Introduzione all'Interpretazione Astratta

14 Settembre 2009

Giuseppe Lettieri

g.lettieri@iet.unipi.it

Motivazione

- Scoprire *automaticamente* proprietà delle possibili esecuzioni di un programma

Motivazione

- Scoprire *automaticamente* proprietà delle possibili esecuzioni di un programma
- per verificarne la correttezza, la sicurezza o per ottimizzarlo

Motivazione

- Scoprire *automaticamente* proprietà delle possibili esecuzioni di un programma
- per verificarne la correttezza, la sicurezza o per ottimizzarlo
- in modo *approssimato*, ma *corretto*.

Motivazione

- Scoprire *automaticamente* proprietà delle possibili esecuzioni di un programma
- per verificarne la correttezza, la sicurezza o per ottimizzarlo
- in modo *approssimato*, ma *corretto*.
- **Automaticamente**: vogliamo costruire programmi che analizzano altri programmi.

Motivazione

- Scoprire *automaticamente* proprietà delle possibili esecuzioni di un programma
- per verificarne la correttezza, la sicurezza o per ottimizzarlo
- in modo *approssimato*, ma *corretto*.
- **Automaticamente**: vogliamo costruire programmi che analizzano altri programmi.
- **Approssimazione**: le analisi possono a volte concludersi con la risposta “non lo so” (falsi positivi).

Motivazione

- Scoprire *automaticamente* proprietà delle possibili esecuzioni di un programma
- per verificarne la correttezza, la sicurezza o per ottimizzarlo
- in modo *approssimato*, ma *corretto*.
- **Automaticamente**: vogliamo costruire programmi che analizzano altri programmi.
- **Approssimazione**: le analisi possono a volte concludersi con la risposta “non lo so” (falsi positivi).
- **Correttezza (dell’analisi)**: nessun falso negativo.

Riferimenti

P. Cousot, R. Cousot. *Abstract interpretation: a unified lattice model for static analysis of programs by constraction or approximation of fixpoints*. In 4th Symp. on Principles of Programming Languages, pages 238–252. ACM Press, 1977.

P. Cousot, R. Cousot. *Systematic design of program analysis frameworks*. In 6th Symp. on Principles of Programming Languages, pages 269–262. ACM Press, 1979.

Corso tenuto da P. Cousot al MIT nel 2005: http://web.mit.edu/16.399/www/

Esempio

Consideriamo la seguente istruzione di un programma P:

$$x = a[i + j];$$

Esempio

Consideriamo la seguente istruzione di un programma P:

$$x = a[i + j];$$

Vogliamo sapere se in *tutte* le esecuzioni di P l'indice $i + j$ è *sempre* all'interno dell'array a (dimensione M):

$$0 \leq i + j < M.$$

Esempio

Consideriamo la seguente istruzione di un programma P:

$$x = a[i + j];$$

Vogliamo sapere se in *tutte* le esecuzioni di P l'indice $i + j$ è *sempre* all'interno dell'array a (dimensione M):

$$0 \leq i + j < M.$$

- *tutte* le esecuzioni: qualunque siano gli ingressi.

Esempio

Consideriamo la seguente istruzione di un programma P:

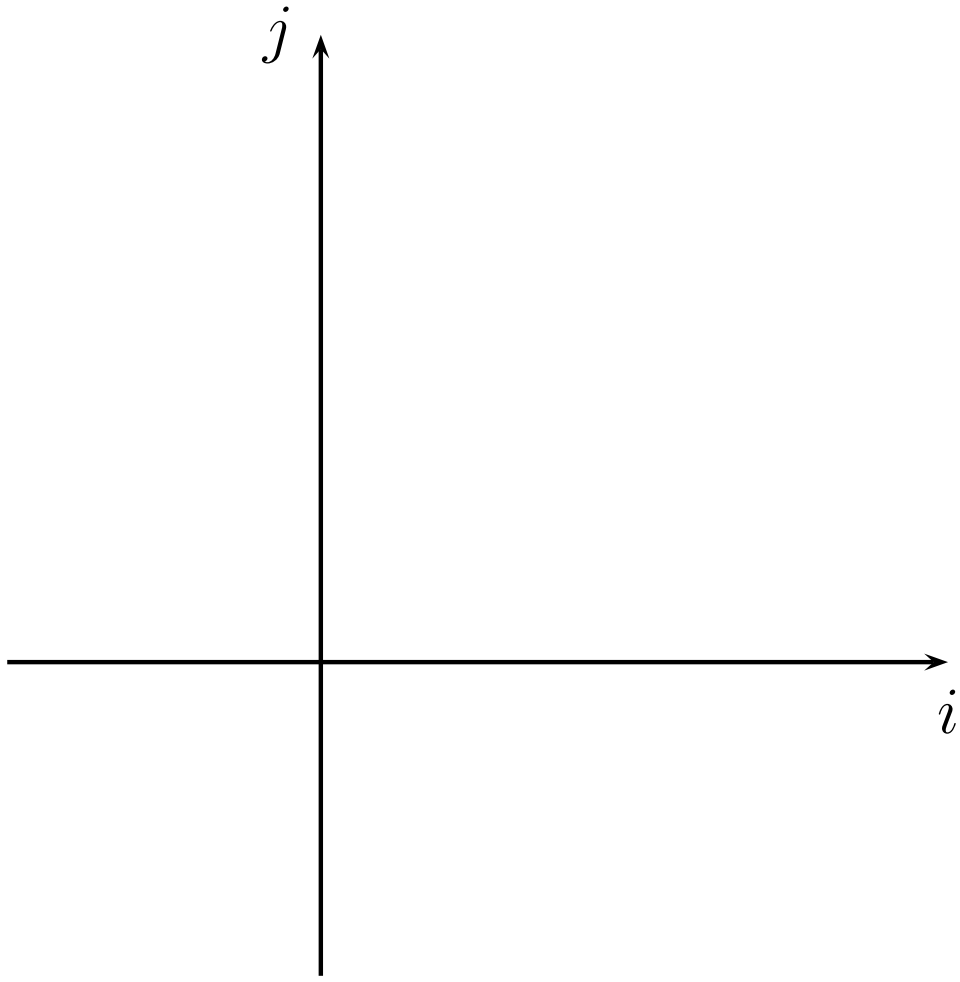
$$x = a[i + j];$$

Vogliamo sapere se in *tutte* le esecuzioni di P l'indice $i + j$ è *sempre* all'interno dell'array a (dimensione M):

$$0 \leq i + j < M.$$

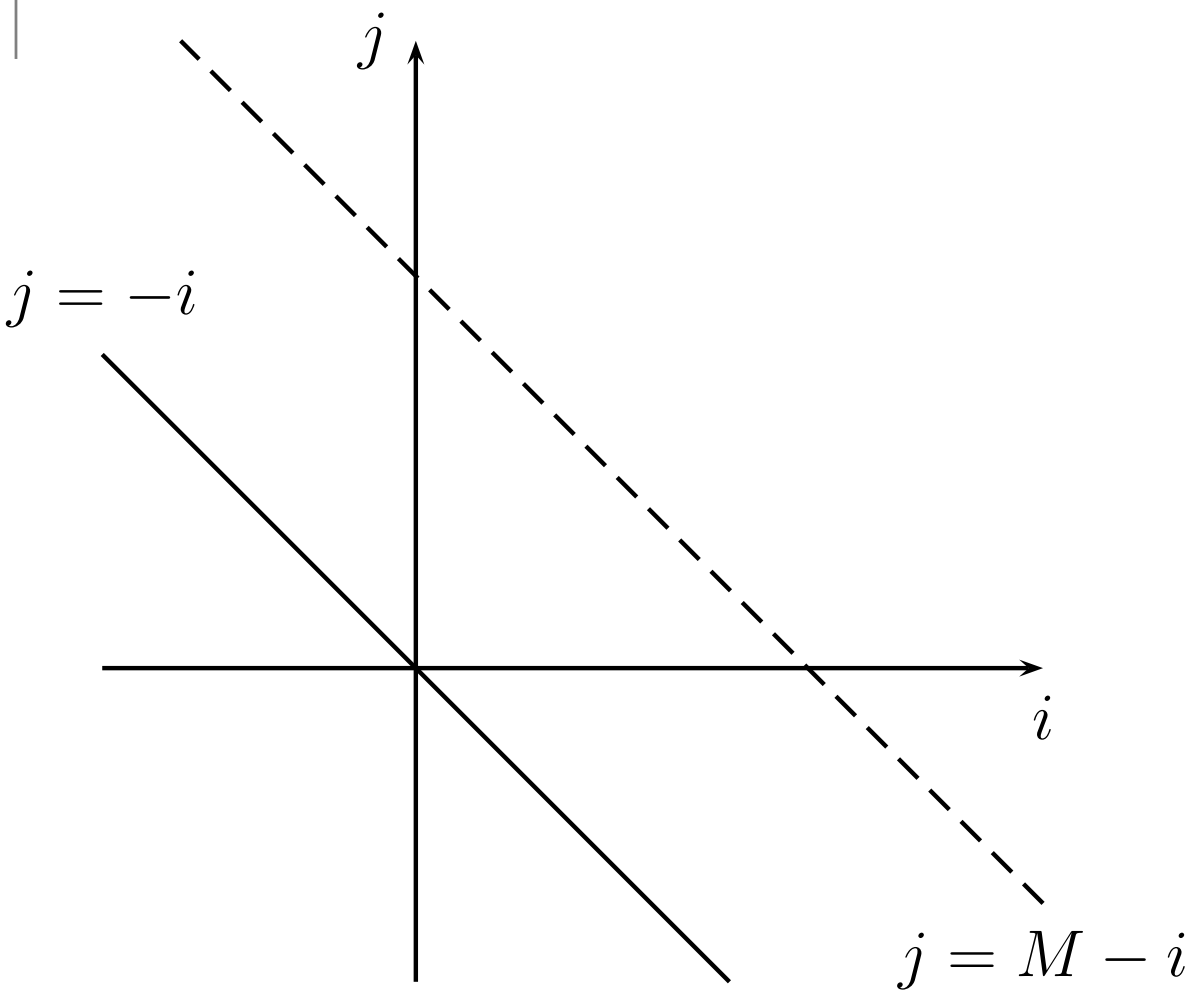
- *tutte* le esecuzioni: qualunque siano gli ingressi.
- *sempre*: ogni volta che l'esecuzione passa per quell'istruzione.

Esempio



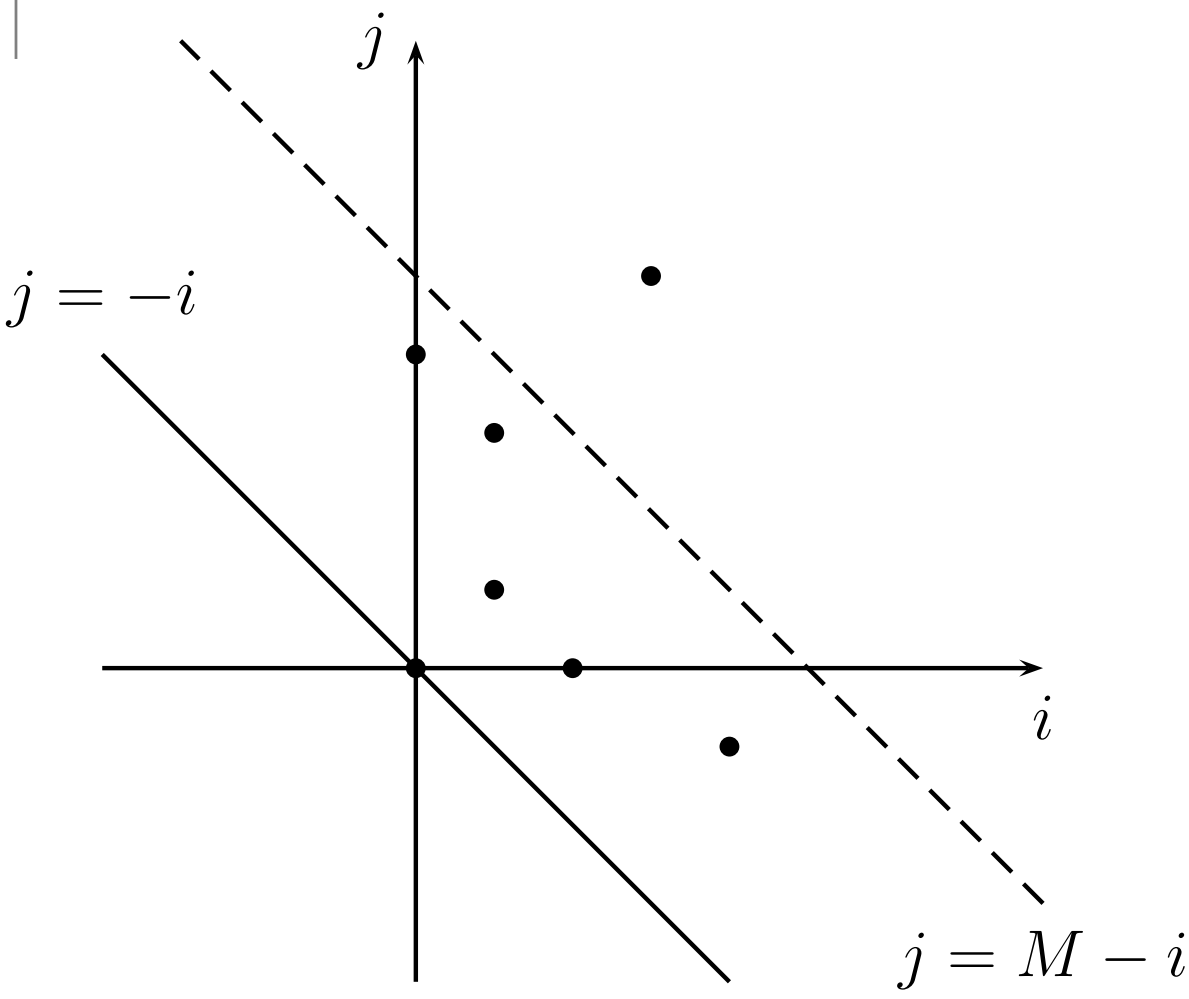
Consideriamo lo spazio delle possibili coppie (i, j)

Esempio



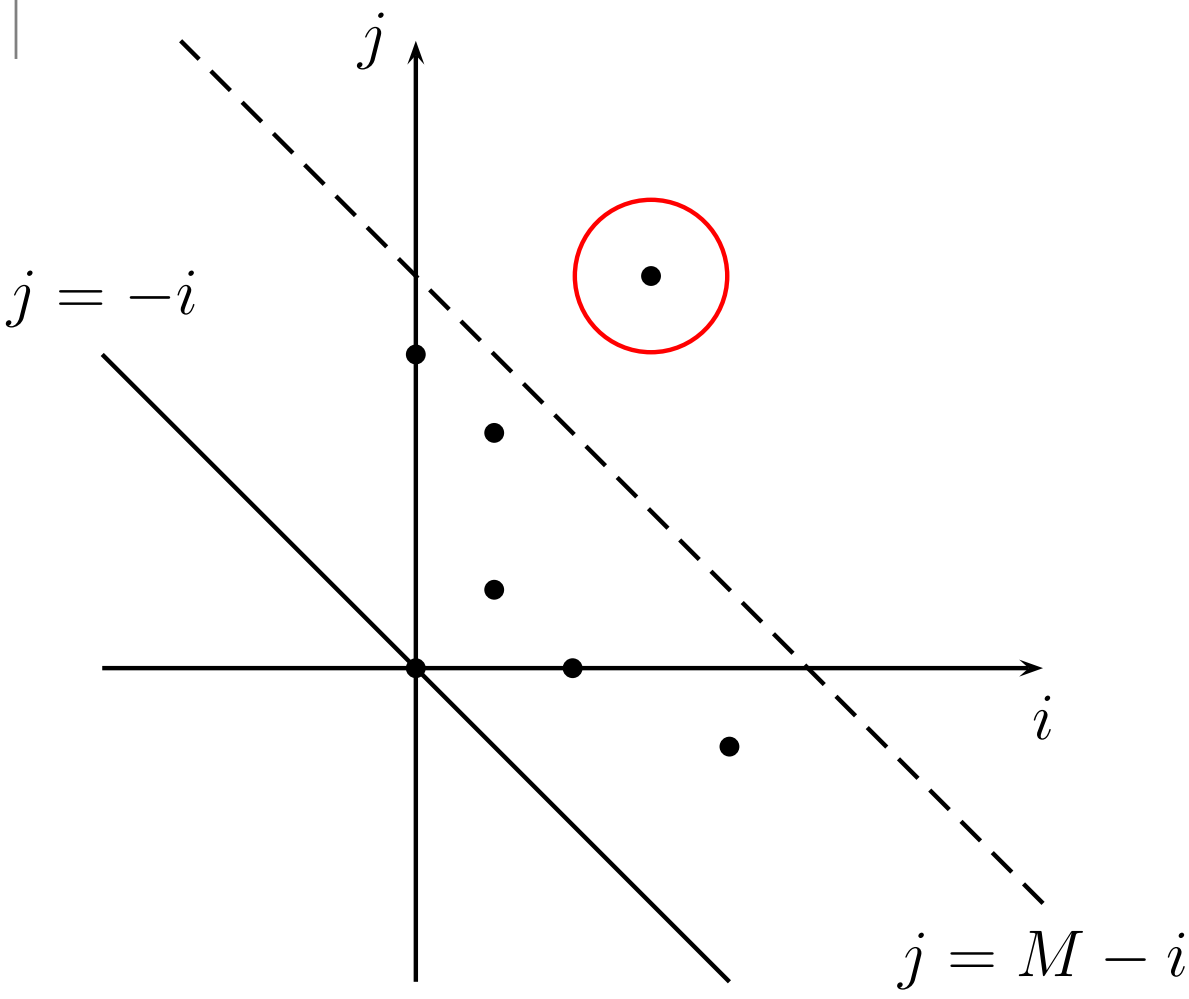
Consideriamo lo spazio delle possibili coppie (i, j) e la regione permessa $0 \leq i + j < M$.

Esempio



Supponiamo di conoscere tutte le coppie (i, j) raggiunte da P.

Esempio



Supponiamo di conoscere tutte le coppie (i, j) raggiunte da P. Troviamo l'errore.

Semantica concreta

Come troviamo tutte le coppie (i, j)?

Semantica concreta

Come troviamo tutte le coppie (i, j)?

Eseguendo il programma P una volta per ogni possibile ingresso e ricordando i valori di i e j ogni volta che sta per essere eseguita l'istruzione $x = a[i + j];$

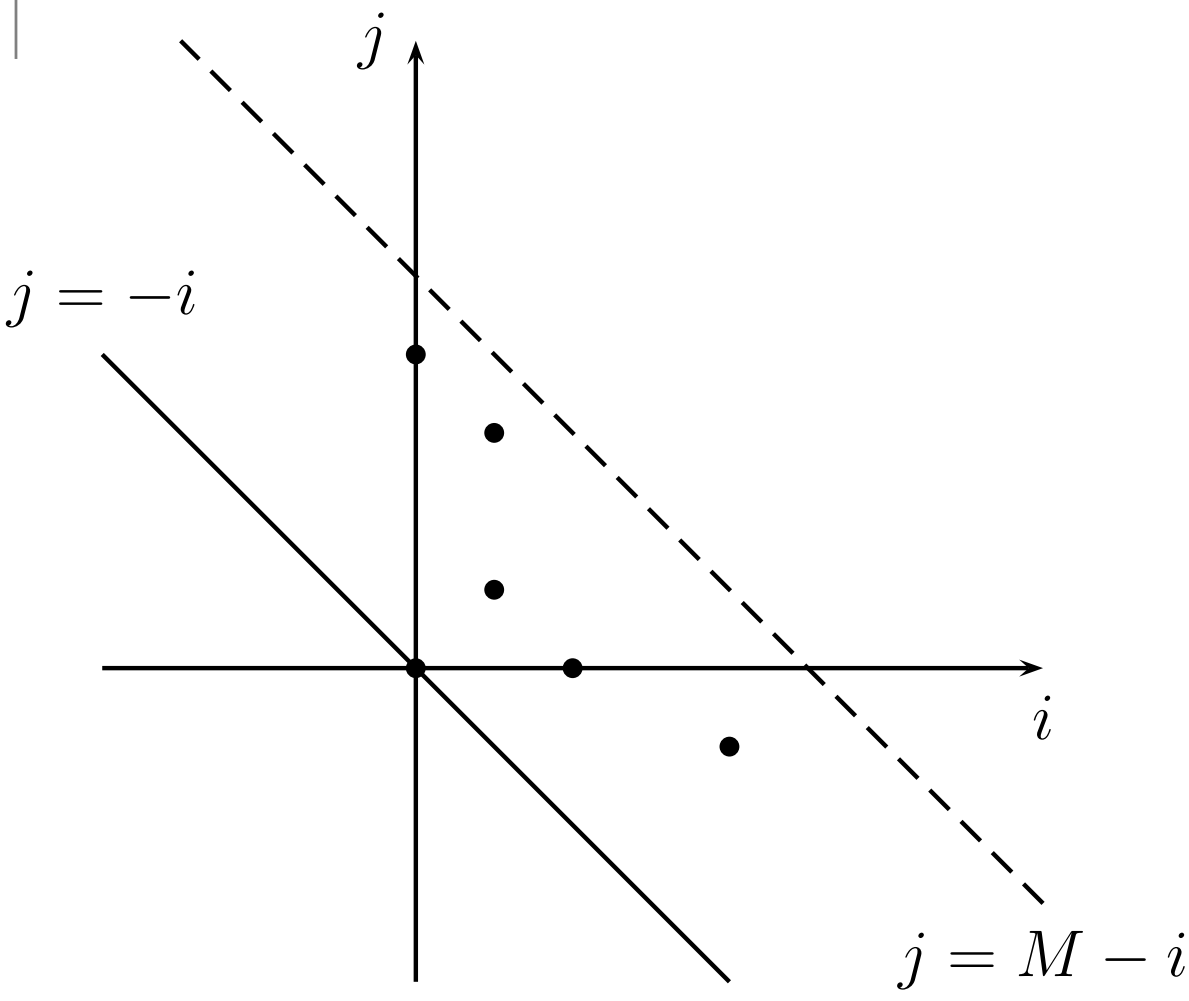
Semantica concreta

Come troviamo tutte le coppie (i, j)?

