

UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

**DIPARTIMENTO DI SCIENZE TEORICHE E
APPLICATE**

CORSO DI LAUREA IN INFORMATICA



**Grafi e strutture ordinate: una implementazione in
Java**

RELATORE: Prof.ssa Brunella Gerla

LAUREANDO: Alessandro Giovannacci

ANNO ACCADEMICO: 2017-2018

Sommario

In questa tesi viene sviluppato un software, GMaker, scritto in Java, che permette di costruire grafi a partire da insiemi ordinati. Il software si appoggia ad una libreria, Graphviz, che consente di generare grafi che inizialmente vengono rappresentati sotto forma di file con estensione DOT. Il software GMaker genera un file con estensione dot, e quest'ultimo viene dato in input ad un programma che renderizza il file, e genera in output il disegno del grafo, o equivalentemente, il diagramma di Hasse; a questo punto GMaker si occupa di mostrare all'utente il grafo appena creato.

Si possono creare grafi che rappresentano diverse strutture algebriche: insieme delle parti di un insieme, insieme dei down-sets di un poset, insieme dei down-sets disgiunti di un poset, insieme degli elementi join-irreducible di un poset.

Nel capitolo 1, vengono introdotte le definizioni formali e i concetti fondamentali, in modo da avere un quadro preciso del contesto che si sta trattando.

Si parte con una descrizione formale di insiemi, insiemi ordinati e reticoli; a questo punto si presentano i reticoli visti come strutture algebriche, e si introducono le algebre finite di Boole, le algebre finite di Heyting, le algebre finite di Kleene e le algebre finite IUML.

Infine, si presenta il teorema di rappresentazione di Birkhoff.

Nel capitolo 2, si descrive in modo dettagliato il software creato, con particolare enfasi sulla descrizione (in pseudolinguaggio) degli algoritmi principali utilizzati per la generazione dei grafi.

Viene presentata, mediante una serie di immagini, l'interfaccia grafica di GMaker, riportando inoltre esempi di utilizzo, di inserimento dati e di visualizzazione dei grafi.

Si dà inoltre una panoramica su Graphviz, strumento utilizzato da GMaker per renderizzare i file che rappresentano il grafo.

Sono presenti, nell'elaborato, esempi di grafi generati dal software, così come linee di codice di file DOT.

Indice

Sommario.....
1 Strutture ordinate.....	1
1.1 Insieme ordinato.....	1
1.2 Catena.....	1
1.3 Diagrammi.....	1
1.3.1 Relazione di copertura.....	1
1.3.2 Diagramma.....	2
1.4 Down-sets e up-sets.....	4
1.5 Prodotto cartesiano fra insiemi.....	6
1.6 Insieme dei down-sets disgiunti.....	6
1.7 Reticoli.....	6
1.7.1 Maggiorante e minorante.....	7
1.7.2 Estremo superiore ed estremo inferiore.....	7
1.7.3 Reticolo.....	8
1.7.4 Reticolo distributivo.....	9
1.8 Reticoli come strutture algebriche.....	9
1.9 Algebra booleana.....	9
1.9.1 Reticolo booleano.....	9
1.9.2 Algebra booleana.....	10
1.10 Algebra di Heyting.....	10
1.11 Algebra di Kleene.....	11
1.11.1 Algebra di De Morgan.....	11
1.11.2 Algebra di Kleene.....	11
1.11.3 Proprietà di interpolazione di un algebra di Kleene.....	11
1.11.4 Costruzione di Kalman.....	12
1.12 IUML-algebra.....	12
1.13 Teoria della rappresentazione: il caso finito.....	12
1.13.1 Atomo.....	12
1.13.2 Corollario: teorema di rappresentazione per algebre booleane.....	13

