Ordini parziali e reticoli

Semantica di punto fisso

Agostino Cortesi

Insiemi parzialmente ordinati

nell'analisi statica, ordini parziali e reticoli giocano un ruolo fondamentale

dato un insieme S, un ordine parziale su S è una relazione

$$\leq \ \mu \ S \times S$$

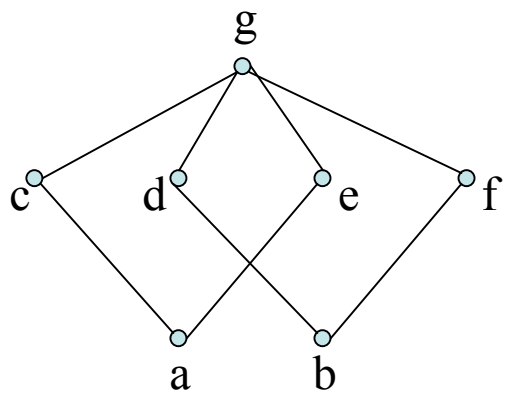
che gode delle proprietà:

- riflessiva: $\forall \ e \in S : \ e \leq e$
- transitiva: $\forall \ e_1, e_2, e_3 \in S : \ e_1 \leq e_2 \ \wedge \ e_2 \leq e_3 \ \Rightarrow \ e_1 \leq e_3$
- antisimmetrica: $\forall \ e_1, e_2 \in S : e_1 \leq e_2 \ \wedge \ e_2 \leq e_1 \Rightarrow e_1 = e_2$

un insieme parzialmente ordinato (S, ≤) (anche detto poset) è un insieme S sul quale è definito un ordine parziale ≤.

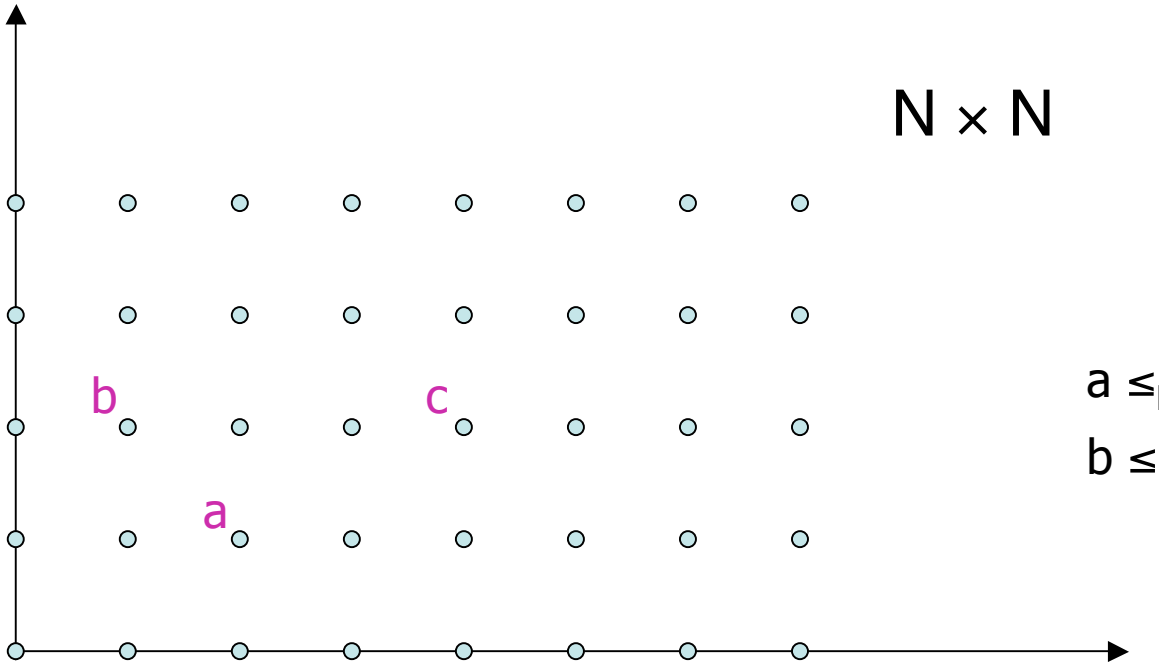
esempio

- S= {a,b,c,d,e,f,g}
- ≤ ={(a,c), (a,e), (b,d), (b,f), (c,g), (d,g), (e,g), (f,g)}^{T} dove (¢)^T e` la chiusura riflessiva e transitiva di una relazione



- (S, ≤) è un poset finito

esempio

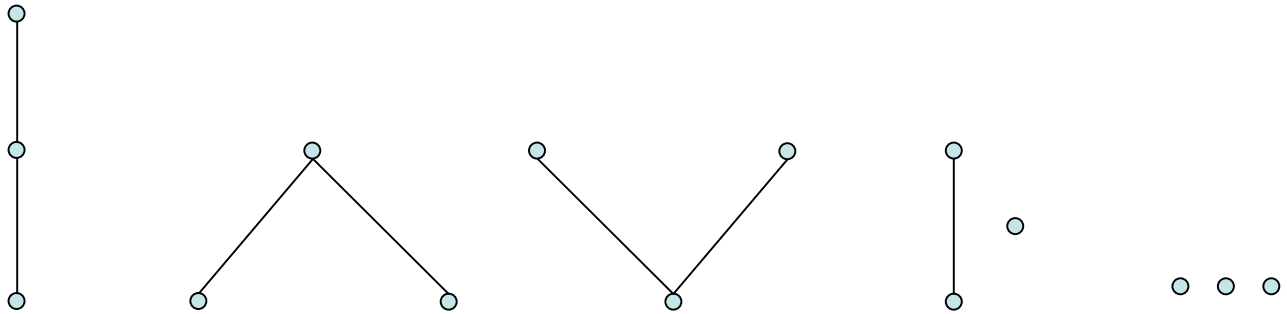


$(x_1,y_1) \leq_{N \times N} (x_2,y_2) \Leftrightarrow x_1 \leq_N x_2 \wedge y_1 \leq_N y_2$

$(N \times N, \leq_{N \times N})$ è un poset infinito

Esempio

Tutti i possibili poset con tre elementi:



lub e glb

dato un insieme parzialmente ordinato (S, ≤),

–un sottoinsieme A μ S ha un elemento u come upper bound se
$\forall a \in A : \; a \leq u$

–un sottoinsieme A μ S ha un elemento l come lower bound se
$\forall a \in A : \; l \leq a$