Eseguendo il programma P una volta per ogni possibile ingresso e ricordando i valori di i e j ogni volta che sta per essere eseguita l'istruzione $x = a[i + j];$

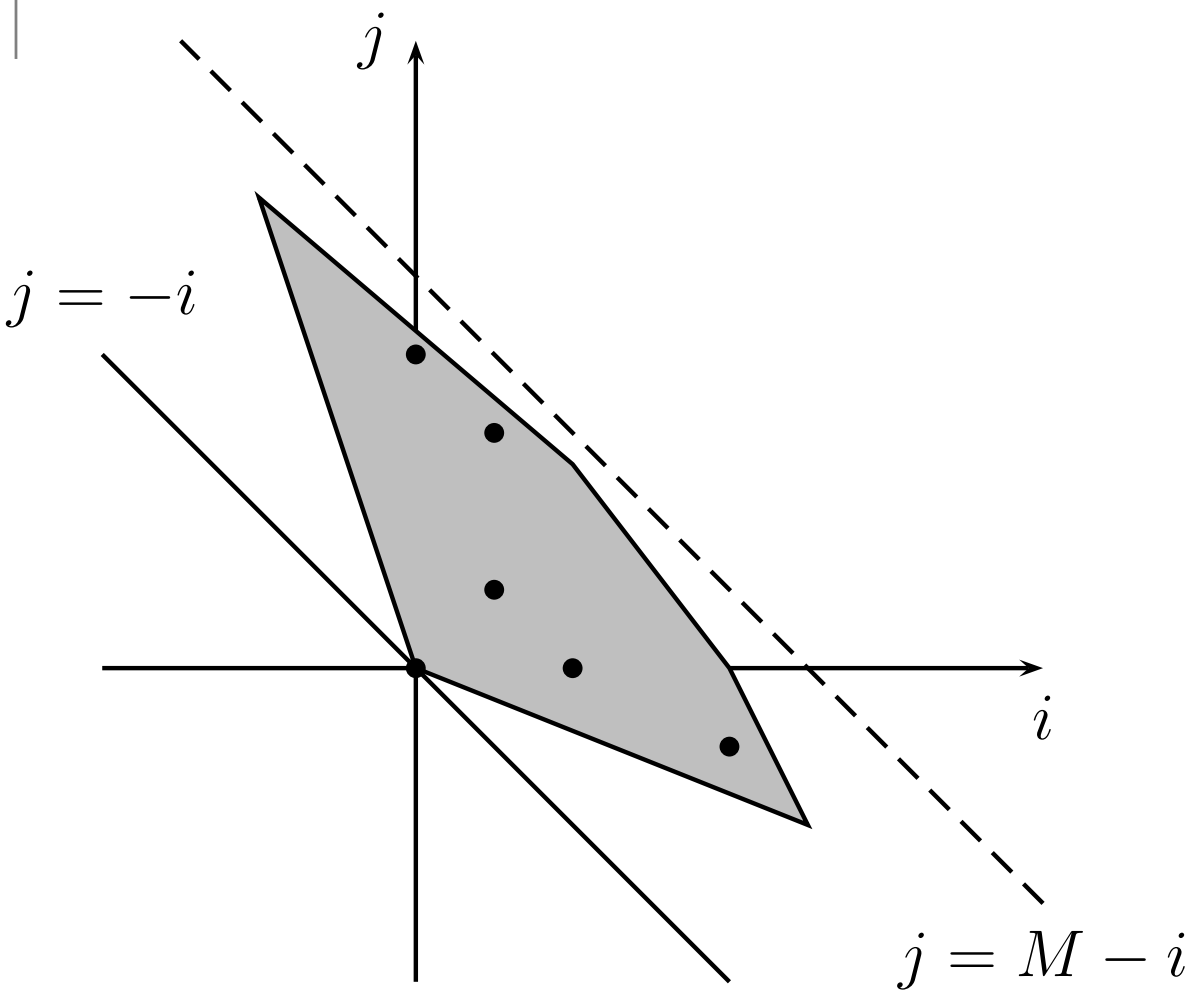
Ovviamente non è possibile farlo davvero.

Approssimazione



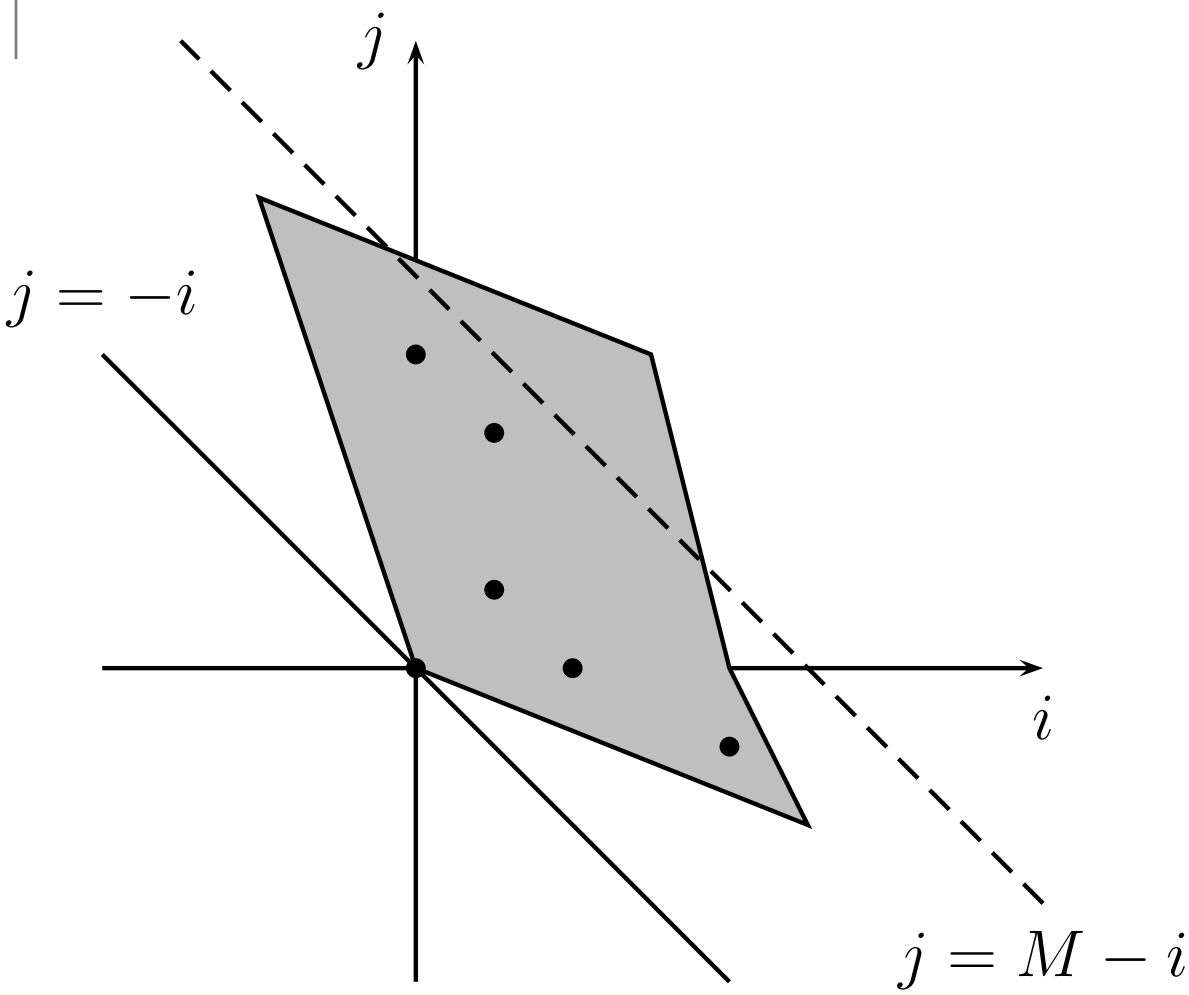
Possiamo però cercare una regione computabile che *contiene* i veri punti.

Approssimazione



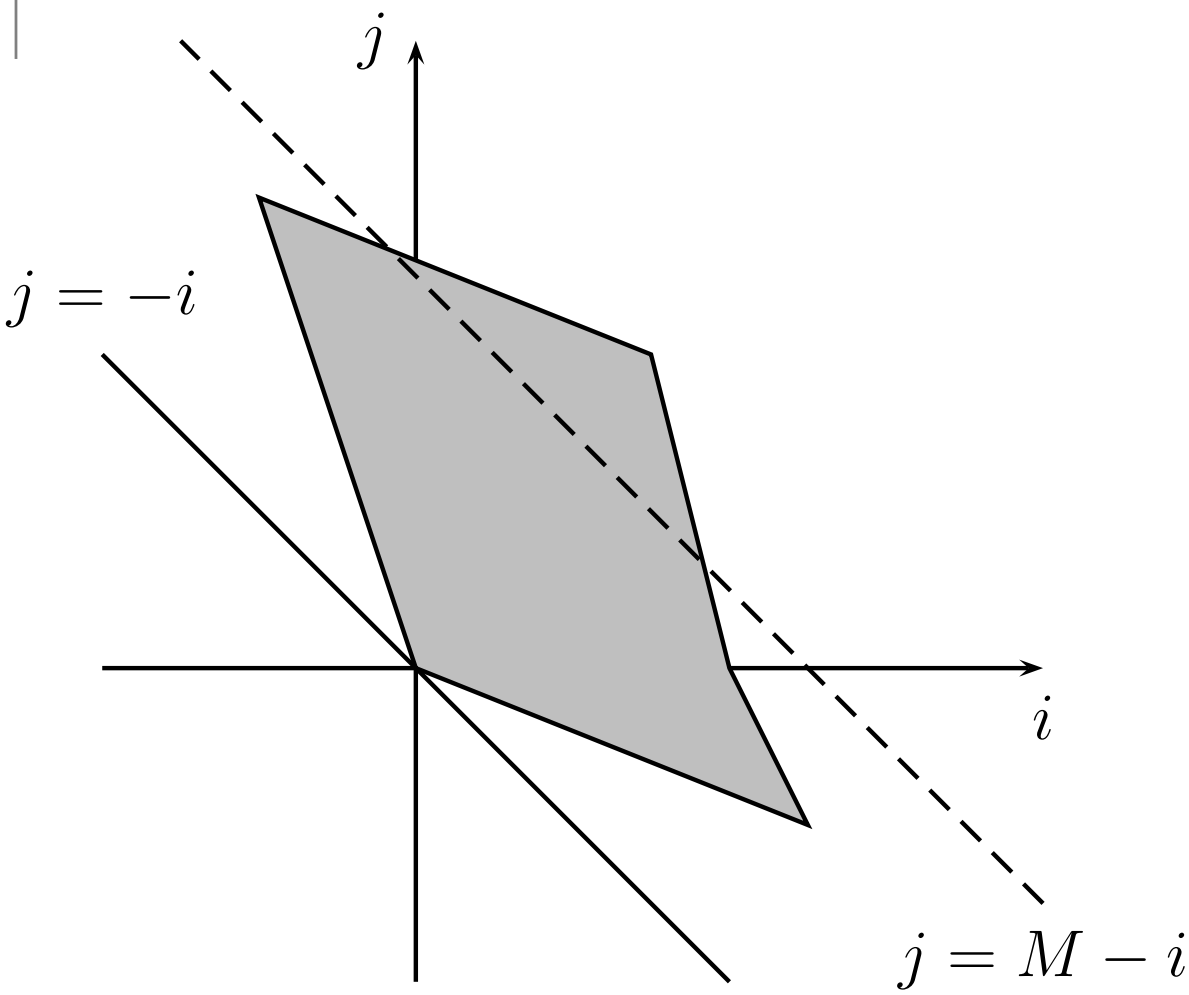
Possiamo però cercare una regione computabile che *contiene* i veri punti. Se la regione è compresa nella zona permessa il programma è corretto.

Approssimazione: falsi positivi



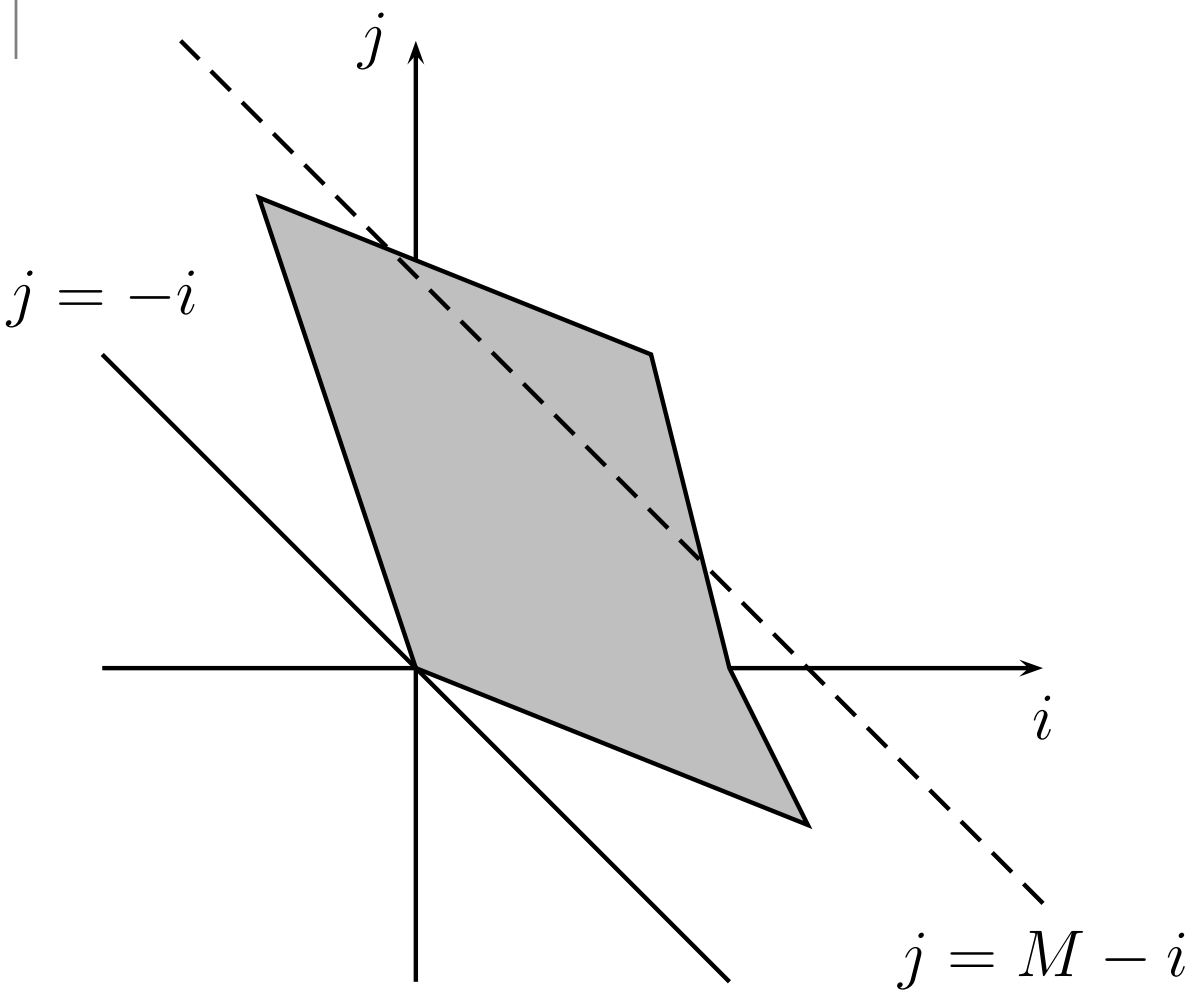
Il programma è corretto

Approssimazione: falsi positivi



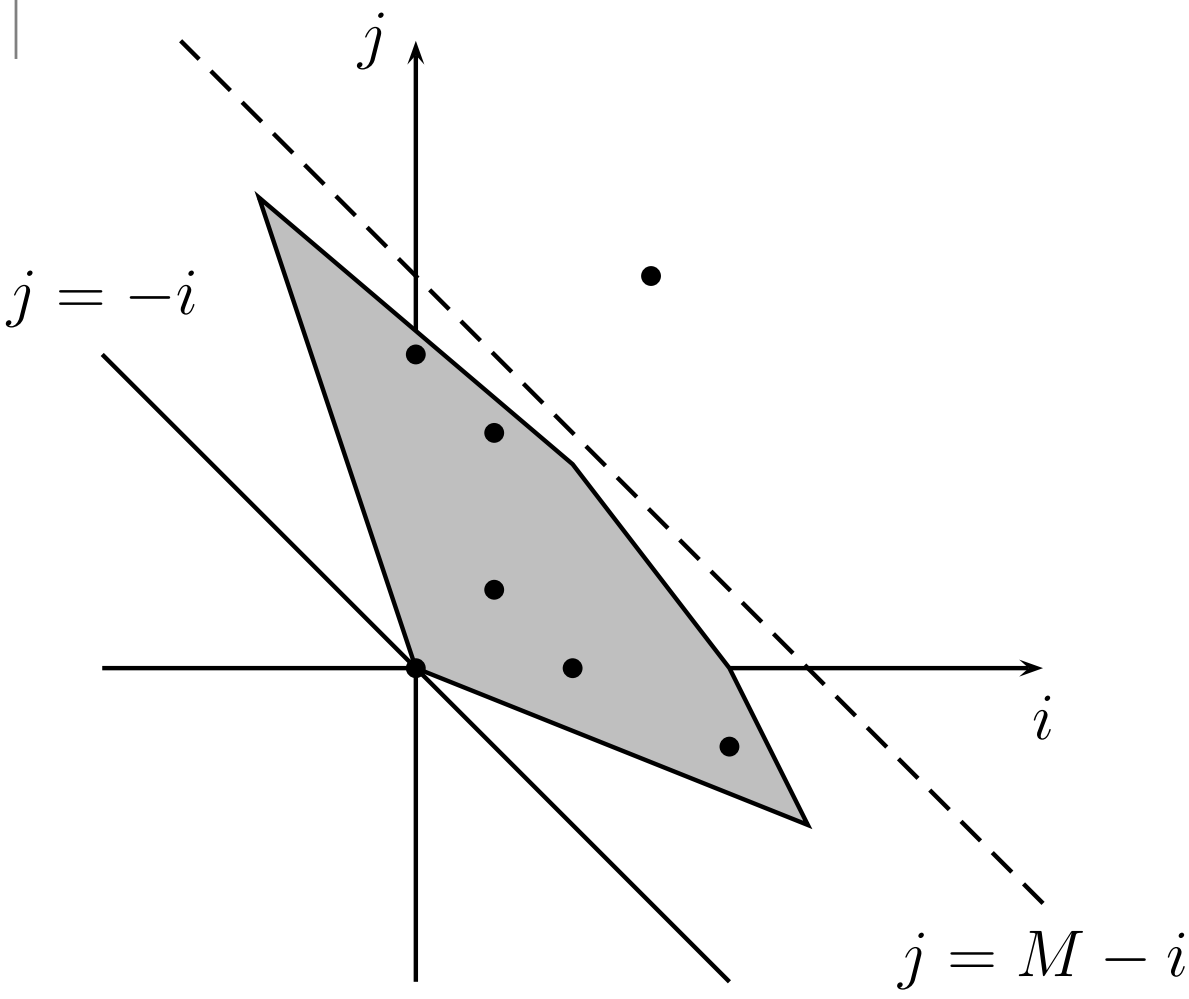
Il programma è corretto, ma non possiamo saperlo.

