Il concetto di calcolatore e di algoritmo

Elementi di Informatica e Programmazione

Percorso di Preparazione agli Studi di Ingegneria

Università degli Studi di Brescia

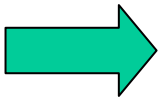
Docente: Massimiliano Giacomin

Informatica e calcolatore

- Informatica: studio sistematico di algoritmi

- Algoritmi "risolvono problemi"
  (per esempio: dato un numero, calcolarne la radice quadrata)

- Algoritmi: destinati ad essere eseguiti da un calcolatore
  (opportunamente espressi in un linguaggio di programmazione)



L'informatica può anche dirsi la scienza dei calcolatori (in una accezione più ampia di quella cui si è abituati)

Per capire bene cosa si intende, bisogna capire cosa si intende per "problema"…

Problema

Classe di domande omogenee

alle quali è possibile dare risposta mediante una *procedura uniforme*

Esempio di problema:

"Quanto vale la radice quadrata intera di un numero intero positivo X?"

Istanza di un problema

Una *specifica domanda* del problema

Esempio:

"Quanto vale la radice quadrata intera di 49?"

Soluzione di una istanza

(detta anche "soluzione attesa"):

la *risposta alla specifica domanda* che l'istanza rappresenta

Esempio (nel caso precedente):

7

Nota bene: problema vs. istanza

"Quanto vale la radice quadrata intera di 49?" non è un problema,

ma una istanza del problema

"Quanto vale la radice intera di un numero intero positivo X?", con X=49

Nota bene: la definizione del problema è data

Dobbiamo conoscere la "soluzione attesa" di tutte le istanze del problema: queste "fanno parte" della descrizione stessa del problema (nel senso che sono deducibili da essa)