1.13.3 Elementi join-irreducible.....	14
1.13.4 Lemma.....	14
1.13.5 Isomorfismo.....	16
1.13.6 Teorema di rappresentazione di Birkhoff per reticoli distributivi.	16
1.13.7 Corollario del teorema di rappresentazione di Birkhoff.....	16
1.13.8 Dualità fra reticoli finiti distributivi ed insiemi finiti ordinati.....	17
1.13.9 Teorema.....	17
2 GMaker.....	18
2.1 Panoramica del software e descrizione dell'interfaccia.....	18
2.1.1 Panoramica.....	18
2.1.2 $P(A)$: insieme delle parti di un insieme A	19
2.1.3 $O(A)$: insieme dei down-sets di un poset A	20
2.1.3.1 Relazione fra elementi dell'insieme.....	23
2.1.4 $U(A)$: insieme dei down-sets disgiunti di un poset A	26
2.1.5 $J(A)$: insieme degli elementi join-irreducible di un poset A	29
2.2 Descrizione degli algoritmi.....	31
2.2.1 Algoritmo per la generazione dell'insieme dei down-sets.....	31
2.2.2 Algoritmo per la generazione del set of disjoint down-sets.....	35
2.2.3 Algoritmo per la generazione del set of join-irreducible elements...	36
2.3 GraphViz.....	37
2.3.1 Descrizione.....	37
2.3.2 Algoritmi per la creazione dei file DOT.....	41
2.3.2.1 File DOT per down-sets.....	41
2.3.2.2 File DOT per set of disjoint down-sets.....	44
3 Bibliografia.....	48

1 Strutture ordinate

1.1 Insieme ordinato

Sia P un insieme.

Un ordine (o ordine parziale) su P è una relazione binaria \leq tale che:

$$\forall x, y, z \in P$$

$$(i) x \leq x$$

$$(ii) x \leq y \wedge y \leq x \Rightarrow x = y$$

$$(iii) x \leq y \wedge y \leq z \Rightarrow x \leq z$$

Queste tre proprietà sono conosciute come riflessività, antisimmetria e transitività.

Un insieme P che soddisfi le tre proprietà di cui sopra, è chiamato insieme ordinato (P, \leq) .

1.2 Catena

Sia P un insieme ordinato.

P viene detto insieme totalmente (o linearmente) ordinato, detto anche catena, se:

$$\forall x, y \in P$$

$$x \leq y \vee y \leq x$$

1.3 Diagrammi

Nel caso finito, un insieme ordinato può essere rappresentato mediante un diagramma. Si introduce il concetto di copertura.

1.3.1 Relazione di copertura

Sia P un insieme ordinato e siano $x, y \in P$.

$x < \cdot y$ (ovvero x è coperto da y) se:

$$(i) x < y$$

$$(ii) x \leq z < y \Rightarrow z = x$$

Esempio

Sia $X = \{a, b, c\}$, e $P(X)$ l'insieme delle parti di X .

$$P(X) = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\} \}$$

Si ha che $A < B \Leftrightarrow B = A \cup \{x\}$, dove $x \in X \setminus A$

1.3.2 Diagramma

Sia P un insieme finito e ordinato.

Un diagramma di Hasse è un grafo in cui:

- (i) i nodi sono gli elementi di P
- (ii) due nodi sono collegati fra loro tramite un arco se e solo se un nodo copre l'altro

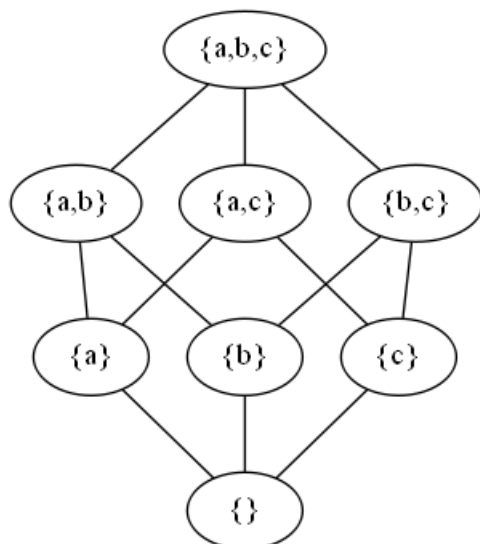
Per convenzione, i nodi che risultano minori rispetto alla relazione vengono disegnati più in basso.

