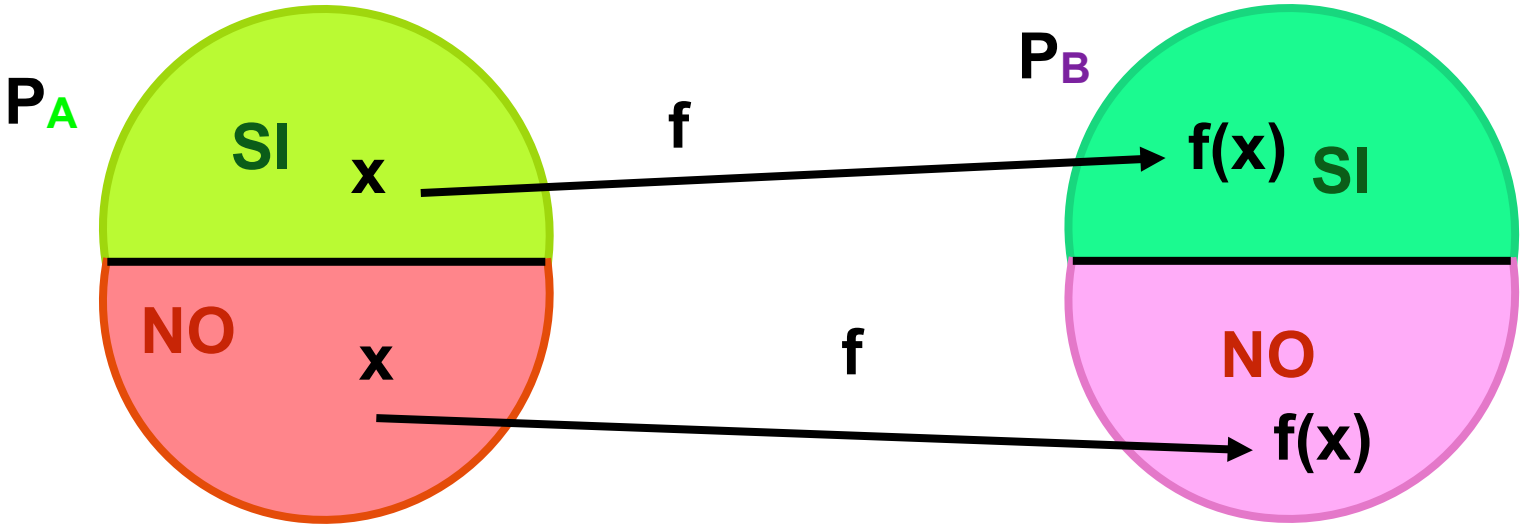


Risposta:
Se y è un input che produce risposta sì dalla TM per B allora anche l'input x corrispondente deve essere un input che deve produrre risposta sì, se invece y produce risposta no allora anche il corrispondente x deve essere un input che deve produrre risposta no