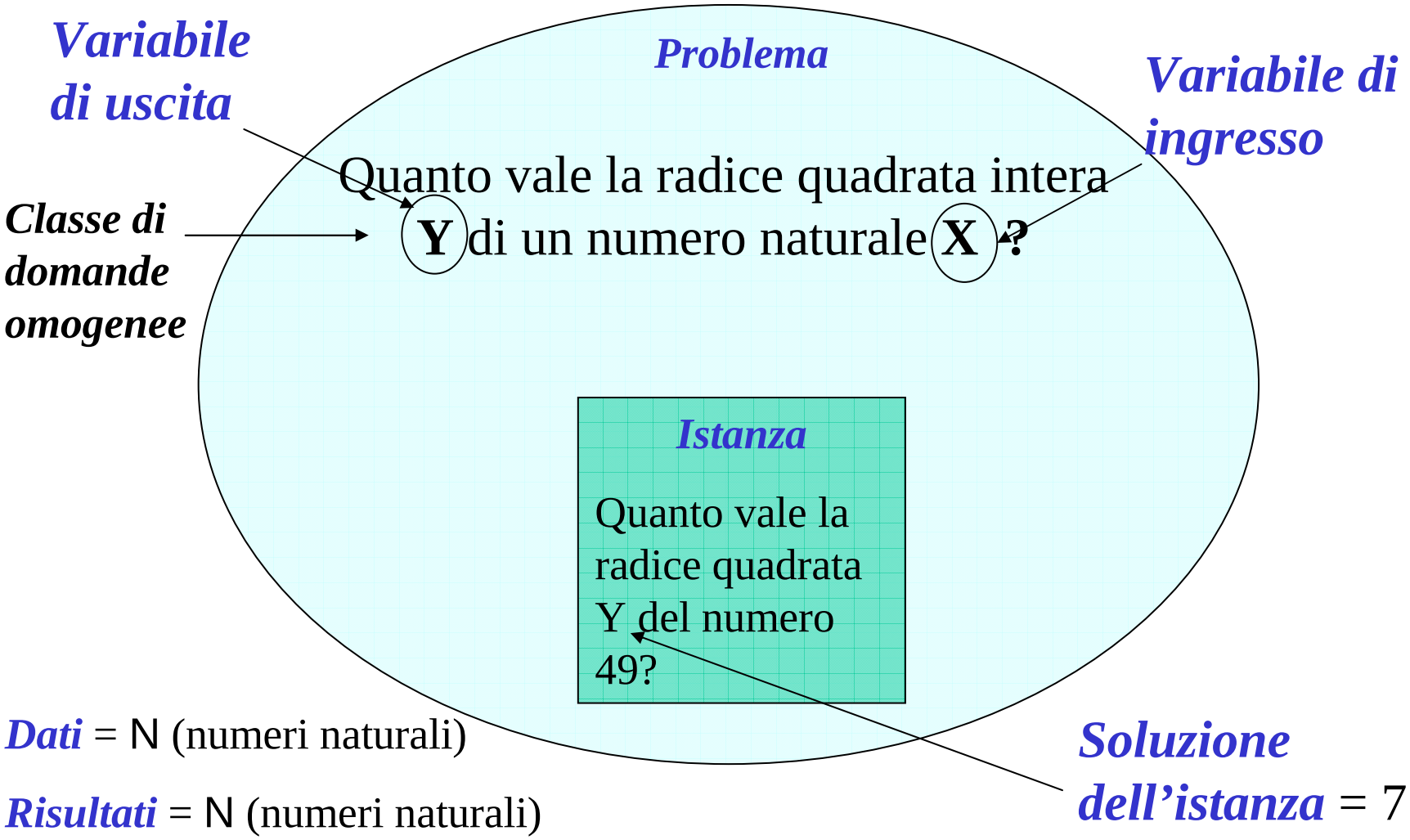
Definizione di problemi e istanze

Un problema è formulato mediante una o più ***variabili*:**

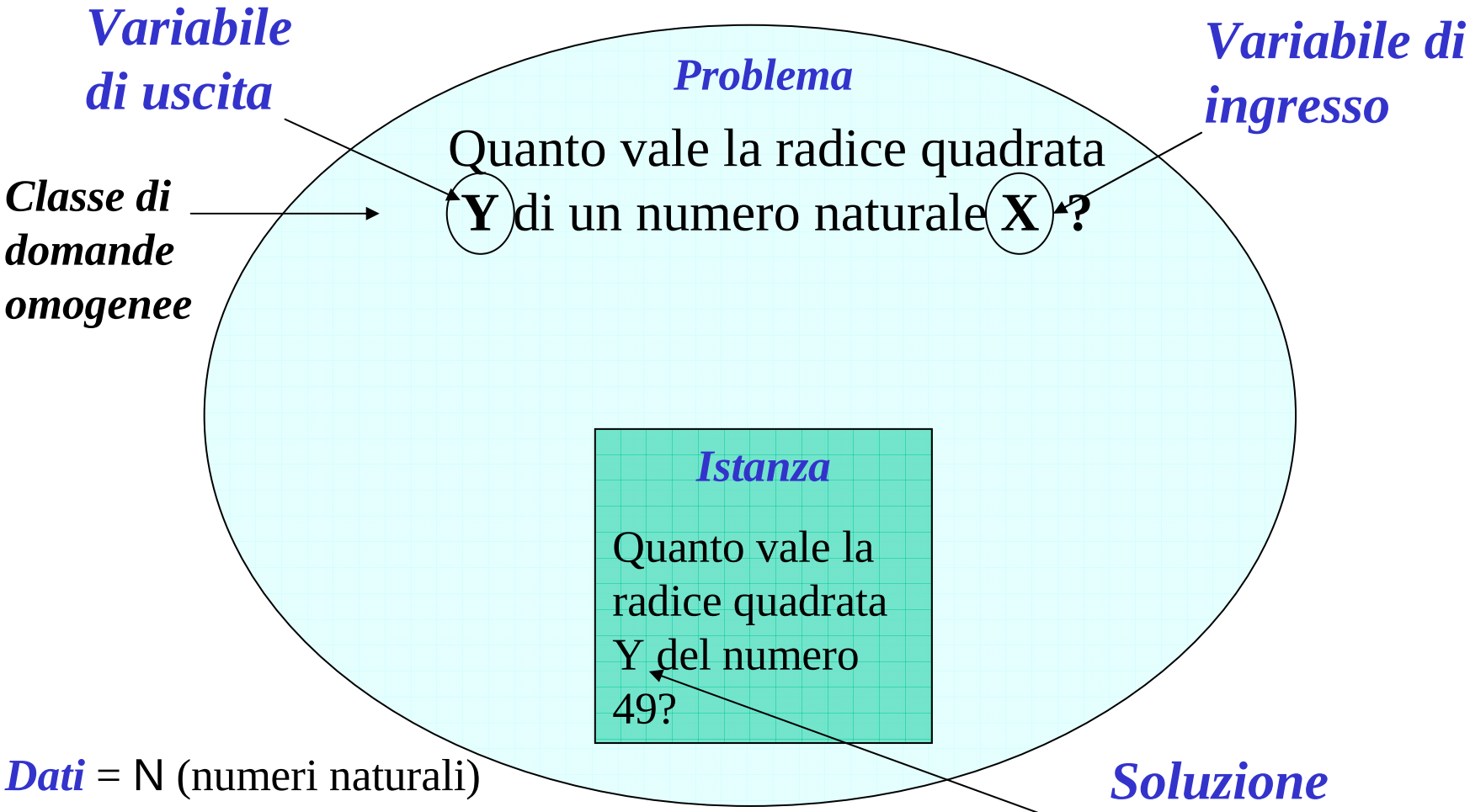
- ***Variabili di ingresso***
  - termini variabili che, assumendo un valore del dominio, permettono di generare le istanze del problema
  - i valori che esse assumono (elementi del loro dominio) vengono chiamati ***dati*** (o, più esplicitamente, dati di ingresso)

- ***Variabili di uscita***
  - termini variabili che caratterizzano le *soluzioni attese* (delle istanze) di un problema
  - i valori che esse assumono vengono chiamati ***risultati***

Esempio

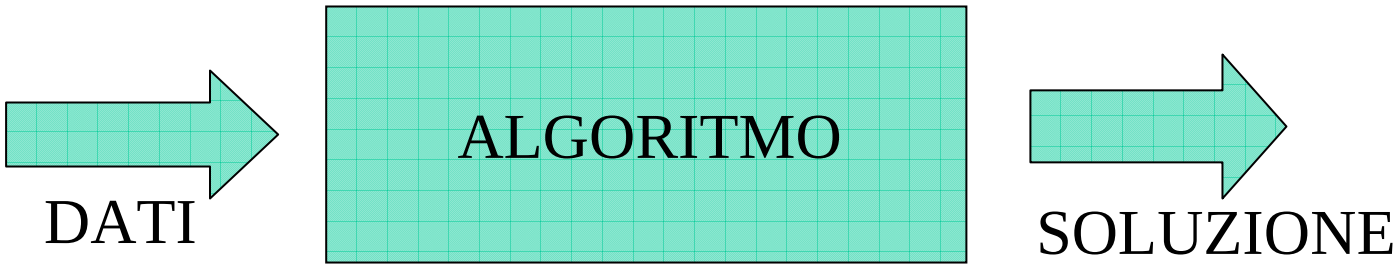

Variabile di uscita
Problema
Variabile di ingresso
Quanto vale la radice quadrata intera
Classe di domande omogenee
Y di un numero naturale X ?
Istanza
Quanto vale la radice quadrata Y del numero 49?
Dati = N (numeri naturali)
Risultati = N (numeri naturali)
Soluzione dell'istanza = 7

Esempio 2


Variabile di uscita
Variabile di ingresso
Problema
Quanto vale la radice quadrata Y di un numero naturale X ?
Classe di domande omogenee
Istanza
Quanto vale la radice quadrata Y del numero 49?
Dati = N (numeri naturali)
Risultati = R (numeri reali)
Soluzione dell'istanza = 7

Il concetto di algoritmo

- Algoritmo risolvente di un problema: metodo che specifica come produrre in modo uniforme una soluzione per ogni possibile istanza mediante una sequenza di ***istruzioni***



Esempio: il problema della radice quadrata intera

- Quale potrebbe essere un algoritmo risolvente?