Esempio

(i) Sia $X = \{a, b, c\}$, e $P(X)$ l'insieme delle parti di X .

$$P(X) = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\} \}$$

Il diagramma di Hasse di $P(X)$ è il seguente:



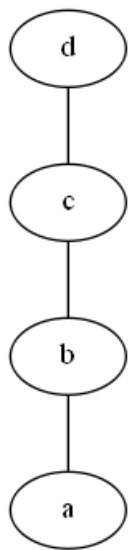
(ii) Sia $A = \{a, b, c, d\}$ con le seguenti relazioni d'ordine:

a) $a \leq b$

b) $b \leq c$

c) $c \leq d$

Il diagramma di Hasse che rappresenta A è il seguente:



(iii) Sia $B = \{a, b, c\}$ con le seguenti relazioni d'ordine:

a) $a \leq c$

b) $b \leq c$

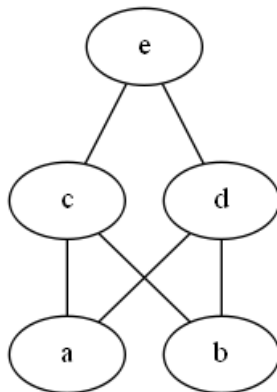
c) $a \leq d$

d) $b \leq d$

e) $c \leq e$

f) $d \leq e$

Il diagramma di Hasse che rappresenta B è il seguente:



1.4 Down-sets e up-sets

Sia P un insieme ordinato e sia $Q \subseteq P$.

(i) Q è un down-set se:

$$\forall x \in Q, y \in P \text{ e } y \leq x$$

$$y \in Q$$

(ii) Per principio di dualità, Q è un up-set se:

$$\forall x \in Q, y \in P \text{ e } y \geq x$$

$$y \in Q$$

Inoltre, Q è un down-set se e solo se $P \setminus Q$ è un up-set (complementarietà).

Dato $Q \subseteq P$ e dato $x \in P$, si definiscono:

$$\downarrow Q = \{ y \in P \mid (\exists x \in Q) y \leq x \}$$

$$\uparrow Q = \{ y \in P \mid (\exists x \in Q) y \geq x \}$$

$$\downarrow x = \{ y \in P \mid y \leq x \}$$

$$\uparrow x = \{ y \in P \mid y \geq x \}$$

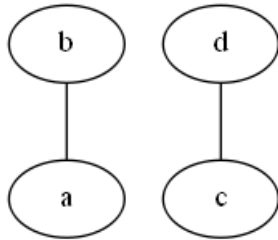
$\downarrow Q$ (down Q) è il più piccolo down-set contenente Q e Q è un down-set se e solo se

$$Q = \downarrow Q \text{ (dualmente per } \uparrow Q \text{)}.$$

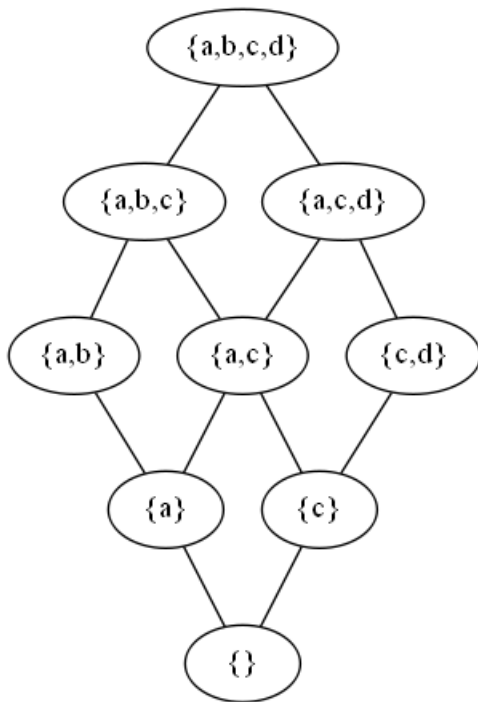
La famiglia di tutti i down-sets di P si denota con $O(P)$. E' a sua volta un insieme ordinato (mediante inclusione).