Riduzione tra problemi

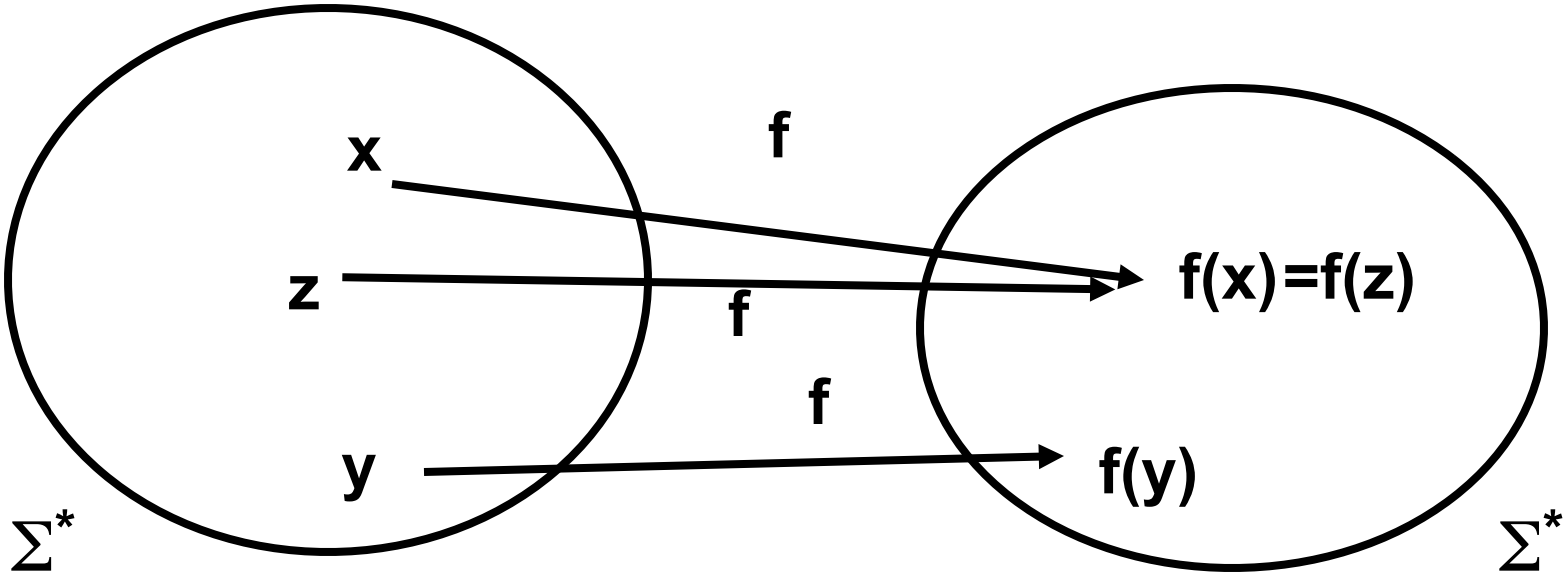
Ricordiamo che per noi un problema P è l'insieme delle (codifiche) delle sue istanze e che questo insieme si partiziona nell'insieme delle istanze SI e in quello delle istanze NO.

Diciamo che P_A si riduce a P_B se esiste una funzione Turing calcolabile che associa a istanze di P_A istanze di P_B in modo tale che istanze SI di P_A siano associate a istanze SI di P_B e a istanze NO di P_A istanze NO di P_B



Funzione Turing calcolabile

una funzione f : $\Sigma^* \rightarrow \Sigma^*$ si dice Turing calcolabile se esiste una TM T che inizializzata con x termina il calcolo su x scrivendo f(x) sul nastro (al posto di x o a seguire, con un opportuno separatore tra x e f(x)).