Alcune istanze (e soluzioni)

5:	2*2 = 4, 3*3 = 9	Y = 2
9:	2*2 = 4, 3*3 = 9	Y = 3
1:	1*1 = 1, 2*2 = 4	Y = 1
10:	2*2 = 4, 3*3 = 9, 4*4 = 16	Y = 3

Un'idea:

Parto da Y=1, controllo se $Y^2 < X$…
se sì allora Y=Y+1 e continuo,
altrimenti significa che $Y^2 \geq X$:
se $Y^2 = X$ ok,
altrimenti la soluzione deve essere Y-1: Y = Y-1

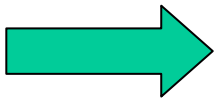
Un algoritmo:

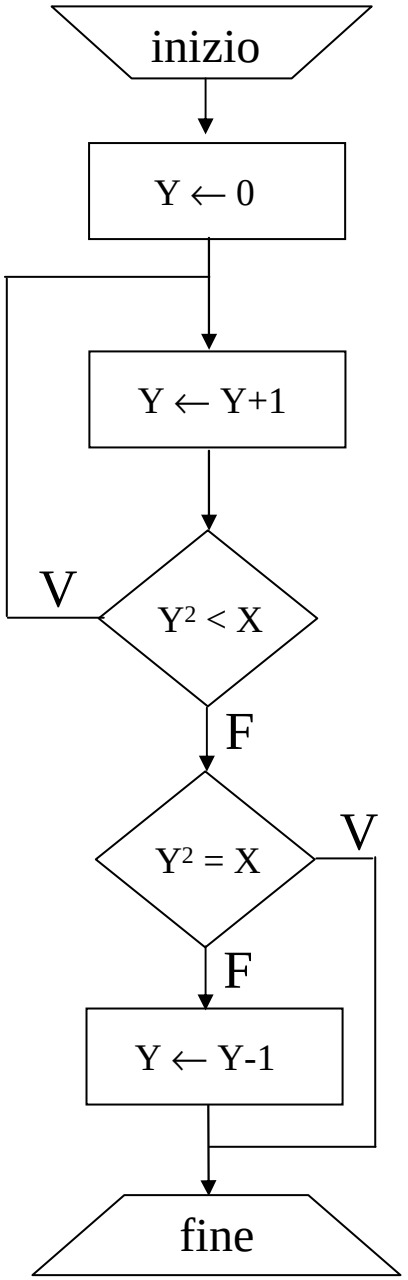
1. Assegna a Y il valore 0
2. Incrementa Y di 1
3. Se $Y^2 < X$: torna all'istruzione 2
4. Se $Y^2 = X$: FINE
5. Se $Y^2 > X$: decrementa Y di 1 : FINE

Per esprimere questo algoritmo (ed altri) in una forma più precisa e più evidente dal punto di vista visivo, si può usare il formalismo degli ***schemi a blocchi***, detti anche "***diagrammi di flusso***" o "***flowchart***": linguaggio *grafico* i cui elementi sono:

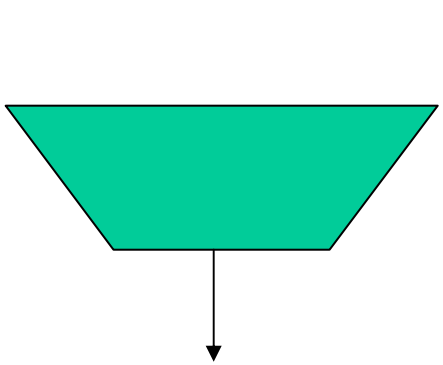
- ***nodi*** (o ***blocchi***) per indicare le istruzioni:
  inizio e fine, elaborazioni, decisioni sulla base di condizioni
- ***archi orientati*** che collegano coppie di nodi, per indicare l'ordine tra i nodi

1. Assegna a Y il valore 0
2. Incrementa Y di 1
3. Se $Y^2 < X$: torna all'istruzione 2
4. Se $Y^2 = X$: FINE
5. Se $Y^2 > X$: decrementa Y di 1 : FINE

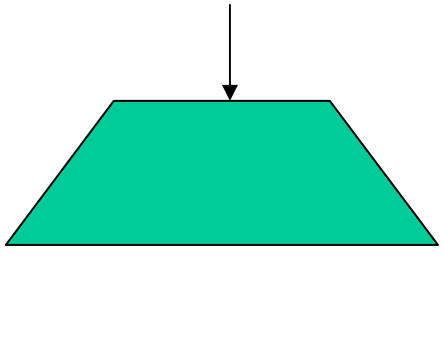




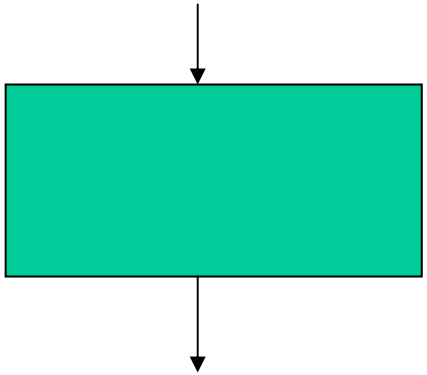
La simbologia comunemente utilizzata



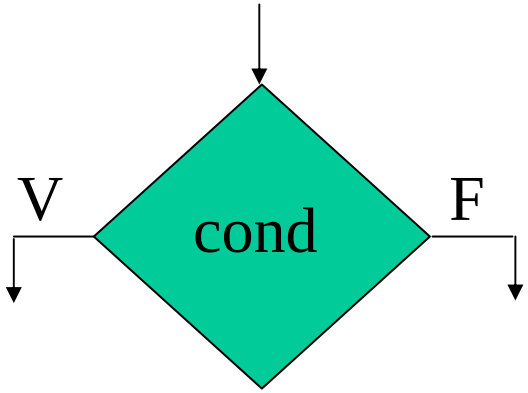
inizio



fine



blocco funzionale



Blocco decisionale (selezione a 2 vie)

NB: nel libro inizio e fine sono indicati sempre con un rettangolo

Variabili, espressioni e assegnamenti

- ***Variabile***: "contenitore" di dati caratterizzata da un ***nome*** es. *x*
- Ad una variabile può essere assegnato un ***valore***: p.es.

  $x \leftarrow 10$

- Le variabili possono comparire in ***espressioni aritmetiche*** (es. *x*-*y*) o ***logiche***, che restituiscono un valore
- Le espressioni possono essere assegnate ad altre variabili: p.es.