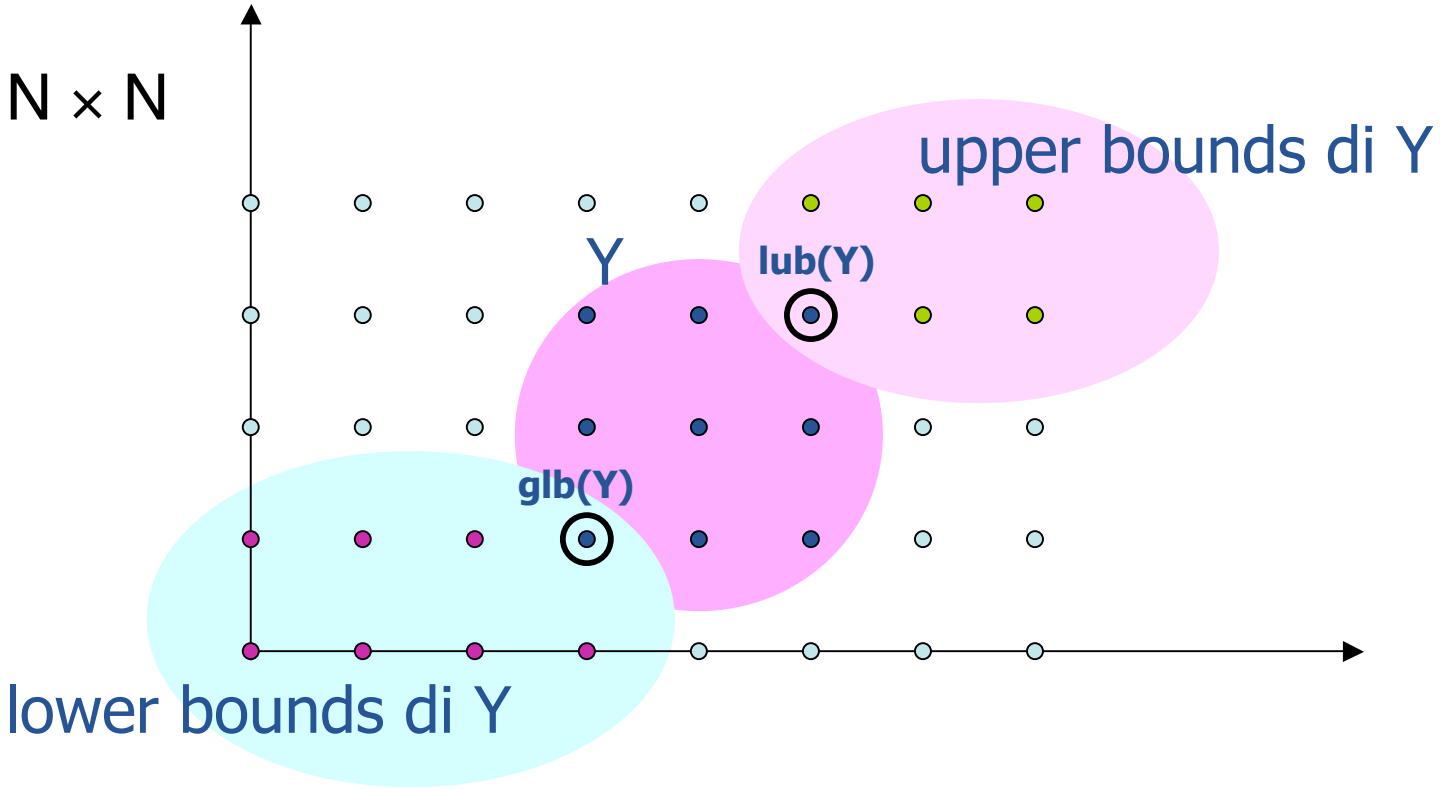
un least upper bound (lub) di A è un upper bound u che soddisfa la seguente proprietà:

u' è un upper bound di A => u ≤ u'

un greatest lower bound (glb) di A è un lower bound l che soddisfa la seguente proprietà:

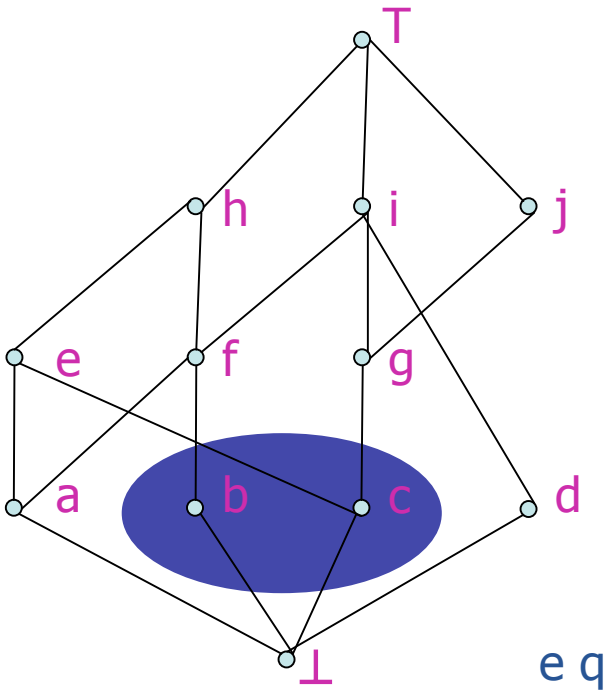
l' è un lower bound di A fi l' => l

Esempio



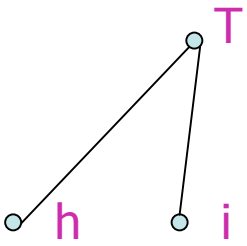
$(x_1,y_1) \leq_{N \times N} (x_2,y_2) \Leftrightarrow x_1 \leq_N x_2 \wedge y_1 \leq_N y_2$

esempio



lub({b,c})= ?

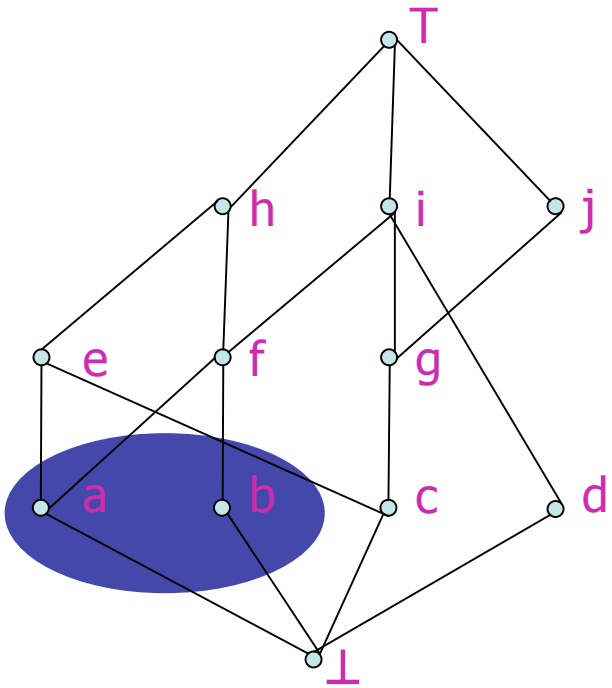
Gli upper bounds dell'insieme {b,c} sono {h,i,T}



e questo insieme non ha un minimo elemento:

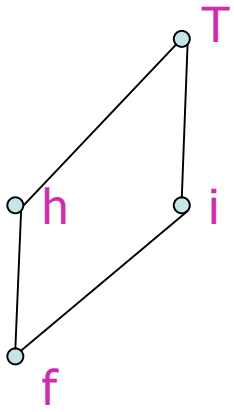
Il lub non c'è !

esempio



lub({a,b})= ?

Gli upper bounds dell'insieme {a,b} sono {T,h,i,f}



e questo insieme ha f come minimo elemento:

lub({a,b})= f

reticoli

un reticolo è un insieme parzialmente ordinato (S, ≤) tale che per ogni coppia di elementi di S esiste il least upper bound ed il greatest lower bound.