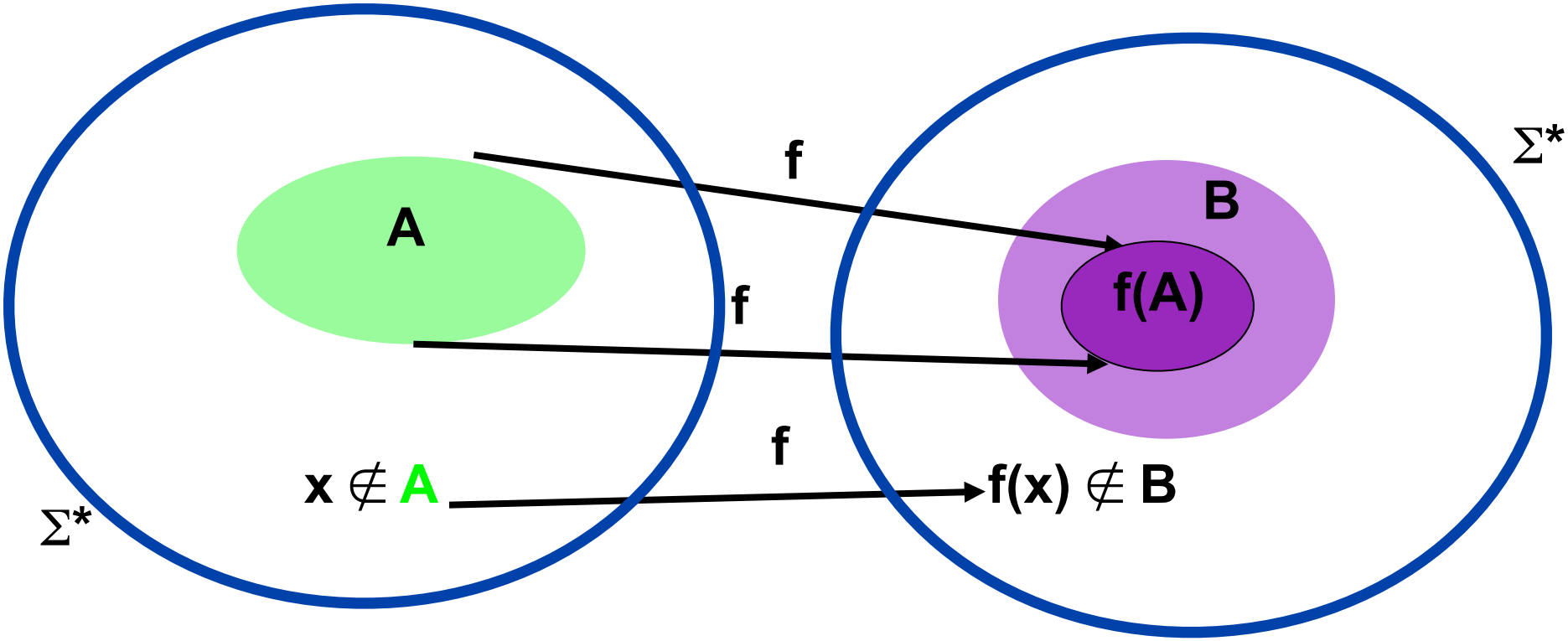


La funzione f può essere o no iniettiva o suriettiva!

Formalmente

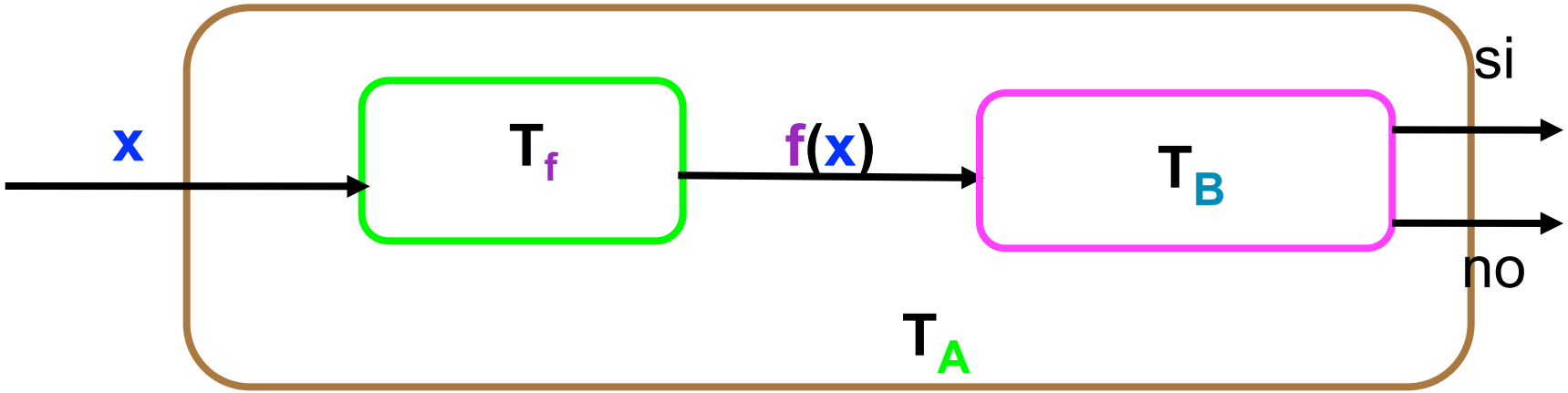
Dati A $\subseteq \Sigma^*$ e B $\subseteq \Sigma^*$ due linguaggi diciamo che A si riduce a B (many-one o mapping reducible), in breve $A \leq_m B$, se esiste una funzione $f : \Sigma^* \rightarrow \Sigma^*$ Turing calcolabile tale che