  $d \leftarrow x\text{-}y$

  d assume il valore dell'espressione *x*-*y*: il precedente valore di *d* viene sovrascritto
- Altro esempio, supponendo che prima dell'istruzione x=10, y=8

  $x \leftarrow x\text{-}y$

  x assume il valore 2

Esecuzione passo passo

Supponendo che la variabile in ingresso X = 8

1 Y ← 0
2 Calcolo di Y+1 e risultato in Y ➔ Y =1
3 Controllo se $Y^2 < X$ ➔ è vero
4 Calcolo di Y+1 e risultato in Y ➔ Y =2
5 Controllo se $Y^2 < X$ ➔ è vero
6 Calcolo di Y+1 e risultato in Y ➔ Y =3
7 Controllo se $Y^2 < X$ ➔ è falso
8 Controllo se $Y^2 = X$ ➔ è falso
9 Calcolo di Y-1 e risultato in Y ➔ Y =2
10 Fine

PROBLEMA, ISTANZA E ALGORITMO (formalmente parlando)

Problema

$P[X, Y]$

- $(X_1, X_2, \ldots, X_n)$: variabili di ingresso, con domini $D_1, D_2, \ldots, D_n$ (insiemi di dati)
- $(Y_1, Y_2, \ldots, Y_m)$: variabili di uscita, con domini $R_1, R_2, \ldots, R_m$ (insiemi di risultati)

Istanza del problema

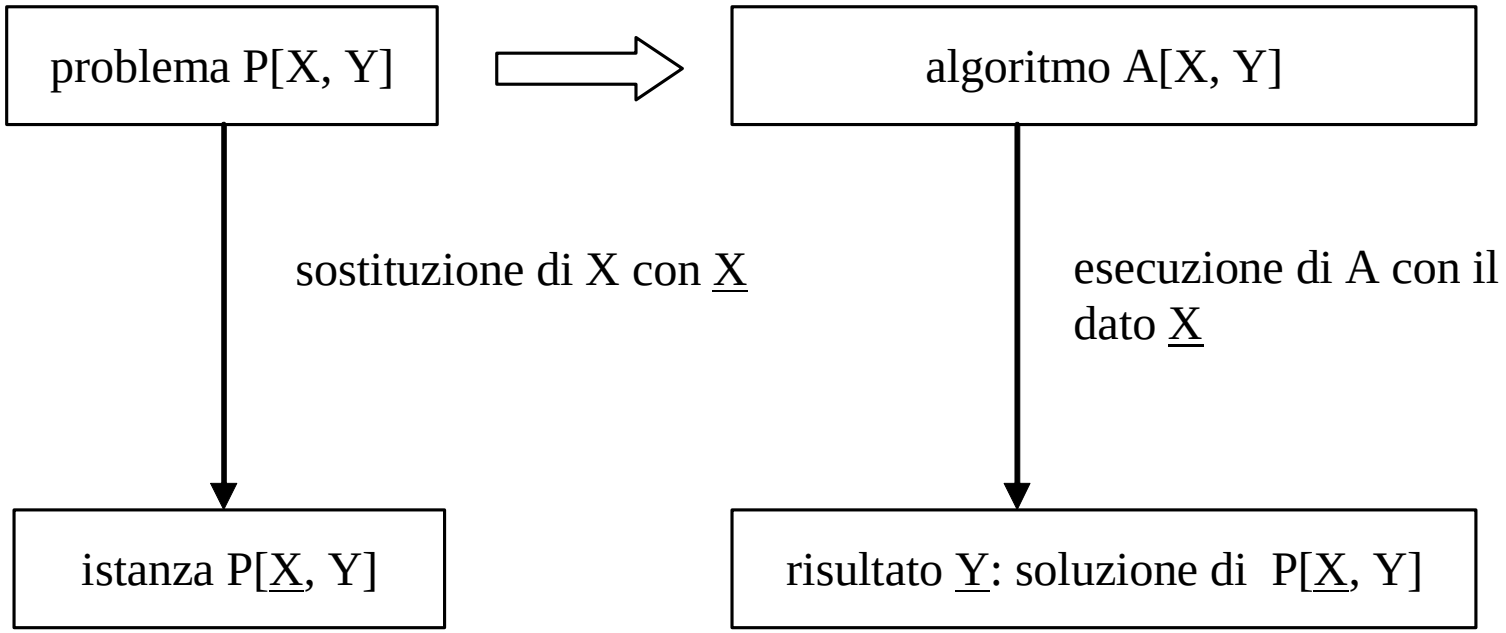
$P[\underline{X}, Y]$ con $\underline{X} \in D = D_1 \times D_2 \times \ldots \times D_n$