Approssimazione: falsi positivi



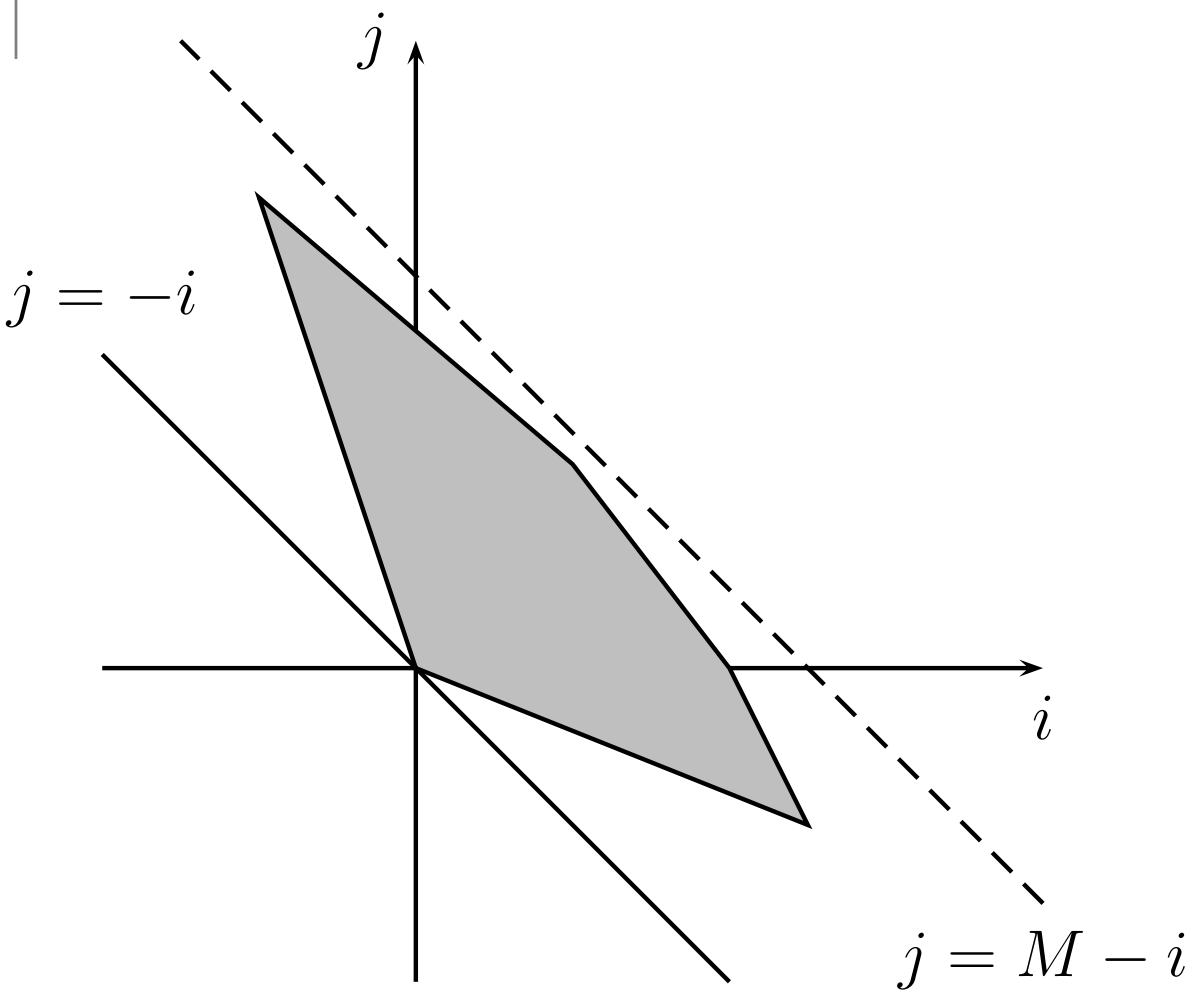
Il programma è corretto, ma non possiamo saperlo. Per sicurezza lo consideriamo *scorretto*.

Approssimazione: falsi negativi



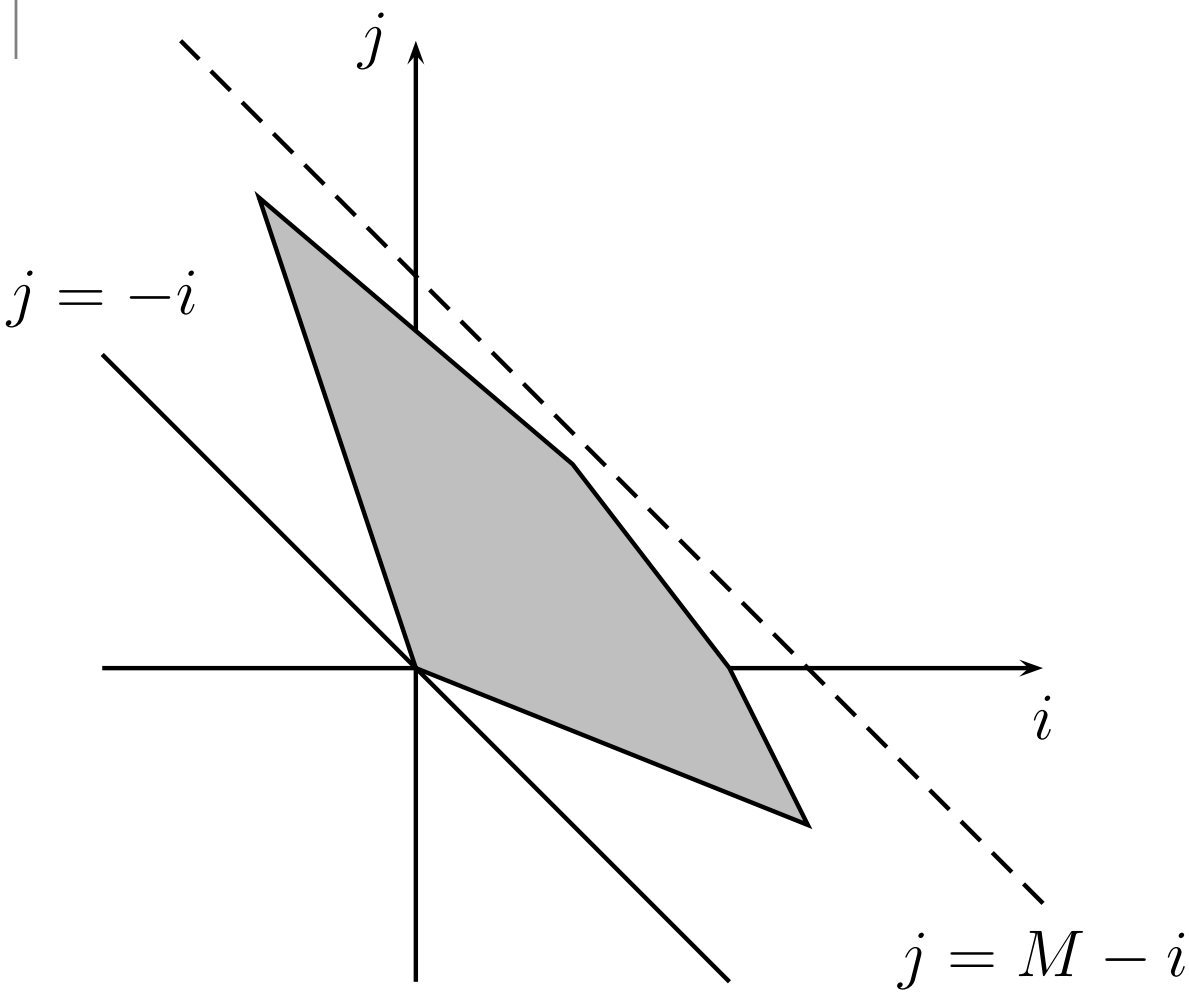
Il programma è scorretto

Approssimazione: falsi negativi



Il programma è scorretto, ma (di nuovo) non possiamo saperlo.

Approssimazione: falsi negativi



Il programma è scorretto, ma (di nuovo) non possiamo saperlo. È la nostra analisi che è *sbagliata*!

Semantica astratta

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Semantica astratta

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Partiamo da una regione iniziale, per es. l'intero piano.

Semantica astratta

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Partiamo da una regione iniziale, per es. l'intero piano.

Per ogni istruzione, deduciamo una regione di uscita data la regione di ingresso. Per esempio:

```
if (i > j) {
    // ...
}
```

All'inizio del ramo *then* intersechiamo la regione ottenuta prima dell'**if** con il semipiano $i - j > 0$.

Semantica astratta (continua)

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Semantica astratta (continua)

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Esempio nel caso di istruzioni di assegnamento:

$$i = i + 1;$$

Semantica astratta (continua)

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Esempio nel caso di istruzioni di assegnamento:

$i = i + 1;$ “Spostiamo” la regione di una unità a destra.

Semantica astratta (continua)

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Esempio nel caso di istruzioni di assegnamento:

$i = i + 1;$ “Spostiamo” la regione di una unità a destra.

Nelle istruzioni raggiungibili da più percorsi (per es. dopo un **if** o all’inizio di un **while**) dobbiamo in qualche modo “unire” le regioni ottenute nei vari percorsi.

Semantica astratta (continua)

Come troviamo una regione che contiene tutti i punti (i, j)?
(supponiamo che P non contenga altre variabili oltre a i e j)

Esempio nel caso di istruzioni di assegnamento:

$i = i + 1;$ “Spostiamo” la regione di una unità a destra.

Nelle istruzioni raggiungibili da più percorsi (per es. dopo un **if** o all’inizio di un **while**) dobbiamo in qualche modo “unire” le regioni ottenute nei vari percorsi.

Ci fermiamo quando non scopriamo più niente di nuovo.

Interpretazione Astratta

Che relazione c’è tra la semantica concreta e quella astratta?

Interpretazione Astratta

Che relazione c’è tra la semantica concreta e quella astratta?

- Sostituiamo il dominio “concreto” (es. insiemi di punti nel piano) con un dominio “astratto” (es. poliedri convessi).

Interpretazione Astratta

Che relazione c’è tra la semantica concreta e quella astratta?

- Sostituiamo il dominio “concreto” (es. insiemi di punti nel piano) con un dominio “astratto” (es. poliedri convessi).
- Reinterpretiamo le operazioni del programma nel dominio astratto.

Interpretazione Astratta

Che relazione c'è tra la semantica concreta e quella astratta?

- Sostituiamo il dominio "concreto" (es. insiemi di punti nel piano) con un dominio "astratto" (es. poliedri convessi).
- Reinterpretiamo le operazioni del programma nel dominio astratto.

I domini concreto e astratto sono normalmente *reticoli* (completi) collegati tra loro da una *connessione di Galois*.

Interpretazione Astratta

Che relazione c'è tra la semantica concreta e quella astratta?

- Sostituiamo il dominio "concreto" (es. insiemi di punti nel piano) con un dominio "astratto" (es. poliedri convessi).
- Reinterpretiamo le operazioni del programma nel dominio astratto.

I domini concreto e astratto sono normalmente *reticoli* (completi) collegati tra loro da una *connessione di Galois*.

L'interpretazione delle operazioni del programma discende dalla scelta del dominio astratto.

Un semplice linguaggio

Gli esempi saranno relativi a un semplice linguaggio di livello intermedio:

- *assegnamenti* della forma "$x = e$";
- *salti condizionali* della forma "**if** b **goto** m".

Dove x è un nome di variabile preso da un insieme dato Var, e è una espressione aritmetica e b una espressione booleana.

Un semplice linguaggio

Supporremo che tutte le variabili assumano valori in $\mathbb{Z}$ (numeri interi).

Le espressioni aritmetiche rispettano la seguente sintassi (c è una costante intera)

$$Aexpr ::= c | Var | - Var | Var + Var | Var * Var.$$

Le espressioni booleane rispettano la seguente sintassi:

$$Bexpr ::= \textbf{true} | \textbf{false} | Var = Var | Var < Var.$$

Un semplice linguaggio

Se $Instr$ è l'insieme delle possibili istruzioni (assegnamenti o salti condizionali) e $Addr = \{1, \ldots, N\}$ è un insieme di indirizzi, un programma è una funzione $P\colon Addr \to Instr$ che associa ad ogni indirizzo una istruzione.

Il programma dispone di un *ambiente* ρ che associa ad ogni variabile il suo valore corrente. L'ambiente è quindi una funzione $\rho\colon Var \to \mathbb{Z}$. Chiamiamo Env l'insieme delle funzioni da Var a $\mathbb{Z}$, in modo che $\rho \in Env$.

Lo *stato* del programma è una coppia (n, ρ) dove n è la prossima istruzione che verrà eseguita e ρ è l'ambiente corrente.

Un semplice linguaggio

Abbiamo bisogno di poter calcolare il valore di una espressione aritmetica. Poichè l'espressione può riferire variabili, il valore dipende dall'ambiente.

Data una espressione $e \in Aexpr$ definiamo una funzione $Aval_e \colon Env \to \mathbb{Z}$ per casi ($c \in \mathbb{Z}$ e $x, y \in Var$):

$$Aval_c(\rho) = c$$

$$Aval_x(\rho) = \rho(x)$$

$$Aval_{-x}(\rho) = -\rho(x)$$

$$Aval_{x+y}(\rho) = \rho(x) + \rho(y)$$

$$Aval_{x*y}(\rho) = \rho(x)\rho(y).$$

Un semplice linguaggio

Per valutare il valore di una espressione booleana $b \in Bexpr$ definiamo la funzione $Bval_b \colon Env \to \{true, false\}$ per casi ($x, y \in Var$):

$$Bval_{\textbf{true}}(\rho) = true$$

$$Bval_{\textbf{false}}(\rho) = false$$

$$Bval_{x=y}(\rho) = (\rho(x) = \rho(y))$$

$$Bval_{x<y}(\rho) = (\rho(x) < \rho(y)).$$

Un semplice linguaggio

Diamo la semantica del linguaggio sotto la forma di una funzione

$$next : (Addr \times Env) \rightarrow (Addr \times Env)$$

che dato lo stato corrente ci restituisce lo stato successivo.

Supporremo che lo stato iniziale sia $(1, \rho_0)$, dove ρ_0 è una funzione che associa il valore 0 ad ogni variabile.

Lo stato finale sarà (N, ρ), dove N è il massimo indirizzo in $Addr$.

Un semplice linguaggio

Data una funzione f denotiamo con $f[v/x]$ la funzione tale che $f[v/x](x) = v$ e $f[v/x](y) = f(y)$ per $x \neq y$.

Fissato un programma $P\colon Addr \to Instr$, avremo:

$$next(n,\rho) = \begin{cases} (n,\rho) & \text{se } n = N, \\ (n+1, \rho[v/x]) & P(n) = \text{“}x := e\text{”} \text{ e } v = Aval_e(\rho), \\ (n+1,\rho) & P(n) = \text{“}\textbf{if } b \textbf{ goto } m\text{”} \text{ e } !Bval_b(\rho), \\ (m,\rho) & P(n) = \text{“}\textbf{if } b \textbf{ goto } m\text{”} \text{ e } Bval_b(\rho). \end{cases}$$

L'esecuzione di P comporta l'applicazione ripetuta di $next$ a partire da $(1, \rho_0)$.
Scriveremo $next^i$ per denotare l'applicazione ripetuta i volte di $next$ ($next^0$ è la funzione identità).

Un semplice linguaggio

Esempio:

$x = 1;$
while $(x \leq 100)$
$\quad x = x + 2;$

$1\colon z = 2$

$2\colon y = 100$

$3\colon x = 1$

$4\colon$ **if** $y < x$ **goto** 7

$5\colon x = x + z$

$6\colon$ **if true goto** 4

$7\colon x = x.$

Un semplice linguaggio

Esecuzione:

$1\colon z = 2$

$2\colon y = 100$

$3\colon x = 1$

$4\colon$ **if** $y < x$ **goto** 7

$5\colon x = x + z$

$6\colon$ **if true goto** 4

$7\colon x = x.$

$(1, \quad [x \mapsto 0, y \mapsto 0, z \mapsto 0])$

$(2, \quad [x \mapsto 0, y \mapsto 0, z \mapsto 2])$

$(3, \ [x \mapsto 0, y \mapsto 100, z \mapsto 2])$

$(4, \ [x \mapsto 1, y \mapsto 100, z \mapsto 2])$

$(5, \ [x \mapsto 1, y \mapsto 100, z \mapsto 2])$

$(6, \ [x \mapsto 3, y \mapsto 100, z \mapsto 2])$

$\ldots$

$(4,[x \mapsto 101, y \mapsto 100, z \mapsto 2])$

$(7,[x \mapsto 101, y \mapsto 100, z \mapsto 2])$

$(7,[x \mapsto 101, y \mapsto 100, z \mapsto 2])$

$\ldots$

Collecting semantics

Siamo interessati a *proprietà* del comportamento a tempo di esecuzione dei nostri programmi.

Definiremo quindi una nuova semantica che associa proprietà a programmi.

Questa sarà la semantica che cercheremo di approssimare tramite interpretatizione astratta, non la semantica data da $next$.

Collecting semantics

Non esiste un'unica semantica che associ proprietà a programmi.

La scelta di questa semantica determina il tipo di analisi che è possibile realizzare tramite interpretazione astratta.

Definiremo una semantica che associa ad ogni punto del programma una proprietà rispettata dall'ambiente ogni volta che l'esecuzione passa da quel punto.

Collecting semantics

Possiamo esprimere una proprietà come un predicato sugli ambienti.

Per esempio

$$p(\rho) = \text{“}\rho(x) \text{ è compreso tra 1 e 101”}$$

è una proprietà verificata dall'ambiente ogni volta che l'esecuzione passa dal punto 4 del programma precedente.

Più precisamente:

$$\exists i \geq 0 : next^i(1, \rho_0) = (4, \rho) \implies p(\rho).$$

Collecting semantics

È conveniente rappresentare le proprietà tramite insiemi: l'insieme degli elementi che verificano la proprietà.

Rappresentiamo quindi un predicato $p(\rho)$ con

$$\{\, \rho \in Env \mid p(\rho) \,\}.$$

L'implicazione tra predicati si trasforma in inclusione tra insiemi. Se

$$p(\rho) \implies q(\rho)$$

allora

$$\{\, \rho \in Env \mid p(\rho) \,\} \subseteq \{\, \rho \in Env \mid q(\rho) \,\}.$$

(E viceversa).

Collecting semantics

Più predicati possono esprimere la stesssa proprietà:

$p_1(\rho) =$ "$\rho(x)\rho(y) > 0$",

$p_2(\rho) =$ "$\rho(x) > 0$ e $\rho(y) > 0$ oppure $\rho(x) < 0$ e $\rho(y) < 0$".

Abbiamo:

$$p_1(\rho) \iff p_2(\rho).$$

Se passiamo agli insiemi, questo diventa:

$$\{\, \rho \in Env \mid p_1(\rho) \,\} = \{\, \rho \in Env \mid p_2(\rho) \,\}.$$

Quindi, quando la proprietà espressa è la stessa, l'insieme è lo stesso.

Collecting semantics

L'esempio precedente:

$$\exists i \geq 0 : next^i(1, \rho_0) = (4, \rho) \implies 1 \leq \rho(x) \leq 101$$

Diventa:

$$\{\, \rho \mid \exists i \geq 0 : next^i(1, \rho_0) = (4, \rho) \,\} \subseteq \{\, \rho \mid 1 \leq \rho(x) \leq 101 \,\}.$$

L'insieme a sinistra è la proprietà più precisa rispettata dall'ambiente nel punto 4.

Collecting semantics

Fissiamo un programma $P\colon Addr \to Instr$, con $Addr = \{1, \ldots, N\}$.

Per ogni $n \in Addr$ definiamo il *contesto* di n.

$$C_n = \{\, \rho \in Env \mid \exists i \geq 0 : next^i(1, \rho_0) = (n, \rho) \,\}$$

Gli insiemi $C_1, \ldots, C_N$ sono la *collecting semantics* del programma P.

Collecting semantics

Per trovare $C_1, \ldots, C_N$ rappresentiamo prima il programma come un grafo.