$$x \in A \Leftrightarrow \quad f(x) \in B$$



La TM

Se $A \leq_m B$ e T_B è la TM che decide B, allora esiste una TM T_A che decide A. Sia T_f la TM che calcola f, allora T_A è la composizione in sequenza di T_f e T_B



Infatti $[\ f(x) \in B \Rightarrow x \in A$ **e** $f(x) \notin B \Rightarrow x \notin A\ ] \Leftrightarrow$

$[\ x \in A \Leftrightarrow\ f(x) \in B\ ]$

Teoremi fondamentali

Teorema 1. Se A $\leq_m$ B e B è decidibile [Turing riconoscibile]

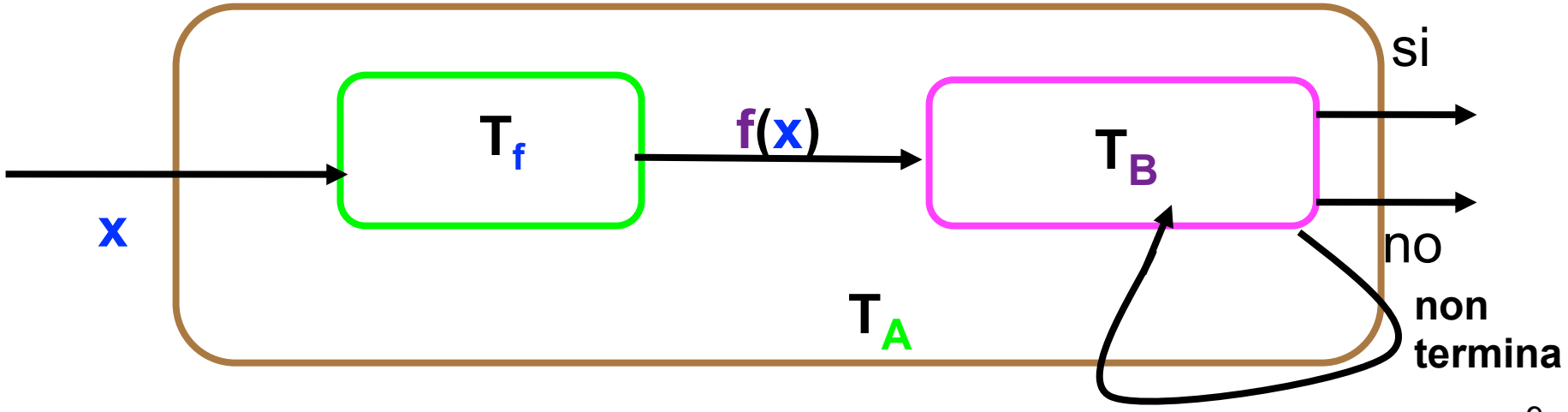
allora A è decidibile. [Turing riconoscibile]

A è <u>al più</u> difficile come B,
infatti otteniamo un algoritmo per A da uno dato per B

Teorema 2. Se A $\leq_m$ B e A è indecidibile [non Turing riconoscibile]

allora B è indecidibile . [non Turing riconoscibile]

[limite inferiore] B è almeno difficile come A: se A è noto essere difficile, allora la riduzione di A a B comporta che anche B è difficile.



Proprietà delle riduzioni

Teorema 3. La relazione $\leq_m$ è riflessiva e transitiva.

Riflessività: $A \leq_m A$

l'identità è una funzione calcolabile!