Esempio

(i) Sia A il seguente insieme ordinato:



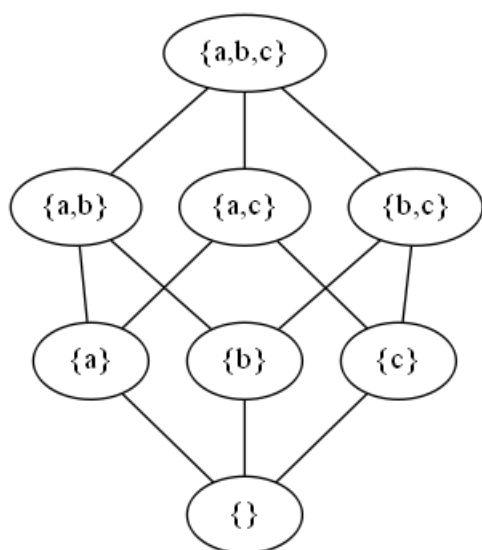
Il diagramma successivo rappresenta $O(A)$:



(ii) Sia B il seguente insieme, detto anche anti-catena:



In questo caso, $O(B) = P(B)$



1.5 Prodotto cartesiano fra insiemi

Siano A, B insiemi t.c. $A \neq \emptyset$ e $B \neq \emptyset$.

Il prodotto cartesiano fra A e B è definito nel modo seguente:

$$A \times B = \{ (a, b) \mid a \in A, b \in B \}$$

1.6 Insieme dei down-sets disgiunti

Si definisce insieme dei down-sets disgiunti di un insieme ordinato A:

$$U(A) = \{ (X, Y) \in O(A) \times O(A) \mid X \cap Y = \emptyset \}$$

1.7 Reticoli

Molte importanti proprietà di un insieme ordinato P sono espresse in termini di esistenza di estremi superiori o estremi inferiori di sottoinsiemi di P.

Due fra le più importanti classi di insiemi ordinati definiti in tal modo sono i reticoli ed i reticoli completi.

1.7.1 Maggiorante e minorante

Sia P un insieme ordinato e sia $S \subseteq P$.

Un elemento $x \in P$ viene detto maggiorante di S se:

$$\forall s \in S$$

$$s \leq x$$

Un elemento minorante è definito in modo duale.

L'insieme di tutti i maggioranti di S si denota con S^u , mentre l'insieme di tutti i minoranti di S si denota con S^ℓ .

$$S^u := \{ x \in P \mid (\forall s \in S) s \leq x \}$$

$$S^\ell := \{ x \in P \mid (\forall s \in S) s \geq x \}$$

Dato che \leq è transitiva, S^u è sempre un up-set mentre S^ℓ è sempre un down-set.

1.7.2 Estremo superiore ed estremo inferiore

Se S^u ha un elemento x tale che $x \leq y$ per ogni $y \in S^u$, allora x viene detto estremo superiore di S . Equivalentemente, x viene detto estremo superiore di S se:

- (i) x è un maggiorante di S
- (ii) $x \leq y$ per ogni maggiorante y di S

Dualmente, se S^ℓ ha un elemento x tale che $x \geq y$ per ogni $y \in S^\ell$, allora x viene detto estremo inferiore di S .

L'estremo superiore di S viene anche indicato con $\sup S$.

L'estremo inferiore di S viene anche indicato con $\inf S$.

$\vee S$ denota $\sup S$, mentre $\wedge S$ denota $\inf S$.