Algoritmo risolvente del problema

$A[X, Y]$ tale che

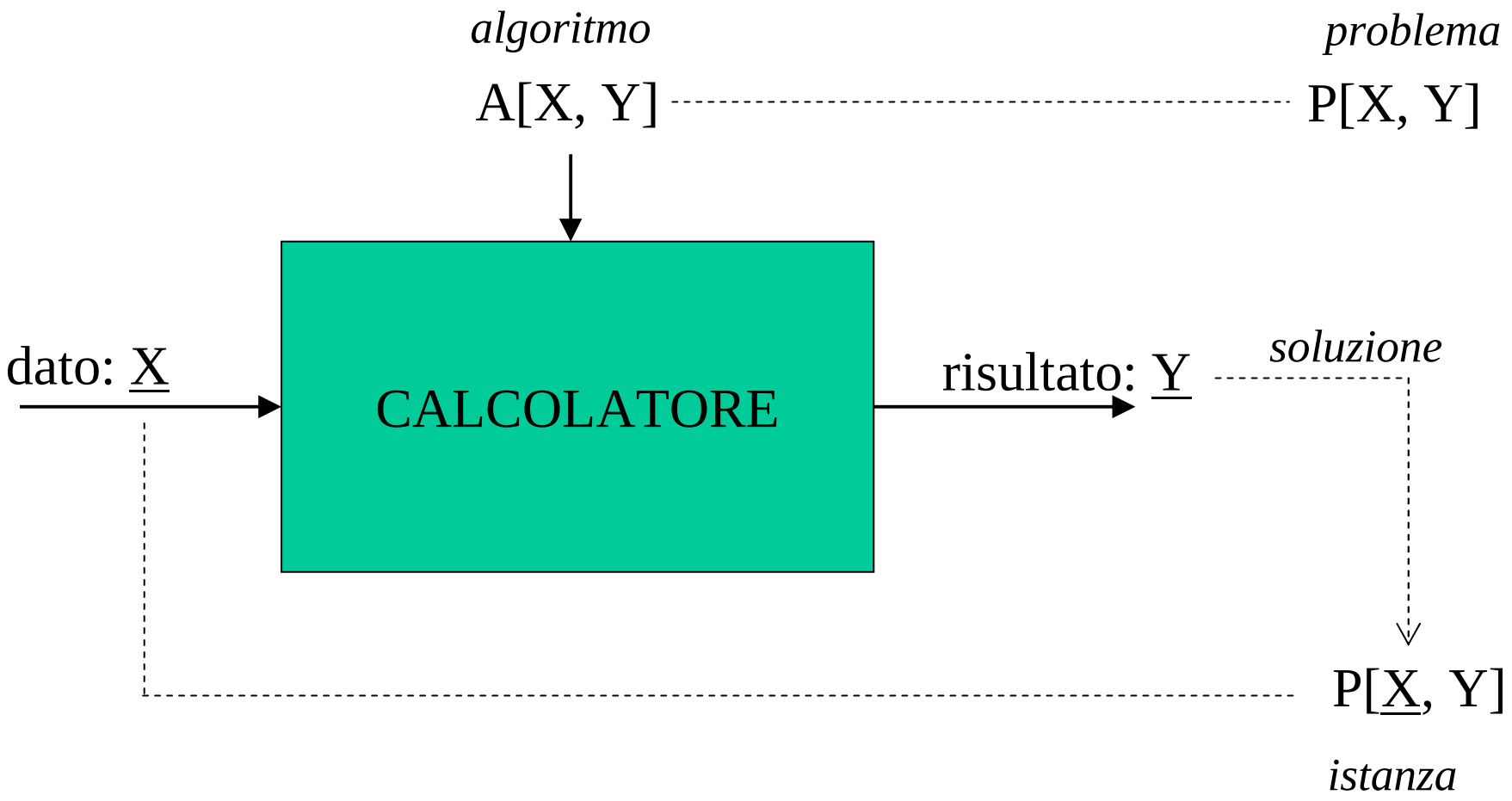
eseguito con il dato $\underline{X}$ produce $\underline{Y}$ soluzione dell'istanza $P[\underline{X}, Y]$

($\underline{Y} \in R = R_1 \times R_2 \times \ldots \times R_m$)


problema P[X, Y]
algoritmo A[X, Y]
sostituzione di X con X
esecuzione di A con il dato X
istanza P[X, Y]
risultato Y: soluzione di P[X, Y]

Il concetto di calcolatore

come esecutore universale di algoritmi



- Un calcolatore è un sistema che, ricevendo in ingresso la descrizione, in un opportuno linguaggio, di un algoritmo risolvente A[X,Y] per un certo problema P[X,Y] e un dato X, produce come risultato la soluzione Y dell'istanza P[X,Y]
- Per essere comprensibili al calcolatore, gli algoritmi devono essere espressi in un ***linguaggio di programmazione***:

  Programma = algoritmo descritto formalmente attraverso un linguaggio di programmazione
- Un calcolatore è quindi un ***esecutore universale di programmi***

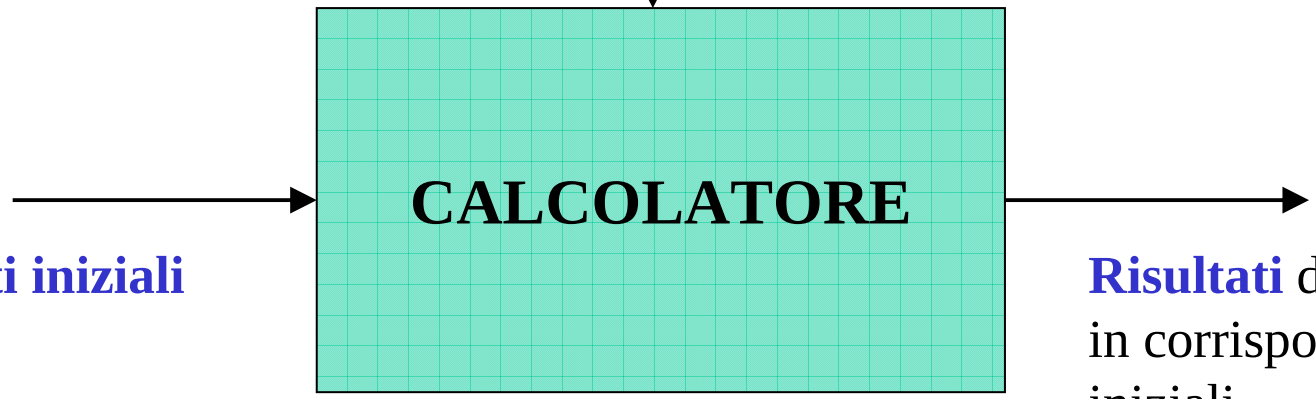
  - elabora puri simboli (per esso "privi di significato")
  - non risolve problemi (il problema non è un suo ingresso)

  ma esegue programmi!

Il calcolatore come esecutore di programmi

***Programma*:**

sequenza di istruzioni di un linguaggio di programmazione (descrive un algoritmo)