Transitività: $A \leq_m B$ e $B \leq_m C$ implica $A \leq_m C$

infatti se $A \leq_m B$ e $B \leq_m C$, allora esistono due funzioni

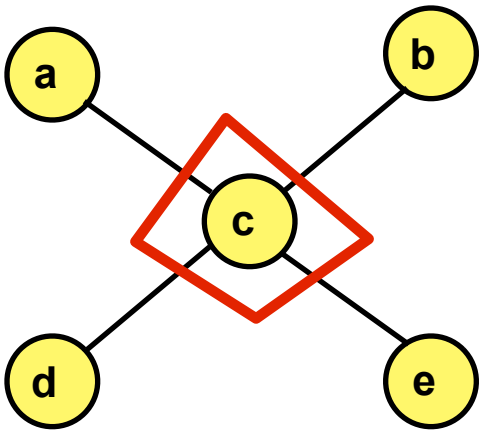
$f : \Sigma^* \rightarrow \Sigma^*$ e $g : \Sigma^* \rightarrow \Sigma^*$ Turing calcolabili tali che
$x \in A \Leftrightarrow f(x) \in B$ e $x \in B \Leftrightarrow g(x) \in C$.

La funzione composta $g \circ f(x) = g(f(x))$ soddisfa la proprietà

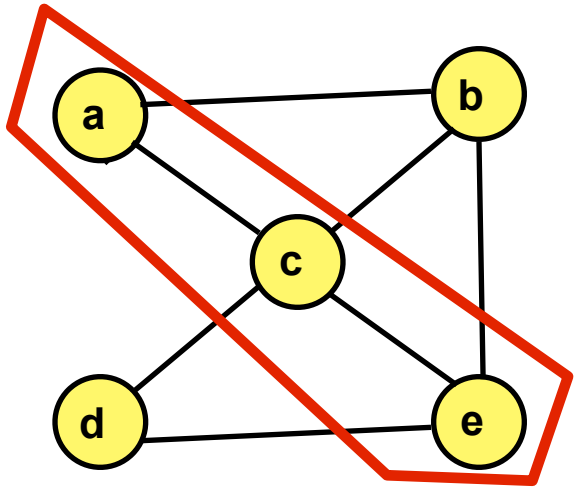
della riduzione perchè $x \in A \Leftrightarrow g \circ f(x) \in C$

VERTEX-COVER

Un vertex cover in un grafo non diretto è un sottoinsieme C dei vertici con la proprietà che ogni arco di G ha un estremo in C. Un k-vertex cover è un vertex cover con k nodi.



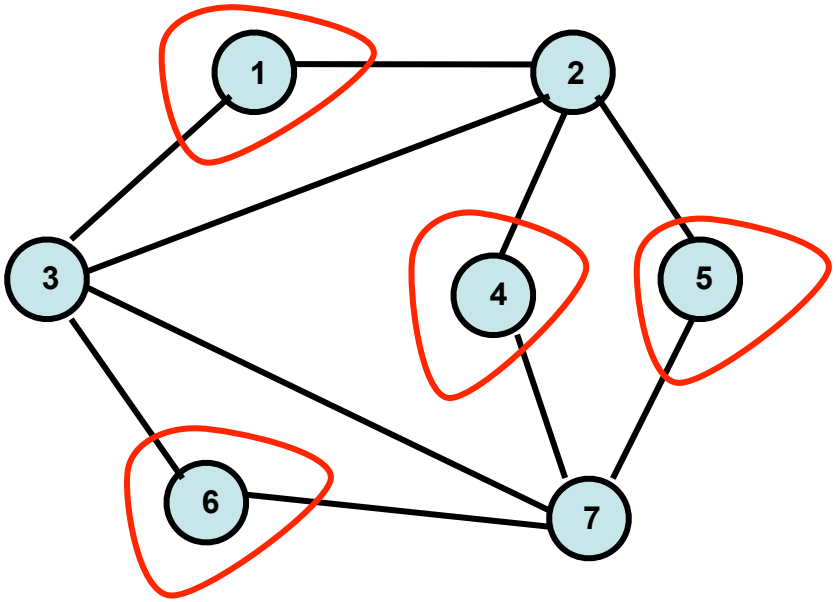
1-vertex cover



3-vertex cover

IndependentSET

Dato un grafo G, un k-independentSet è un sottoinsieme di vertici nessuna coppia dei quali è collegata da un arco



un 4-IndependentSet

IndependentSET $\leq_m$ VERTEX-COVER

IndependentSET
istanze: <G,k>
istanze si: <G,k> tali che G ha un independentSet di k vertici