$1\colon z = 2$

$2\colon y = 100$

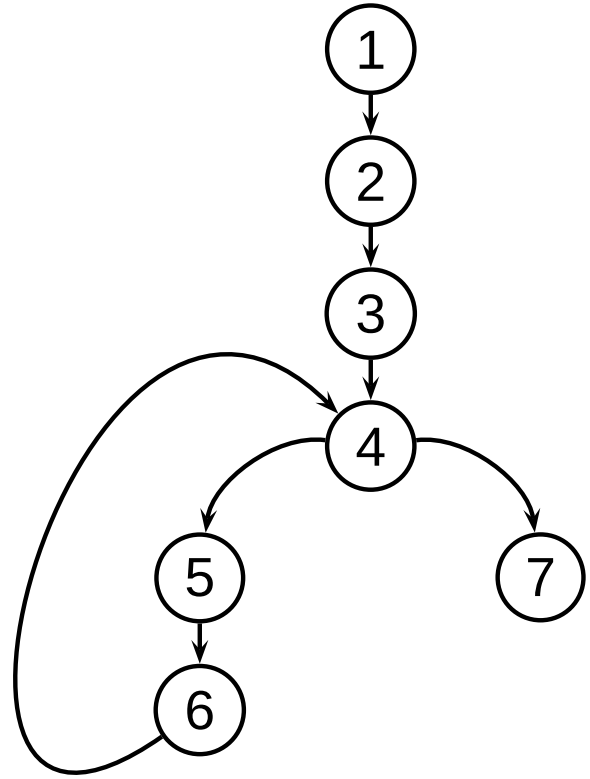
$3\colon x = 1$

$4\colon$ **if** $y < x$ **goto** 7

$5\colon x = x + z$

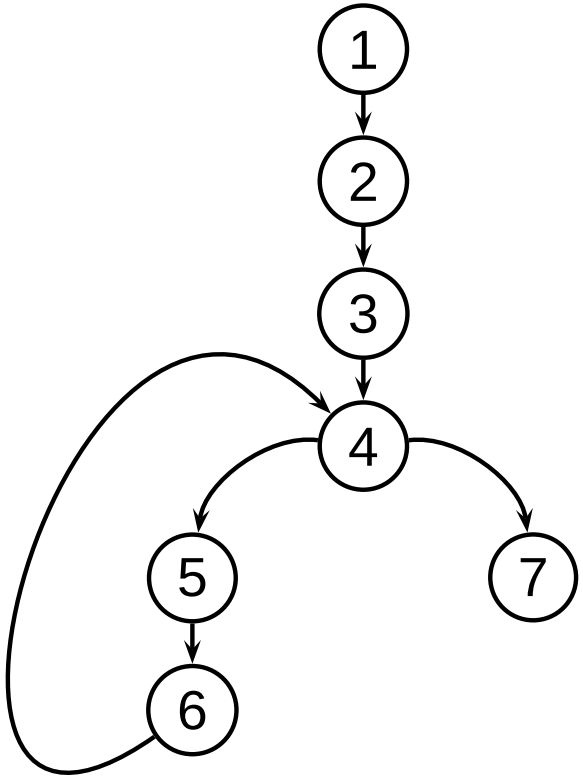
$6\colon$ **if true goto** 4

$7\colon x = x.$



Collecting semantics

Per ogni nodo $n \in Addr$ troviamo l'insieme dei suoi predecessori $prec(n)$.



$$prec(1) = \varnothing$$
$$prec(2) = \{1\}$$
$$prec(3) = \{2\}$$
$$prec(4) = \{3, 6\}$$
$$prec(5) = \{4\}$$
$$prec(6) = \{5\}$$
$$prec(7) = \{4\}.$$

Collecting semantics

Vogliamo esprimere C_n in funzione dei contesti dei nodi predecessori di n.

Per esempio, se $\rho \in C_3$ sappiamo che dovrà essere $\rho[1/x] \in C_4$, perché il nodo 3 contiene l'istruzione $x = 1$.

In generale, per ogni $n \in Addr$ e $m \in prec(n)$ possiamo definire

$$F_{m \to n}(R) = \{\, \rho' \mid \exists \rho \in R : next(m, \rho) = (n, \rho') \,\},$$

con $R \subseteq Env$.

Collecting semantics

Per esempio, prendiamo un programma P in cui $P(3) =$ "$x = 1$". Allora

$$\begin{aligned} F_{3\to 4}(R) &= \{\, \rho' \mid \exists \rho \in R : next(3,\rho) = (4,\rho') \,\} \\ &= \{\, \rho' \mid \exists \rho \in R : (4, \rho[1/x]) = (4,\rho') \,\} \\ &= \{\, \rho[1/x] \mid \rho \in R \,\}. \end{aligned}$$

Se, per esempio,

$$R = \{[x \mapsto 0, y \mapsto 100, z \mapsto 2], [x \mapsto -20, y \mapsto 13, z \mapsto 0]\}$$

avremo

$$F_{3\to 4}(R) = \{[x \mapsto 1, y \mapsto 100, z \mapsto 2], [x \mapsto 1, y \mapsto 13, z \mapsto 0]\}.$$

Collecting semantics

Per ogni nodo $n \in Addr$ con $n > 1$ scriveremo

$$C_n = \bigcup_{m \in prec(n)} F_{m \to n}(C_m).$$

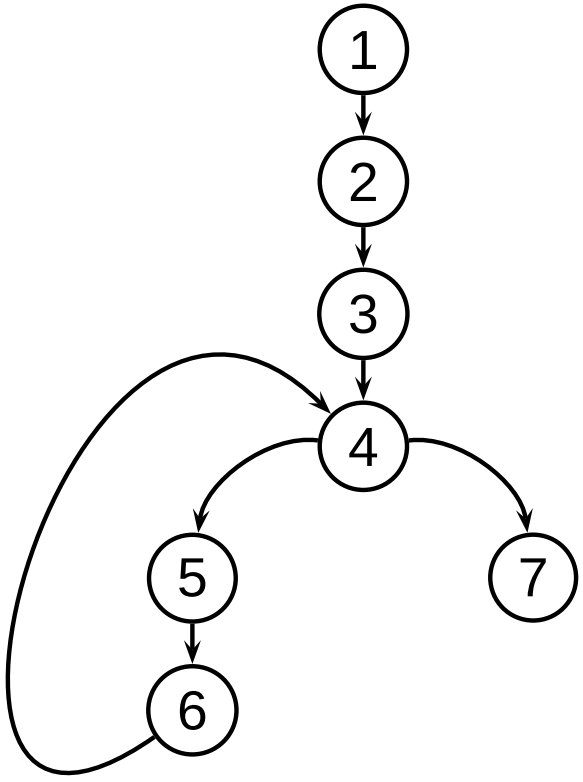
Per $n = 1$ dobbiamo tener conto dell'insieme iniziale:

$$C_1 = \{\rho_0\} \cup \bigcup_{m \in prec(1)} F_{m \to 1}(C_m).$$

Questo ci fornisce n equazioni in $C_1, \ldots, C_N$ che dobbiamo risolvere per trovare i contesti.

Collecting semantics

Esempio:



$$C_1 = \{\rho_0\}$$
$$C_2 = F_{1\to 2}(C_1)$$
$$C_3 = F_{2\to 3}(C_2)$$
$$C_4 = F_{3\to 4}(C_3) \cup F_{6\to 4}(C_6)$$
$$C_5 = F_{4\to 5}(C_4)$$
$$C_6 = F_{5\to 6}(C_5)$$
$$C_7 = F_{4\to 7}(C_4).$$

Collecting semantics

Le funzioni $F_{m \to n}$ dipendono dall'istruzione al nodo m e da qual è il nodo n di destinazione.

Nel nostro semplice linguaggio abbiamo solo tre casi diversi:

- m contiene un assegnamento (quindi $n = m + 1$);
- m contiene un salto condizionale e n è sul ramo "$true$";
- m contiene un salto condizionale e n è sul ramo "$false$" (quindi $n = m + 1$).

Collecting semantics

Definiamo tre funzioni generiche per i tre casi

$$Assign_{x=e}(R) = \{\, \rho[v/x] \mid \rho \in R \text{ e } v = Aval_e(\rho) \,\}$$
$$tBranch_b(R) = R \cap \{\, \rho \mid Bval_b(\rho) \,\}$$
$$fBranch_b(R) = R \cap \overline{\{\, \rho \mid Bval_b(\rho) \,\}}.$$

(abbiamo denotato con $\overline{X}$ il complemento dell'insieme X).

Collecting semantics

Torniamo all'esempio:

$1\colon z = 2$

$2\colon y = 100$

$3\colon x = 1$

$4\colon$ **if** $y < x$ **goto** 7

$5\colon x = x + z$

$6\colon$ **if true goto** 4

$7\colon x = x.$

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

(abbiamo semplificato $tBranch_{\textbf{true}}(R) = R$).

Collecting semantics

Interpretando $C_1, \ldots, C_N$ come un vettore $\underline{C}$, possiamo ridurre il problema a trovare $\underline{C}$ tale che

$$\underline{C} = F(\underline{C}).$$

Sotto opportune condizioni, l'equazione può essere risolta iterativamente definendo la sequenza

$$\underline{C}^0 = (\varnothing, \ldots, \varnothing)$$

$$\underline{C}^{k+1} = F(\underline{C}^k) \qquad (k \geq 0)$$

$$\underline{C} = \bigcup_{k=0}^{\infty} \underline{C}^k.$$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	0	1
C_1	$\varnothing$	$\{[0,0,0]\}$
C_2	$\varnothing$	$\varnothing$
C_3	$\varnothing$	$\varnothing$
C_4	$\varnothing$	$\varnothing$
C_5	$\varnothing$	$\varnothing$
C_6	$\varnothing$	$\varnothing$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	1	2
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\varnothing$	$\{[0,0,2]\}$
C_3	$\varnothing$	$\varnothing$
C_4	$\varnothing$	$\varnothing$
C_5	$\varnothing$	$\varnothing$
C_6	$\varnothing$	$\varnothing$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	2	3
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\varnothing$	$\{[0,100,2]\}$
C_4	$\varnothing$	$\varnothing$
C_5	$\varnothing$	$\varnothing$
C_6	$\varnothing$	$\varnothing$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	3	4
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\{[0,100,2]\}$	$\{[0,100,2]\}$
C_4	$\varnothing$	$\{[1,100,2]\}$
C_5	$\varnothing$	$\varnothing$
C_6	$\varnothing$	$\varnothing$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	4	5
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\{[0,100,2]\}$	$\{[0,100,2]\}$
C_4	$\{[1,100,2]\}$	$\{[1,100,2]\}$
C_5	$\varnothing$	$\{[1,100,2]\}$
C_6	$\varnothing$	$\varnothing$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	5	6
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\{[0,100,2]\}$	$\{[0,100,2]\}$
C_4	$\{[1,100,2]\}$	$\{[1,100,2]\}$
C_5	$\{[1,100,2]\}$	$\{[1,100,2]\}$
C_6	$\varnothing$	$\{[3,100,2]\}$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	6	7
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\{[0,100,2]\}$	$\{[0,100,2]\}$
C_4	$\{[1,100,2]\}$	$\{[1,100,2],[3,100,2]\}$
C_5	$\{[1,100,2]\}$	$\{[1,100,2]\}$
C_6	$\{[3,100,2]\}$	$\{[3,100,2]\}$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	7	8
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\{[0,100,2]\}$	$\{[0,100,2]$
C_4	$\{[1,100,2],[3,100,2]\}$	$\{[1,100,2],[3,1$
C_5	$\{[1,100,2]\}$	$\{[1,100,2],[3,1$
C_6	$\{[3,100,2]\}$	$\{[3,100,2]$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

k	8	9
C_1	$\{[0,0,0]\}$	$\{[0,0,0]\}$
C_2	$\{[0,0,2]\}$	$\{[0,0,2]\}$
C_3	$\{[0,100,2]\}$	$\{[0,100,2]$
C_4	$\{[1,100,2],[3,100,2]\}$	$\{[1,100,2],[3,1$
C_5	$\{[1,100,2],[3,100,2]\}$	$\{[1,100,2],[3,1$
C_6	$\{[3,100,2]\}$	$\{[3,100,2],[5,1$
C_7	$\varnothing$	$\varnothing$

Collecting semantics

Esempio (scriviamo $[a, b, c]$ per denotare $[x \mapsto a, y \mapsto b, z \mapsto c]$):

$$C_1 = \{\rho_0\}$$
$$C_2 = Assign_{z=2}(C_1)$$
$$C_3 = Assign_{y=100}(C_2)$$
$$C_4 = Assign_{x=1}(C_3) \cup C_6$$
$$C_5 = fBranch_{y<x}(C_4)$$
$$C_6 = Assign_{x=x+z}(C_5)$$
$$C_7 = tBranch_{y<x}(C_4).$$

C_1	$\{[0, 0, 0]\}$
C_2	$\{[0, 0, 2]\}$
C_3	$\{[0, 100, 2]\}$
C_4	$\{[1, 100, 2], [3, 100, 2], [5, 100, 2], \ldots, [99,$
C_5	$\{[1, 100, 2], [3, 100, 2], [5, 100, 2], ..$
C_6	$\{[3, 100, 2], [5, 100, 2], [7, 100, 2], ..$
C_7	$\{[101, 100, 2]\}$

Collecting semantics

Il vettore $\underline{C} = (C_1, \ldots, C_N)$ tale che

$$\underline{C} = F(\underline{C})$$

trovato iterativamente partendo da $(\varnothing, \ldots, \varnothing)$ è quello che cercavamo. Infatti, per ogni $n \in Addr$,

$$C_n = \{\, \rho \in Env \mid \exists i \geq 0 : next^i(1, \rho_0) = (n, \rho) \,\}.$$

Collecting semantics

Approssimazione

In generale la soluzione esatta non può essere trovata.

Dobbiamo accontentarci di una soluzione approssimata.

L'approssimazione più spesso utile è quella per *eccesso*: trovare un vettore $\underline{C'}$ che "contiene" $\underline{C}$ (ogni elemento di $\underline{C'}$ contiene il corrispondente di $\underline{C}$).

In questo senso, l'inclusione tra insiemi rappresenta la relazione "è più preciso di".

Collecting semantics

Abbiamo definito una semantica che associa proprietà degli ambienti ad ogni punto del programma.

- Rappresentiamo le proprietà come *insiemi* di ambienti e le organizziamo in un vettore (un elemento per ogni punto del programma).
- Dal programma ricaviamo un insieme di equazioni che il vettore cercato deve rispettare. Le equazioni usano operazioni come $\cup$ o funzioni come $Assign_{x=e}$, $tBranch_b$ e $fBranch_b$.

Per definire una intepretazione astratta dobbiamo trovare i corrispondenti astratti delle proprietà e delle operazioni $\cup$, $Assign_{x=e}$ etc., quindi ottenere un corrispondente vettore di proprietà e un sistema di equazioni.

Abstraction

La struttura delle equazioni sarà la stessa, ma con gli operatori astratti e le proprietà astratte al posto dei corrispondenti concreti.

$$C_1 = \{\rho_0\} \qquad A_1 = \rho_0^\sharp$$

$$C_2 = Assign_{z=2}(C_1) \qquad A_2 = Assign^\sharp_{z=2}(A_1)$$

$$C_3 = Assign_{y=100}(C_2) \qquad A_3 = Assign^\sharp_{y=100}(A_2)$$

$$C_4 = Assign_{x=1}(C_3) \cup C_6 \qquad A_4 = Assign^\sharp_{x=1}(A_3) \sqcup A_6$$

$$C_5 = fBranch_{y<x}(C_4) \qquad A_5 = fBranch^\sharp_{y<x}(A_4)$$

$$C_6 = Assign_{x=x+z}(C_5) \qquad A_6 = Assign^\sharp_{x=x+z}(A_5)$$

$$C_7 = tBranch_{y<x}(C_4) \qquad A_7 = tBranch^\sharp_{y<x}(A_4).$$

Abstraction

La soluzione $\underline{A}$ del sistema di equazioni astratte dovrà in qualche senso “contenere” la soluzione $\underline{C}$ del sistema di equazioni concrete.

Per rendere precise queste idee dobbiamo introdurre i concetti di reticolo, connessione di Galois e interpretazione astratta.

Poset

Un insieme parzialmente ordinato (*poset*) è un insieme P fornito di una relazione $\leq$ che sia

- *riflessiva*: $x \leq x$ per ogni $x \in P$,
- *transitiva*: $x \leq y$ e $y \leq z$ implicano $x \leq z$ per ogni $x, y, z \in P$,
- *antisimmetrica*: $x \leq y$ e $y \leq x$ implicano $x = y$ per ogni $x, y \in P$.

Indichiamo un poset con l'espressione $\langle P; \leq \rangle$.

Esempi di poset

Prendiamo l'insieme $\mathbb{N}$ dei naturali (incluso 0). $\langle \mathbb{N}; \leq \rangle$ è ovviamente un poset (una catena).

Esempi di poset

Prendiamo l'insieme $\mathbb{N}$ dei naturali (incluso 0). $\langle \mathbb{N}; \leq \rangle$ è ovviamente un poset (una catena).
Consideriamo la relazione $\preccurlyeq$ tale che $x \preccurlyeq y$ se e solo se x divide y:

$$x \preccurlyeq y \iff y = kx \quad \text{per qualche } k \in \mathbb{N}.$$

Esempi di poset

Prendiamo l'insieme $\mathbb{N}$ dei naturali (incluso 0). $\langle \mathbb{N}; \leq \rangle$ è ovviamente un poset (una catena).
Consideriamo la relazione $\preccurlyeq$ tale che $x \preccurlyeq y$ se e solo se x divide y:

$$x \preccurlyeq y \iff y = kx \quad \text{per qualche } k \in \mathbb{N}.$$

- $x = kx$ con $k = 1$,

Esempi di poset

Prendiamo l'insieme $\mathbb{N}$ dei naturali (incluso 0). $\langle \mathbb{N}; \leq \rangle$ è ovviamente un poset (una catena).
Consideriamo la relazione $\preccurlyeq$ tale che $x \preccurlyeq y$ se e solo se x divide y:

$$x \preccurlyeq y \iff y = kx \quad \text{per qualche } k \in \mathbb{N}.$$

- $x = kx$ con $k = 1$,
- $y = k_1 x$ e $z = k_2 y$ allora $z = (k_2 k_1)x$,

Esempi di poset

Prendiamo l'insieme $\mathbb{N}$ dei naturali (incluso 0). $\langle \mathbb{N}; \leq \rangle$ è ovviamente un poset (una catena).
Consideriamo la relazione $\preccurlyeq$ tale che $x \preccurlyeq y$ se e solo se x divide y:

$$x \preccurlyeq y \iff y = kx \quad \text{per qualche } k \in \mathbb{N}.$$

- $x = kx$ con $k = 1$,
- $y = k_1 x$ e $z = k_2 y$ allora $z = (k_2 k_1)x$,
- $y = k_1 x$ e $x = k_2 y$ implica $x = (k_2 k_1)x$, quindi:
  - se $x = 0$ anche $y = k_1 x = 0$,
  - se $x \neq 0$ deve essere $k_2 k_1 = 1$ e dunque $k_1 = k_2 = 1$.

Esempi di poset

Prendiamo l'insieme $\mathbb{N}$ dei naturali (incluso 0). $\langle \mathbb{N}; \leq \rangle$ è ovviamente un poset (una catena).
Consideriamo la relazione $\preccurlyeq$ tale che $x \preccurlyeq y$ se e solo se x divide y:

$$x \preccurlyeq y \iff y = kx \quad \text{per qualche } k \in \mathbb{N}.$$

- $x = kx$ con $k = 1$,
- $y = k_1 x$ e $z = k_2 y$ allora $z = (k_2 k_1)x$,
- $y = k_1 x$ e $x = k_2 y$ implica $x = (k_2 k_1)x$, quindi:
  - se $x = 0$ anche $y = k_1 x = 0$,
  - se $x \neq 0$ deve essere $k_2 k_1 = 1$ e dunque $k_1 = k_2 = 1$.

Quindi anche $\langle \mathbb{N}; \preccurlyeq \rangle$ è un poset.

Insieme delle parti

Un esempio fondamentale è $\langle \wp(X); \subseteq \rangle$, l'insieme dei sottoinsiemi di un insieme X, ordinato da $\subseteq$. Se $A, B, C \in \wp(X)$ abbiamo

- $A \subseteq A$,
- $A \subseteq B$ e $B \subseteq C$ implica $A \subseteq C$,
- $A \subseteq B$ e $B \subseteq A$ implica $A = B$.

Insieme delle parti

Un esempio fondamentale è $\langle \wp(X); \subseteq \rangle$, l'insieme dei sottoinsiemi di un insieme X, ordinato da $\subseteq$. Se $A, B, C \in \wp(X)$ abbiamo

- $A \subseteq A$,
- $A \subseteq B$ e $B \subseteq C$ implica $A \subseteq C$,
- $A \subseteq B$ e $B \subseteq A$ implica $A = B$.

Per esempio, se $X = \{a, b, c\}$ avremo

$$\wp(X) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, X\}$$

dove $\{a, b\} \not\subseteq \{a, c\}$ e $\{a, c\} \not\subseteq \{a, b\}$.

Predicati

Consideriamo i predicati su una variabile $p(x)$, dove $x \in X$ e X è il nostro universo del discorso.

Possiamo usare $\Longrightarrow$ come relazione tra i predicati.
Avremo:

Predicati

Consideriamo i predicati su una variabile $p(x)$, dove $x \in X$ e X è il nostro universo del discorso.