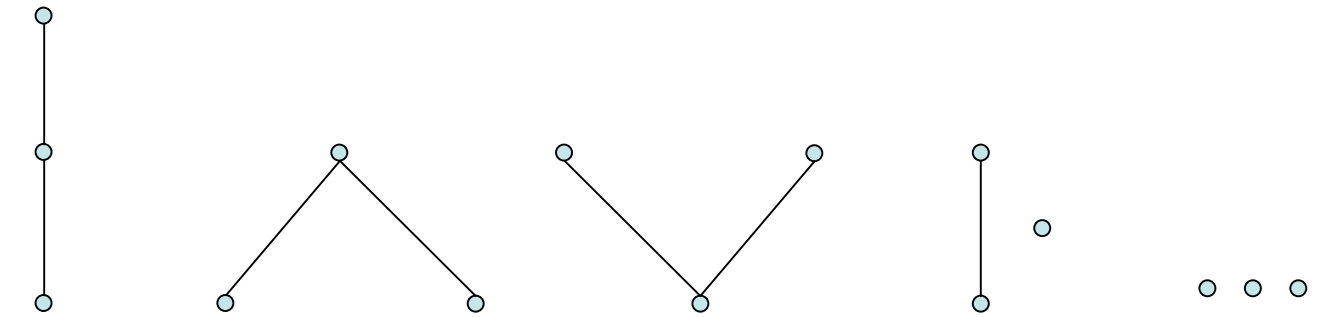
se S è un poset non vuoto e $x \leq y$, allora

$$\text{lub}(\{x,y\}) = y$$
$$\text{glb}(\{x,y\}) = x.$$

per dimostrare che L è un reticolo basterà quindi verificare che per ogni coppia di elementi incomparabili esistono sia lub che glb.

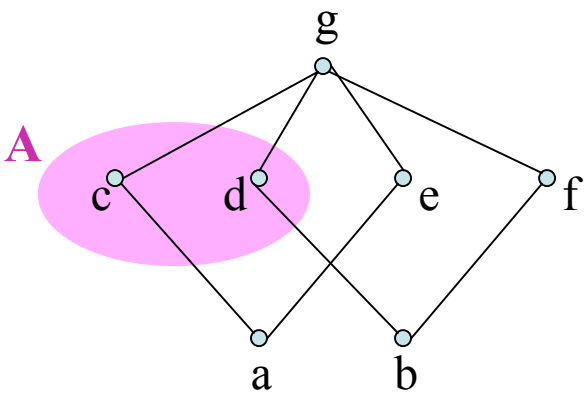
esempio

rivediamo tutti i possibili insiemi ordinati con tre elementi: sono reticoli?



SI NO NO NO NO

esempio



- S= {a,b,c,d,e,f,g}
- ≤ ={(a,c), (a,e), (b,d), (b,f), (c,g), (d,g), (e,g), (f,g)}^{T}
- (S,≤) non è un reticolo: Y non ha lower bounds

catene

dato un poset (S,≤), un sottoinsieme AμS è una catena se

$$\forall\, a_1, a_2 \in A : \quad (a_1 \leq a_2) \vee (a_2 \leq a_1)$$

(una catena è un sottoinsieme di S totalmente ordinato)

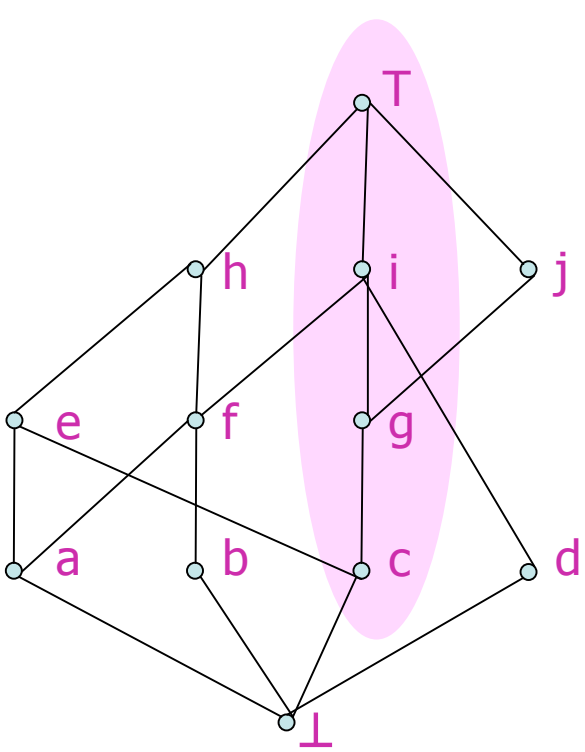
un poset (S,≤) ha altezza finita n 2 N se la cardinalita` della piu` lunga catena in S e` n

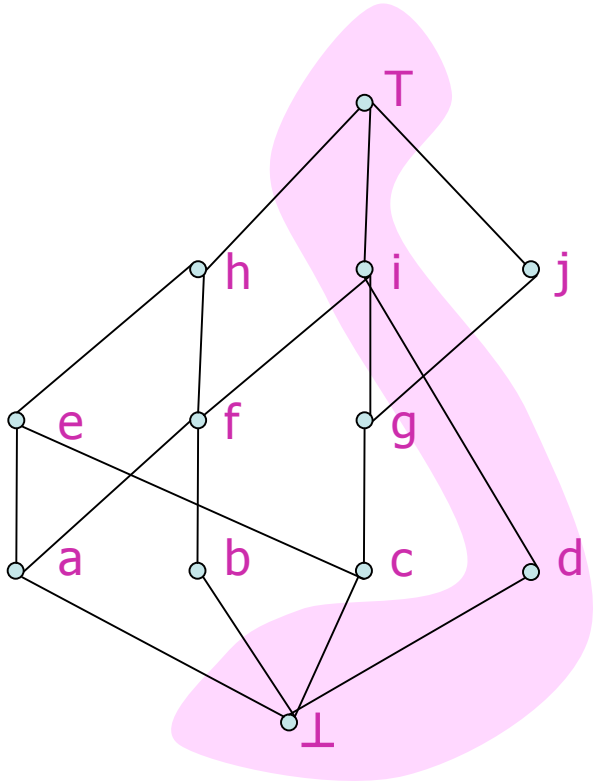
un poset (S,≤) non ha catene infinite se ogni catena in L e` finita

una sequenza $(a_n)_{n\in N}$ numerabile (quindi indiciabile sui naturali) di elementi di S è una catena ascendente se

$$n \leq m \quad \text{fi} \quad a_n \leq a_m$$

esempio: catene





CPO

un poset $(P,\leq_P)$ si dice CPO (complete partial order) se esiste il lub di ogni catena in P

in particolare, in ogni CPO esiste un un elemento minimo (bottom) ottenuto come lub della catena

vuota e denotato con ?.

reticoli e reticoli completi

un reticolo è un poset per cui ogni suo sottoinsieme finito e non vuoto ha least upper bound e greatest lower bound

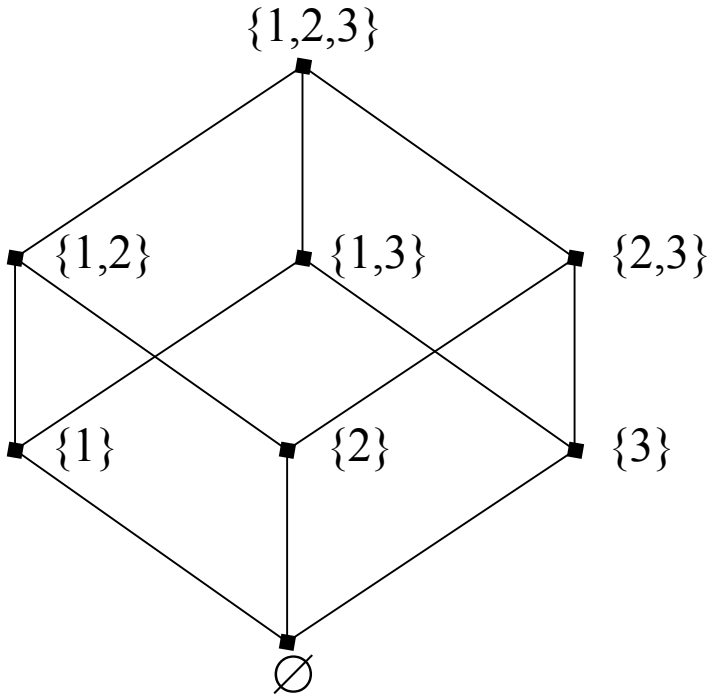
un reticolo e` completo quando ogni suo sottoinsieme ha least upper bound e greatest lower bound

se (S, ≤) è un reticolo completo, si denotano:

$\bot = \text{lub}(\varnothing)$ bottom element
$T = \text{glb}(S)$ top element

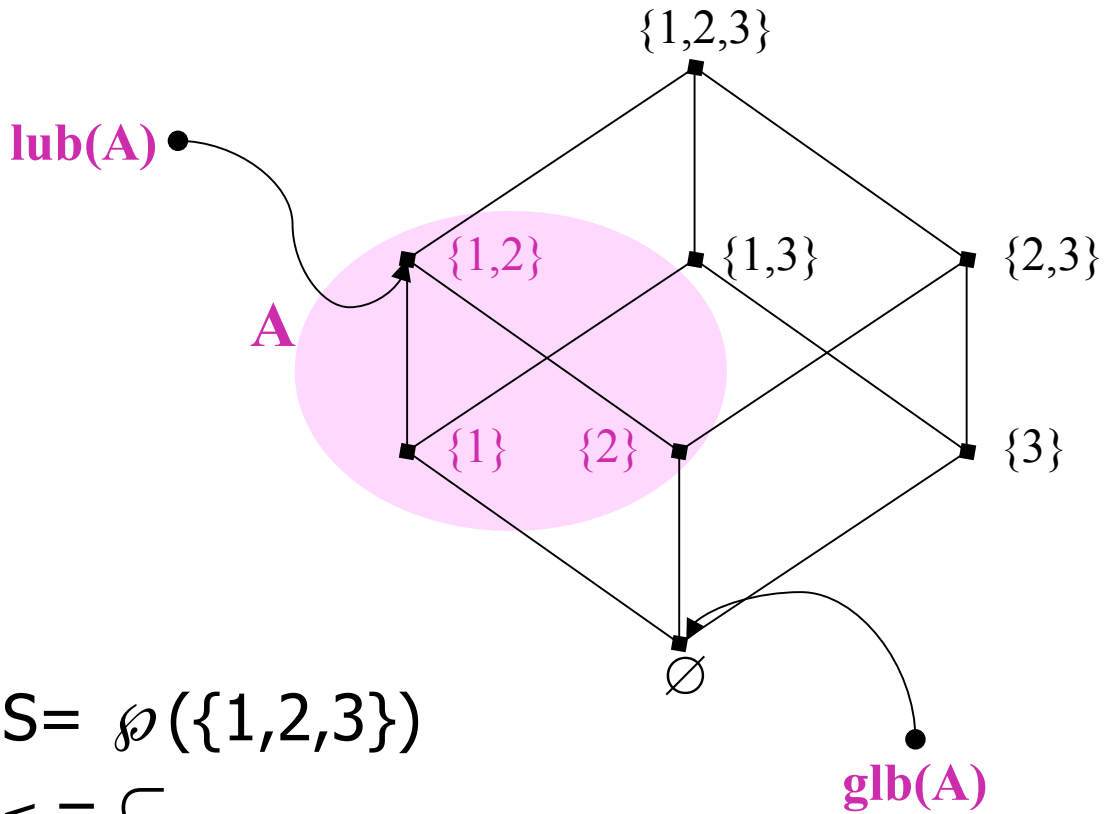
ogni reticolo finito è un reticolo completo

esempio



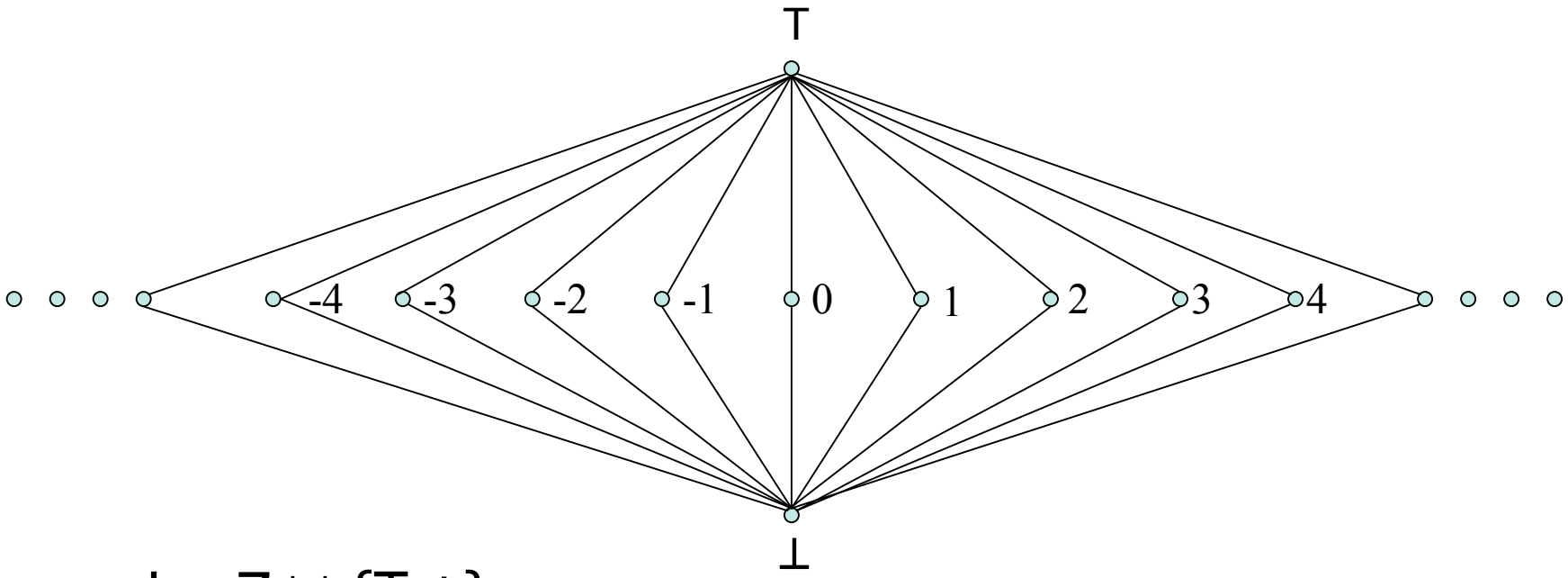
- S= $\wp(\{1,2,3\})$
- $\leq = \subseteq$
- lub(A) = $\cup A$
- glb(A) = $\cap A$

esempio



- S= $\wp(\{1,2,3\})$
- $\leq = \subseteq$
- lub(A) = $\cup A$
- glb(A) = $\cap A$

esempio

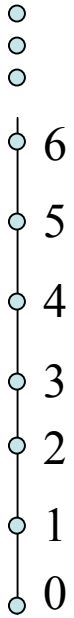


- $L = Z \cup \{\top, \bot\}$
- $\forall n \in Z : \bot \leq n \leq \top$

esempio

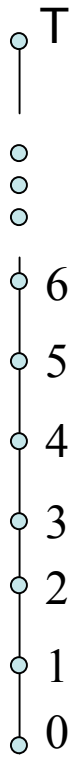
- L= N
- ≤ ordine totale su N
- lub = max
- glb = min

e’ un reticolo, ma non completo



esempio

- S= N∪{T}
- ≤ ordine totale su N∪{T}
- lub = max
- glb = min



e’ un reticolo completo

esempi

- R (numeri reali) con $\leq$ ordine totale
- (R, $\leq$) non è un reticolo completo:
  ad esempio $\{x \in R \mid x > 2\}$ non ha lub
- per ogni x<y in R, ([x,y], $\leq$) è un reticolo completo

- Q (numeri razionali) con $\leq$ ordine totale
- (Q, $\leq$) non è un reticolo completo
- e non basta aggiungere un top ed un bottom per ottenere la completezza:
  l'insieme $\{x \in Q \mid x^2 < 2\}$ ha upper bounds ma non ha un least upper bound.