VERTEXCOVER
istanze: <G,m>
istanze si: <G,m> tali che G ha un vertex-cover di m vertici

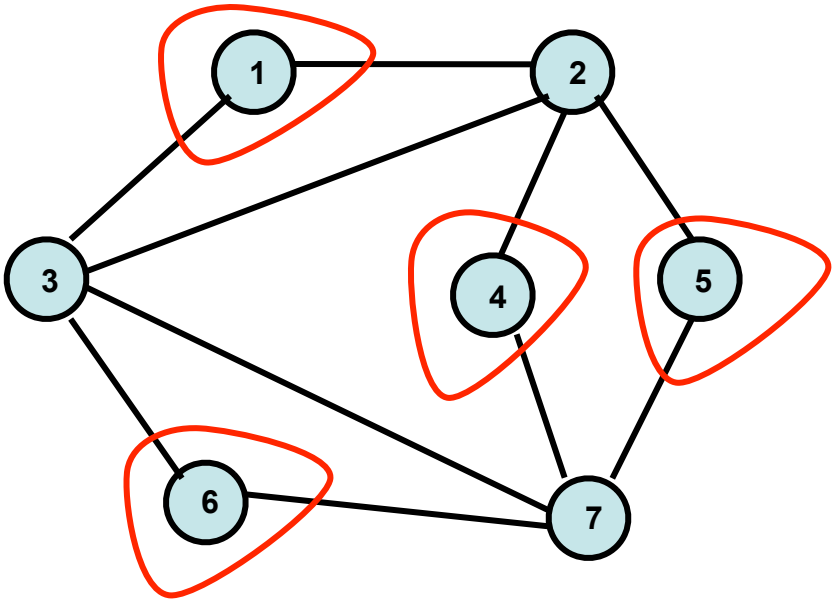
Faremo vedere che, per ogni grafo G=(V,E),

G ha un k-independentSet sse G ha un (|V|-k)-vertex-cover

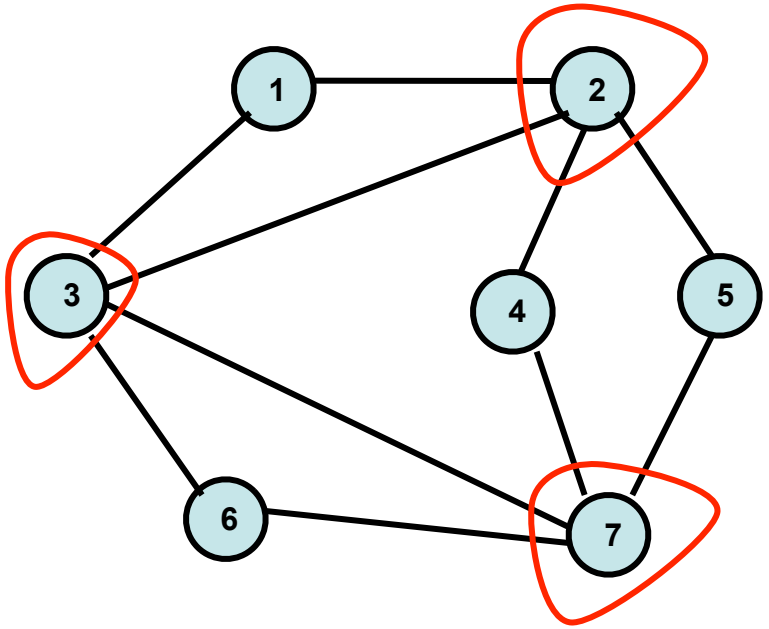
Vedi cap 8 Algorithm Design di Kleinberg-Tardòs

IndependentSET $\leq_m$ VERTEX-COVER

Per ogni grafo G=(V,E), G ha un k-independentSet sse G ha un (|V|-k)-vertex-cover



un 4-IndependentSet



un 3-vertexCover

IndependentSET $\leq_m$ VERTEX-COVER

S è un independentSet di k vertici sse il suo complemento V-S è un (|V|-k)-vertex-cover.