Possiamo usare $\Longrightarrow$ come relazione tra i predicati. Avremo:

- $p(x) \Longrightarrow p(x)$ (propr. riflessiva);

Predicati

Consideriamo i predicati su una variabile $p(x)$, dove $x \in X$ e X è il nostro universo del discorso.

Possiamo usare $\Longrightarrow$ come relazione tra i predicati. Avremo:

- $p(x) \Longrightarrow p(x)$ (propr. riflessiva);
- $p(x) \Longrightarrow q(x)$ e $q(x) \Longrightarrow r(x)$ implica che $p(x) \Longrightarrow r(x)$ (propr. transitiva).

Predicati

Consideriamo i predicati su una variabile $p(x)$, dove $x \in X$ e X è il nostro universo del discorso.

Possiamo usare $\Longrightarrow$ come relazione tra i predicati. Avremo:

- $p(x) \Longrightarrow p(x)$ (propr. riflessiva);
- $p(x) \Longrightarrow q(x)$ e $q(x) \Longrightarrow r(x)$ implica che $p(x) \Longrightarrow r(x)$ (propr. transitiva).

Manca la proprietà antisimmetrica: $p(x) \Longrightarrow q(x)$ e $q(x) \Longrightarrow p(x)$ non implica $p(x) = q(x)$ ma solo che $p(x)$ e $q(x)$ sono equivalenti.

Intervalli di interi

Consideriamo l'insieme

$$\mathbb{I} = \{\, [a, b] \mid a, b \in \mathbb{Z} \text{ con } a \leq b \,\}.$$

Dal momento che

$$[a, b] = \{\, x \in \mathbb{Z} \mid a \leq x \leq b \,\},$$

$\mathbb{I}$ è un sottoinsieme di $\wp(\mathbb{Z})$.

Lo ordiniamo in base all'ordinamento indotto da $\langle \wp(\mathbb{Z}); \subseteq \rangle$.

$$[a, b] \leq [c, d] \iff [a, b] \subseteq [c, d] \iff c \leq a \leq b \leq d.$$

Poset: motivazione

Le proprietà che cerchiamo per ogni punto di un programma sono elementi del poset $\langle \wp(Env); \subseteq \rangle$.

Poset: motivazione

Le proprietà che cerchiamo per ogni punto di un programma sono elementi del poset $\langle \wp(Env); \subseteq \rangle$.

La relazione d'ordine in $\langle \wp(Env); \subseteq \rangle$ ci fornisce la relazione di implicazione tra le varie proprietà e la relazione di precisione relativa.

Poset: motivazione

Le proprietà che cerchiamo per ogni punto di un programma sono elementi del poset $\langle \wp(Env); \subseteq \rangle$.

La relazione d'ordine in $\langle \wp(Env); \subseteq \rangle$ ci fornisce la relazione di implicazione tra le varie proprietà e la relazione di precisione relativa.

Nell'astrazione dobbiamo sostituire questo dominio concreto con un altro, astratto.

Poset: motivazione

Le proprietà che cerchiamo per ogni punto di un programma sono elementi del poset $\langle \wp(Env); \subseteq \rangle$.

La relazione d'ordine in $\langle \wp(Env); \subseteq \rangle$ ci fornisce la relazione di implicazione tra le varie proprietà e la relazione di precisione relativa.

Nell'astrazione dobbiamo sostituire questo dominio concreto con un altro, astratto.

Vogliamo che anche nel dominio astratto ci sia una relazione d'ordine che "modelli" quella concreta.

Poset: motivazione

Le proprietà che cerchiamo per ogni punto di un programma sono elementi del poset $\langle \wp(Env); \subseteq \rangle$.

La relazione d'ordine in $\langle \wp(Env); \subseteq \rangle$ ci fornisce la relazione di implicazione tra le varie proprietà e la relazione di precisione relativa.

Nell'astrazione dobbiamo sostituire questo dominio concreto con un altro, astratto.

Vogliamo che anche nel dominio astratto ci sia una relazione d'ordine che "modelli" quella concreta.

Vogliamo quindi che anche il dominio astratto sia un poset.

Diagrammi di Hasse

Se P è finito $\langle P; \leq \rangle$ può essere rappresentato con un *diagramma di Hasse*.

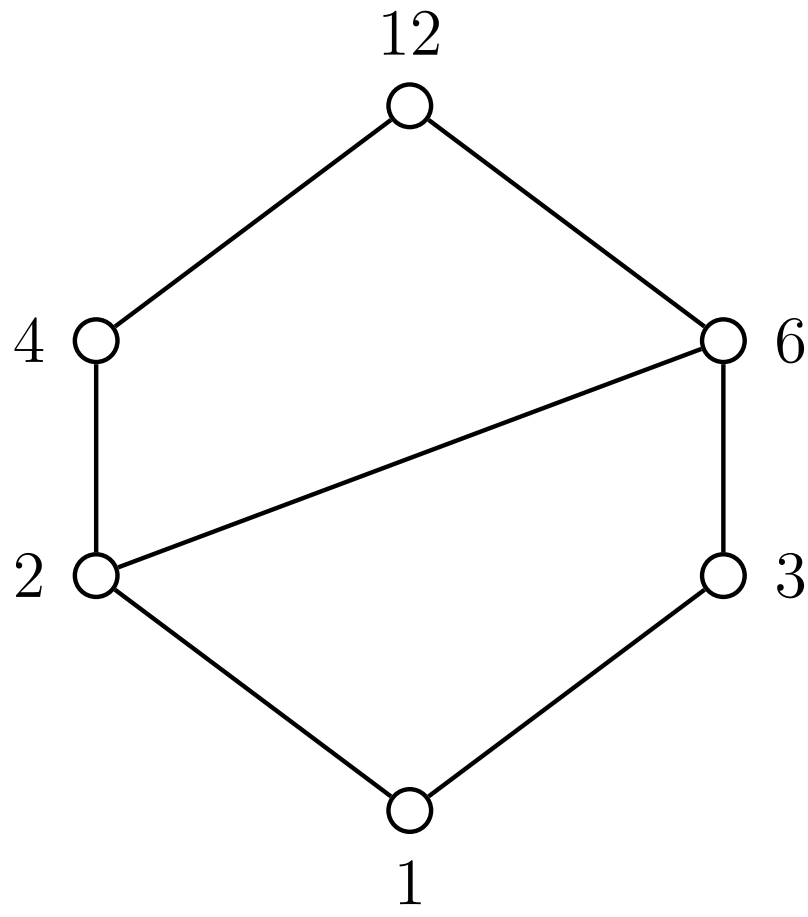
Gli elementi di P sono rappresentati con piccoli cerchi, uniti da una linea se uno “copre” l’altro (è maggiore e non ci sono altri elementi tra i due)

Se y copre x, allora il cerchio di y deve stare più in alto di quello di x.

Nessun cerchio deve toccare una linea se non c’è relazione tra l’elemento corrispondente e i due connessi dalla linea.

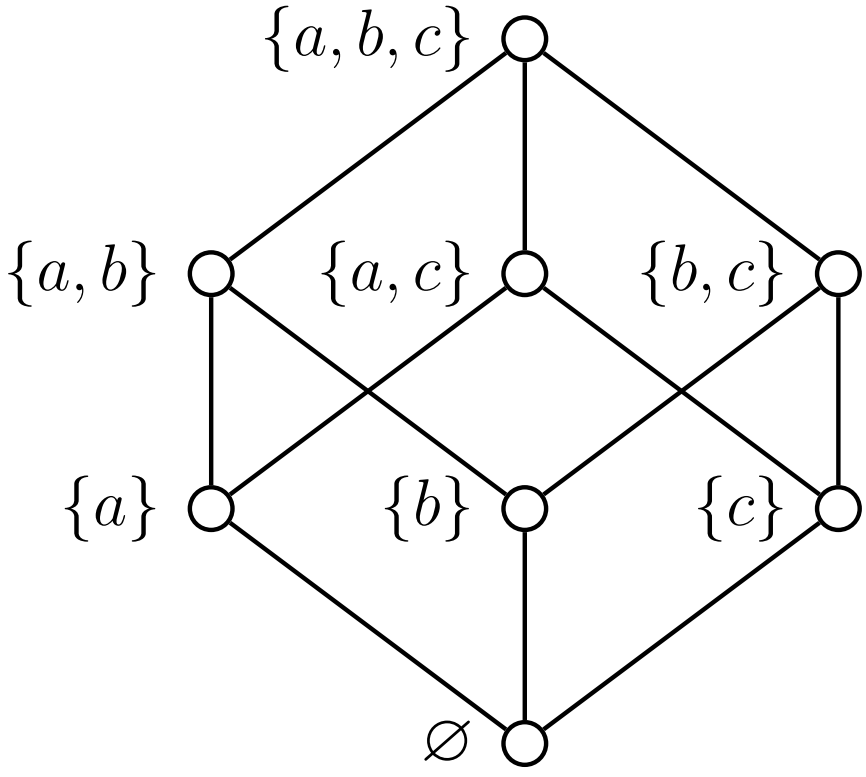
Diagrammi di Hasse: esempi

Diagramma di Hasse di $\downarrow 12$ in $\langle \mathbb{N}; \preccurlyeq \rangle$.



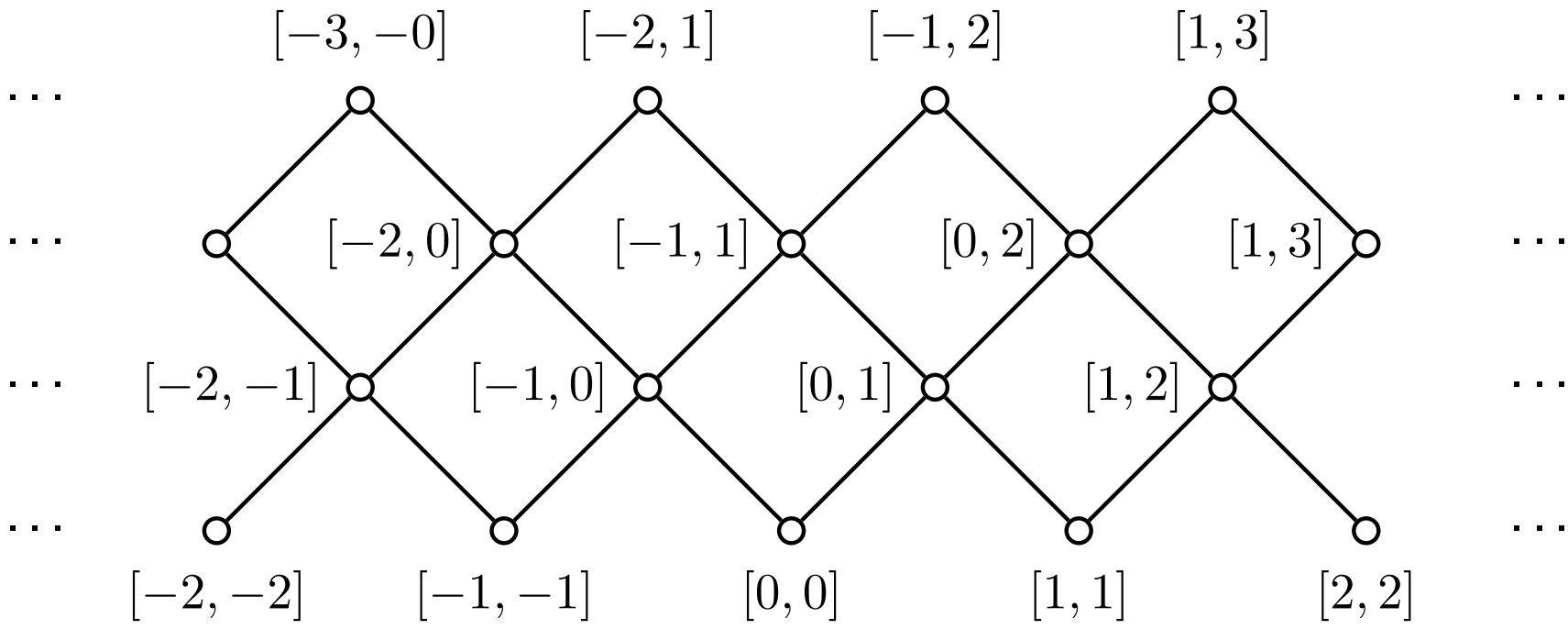
Diagrammi di Hasse: esempi

Diagramma di Hasse di $\langle \wp(\{a, b, c\}); \subseteq \rangle$.



(Pseudo-)diagramma di Hasse

Poset $\langle \mathbb{I}; \leq \rangle$.



Funzioni tra poset

Siano $\langle P; \leq \rangle$ e $\langle Q; \sqsubseteq \rangle$ due poset.

Una funzione $f : P \to Q$ è *monotona* se, per ogni $x, y \in P$,

$$x \leq y \implies f(x) \sqsubseteq f(y).$$

Funzioni tra poset

Siano $\langle P; \leq \rangle$ e $\langle Q; \sqsubseteq \rangle$ due poset.

Una funzione $f : P \to Q$ è *monotona* se, per ogni $x, y \in P$,

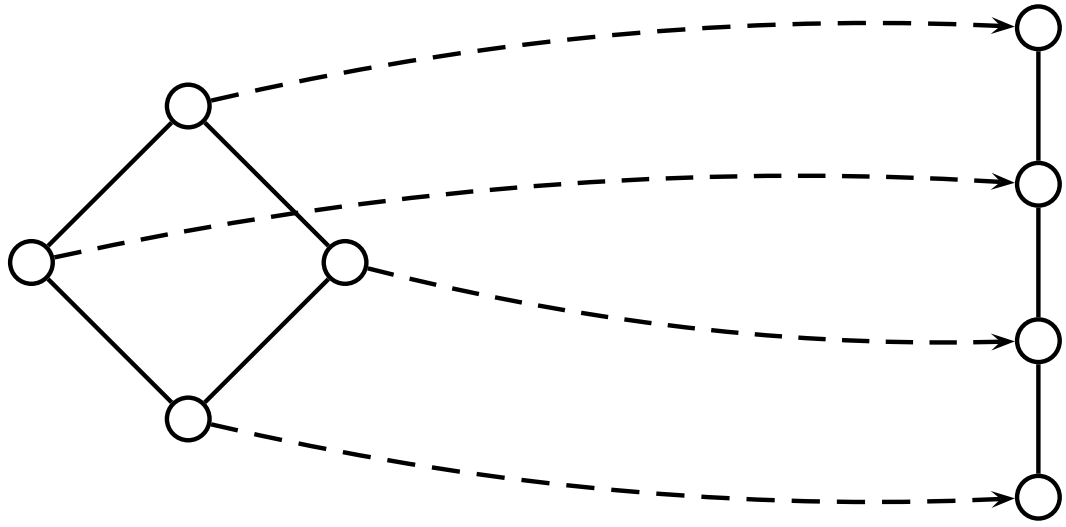
$$x \leq y \implies f(x) \sqsubseteq f(y).$$

Una funzione bigettiva $f : P \to Q$ è un *isomorfismo* se, per ogni $x, y \in P$,

$$x \leq y \iff f(x) \sqsubseteq f(y).$$

(equivalentemente se è monotona e anche la sua inversa è monotona).

Funzioni tra poset: esempio



La funzione è monotona e bigettiva, ma non è un isomorfismo.

Funzioni monotone: motivazione

Tutte le funzioni usate nella definizione della collecting semantics sono monotone.

Per esempio, siano $R_1, R_2 \in \wp(Env)$. Allora

$$R_1 \subseteq R_2 \implies Assign_{x=e}(R_1) \subseteq Assign_{x=e}(R_2).$$

Questo è una proprietà molto naturale, che in più facilita la definizione e la soluzione del sistema di equazioni.

Estremo superiore

Sia $\langle P; \leq \rangle$ un poset e $X \subseteq P$.

Un elemento $a \in P$ è un *maggiorante* di X se

$$x \leq a \quad \text{per ogni } x \in X.$$

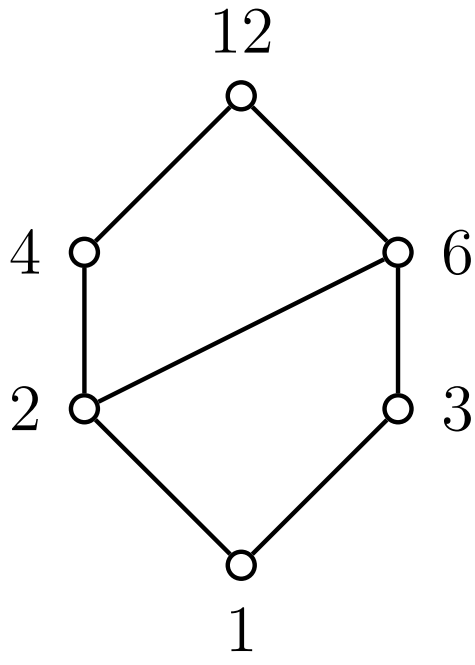
Diremo che $s \in P$ è l'estremo superiore di X, scritto

$$s = \sup X \qquad \text{oppure} \qquad s = \bigvee X$$

se s è il minimo dei maggioranti di X.

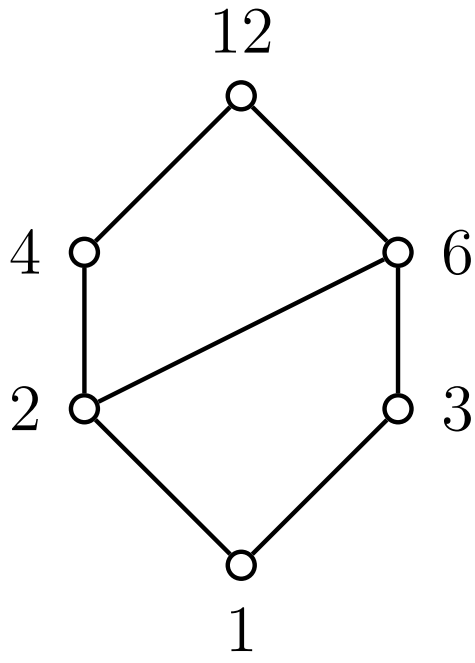
Se $X = \{a, b\}$ si scrive anche $s = a \vee b$.