Teorema: Sia (S, ≤) un poset. Sono equivalenti:

1. (S, ≤) è un reticolo completo
2. ogni sottoinsieme di S ha un least upper bound
3. ogni sottoinsieme di S ha un greatest lower bound

Dimostrazione:

– 1 fi 2 e 1 fi 3 seguono immediatamente dalla definizione

– Per mostrare che 2 fi 1, basta definire per ogni A ⊆ S

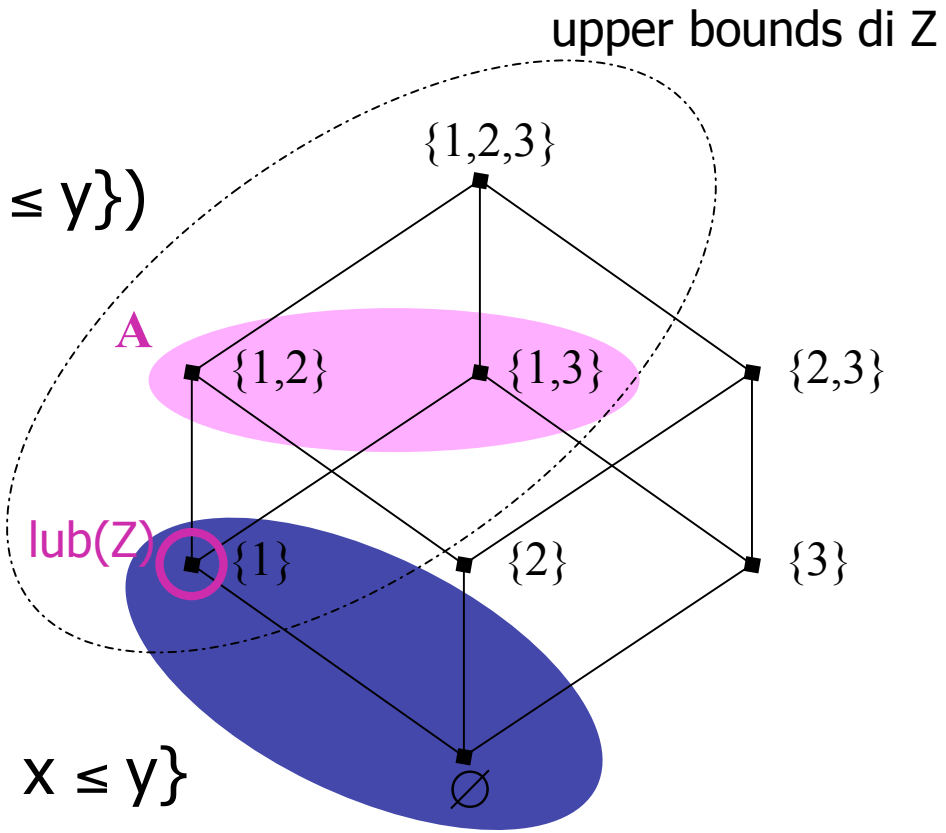
$$glb(A) = lub(\{x \in S \mid \forall y \in A : x \leq y\})$$

- tutti gli elementi dell'insieme a destra sono lower bounds dell'insieme A
- Quindi lub({...}) definisce un lower bound di A.

- poiché tutti i lower bound di A appartengono all'insieme a destra, lub({...}) definisce il greatest lower bound di A.

esempio

glb(A)= lub({x2S | ∀ y2A : x ≤ y})



Z= {x2L | ∀ y2A : x ≤ y}

funzioni tra poset

siano $(P, \leq_P)$ e $(Q, \leq_Q)$ due insiemi parzialmente ordinati

una funzione $f: P \rightarrow Q$ si dice:

–monotona (che preserva l'ordine) se

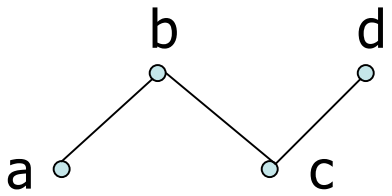
$$p_1 \leq_P p_2 \Rightarrow f(p_1) \leq_Q f(p_2)$$

–un embedding se

$$p_1 \leq_P p_2 \quad \Leftrightarrow \quad f(p_1) \leq_Q f(p_2)$$

–un isomorfismo se è un embedding suriettivo

esempi

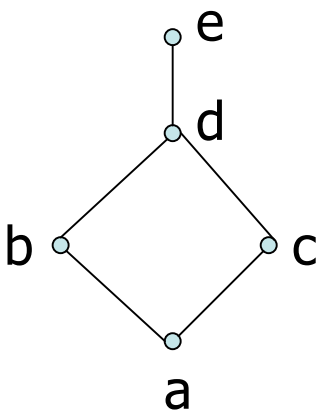


$\varphi_1(a)$

$\varphi_1(d)$

$\varphi_1(b)=\varphi_1(c)$

φ_1 non è una funzione monotona



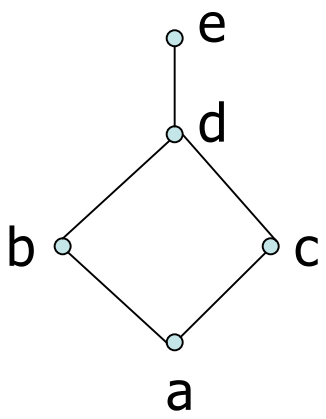
$\varphi_2(d)=\varphi_2(e)$

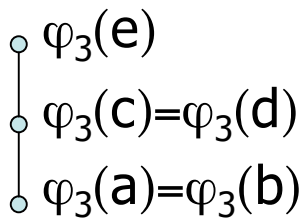
$\varphi_2(b)=\varphi_2(c)$

$\varphi_2(a)$

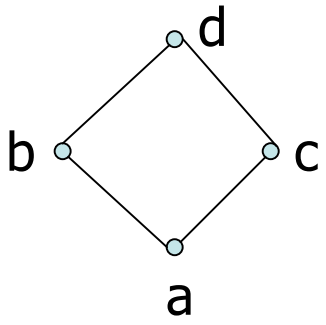
φ_2 è una funzione monotona, ma non è un embedding: $\varphi_2(b) \leq_Q \varphi_2(c)$ ma non è vero che $b \leq_P c$

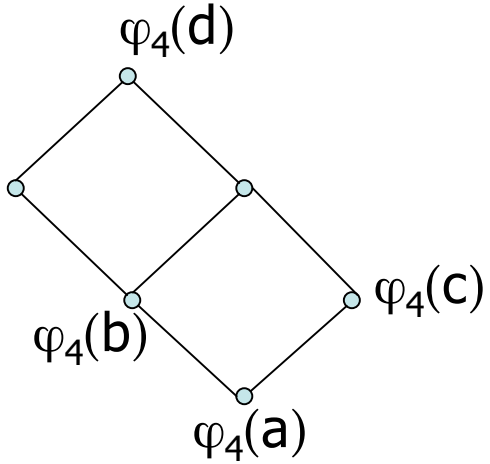
esempi





φ_3 è monotona, ma non è un embedding:
$\varphi_3(b) \leq_Q \varphi_3(c)$ ma non è vero che $b \leq_P c$