Prova.

⇒

S è un independentSet di k vertici ,

se prendo un arco {u,v} allora u e v non possono essere entrambi in S, quindi l'arco è coperto da un vertice in V-S

⇐

C è un (|V|-k)-vertex-cover, prendiamo due nodi u e v in V-C.

Se u e v fossero connessi da un arco, sarebbe un arco non coperto da C,

quindi V-C è un independentSet di k elementi

IndependentSET $\leq_m$ VERTEX-COVER

G ha un k-independentSet sse G ha un (|V|-k)-vertex-cover

La riduzione?

Implicitamente abbiamo associato a ogni istanza <G=(V,E),k> del problema dell'IndependentSET un'istanza <G'=(V',E'),m> del problema del VERTEXCOVER così definita:

G'=G e m=|V|-k

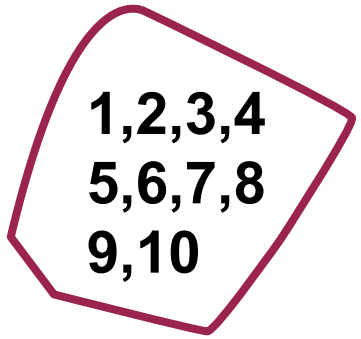
Questa riduzione è ovviamente calcolabile

SET-COVER

Un SET-COVER è una famiglia di sottoinsiemi di un insieme la cui unione è tutto l'insieme.

Problema: Dato un insieme U, una famiglia di suoi sottoinsiemi $S_1,\ldots, S_n$ e un intero k, vogliamo sapere se c'è una famiglia di $k \leq n$ sottoinsiemi la cui unione è U.

Esempio:



S_1 = 1,2

S_2 = 2,3, 4,5

S_3 = 4,7

S_4 = 5,8

S_5 = 6,7, 8,10

S_7 = 9,10

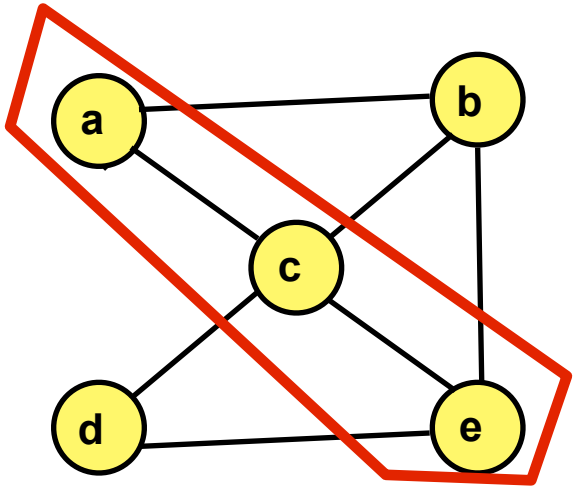
S_6 = 1,3 6,9

k = 3

$\{S_2, S_5, S_6\}$ è un 3-set-cover.

VERTEX-COVER $\leq_m$ SETCOVER

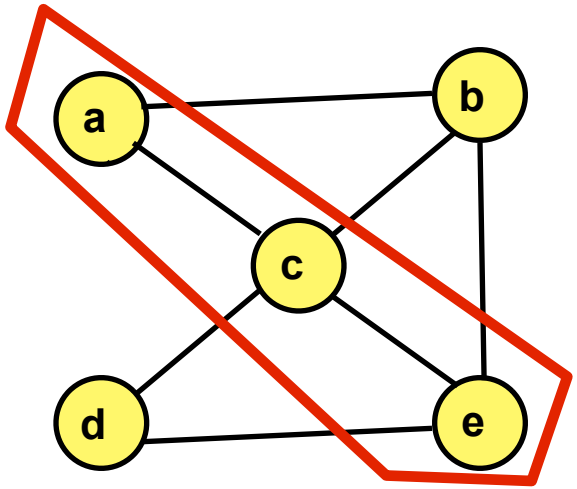
Un vertex cover in un grafo non diretto è un sottoinsieme dei vertici con la proprietà che ogni arco di G ne tocca uno. Un k-vertex cover è un vertex cover con k nodi.