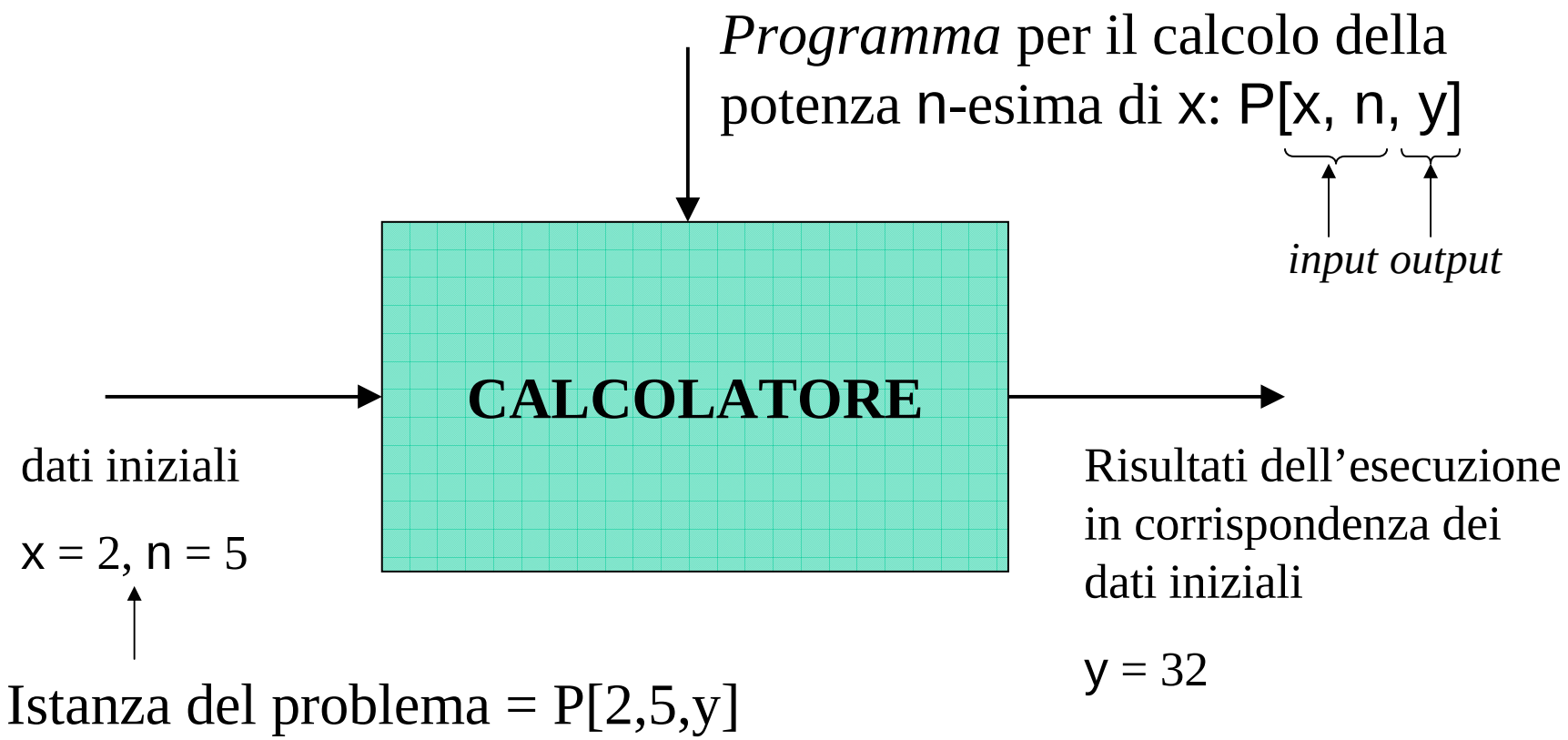
Dati iniziali

Assegnati a **variabili di ingresso (input)**

Risultati dell'esecuzione in corrispondenza dei dati iniziali

Assegnati a **variabili di uscita (output)**

Esempio


Programma per il calcolo della potenza n-esima di x: P[x, n, y]
input output
CALCOLATORE
dati iniziali
x = 2, n = 5
Istanza del problema = P[2,5,y]
Risultati dell'esecuzione in corrispondenza dei dati iniziali
y = 32

Proprietà di un algoritmo

La definizione di algoritmo presuppone che esso possa essere espresso in termini linguistici ben definiti, interpretato ed eseguito da un soggetto esecutore (il calcolatore). Da ciò conseguono le seguenti proprietà:

- ***Finitezza***: un algoritmo deve essere costituito da un numero finito di istruzioni
- ***Definitezza***: le istruzioni di cui un algoritmo è costituito devono appartenere a un insieme finito e prefissato di tipi elementari
- ***Univocità***: ogni istruzione deve essere univocamente interpretabile ed eseguibile
- ***Effettività***: deve esistere un esecutore in grado di eseguire ogni istruzione dell'algoritmo in un tempo finito

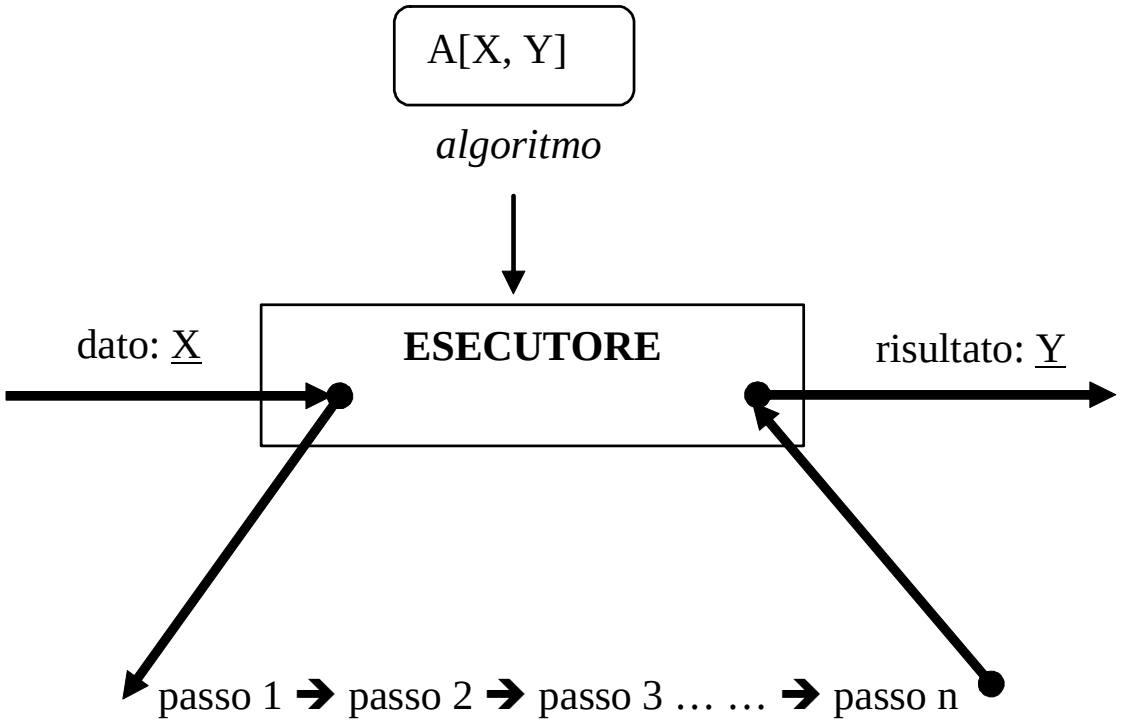
Computazione

- ***Computazione***: esecuzione di un algoritmo da parte del calcolatore in corrispondenza di certi dati in ingresso
- ***Passo di computazione***: insieme delle azioni che l'esecutore compie (durante l'esecuzione di un algoritmo) per eseguire una singola istruzione
- ***Sequenza di computazione***: sequenza di passi di computazione che l'esecutore compie in corrispondenza di certi dati in ingresso durante l'esecuzione di un algoritmo