φ_2 è un embedding, ma non è un isomorfismo

catene convergenti

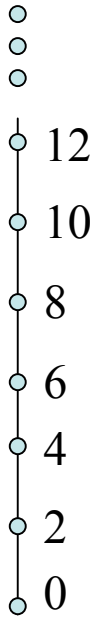
(una sequenza numerabile $(x_n)_{n\in N}$ di elementi di S è una catena ascendente se $n \leq m \Rightarrow x_n \leq x_m$)

una sequenza $(x_n)_{n\in N}$ converge se e solo se

$$\exists\, n_0 \in N.\ \forall\, n \in N.\ n_0 \leq n \Rightarrow x_{n_0} = x_n$$

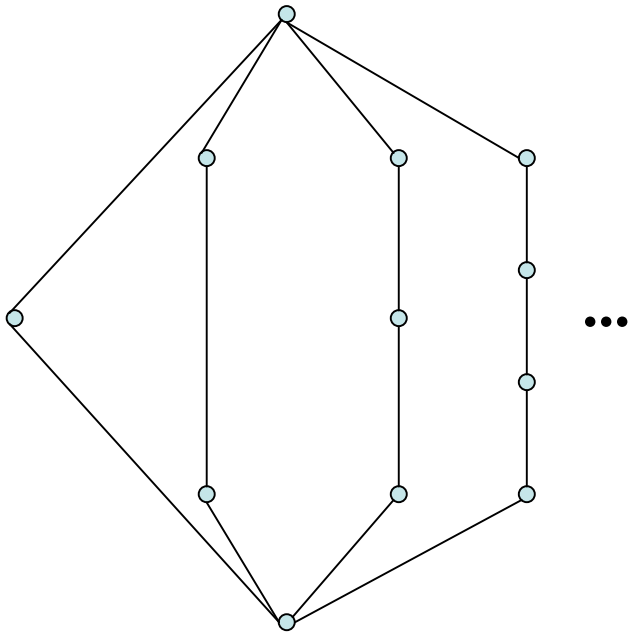
un poset $(S,\leq)$ soddisfa la condizione delle catene ascendenti (ACC) se e solo se ogni catena ascendente di S converge (ovvero se S non ha catene ascendenti infinite)

esempio



L’insieme ordinato dei numeri pari non soddisfa la ACC

esempio



...

poset infinito che soddisfa
la ACC ma non ha altezza finita

punti fissi

sia f una funzione $(P,\leq_P) \rightarrow (P,\leq_P)$, un elemento x2P si dice

punto fisso di f se $f(x) = x$

l'insieme dei punti fissi di f è un sottoinsieme di P denotato da Fix(f) :

$$Fix(f) = \{ a \in P \mid f(a) = a \}$$

Teorema di Tarski: *una funzione monotona crescente su un reticolo completo ha un reticolo completo di punti fissi*

Teorema di Tarski: *una funzione monotona crescente su un reticolo completo ha un reticolo completo di punti fissi*

Prova: dimostrare che

1. **un punto fisso almeno esiste**: Si consideri l'insieme $P = \{ x \mid x \geq f(x) \}$ (insieme dei post-fixed points) e sia $H = \bigcap_{x \geq f(x)} x$, allora $H \leq x$, per ogni $x \in P$. Quindi per monotonia e per definizione di P,

$$f(H) \leq f(x) \leq x, \text{ per ogni } x \in P$$

Poichè H è il più grande dei minoranti di P, deve essere $H \geq f(H)$. Ma da questa equazione deriva che $f(H) \geq f(f(H))$, quindi anche $f(H) \in P$. Perciò $H \leq f(H)$.

2. **questo è il minimo punto fisso** (facile perchè ogni punto fisso appartiene a P);

3. **se una collezione A di punti fissi esiste allora il loro least upper bound (unione) e il loro greatest lower bound (intersezione) sono**

punti fissi su CPO

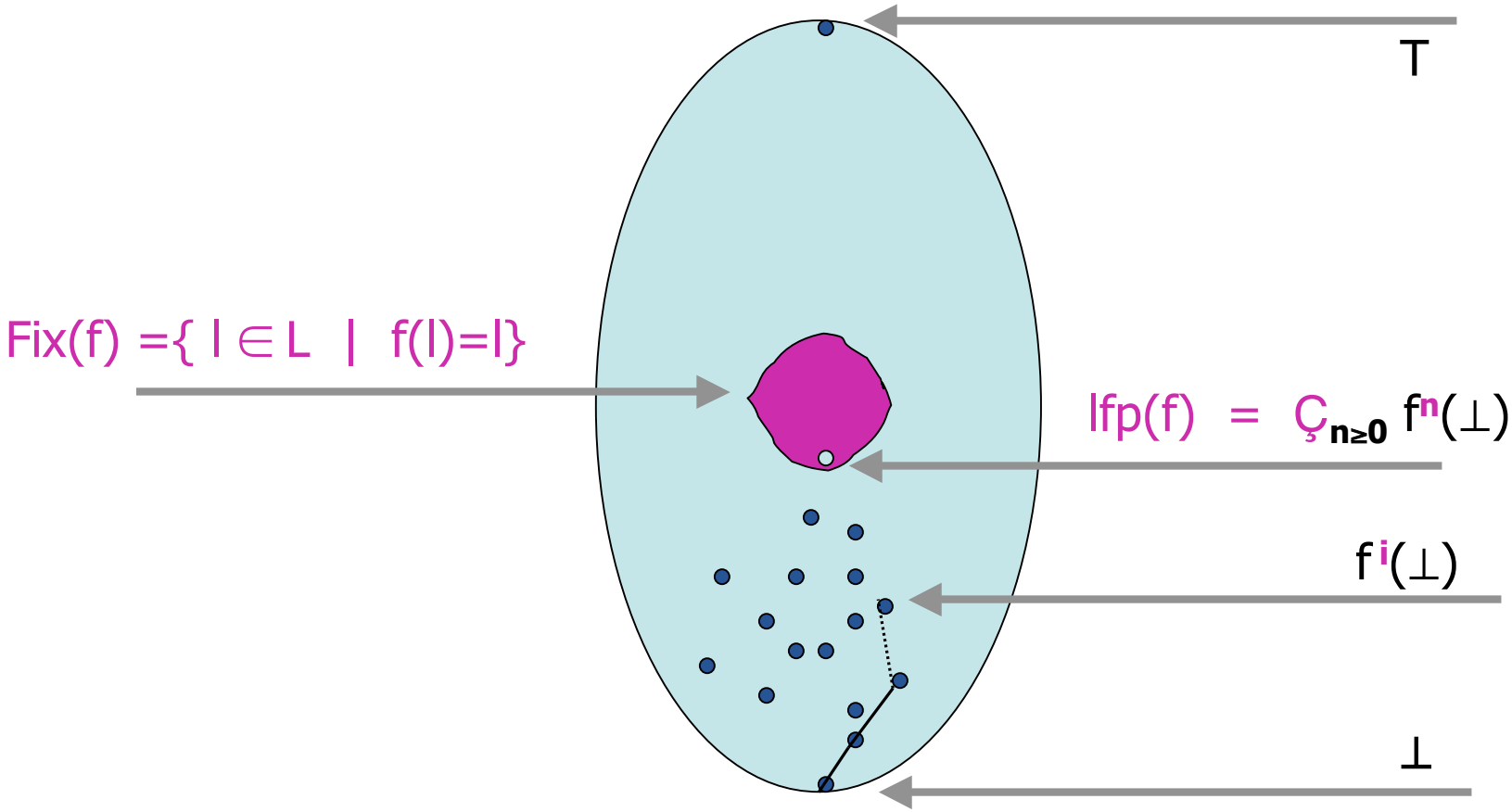
sia f una funzione monotona f: $(P,\leq_P) \rightarrow (P,\leq_P)$ su un CPO P.

sia $\alpha = \cap_{n\geq 0} f^n(\perp)$

–se $\alpha \in Fix(f)$ allora $\alpha = lfp(f)$

–Teorema di Knaster-Tarski-Kleene
se f è continua allora il minimo punto fisso di f esiste ed è uguale ad α.