Estremo superiore: esempi



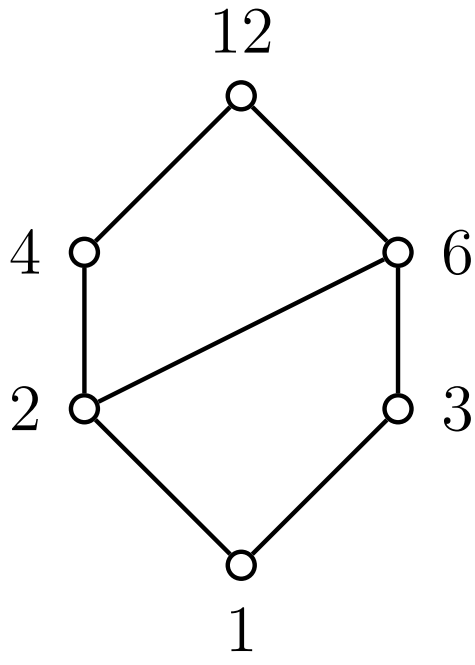
- $2 \vee 4 =$
- $2 \vee 12 =$
- $2 \vee 3 =$
- $4 \vee 6 =$
- $4 \vee 3 =$

Estremo superiore: esempi



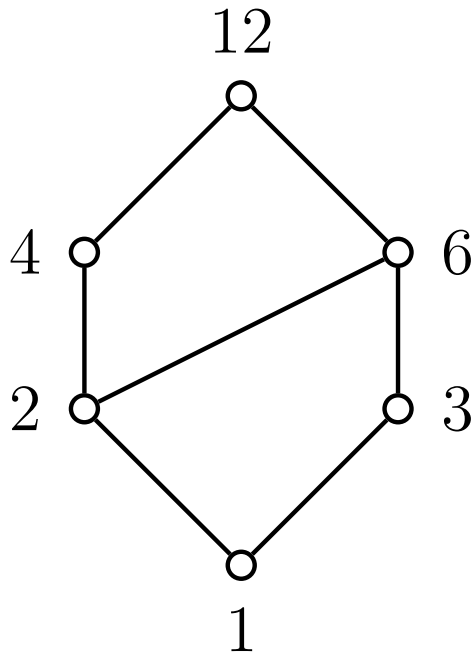
- $2 \vee 4 = 4$
- $2 \vee 12 =$
- $2 \vee 3 =$
- $4 \vee 6 =$
- $4 \vee 3 =$

Estremo superiore: esempi



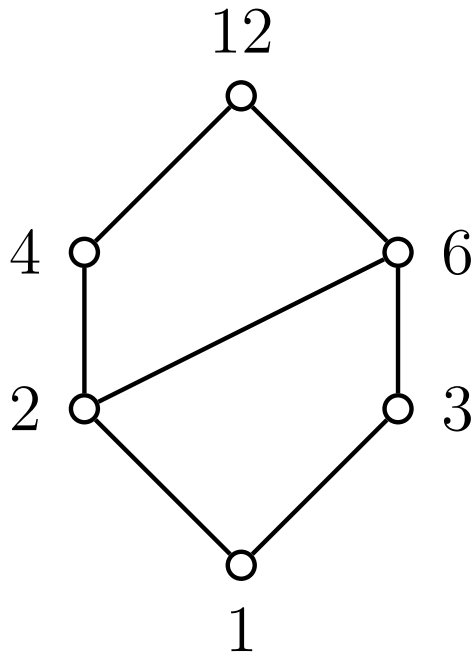
- $2 \vee 4 = 4$
- $2 \vee 12 = 12$
- $2 \vee 3 =$
- $4 \vee 6 =$
- $4 \vee 3 =$

Estremo superiore: esempi



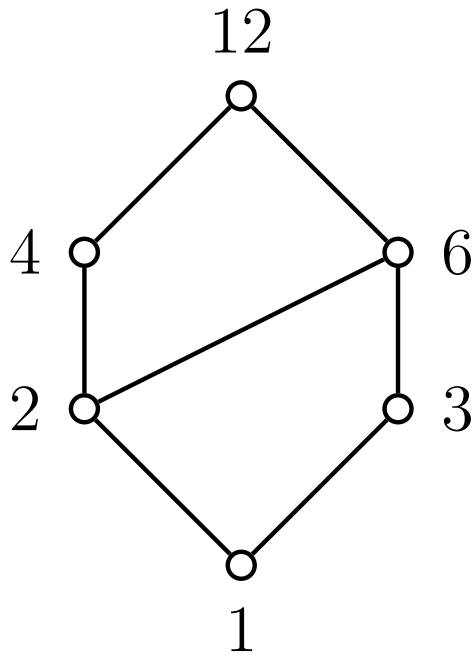
- $2 \vee 4 = 4$
- $2 \vee 12 = 12$
- $2 \vee 3 = 6$
- $4 \vee 6 =$
- $4 \vee 3 =$

Estremo superiore: esempi



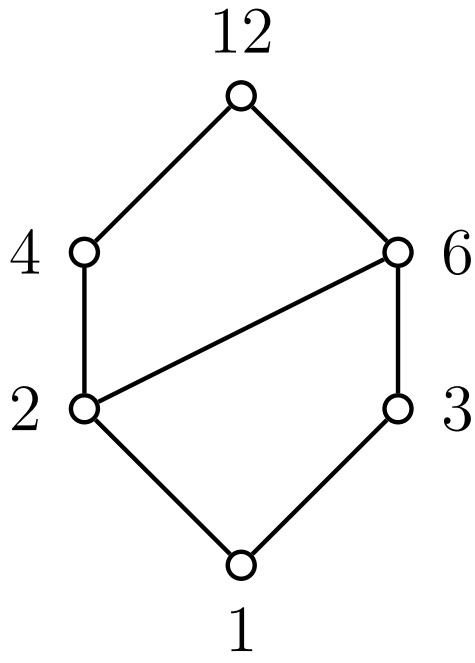
- $2 \vee 4 = 4$
- $2 \vee 12 = 12$
- $2 \vee 3 = 6$
- $4 \vee 6 = 12$
- $4 \vee 3 =$

Estremo superiore: esempi



- $2 \vee 4 = 4$
- $2 \vee 12 = 12$
- $2 \vee 3 = 6$
- $4 \vee 6 = 12$
- $4 \vee 3 = 12$

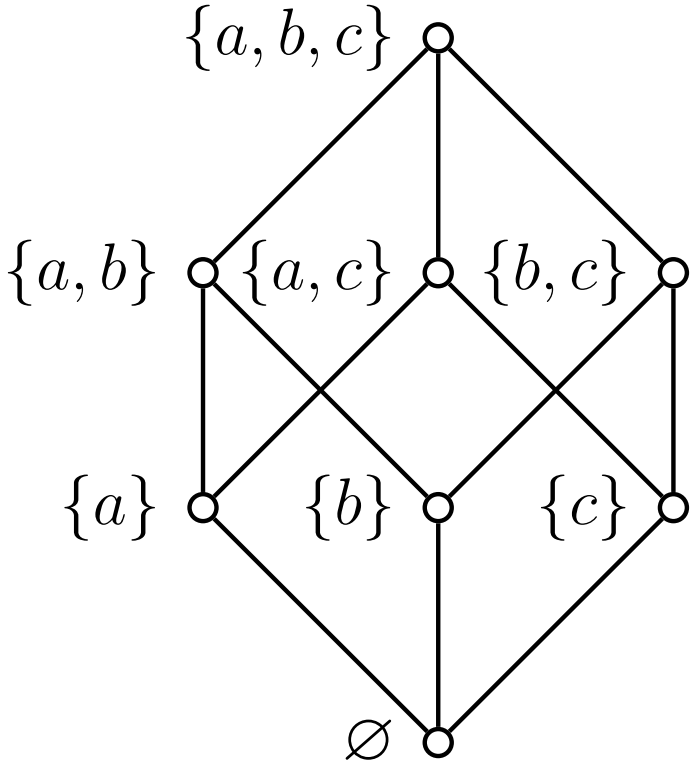
Estremo superiore: esempi



- $2 \vee 4 = 4$
- $2 \vee 12 = 12$
- $2 \vee 3 = 6$
- $4 \vee 6 = 12$
- $4 \vee 3 = 12$

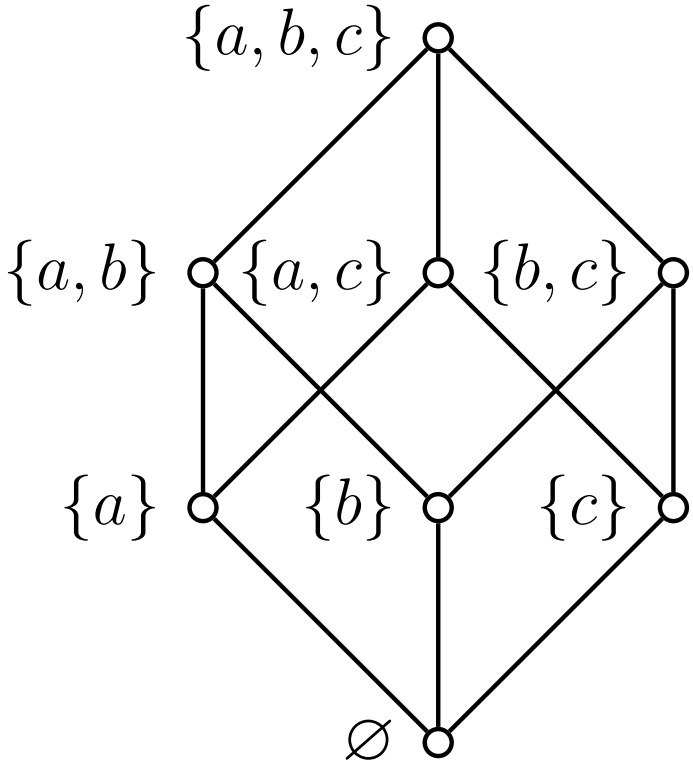
Cosa rappresenta l'estremo superiore in $\langle \mathbb{N}; \preccurlyeq \rangle$?

Estremo superiore: esempi



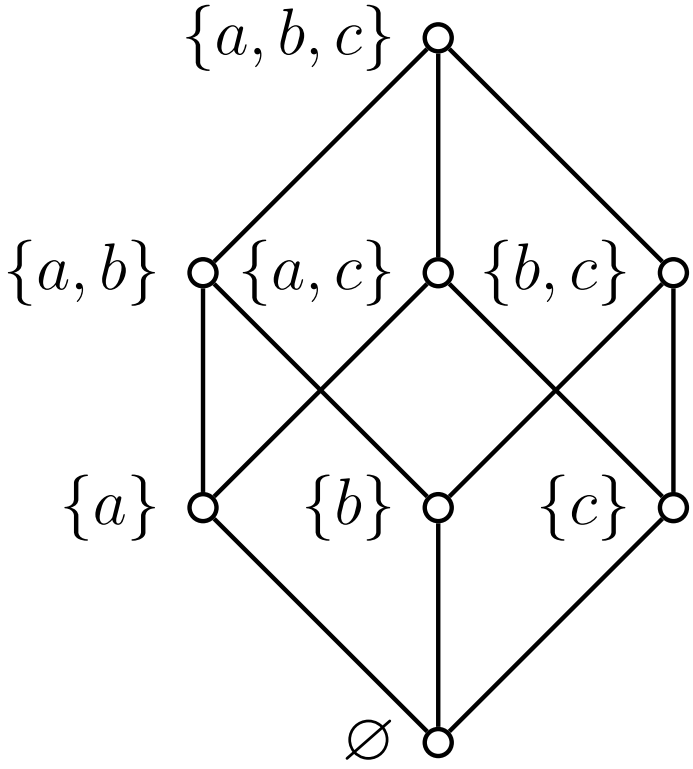
- $\{a\} \vee \{a,b\} =$
- $\{a\} \vee \{c\} =$
- $\{b\} \vee \{a,c\} =$
- $\varnothing \vee \{a,c\} =$

Estremo superiore: esempi



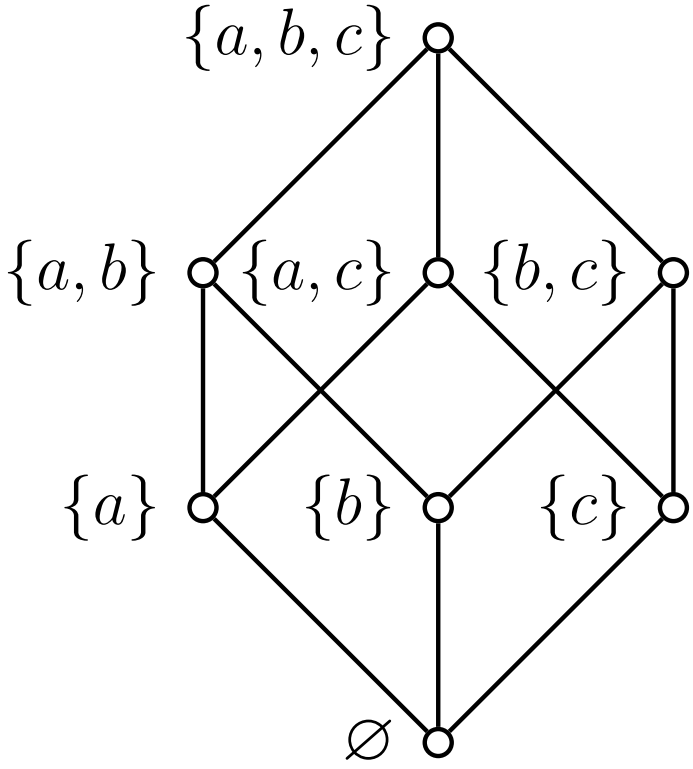
- $\{a\} \vee \{a,b\} = \{a,b\}$
- $\{a\} \vee \{c\} =$
- $\{b\} \vee \{a,c\} =$
- $\varnothing \vee \{a,c\} =$

Estremo superiore: esempi



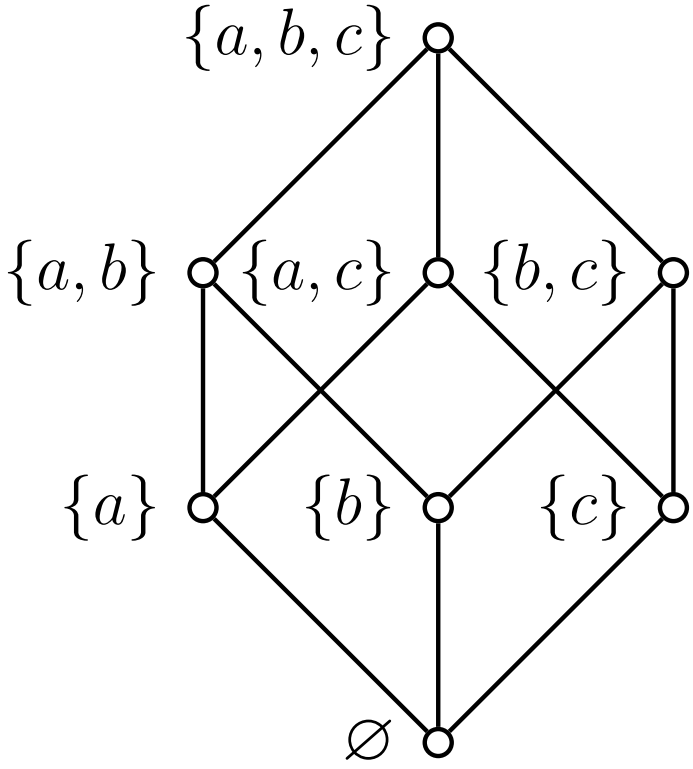
- $\{a\} \vee \{a,b\} = \{a,b\}$
- $\{a\} \vee \{c\} = \{a,c\}$
- $\{b\} \vee \{a,c\} =$
- $\varnothing \vee \{a,c\} =$

Estremo superiore: esempi



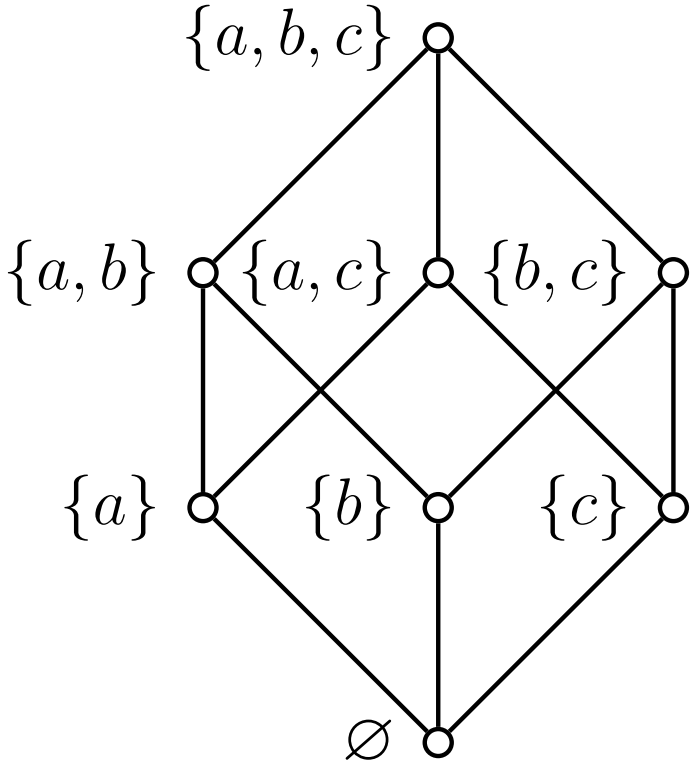
- $\{a\} \vee \{a,b\} = \{a,b\}$
- $\{a\} \vee \{c\} = \{a,c\}$
- $\{b\} \vee \{a,c\} = \{a,b,c\}$
- $\varnothing \vee \{a,c\} =$

Estremo superiore: esempi



- $\{a\} \vee \{a,b\} = \{a,b\}$
- $\{a\} \vee \{c\} = \{a,c\}$
- $\{b\} \vee \{a,c\} = \{a,b,c\}$
- $\varnothing \vee \{a,c\} = \{a,c\}$

Estremo superiore: esempi

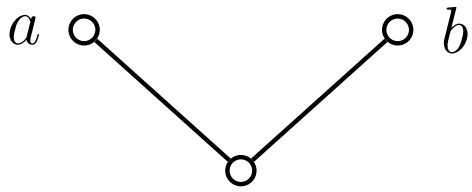


- $\{a\} \vee \{a,b\} = \{a,b\}$
- $\{a\} \vee \{c\} = \{a,c\}$
- $\{b\} \vee \{a,c\} = \{a,b,c\}$
- $\varnothing \vee \{a,c\} = \{a,c\}$

Cosa rappresenta l'estremo superiore in $\langle \wp(X); \subseteq \rangle$?

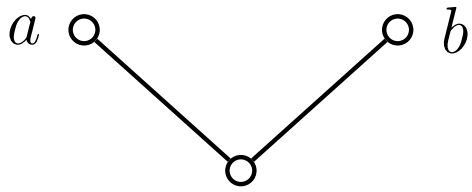
Estremo superiore: controesempi

Estremo superiore: controesempi

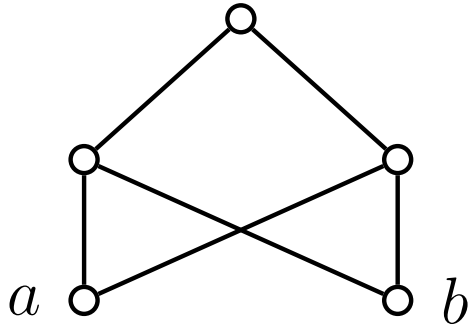


$$a \vee b = ?$$

Estremo superiore: controesempi

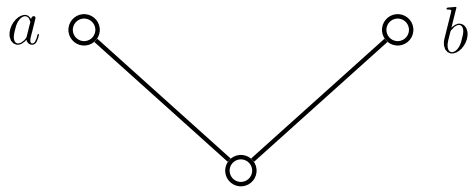


$a \vee b = ?$

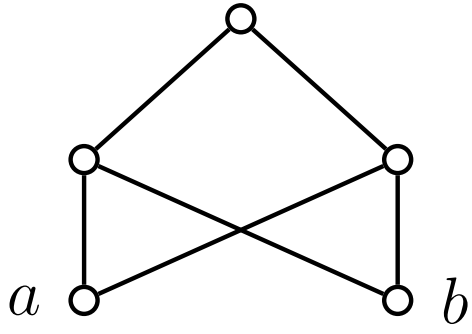


$a \vee b = ?$

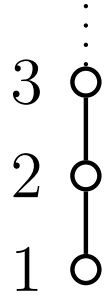
Estremo superiore: controesempi



$a \vee b = ?$



$a \vee b = ?$



$\bigvee\{1, 2, 3, \ldots\} = ?$

Estremo superiore: casi particolari

Sia $\langle P; \leq \rangle$ un poset.

Estremo superiore: casi particolari

Sia $\langle P; \leq \rangle$ un poset.

$\bigvee P$, se esiste, è il *massimo* di P.

Estremo superiore: casi particolari

Sia $\langle P; \leq \rangle$ un poset.

$\bigvee P$, se esiste, è il *massimo* di P.

Cos'è $\bigvee \varnothing$?

Estremo superiore: casi particolari

Sia $\langle P; \leq \rangle$ un poset.

$\bigvee P$, se esiste, è il *massimo* di P.

Cos'è $\bigvee \varnothing$?

Per capirlo, chiediamoci chi sono i maggioranti di $\varnothing$. Sono quegli elementi $m \in P$ tali che

$$(\forall x \in \varnothing)\ x \leq m.$$

Estremo superiore: casi particolari

Sia $\langle P; \leq \rangle$ un poset.

$\bigvee P$, se esiste, è il *massimo* di P.

Cos'è $\bigvee \varnothing$?

Per capirlo, chiediamoci chi sono i maggioranti di $\varnothing$. Sono quegli elementi $m \in P$ tali che

$$(\forall x \in \varnothing)\ x \leq m.$$

Quindi tutti gli $m \in P$!

Estremo superiore: casi particolari

Sia $\langle P; \leq \rangle$ un poset.

$\bigvee P$, se esiste, è il *massimo* di P.

Cos'è $\bigvee \varnothing$?

Per capirlo, chiediamoci chi sono i maggioranti di $\varnothing$. Sono quegli elementi $m \in P$ tali che

$$(\forall x \in \varnothing)\ x \leq m.$$

Quindi tutti gli $m \in P$!

Ne deduciamo che $\bigvee \varnothing$, se esiste, è il *minimo* di P.

Estremo inferiore

Dualmente, il massimo dei minoranti di X, se esiste, è detto *estremo inferiore* e denotato

$$\inf X \qquad \text{oppure} \qquad \bigwedge X.$$

Se $X = \{a, b\}$ si scrive anche $a \wedge b$.

Estremo inferiore

Dualmente, il massimo dei minoranti di X, se esiste, è detto *estremo inferiore* e denotato

$$\inf X \qquad \text{oppure} \qquad \bigwedge X.$$

Se $X = \{a, b\}$ si scrive anche $a \wedge b$.

Cosa rappresenta l'estremo inferiore in $\langle \mathbb{N}; \preccurlyeq \rangle$?

Estremo inferiore

Dualmente, il massimo dei minoranti di X, se esiste, è detto *estremo inferiore* e denotato

$$\inf X \qquad \text{oppure} \qquad \bigwedge X.$$

Se $X = \{a, b\}$ si scrive anche $a \wedge b$.

Cosa rappresenta l'estremo inferiore in $\langle \mathbb{N}; \preccurlyeq \rangle$?