***Algoritmo* = concetto statico**

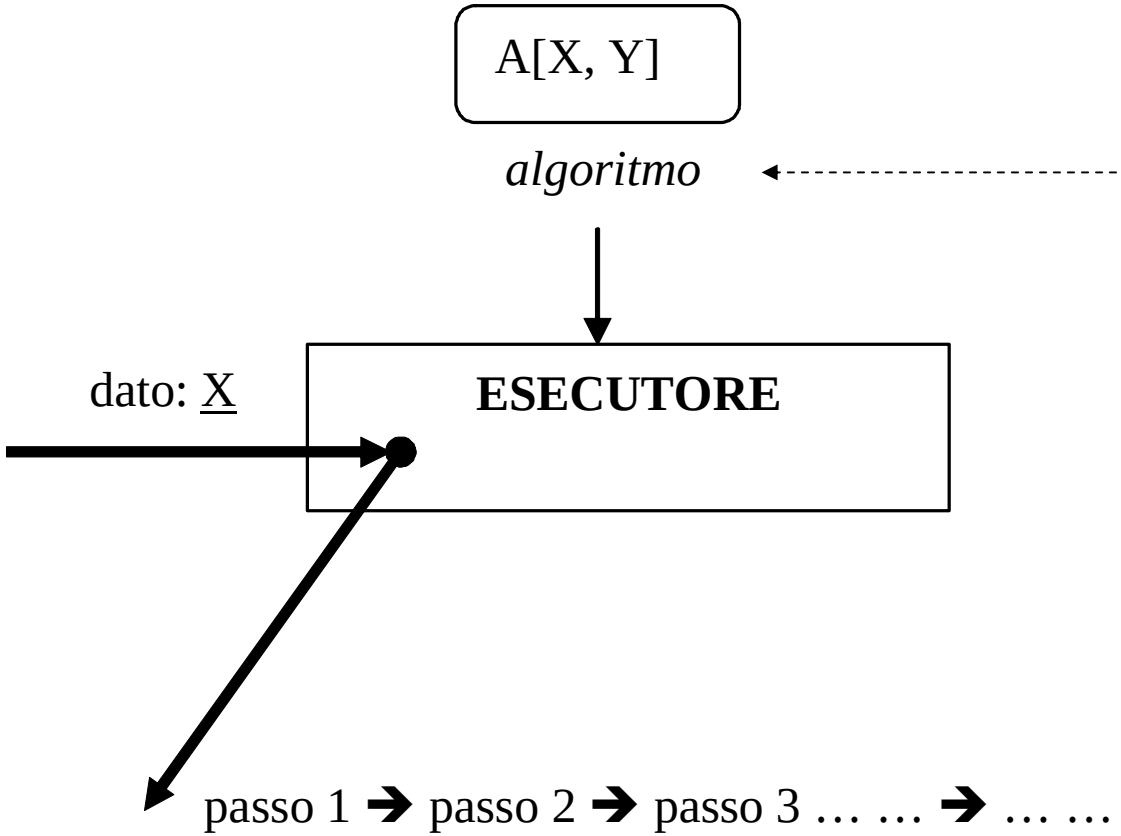
***Computazione* = concetto dinamico**

Sequenza di computazione finita



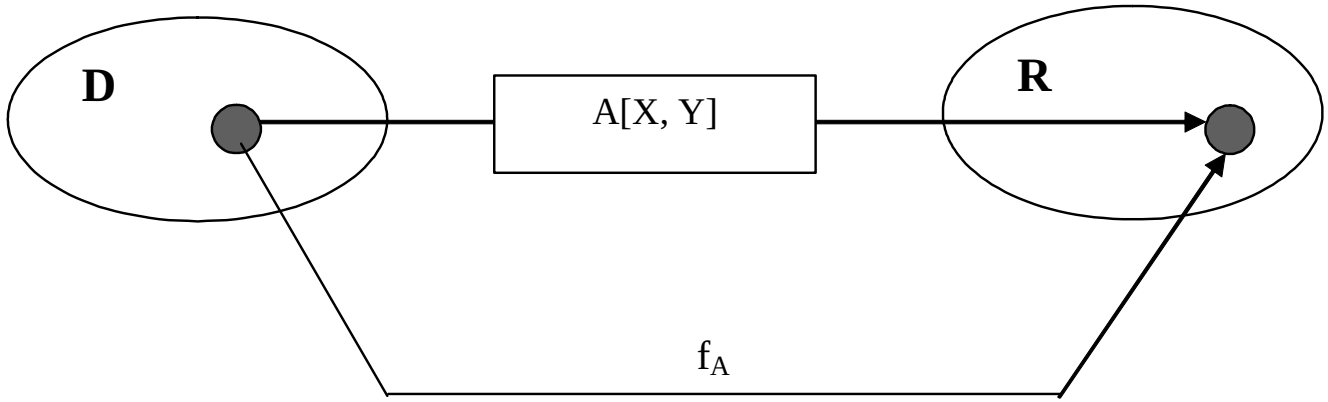
In questo caso, la computazione produce un risultato

Sequenza di computazione infinita



In questo caso, il risultato rimane indefinito

Funzione calcolata da un algoritmo



- Un algoritmo A[X, Y] calcola una funzione da D (dominio delle variabili di ingresso) a R (dominio delle variabili di uscita):
  - $f_A: D \rightarrow R$

    tale che $f_A(\underline{X}) = \underline{Y}$

    con $\underline{Y}$ prodotto dalla computazione di A[X, Y] con il dato $\underline{X}$
- La funzione è in generale parziale!