Punti fissi su CPO



continuità

Dati due poset $(P,<_P)$ e $(Q,<_Q)$, una funzione φ:P-> Q si dice continua se per ogni catena non vuota S in P

$$\varphi(\text{lub}(S)) = \text{lub}\{\ \varphi(x) \mid x \in S\ \}$$

continuità

Non tutte le funzioni monotone sono continue.

Ad esempio,

$$\varphi : \wp(N) \rightarrow \wp(N)$$

$$\varphi(S) = \begin{cases} \varnothing & \text{se } S \text{ è finito,} \\ N & \text{altrimenti} \end{cases}$$

è monotona ma non è continua: se si prende la catena

$$\Delta = \{[0,k] \text{ Í } N \mid k \text{ ¸ } 0\}$$

si ha:

- lub $\{\varphi([0,k]) \mid [0,k] \text{ 2 } D\} = \text{Æ}$ perché ogni [0,k] in D è finito
- φ (lub (D)) = $\varphi(N) = N$

punti fissi su reticoli completi

sia f una funzione monotona $f:L \rightarrow L$ su un reticolo completo L.

$(Fix(f), \cdot_L)$ è pure un reticolo completo:

$$lfp(f) = glb(Fix(f)) \in Fix(f)$$

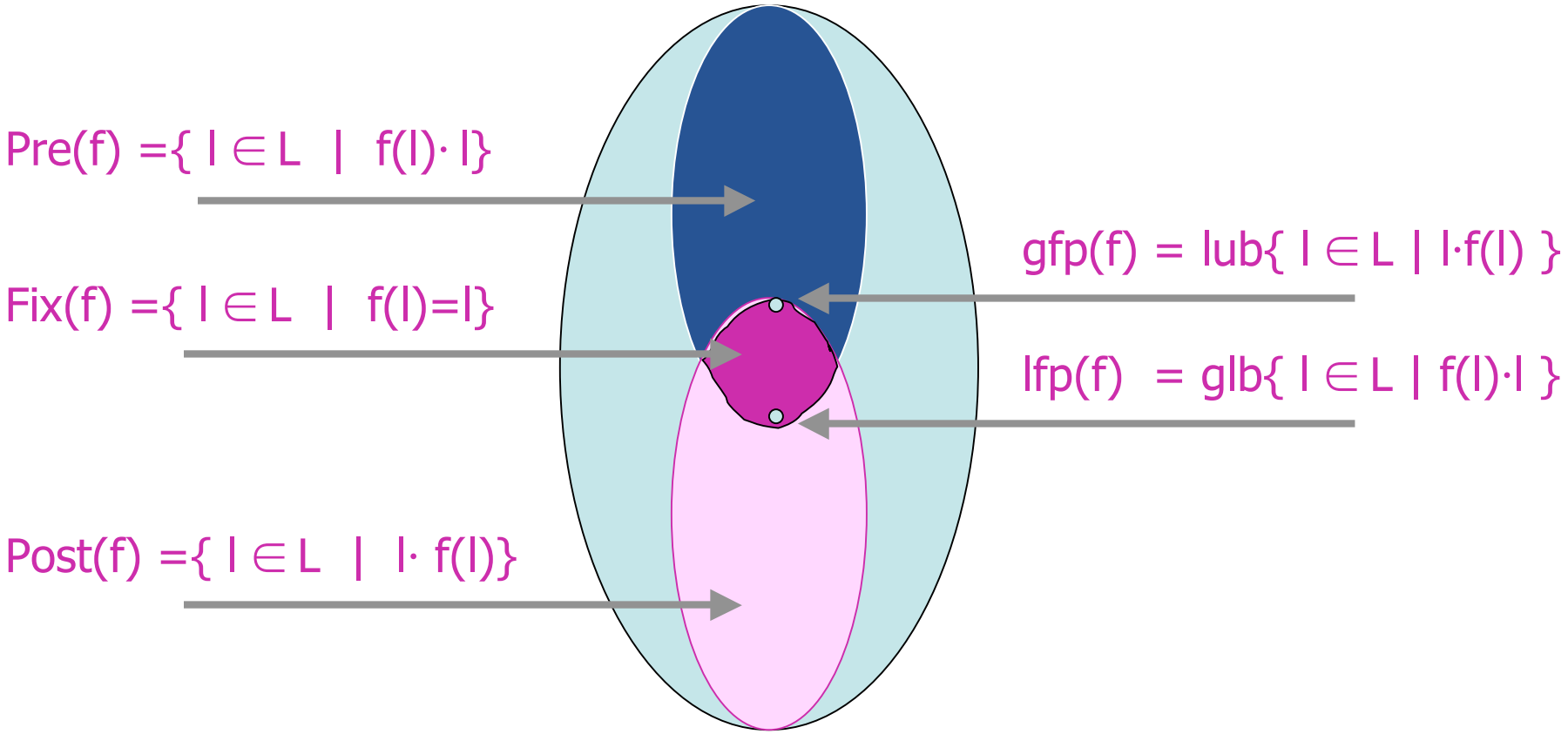
$$gfp(f) = lub(Fix(f)) \in Fix(f)$$

Teorema: Sia L un reticolo completo. Se f:L –> L è una funzione monotona allora

$$lfp(f) = glb \{ l \in L \mid f(l) \leq l \} \quad \text{(glb dei pre-punti fissi)}$$

$$gfp(f) = lub\{ l \in L \mid l \leq f(l) \} \quad \text{(lub dei post-punti fissi)}$$

punti fissi su reticoli completi



Dimostriamo che se L è un reticolo completo allora

gfp(f)=Ç{xÎL | x £ f(x)}.

Sia H={xÎL | x £ f(x)}, e sia a=ÇH. Dimostriamo che a=f(a).

Per ogni hÎH, h £ f(h), per definizione di H. E vale anche h £ a perché a è un upper bound di H. Quindi h £ f(h) £ f(a): la seconda relazione segue dalla monotonia di f. Poiché h £ f(a) vale per ogni hÎH, f(a) è un upper bound dell'insieme H. E poiché a = ÇH, ne segue che a £ f(a).

Dimostriamo ora che f(a) £ a. Applichiamo f ad entrambi i termini dell'espressione a £ f(a) che abbiamo dimostrato essere vera e per monotonia di f otteniamo f(a)£ f(f(a)). Ma allora f(a)ÎH, e quindi f(a) £ ÇH = a.

Punti fissi su CPO

Teorema I

Sia f: $(P,<_P)$ –> $(P,<_P)$ una funzione monotona su un CPO P. Allora f ha minimo punto fisso.

Teorema II

Sia f: $(P,£_P)$ –> $(P,£_P)$ una funzione su un CPO P tale che per ogni x in P, x $£_P$ f(x) (f viene detta increasing). Allora f ha un punto fisso.

Punti Fissi

Ci sono quindi tre risultati che garantiscono l'esistenza del minimo punto fisso:

1. Funzione continua su CPO
2. Funzione monotona su reticoli completi
3. Funzione monotona su CPO

I primi due hanno ipotesi più forti e forniscono una caratterizzazione del minimo punto fisso. Il terzo garantisce solo l'esistenza di un punto fisso.

a che serve tutto questo?

l'approccio all'analisi statica che consideriamo è basato sulla semantica

ci sono modi diversi per definire la semantica di un programma scritto in un dato linguaggio di programmazione:

- **semantica operazionale**:
  la semantica viene espressa in termini dei passi di computazione che possono aver luogo durante l'esecuzione
- **semantica assiomatica**
  la semantica viene definita attraverso assiomi e regole di una qualche logica
- **semantica denotazionale**
  fornisce modelli matematici ai linguaggi di programmazione: associa ad ogni statement del programma un elemento di una struttura matematica

la semantica denotazionale di un programma può essere espressa come soluzione di un'equazione di minimo punto fisso.