E in $\langle \wp(X); \subseteq \rangle$?

Estremo inferiore

Dualmente, il massimo dei minoranti di X, se esiste, è detto *estremo inferiore* e denotato

$$\inf X \qquad \text{oppure} \qquad \bigwedge X.$$

Se $X = \{a, b\}$ si scrive anche $a \wedge b$.

Cosa rappresenta l'estremo inferiore in $\langle \mathbb{N}; \preccurlyeq \rangle$?

E in $\langle \wp(X); \subseteq \rangle$?

Dato un poset $\langle P; \leq \rangle$, cosa sono $\bigwedge P$ e $\bigwedge \varnothing$?

$\sup$ **e** $\inf$**: motivazione**

Nella collecting semantics abbiamo usato l'operazione $\cup$ nel congiungimento dei cammini:

$$C_n = \bigcup_{m \in prec(n)} F_{m \to n}(C_m),$$

e l'operazione $\cap$ nella funzione associata ai test:

$$tBranch_b(R) = R \cap \{\, \rho \mid Bval(b, \rho) \,\}.$$

Queste rappresentano $\sup$ e $\inf$ nel poset $\langle \wp(Env); \subseteq \rangle$.

Avremo bisogno di operazioni corrispondenti anche nel dominio astratto.

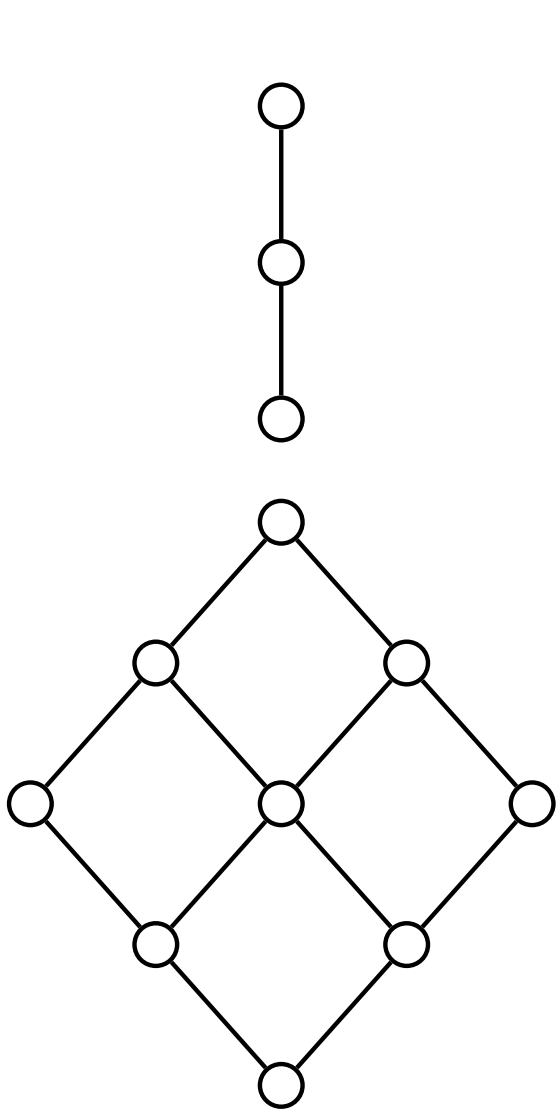
Reticoli

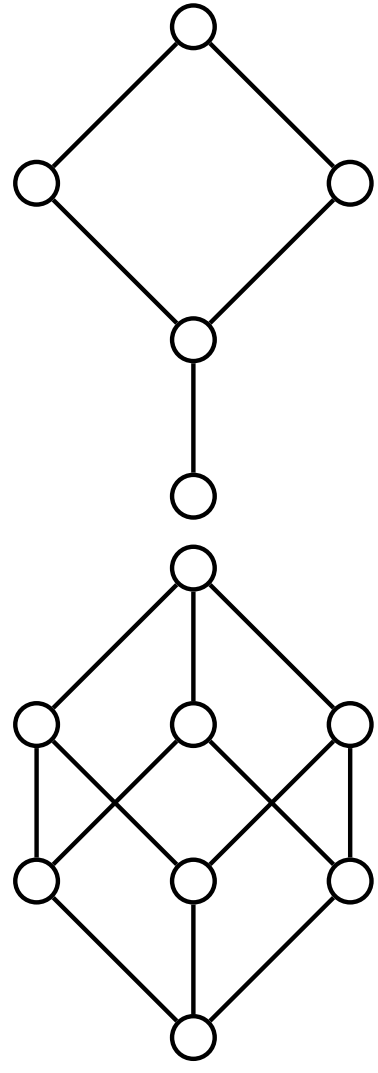
Un *reticolo* è un poset non vuoto in cui ogni coppia di elementi possiede un estremo superiore e inferiore.

Può anche essere visto come un'algebra con due operazioni binarie. In questo caso $\vee$ viene chiamato *join* e $\wedge$ *meet*.
Le due operazioni soddisfano le seguenti proprietà (più quelle duali, scambiando join e meet)

- *idempotenza*: $x \vee x = x$
- *simmetria*: $x \vee y = y \vee x$
- *associatività*: $(x \vee y) \vee z = x \vee (y \vee z)$
- *assorbimento*: $x \vee (y \wedge x) = x$

Reticoli: esempi





Funzioni tra reticoli

Siano $\langle P; \leq \rangle$ e $\langle Q; \leq \rangle$ sue reticoli.

Una funzione $f : P \to Q$ preserva i join se, per ogni $x, y \in P$,

$$f(x \vee y) = f(x) \vee f(y).$$

Dualmente f preserva i meet se, per ogni $x, y \in P$,

$$f(x \wedge y) = f(x) \wedge f(y).$$

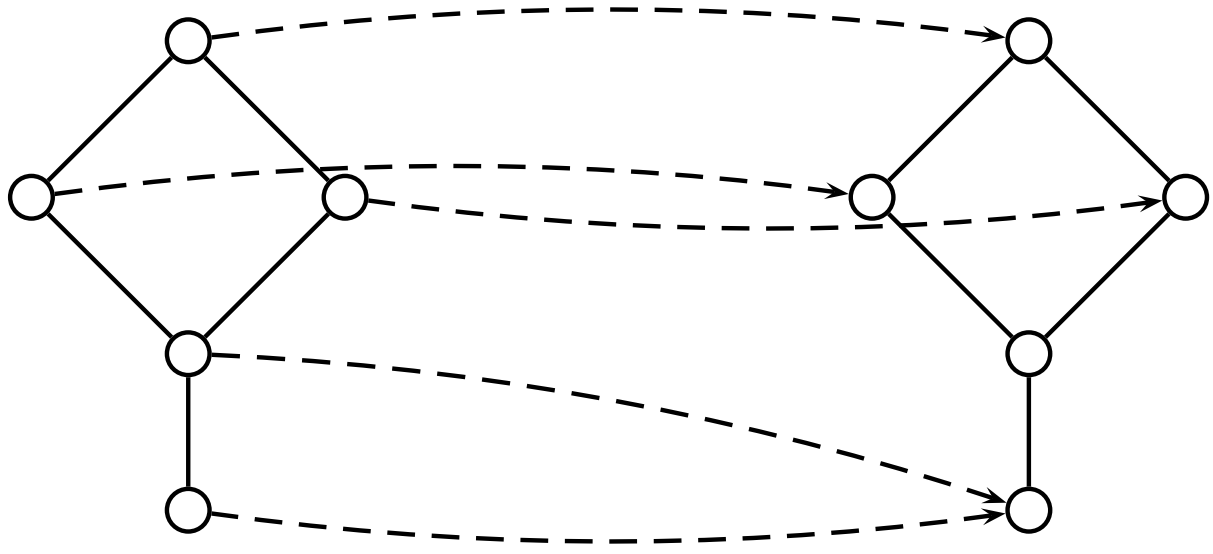
Funzioni tra reticoli

Siano $\langle P; \leq \rangle$ e $\langle Q; \leq \rangle$ sue reticoli, $f : P \to Q$ e $x, y \in P$. Le tre proposizioni seguenti sono equivalenti.

- f è monotona;
- $f(x) \vee f(y) \leq f(x \vee y)$;
- $f(x \wedge y) \leq f(x) \wedge f(y)$.

In particolare, se f preserva i join (o i meet) è monotona.

Funzioni tra reticoli: esempio



La funzione è monotona, preserva i join, ma non i meet.

Funzioni tra reticoli: esempio

Le funzioni $Assign_{x=e}$ preservano i join. Comunque si prendano $R_1, R_2 \in \wp(Env)$ avremo

$$Assign_{x=e}(R_1) \cup Assign_{x=e}(R_2) = Assign_{x=e}(R_1 \cup R_2).$$

Funzioni tra reticoli: esempio

Le funzioni $Assign_{x=e}$ preservano i join. Comunque si prendano $R_1, R_2 \in \wp(Env)$ avremo

$$Assign_{x=e}(R_1) \cup Assign_{x=e}(R_2) = Assign_{x=e}(R_1 \cup R_2).$$

Non preservano i meet. Per esempio:

$$Assign_{x=2}(\{[x \mapsto 0]\} \cap \{[x \mapsto 1]\}) = Assign_{x=2}(\varnothing) = \varnothing,$$

ma

$$Assign_{x=2}(\{[x \mapsto 0]\}) \cap Assign_{x=2}(\{[x \mapsto 1]\}) =$$
$$\{[x \mapsto 2]\} \cap \{[x \mapsto 2]\} = \{[x \mapsto 2]\}$$

Funzioni tra reticoli: esempio

Le funzioni $Assign_{x=e}$ preservano i join. Comunque si prendano $R_1, R_2 \in \wp(Env)$ avremo

$$Assign_{x=e}(R_1) \cup Assign_{x=e}(R_2) = Assign_{x=e}(R_1 \cup R_2).$$

Non preservano i meet. Per esempio:

$$Assign_{x=2}(\{[x \mapsto 0]\} \cap \{[x \mapsto 1]\}) = Assign_{x=2}(\varnothing) = \varnothing,$$

ma

$$Assign_{x=2}(\{[x \mapsto 0]\}) \cap Assign_{x=2}(\{[x \mapsto 1]\}) =$$
$$\{[x \mapsto 2]\} \cap \{[x \mapsto 2]\} = \{[x \mapsto 2]\}$$

Le funzioni $tBranch_b$ e $fBranch_b$ preservano join e meet.

Sottoreticoli

Sia $\langle P; \leq \rangle$ un reticolo e $S \subseteq P$. $\langle S; \leq \rangle$ è un sottoreticolo di P se i meet e i join calcolati in S coincidono con quelli calcolati in P.

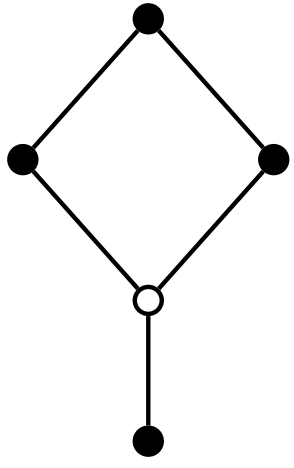
Equivalentemente, se S è un sottoinsieme di P *chiuso* rispetto a meet e join.

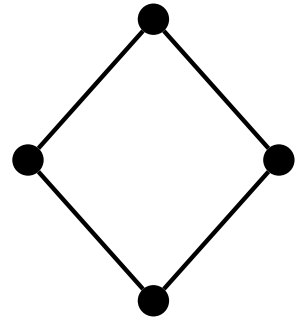
Sottoreticoli

Sia $\langle P; \leq \rangle$ un reticolo e $S \subseteq P$. $\langle S; \leq \rangle$ è un sottoreticolo di P se i meet e i join calcolati in S coincidono con quelli calcolati in P.

Equivalentemente, se S è un sottoinsieme di P *chiuso* rispetto a meet e join.

Notare che $\langle S; \leq \rangle$ può essere un reticolo senza essere un sottoreticolo:

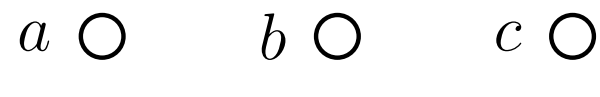




Costruzione di reticoli

Se S è un insieme qualunque (non ordinato), denotiamo con $\overline{S}$ l'insieme ordinato $\langle S; \leq \rangle$ in cui $x \leq x$ per ogni $x \in S$ e non ci sono altre relazioni.

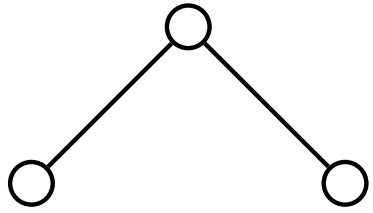
$S = \{a, b, c\}$



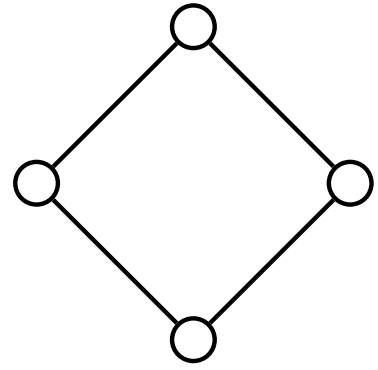
$\overline{S}$

Costruzione di reticoli

Dato un insieme ordinato $\langle P; \leq \rangle$ possiamo ottenere un nuovo insieme ordinato $P_\perp$ aggiungendo un nuovo elemento $\perp \notin P$ tale che $\perp \leq x$ per ogni $x \in P_\perp$.
(Dualmente per $P^\top$).



P



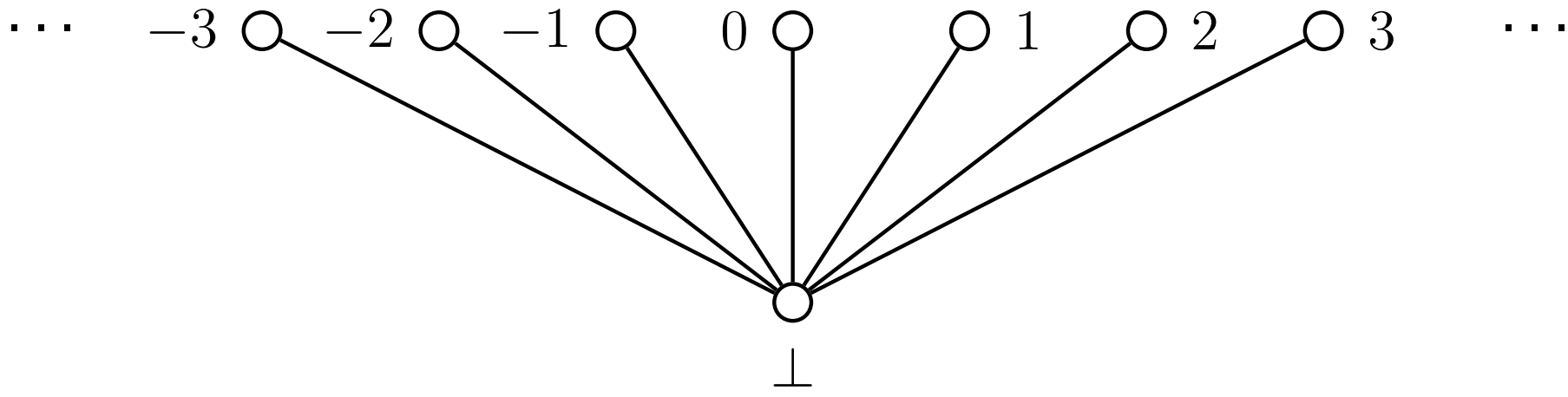
$P_\perp$

Costruzione di reticoli: esempio

Pseudo-diagramma di $\overline{\mathbb{Z}}$.

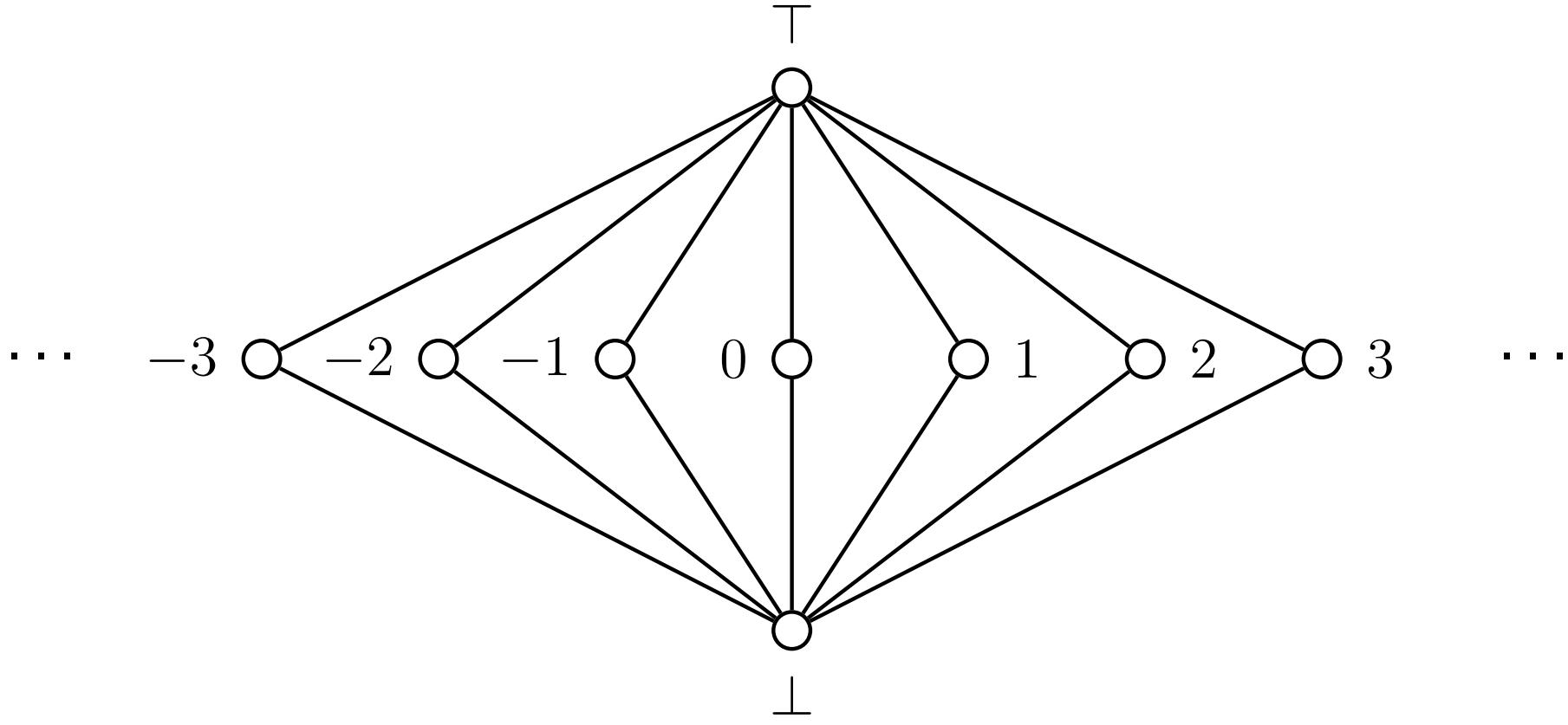
Costruzione di reticoli: esempio

Pseudo-diagramma di $\overline{\mathbb{Z}}_{\perp}$.



Costruzione di reticoli: esempio

Pseudo-diagramma di $\overline{\mathbb{Z}}_{\perp}^{\top}$.



Intervalli

L'insieme $\langle \mathbb{I}; \leq \rangle$ è un reticolo?

Intervalli

L'insieme $\langle \mathbb{I}; \leq \rangle$ è un reticolo?

No, se $[a, b] \cap [c, d] = \varnothing$ non esiste l'estremo inferiore.

Intervalli

L'insieme $\langle \mathbb{I}; \leq \rangle$ è un reticolo?

No, se $[a, b] \cap [c, d] = \varnothing$ non esiste l'estremo inferiore.

Possiamo farlo diventare un reticolo aggiungendo un minimo $\bot$ a $\mathbb{I}$. Denotiamo il nuovo insieme $\langle \mathbb{I}_\bot; \leq \rangle$, che è un reticolo.

Intervalli

L'insieme $\langle \mathbb{I}; \leq \rangle$ è un reticolo?

No, se $[a, b] \cap [c, d] = \varnothing$ non esiste l'estremo inferiore.

Possiamo farlo diventare un reticolo aggiungendo un minimo $\bot$ a $\mathbb{I}$. Denotiamo il nuovo insieme $\langle \mathbb{I}_\bot; \leq \rangle$, che è un reticolo.

Il reticolo $\langle \mathbb{I}_\bot; \leq \rangle$ è un *sottoreticolo* di $\langle \wp(\mathbb{Z}); \subseteq \rangle$?

Intervalli

L'insieme $\langle \mathbb{I}; \leq \rangle$ è un reticolo?