Nota importante

- Un algoritmo risolve un problema (calcola una funzione)
  e ne risolve uno solo (ne calcola solo una)
- Viceversa, per un problema risolubile (ovvero, se esiste un algoritmo che lo risolve) esistono infiniti algoritmi che lo risolvono!
  - infatti, un algoritmo è descritto da una sequenza di istruzioni
  - per convincersene, è sufficiente pensare che possiamo sempre aggiungere sequenze di istruzioni che non hanno effetto sul risultato
  - e possiamo farlo in infiniti modi (es. sommare e sottrarre 1 a/da una variabile, sommare e sottrarre 2, ecc. ecc.)

L'informatica teorica

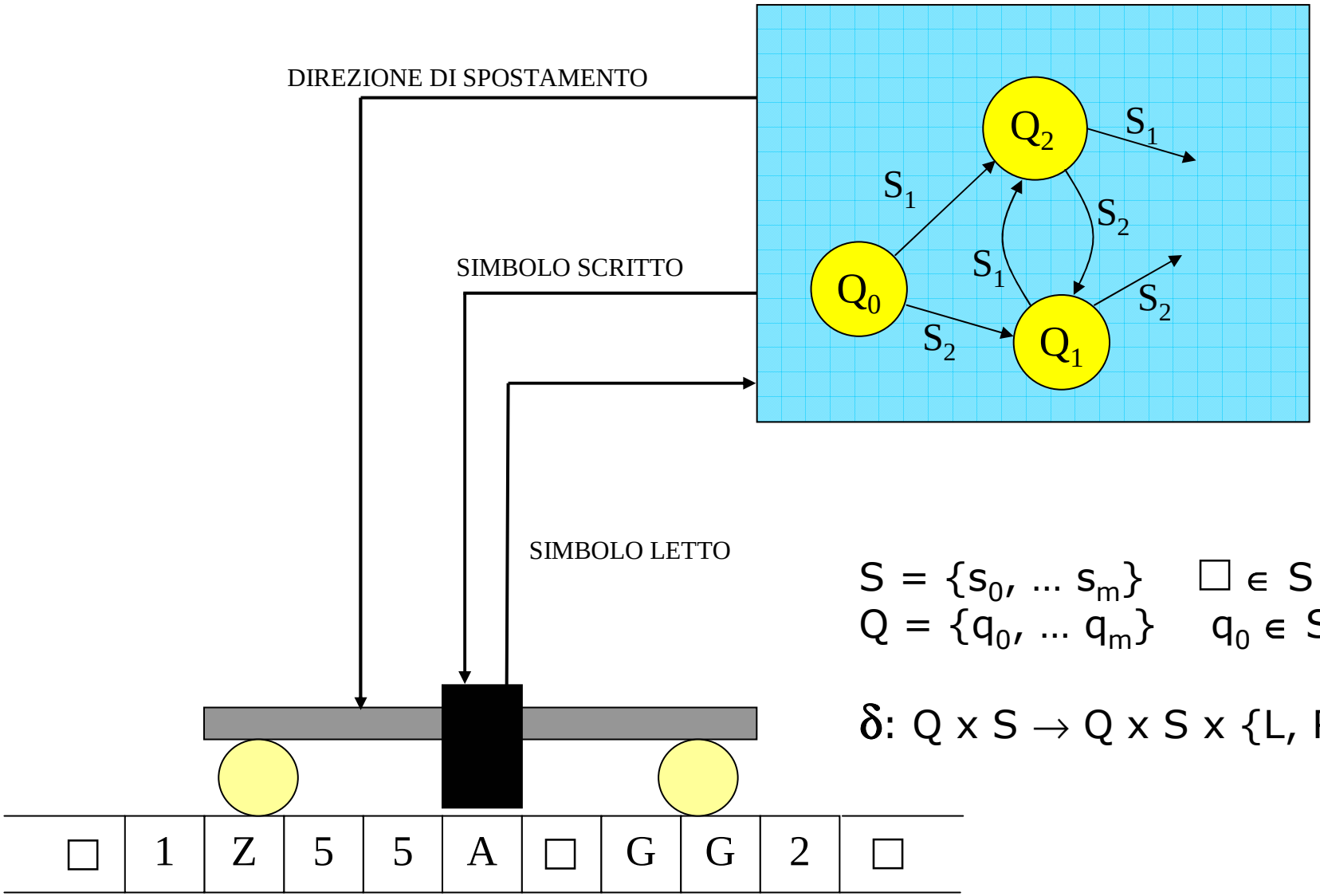
- Utilizzando strumenti matematici, studia **macchine astratte** descritte formalmente, anziché macchine concrete.
- Permette di ottenere risultati su "cosa una macchina è in grado di calcolare" e quindi "quali problemi possono essere risolti" a prescindere dalla tecnologia impiegata per realizzare i calcolatori
- La tecnologia cambia, i risultati generali no:
  - La "macchina analitica" di Charles Babbage (mai realizzata) è stata progettata nel 1830
  - La "macchina di Turing" (di Alan Turing) è stata pubblicata nel 1936
  - Il primo calcolatore elettronico solo nel 1943

Alcuni risultati (cenni):
se qualcuno è interessato, può consultare i capitoli 15, 16, 17

APPROFONDIMENTO

(NON RICHIESTO ALL'ESAME)

La macchina di Turing



$S = \{s_0, \ldots s_m\}$ $\square \in S$

$Q = \{q_0, \ldots q_m\}$ $q_0 \in S$

$\delta: Q \times S \rightarrow Q \times S \times \{L, R\}$

La tesi di Church-Turing

Tutte le funzioni che si possono calcolare sono calcolabili mediante una Macchina di Turing

⇒ Il più potente calcolatore esistente al mondo e tutti i calcolatori che saranno costruiti in futuro sono equivalenti ad una Macchina di Turing

Alcuni risultati formali

- Le macchine di Turing sono numerabili
  ⇒ tutti i possibili programmi (infiniti) si possono contare!
- Esistono problemi (funzioni) non computabili, ovvero per i quali non esistono algoritmi risolventi

- i problemi (le funzioni) esistenti sono non numerabili (non si possono contare) e quindi… sono di più!
- un esempio: non esiste un algoritmo (programma) che, dato un qualunque algoritmo (programma) e un dato, può decidere se esso termina oppure no
- un altro esempio: non possiamo sapere in generale se un fatto è "conseguenza logica" di un insieme di fatti