Il linguaggio **While**

n will range over numerals, **Num**,

x will range over variables, **Var**,

a will range over arithmetic expressions, **Aexp**,

b will range over boolean expressions, **Bexp**, and

S will range over statements, **Stm**.

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2 \\
b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & x := a \mid \texttt{skip} \mid S_1 \,;\, S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \\
 & \mid & \texttt{while } b \texttt{ do } S
\end{array}
$$

Semantica dei numerali

$$n ::= 0 \mid 1 \mid n\ 0 \mid n\ 1$$

$$\mathcal{N}: \mathbf{Num} \to \mathbf{Z}$$

$$\begin{aligned}
\mathcal{N}[\![0]\!] &= 0 \\
\mathcal{N}[\![1]\!] &= 1 \\
\mathcal{N}[\![n\ 0]\!] &= 2 \star \mathcal{N}[\![n]\!] \\
\mathcal{N}[\![n\ 1]\!] &= 2 \star \mathcal{N}[\![n]\!] + 1
\end{aligned}$$

La funzione semantica associa ad ogni numerale (in forma binaria) il suo significato (un elemento dell'insieme dei numeri interi Z)

Ad esempio,

$$\begin{aligned}
\mathcal{N}[\![101]\!] &= 2 \star \mathcal{N}[\![10]\!] + 1 \\
&= 2 \star (2 \star \mathcal{N}[\![1]\!]) + 1 \\
&= 2 \star (2 \star 1) + 1 \\
&= 5
\end{aligned}$$

Stato ed Espressioni

Il significato di una espressione aritmetica dipende dallo stato delle variabili che vi appaiono.

Uno stato è una funzione che associa valori (interi) alle variabili (intere)

$$\mathbf{State} = \mathbf{Var} \to \mathbf{Z}$$

Data un'espressione aritmetica ed uno stato possiamo determinare la semantica di questa espressione

$$\mathcal{A}\colon \mathbf{Aexp} \to (\mathbf{State} \to \mathbf{Z})$$

Semantica delle Espressioni

$$
\begin{array}{lcl}
\mathcal{A}[\![n]\!]s & = & \mathcal{N}[\![n]\!] \\
\mathcal{A}[\![x]\!]s & = & s\ x \\
\mathcal{A}[\![a_1 + a_2]\!]s & = & \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 \star a_2]\!]s & = & \mathcal{A}[\![a_1]\!]s \star \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 - a_2]\!]s & = & \mathcal{A}[\![a_1]\!]s - \mathcal{A}[\![a_2]\!]s
\end{array}
$$

Semantica delle espressioni booleane

$$\mathcal{B}[\![\texttt{true}]\!]s = \mathbf{tt}$$

$$\mathcal{B}[\![\texttt{false}]\!]s = \mathbf{ff}$$

B : **Bexp -> State ->** {tt,ff}

$$\mathcal{B}[\![a_1 = a_2]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s = \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s \neq \mathcal{A}[\![a_2]\!]s \end{cases}$$

$$\mathcal{B}[\![a_1 \leq a_2]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s \leq \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s > \mathcal{A}[\![a_2]\!]s \end{cases}$$

$$\mathcal{B}[\![\neg\, b]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \end{cases}$$

$$\mathcal{B}[\![b_1 \wedge b_2]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{tt} \text{ and } \mathcal{B}[\![b_2]\!]s = \mathbf{tt} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{ff} \text{ or } \mathcal{B}[\![b_2]\!]s = \mathbf{ff} \end{cases}$$

Semantica dei comandi

Un comando è un costrutto che può modificare lo stato della memoria: la semantica di uno statement è quindi una funzione parziale tra stati

$$S_{ds} : \textbf{Stm -> (State -> State)}$$

La semantica dell'assegnamento

assicura che se

$$\mathcal{S}_{ds}[\![x := a]\!]s = s[x \mapsto \mathcal{A}[\![a]\!]s]$$

allora

$$\mathcal{S}_{ds}[\![x := a]\!]s = s'$$

$$s'\ x = \mathcal{A}[\![a]\!]s \text{ and } s'\ y = s\ y \text{ for } y \neq x$$

Composizione sequenziale

La semantica della composizione sequenziale considera il caso In cui una delle due funzioni non sia definita rispetto agli argomenti:

$$
\begin{aligned}
&\mathcal{S}_{\mathrm{ds}}[\![S_1 \,;\, S_2]\!]s \\
&\qquad = \quad (\mathcal{S}_{\mathrm{ds}}[\![S_2]\!] \circ \mathcal{S}_{\mathrm{ds}}[\![S_1]\!])s \\
&\qquad = \quad \begin{cases} s'' & \text{if there exists } s' \text{ such that } \mathcal{S}_{\mathrm{ds}}[\![S_1]\!]s = s' \\ & \text{and } \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]s' = s'' \\ \underline{\mathrm{undef}} & \text{if } \mathcal{S}_{\mathrm{ds}}[\![S_1]\!]s = \underline{\mathrm{undef}} \\ & \text{or if there exists } s' \text{ such that } \mathcal{S}_{\mathrm{ds}}[\![S_1]\!]s = s' \\ & \text{but } \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]s' = \underline{\mathrm{undef}} \end{cases}
\end{aligned}
$$

Comando condizionale

$$
\begin{aligned}
&\mathcal{S}_{\mathrm{ds}}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!]\ s \\
&\quad = \quad \mathrm{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{\mathrm{ds}}[\![S_1]\!], \mathcal{S}_{\mathrm{ds}}[\![S_2]\!])\ s \\
&\quad = \quad \begin{cases}
s' & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \text{ and } \mathcal{S}_{\mathrm{ds}}[\![S_1]\!]s = s' \\
 & \text{or if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \text{ and } \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]s = s' \\
\underline{\text{undef}} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \text{ and } \mathcal{S}_{\mathrm{ds}}[\![S_1]\!]s = \underline{\text{undef}} \\
 & \text{or if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \text{ and } \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]s = \underline{\text{undef}}
\end{cases}
\end{aligned}
$$

Comando **`while`**

La semantica del comando

$$\texttt{while}\ b\ \texttt{do}\ S$$

deve essere la stessa di

$$\texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else skip}$$

Quindi la semantica di

$$\mathcal{S}_{\mathrm{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S]\!]$$

deve essere il punto fisso del funzionale

F : (**State** -> **State**) -> (**State** -> **State**) definito da

$$F\ g = \mathrm{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{\mathrm{ds}}[\![S]\!], \mathrm{id})$$

Comando `while`

La semantica del comando while è quindi:

$$\mathcal{S}_{\mathrm{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S]\!] = \mathrm{FIX}\ F$$
$$\text{where } F\ g = \mathrm{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{\mathrm{ds}}[\![S]\!], \mathrm{id})$$

dove l'operatore di punto fisso FIX ha tipo:

FIX: ((**State ->State**) -> (**StateÃState**)) -> (**State->State**)

Punti Fissi

Per garantire l'esistenza di soluzioni a queste equazioni di punto fisso sono necessarie le seguenti condizioni:

- –**State** -> **State** e` un CPO rispetto all'ordine puntuale tra funzioni parziali
- –Il funzionale F su **State** -> **State** e` continuo