3-vertex cover

Notiamo che se si prende per ogni vertice l'insieme degli archi incidenti su quel vertice e poi si prende l'unione degli insiemi determinati dai vertici in un vertex cover si ottiene tutto E!

VERTEX-COVER $\leq_m$ SETCOVER



k = 3

3-vertex cover

U = E = {{a,b},{a,c},{b,c},{b,e},
{c,d},{c,e},{d,e}}

S_a = {{a,b},{a,c}}

S_b = {{a,b},{b,c},{b,e}}

S_c = {{a,c},{b,c},{c,d},{c,e}}

S_d = {{c,d},{c,d}}

S_e = {{b,e},{c,e},{d,e}}

k = 3

$S_a \cup S_c \cup S_e = E$

3-SETCOVER

Vedi cap 8 Algorithm Design di Kleinberg-Tardòs

VERTEX-COVER $\leq_m$ SETCOVER

Data un'istanza del problema del VERTEX-COVER <G=(V,E),k>, costruiamo un'istanza del problema del SETCOVER <U,F,m> così definita:

U=E, F = {S_u | u è in V} dove S_u={{u,v} | {u,v} è in E}, l'insieme degli archi incidenti u, e m=k.

dimostreremo che

<G=(V,E),k> è un istanza sì per il VERTEX-COVER

sse

< E,{S_u | u è in V},k > lo è per SETCOVER

VERTEX-COVER $\leq_m$ SETCOVER

<G=(V,E),k> ha un k-vertex-cover sse

< E,{S_u | u è in V},k > ha un k-set-cover.

⇒

Sia C = {u_1,...,u_k} un k-vertex-cover per G, allora D= {S_u | u è in C} è un k-set-cover per E,

perchè preso un qualsiasi arco {u,v} uno dei due vertici è in C, ma allora l'insieme dei suoi archi incidenti è in D e quindi nell'unione degli elementi di D.

⇐

Sia D= {S_{u_1},...,S_{u_k} } un k-set-cover per E,

allora ogni arco di G è in uno degli insiemi di D, e quindi incidente su uno dei vertici di C = {u_1,...,u_k} che quindi è un k-vertex-cover per G

$$\text{IndependentSET} \leq_m \text{VERTEX-COVER}$$

e

$$\text{VERTEX-COVER} \leq_m \text{SETCOVER}$$

quindi

$$\text{IndependentSET} \leq_m \text{SETCOVER}$$

una riduzione sbagliata

Si consideri il problema

CLIQUE/INDSET= {<G,k,m> | G è un grafo non diretto che ha un k-clique o un independent set di m vertici}

(k-clique= sottografo indotto completo di k vertici)

si vuole dimostrare che

$$\text{IndependentSET} \leq_m \text{CLIQUE/INDSET}$$

A <G=(V,E),k> associamo <G=(V,E),k,k>

Prova (sbagliata!) che la riduzione è corretta: Se G ha un k independentSet allora ovviamente anche G stesso l'ha e viceversa.

Una correzione

Si consideri il problema

CLIQUE/INDSET= {<G,k,m> | G è un grafo non diretto che ha un k-clique o un independent set di m vertici}

(k-clique= sottografo indotto completo di k vertici)

si vuole dimostrare che

$$\text{IndependentSET} \leq_m \text{CLIQUE/INDSET}$$

A <G=(V,E),k> associamo <G=(V,E),|V|+1,k>

Prova che la riduzione è corretta: Se G ha un k independentSet allora ovviamente anche G stesso l'ha e viceversa. Nessun grafo può avere un clique con un numero di vertici superiore a quelli che ha.