No, se $[a, b] \cap [c, d] = \varnothing$ non esiste l'estremo inferiore.

Possiamo farlo diventare un reticolo aggiungendo un minimo $\bot$ a $\mathbb{I}$. Denotiamo il nuovo insieme $\langle \mathbb{I}_\bot; \leq \rangle$, che è un reticolo.

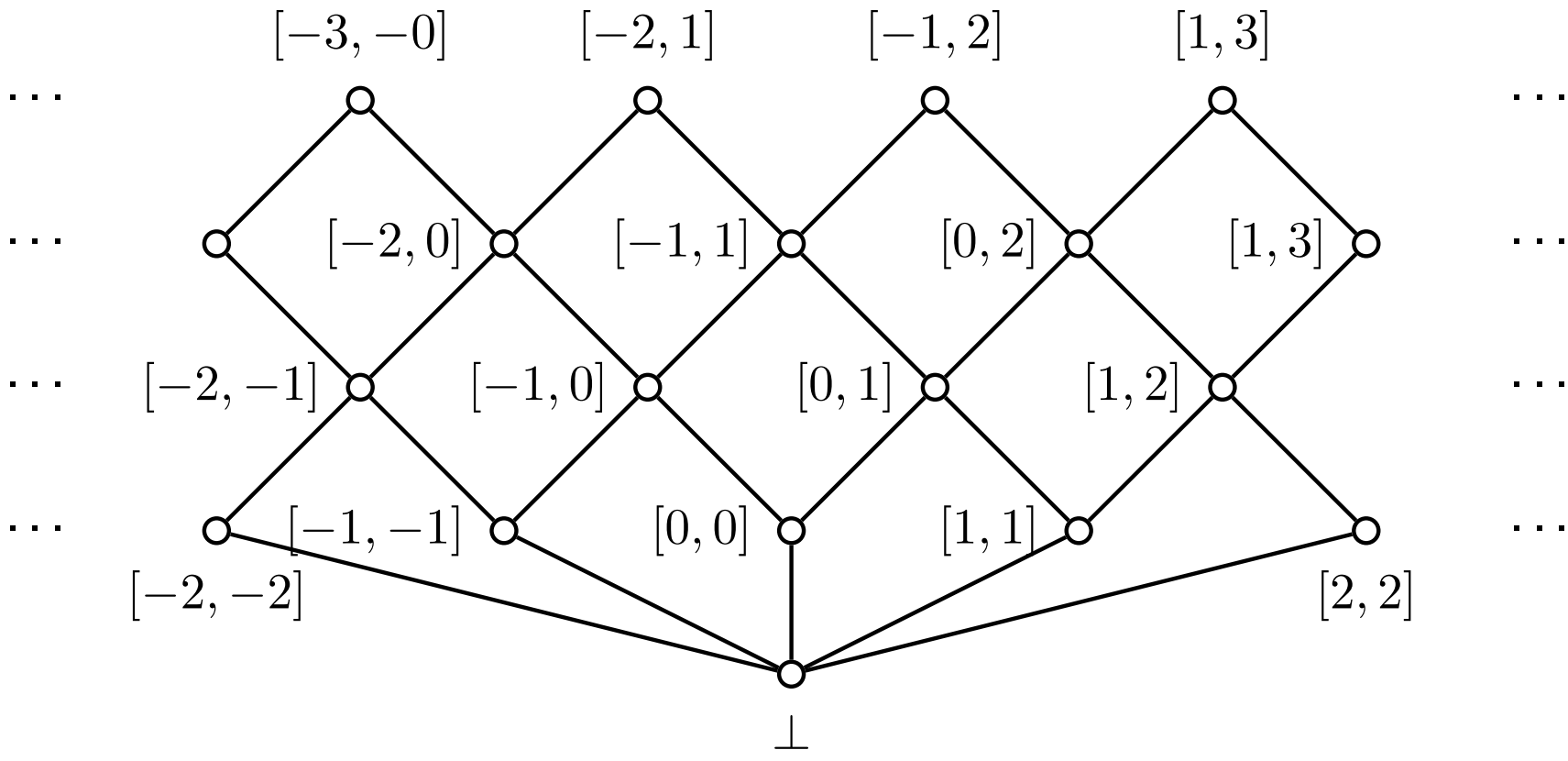
Il reticolo $\langle \mathbb{I}_\bot; \leq \rangle$ è un *sottoreticolo* di $\langle \wp(\mathbb{Z}); \subseteq \rangle$?

No. I meet coincidono, ma per es.

$$[-1, -1] \vee [1, 1] = [-1, 1] \neq \{-1, 1\} = [-1, -1] \cup [1, 1].$$

Intervalli: diagramma

Il reticolo $\langle \mathbb{I}_\perp ; \leq \rangle$.



Costruzione di reticoli: funzioni

Se $\langle P; \leq \rangle$ è un reticolo e S è un insieme, allora l'insieme delle funzioni da S in P, denotato $(S \to P)$, ordinato *puntualmente* è un reticolo.

Ordinamento puntuale. Siano $f, g \in (S \to P)$. Poniamo

$$f \leq g \iff f(x) \leq g(x)$$

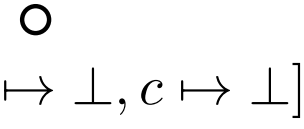
per ogni $x \in S$.

Costruzione di reticoli: funzioni

Prendiamo $S = \{a, b, c\}$ e $P = \{\bot, \top\}$.

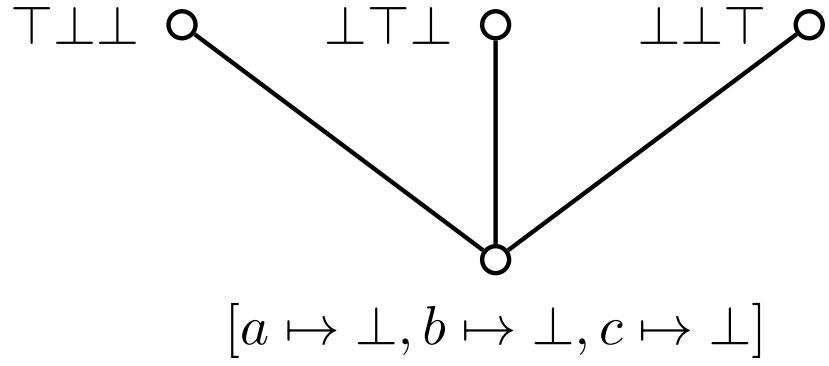
Costruzione di reticoli: funzioni

Prendiamo $S = \{a, b, c\}$ e $P = \{\bot, \top\}$.



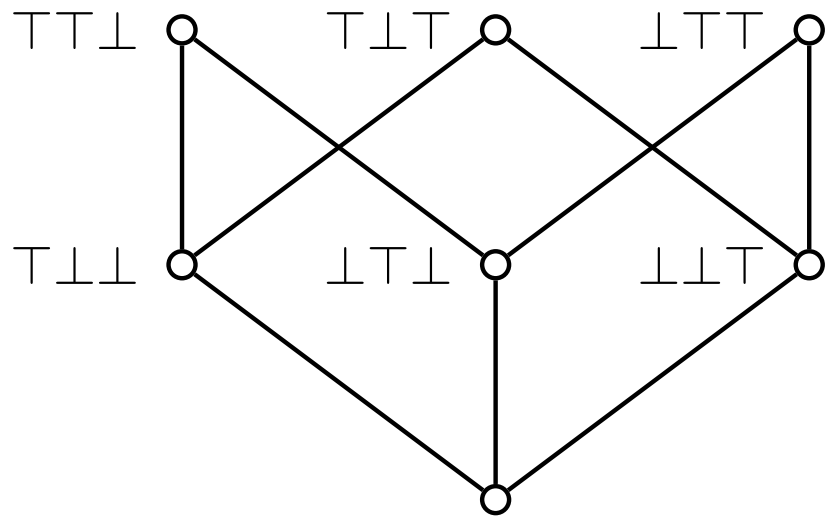
Costruzione di reticoli: funzioni

Prendiamo $S = \{a, b, c\}$ e $P = \{\bot, \top\}$.



Costruzione di reticoli: funzioni

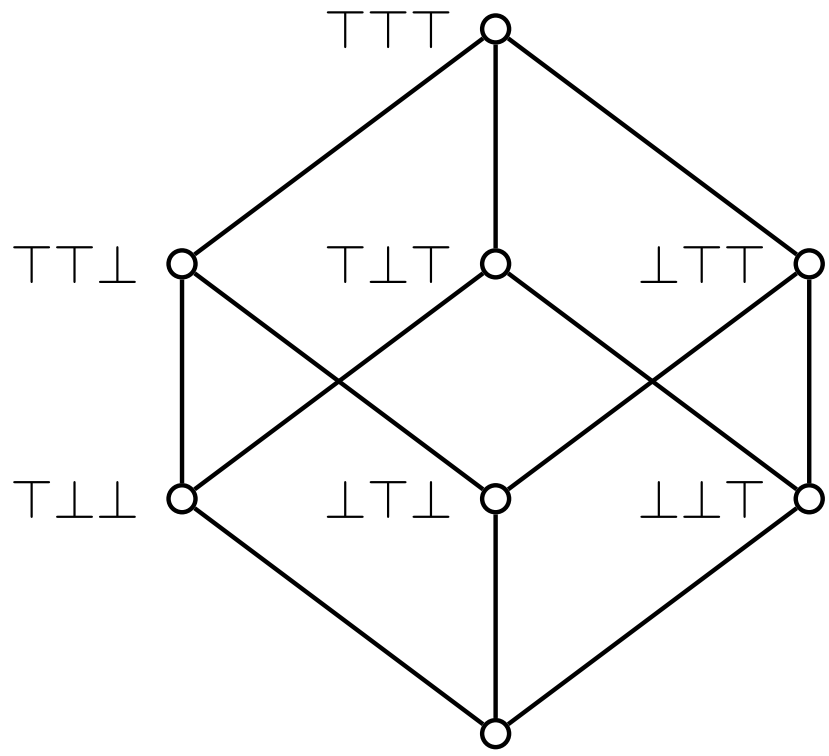
Prendiamo $S = \{a, b, c\}$ e $P = \{\bot, \top\}$.



$[a \mapsto \bot, b \mapsto \bot, c \mapsto \bot]$

Costruzione di reticoli: funzioni

Prendiamo $S = \{a, b, c\}$ e $P = \{\bot, \top\}$.



$[a \mapsto \bot, b \mapsto \bot, c \mapsto \bot]$

Costruzione di reticoli: prodotti

Se $\langle P; \leq \rangle$ e $\langle Q; \leq \rangle$ sono reticoli, allora l'insieme

$$P \times Q = \{ (a, b) \mid a \in P \text{ e } b \in Q \}$$

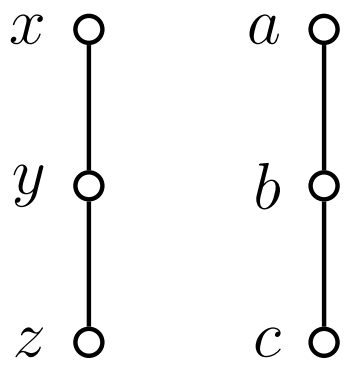
ordinato *puntualmente* è un reticolo.

Ordinamento puntuale. Siano $(a_1, b_1), (a_2, b_2) \in P \times Q$. Poniamo

$$(a_1, b_1) \leq (a_2, b_2) \iff a_1 \leq a_2 \text{ e } b_1 \leq b_2.$$

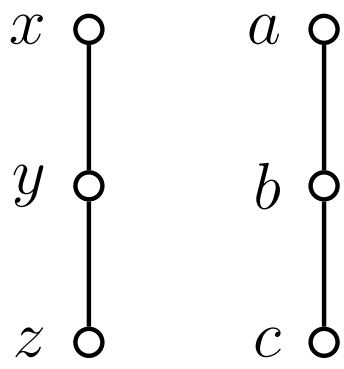
Costruzione di reticoli: prodotti

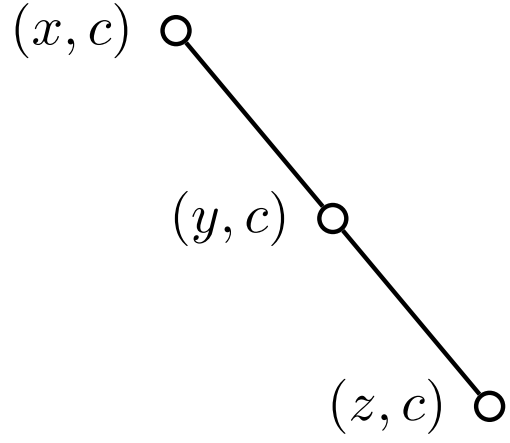
Prendiamo $P = \{x, y, z\}$ e $Q = \{a, b, c\}$.



Costruzione di reticoli: prodotti

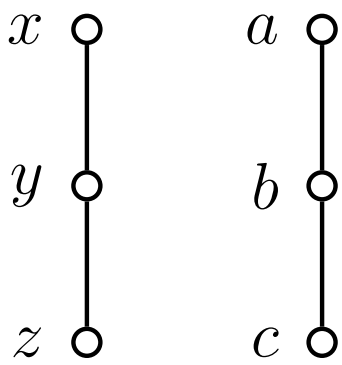
Prendiamo $P = \{x, y, z\}$ e $Q = \{a, b, c\}$.

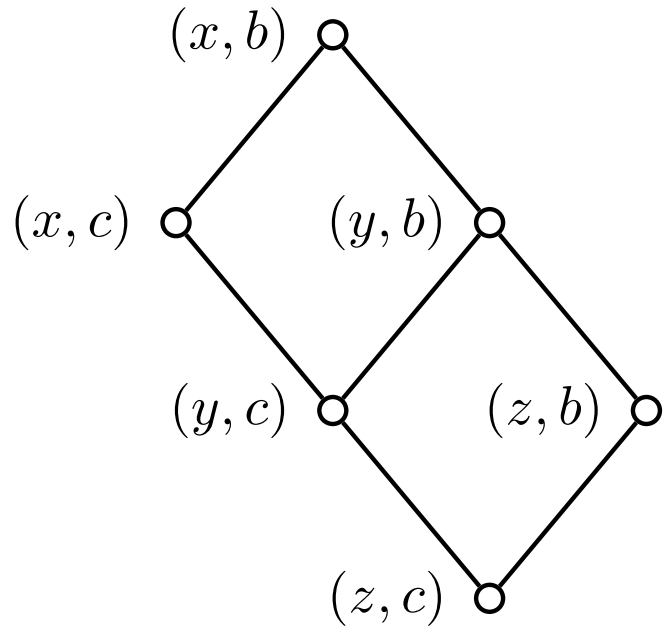




Costruzione di reticoli: prodotti

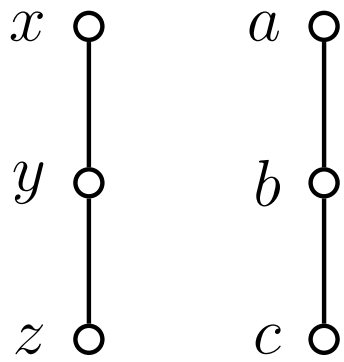
Prendiamo $P = \{x, y, z\}$ e $Q = \{a, b, c\}$.

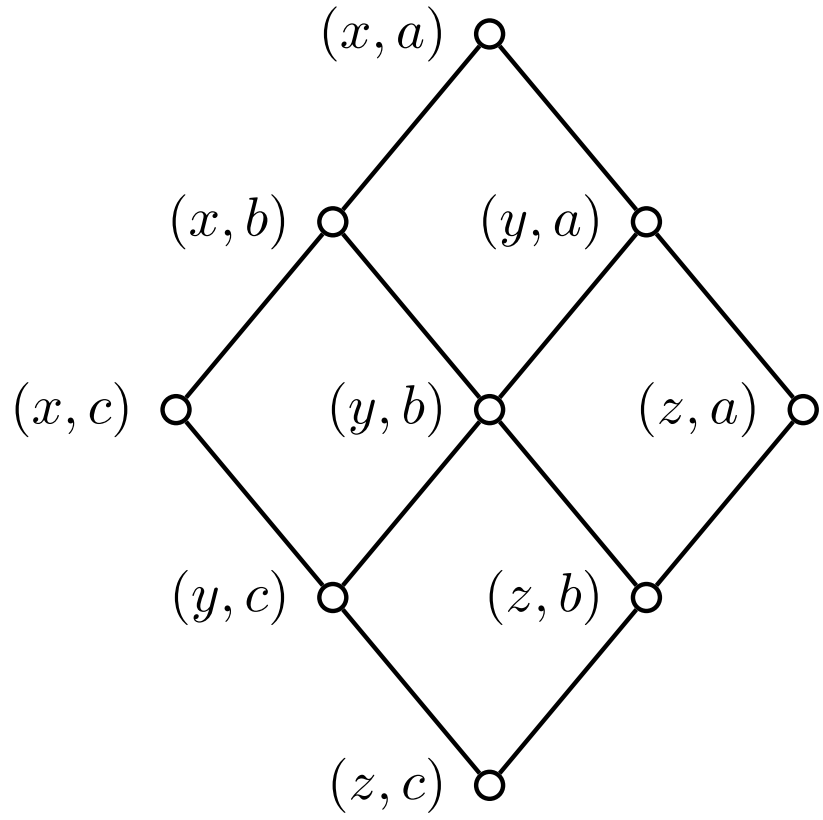




Costruzione di reticoli: prodotti

Prendiamo $P = \{x, y, z\}$ e $Q = \{a, b, c\}$.





Prodotti: motivazione

I contesti della collecting semantics sono elementi del reticolo $\langle \wp(Env); \subseteq \rangle$.

I vettori $\underline{C} = (C_1, \ldots, C_N)$ sono elementi di

$$\langle \wp(Env) \times \cdots \times \wp(Env); \subseteq \rangle$$

che è anch'esso un reticolo, con l'ordinamento puntuale.

Reticoli completi

Un reticolo si dice *completo* se ogni suo sottoinsieme possiede un estremo superiore e un estremo inferiore.

In un reticolo ogni insieme di uno o due elementi possiede $\sup$ e $\inf$. Si mostra per induzione che allora ogni sottoinsieme *finito* e *non vuoto* possiede $\sup$ e $\inf$.

Per avere un reticolo completo bisogna avere anche $\sup$ e $\inf$ di $\varnothing$ (quindi un massimo e un minimo) e dei sottoinsiemi infiniti.

I reticoli finiti sono sempre completi.

Condizioni sufficienti

- Un reticolo che possiede tutti gli $\inf$ possiede anche tutti i $\sup$, quindi è completo.
- Un reticolo che possiede un minimo e i $\sup$ di tutti gli insiemi non vuoti è completo.
- Un reticolo che possiede un minimo e non contiene catene ascendenti infinite è completo.

Valgono anche le condizioni duali.

Reticoli completi: esempi

- $\langle \mathbb{N}; \leq \rangle$ è un reticolo, ma non è un reticolo completo.
- $\langle \mathbb{N}; \preccurlyeq \rangle$ è un reticolo completo.
- $\langle \wp(X); \subseteq \rangle$ è un reticolo completo per qualunque insieme X.

Quindi $\langle \wp(Env); \subseteq \rangle$ è un reticolo completo.

Lo stesso vale per $\langle \wp(Env) \times \cdots \times \wp(Env); \subseteq \rangle$.

Questo esempio ci spinge a richiedere che anche i domini astratti siano reticoli completi (se possibile).

Funzioni ed estremi arbitrari

Siano $\langle P; \leq \rangle$ e $\langle Q; \leq \rangle$ due reticoli completi e $f : P \to Q$. Si dice f preserva join *arbitrari* se, per ogni $A \subseteq P$, abbiamo

$$f(\bigvee A) = \bigvee f(A)$$

(dove $f(A) = \{ f(a) \mid a \in A \}$).

Dualmente per i meet arbitrari.

Notare che è ammesso $A = \varnothing$, quindi se f preserva join (meet) arbitrari preserva anche il minimo (massimo) di P.

Intervalli

Il reticolo $\langle \mathbb{I}_\perp ; \leq \rangle$ è completo?

Intervalli

Il reticolo $\langle \mathbb{I}_\perp ; \leq \rangle$ è completo? No. Per esempio non esiste

$$\bigvee \{ \, [0, n] \mid 0 \leq n \, \}.$$

Possiamo renderlo completo se prendiamo gli estremi degli intervalli da

$$\mathbb{Z}^* = \mathbb{Z} \cup \{-\infty, +\infty\}$$

ponendo

$$-\infty \leq x \leq +\infty$$

per ogni $x \in \mathbb{Z}^*$. Otteniamo così il reticolo completo $\langle \mathbb{I}^*, \leq \rangle$, dove

$$\mathbb{I}^* = \{\perp\} \cup \{ \, [a, b] \mid a, b \in \mathbb{Z}^* \text{ con } a \leq b \, \}$$