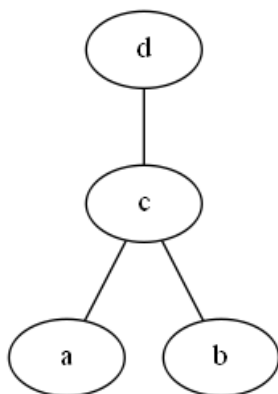
$x \vee y$ denota $\sup\{x,y\}$, mentre $x \wedge y$ denota $\inf\{x,y\}$.

Esempio

(i) Nel seguente insieme ordinato, $\{a, b\}^u = \emptyset$, pertanto $a \vee b$ non esiste.



(ii) Nel seguente insieme ordinato $\{a, b\}^u = \{c, d\}$, in questo caso $a \vee b = c$, poiché il minor elemento di $\{a, b\}^u$ è c .



1.7.3 Reticolo

Sia P un insieme ordinato non vuoto.

Se esistono $(x \vee y)$ e $(x \wedge y)$ per ogni $x, y \in P$, allora P viene chiamato reticolo.

Se esistono $\vee S$ e $\wedge S$ per ogni $S \subseteq P$, allora P viene chiamato reticolo completo.

Se P è un reticolo, allora \vee e \wedge sono operazioni binarie su P , e si ha una struttura algebrica:

$$\langle P; \vee; \wedge \rangle$$

Esempio

(i) Per ogni insieme X , l'insieme ordinato $P(X)$ è un reticolo completo nel quale:

$$\vee \{ A_i \mid i \in I \} = \cup A_i \quad \forall i \in I.$$

$$\wedge \{ A_i \mid i \in I \} = \cap A_i \quad \forall i \in I.$$

dove I è un intervallo chiuso e limitato inferiormente e superiormente.

1.7.4 Reticolo distributivo

Sia P un reticolo, $\langle P; \vee; \wedge \rangle$

P viene detto reticolo distributivo se:

$\forall x, y, z \in L$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Dualmente:

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

1.8 Reticoli come strutture algebriche

Sia L un reticolo.

L ha un elemento unità, o elemento identità, se:

$$\exists 1 \in L \text{ tale che } a = a \wedge 1, \forall a \in L$$

Dualmente, L ha un elemento zero se:

$$\exists 0 \in L \mid a = a \vee 0, \forall a \in L$$

Il reticolo $\langle L; \vee; \wedge \rangle$ ha un elemento unità 1 se e solo se $\langle L; \leq \rangle$ ha un elemento superiore T , ed in questo caso $1 = T$. Dualmente per 0 e \perp .

Un reticolo $\langle L; \vee; \wedge \rangle$ che possiede 0 e 1 viene chiamato limitato.

1.9 Algebra booleana

1.9.1 Reticolo booleano

Sia L un reticolo.

L viene detto reticolo booleano se:

- (i) L è distributivo
- (ii) L ha gli elementi 0 e 1
- (iii) ogni elemento $a \in L$ ha il suo complemento $a' \in L$

Sia L un reticolo booleano. Allora:

$$(i) 0' = 1 \text{ e } 1' = 0$$

$$(ii) a'' = a \quad \forall a \in L$$

(iii) per le leggi di De Morgan, $\forall a, b \in L$:

$$(a \vee b)' = a' \wedge b'$$

$$(a \wedge b)' = a' \vee b'$$

(iv) $\forall a, b \in L$

$$a \wedge b = (a' \vee b')'$$

$$a \vee b = (a' \wedge b')'$$

(v) $\forall a, b \in L$

$$a \wedge b' = 0 \Leftrightarrow a \leq b$$

1.9.2 Algebra booleana

Un'algebra booleana è una struttura $\langle B; \vee; \wedge; ', 0, 1 \rangle$ tale che:

(i) $\langle B; \vee; \wedge \rangle$ è un reticolo distributivo

$$(ii) \forall a \in B (a \vee 0 = a, a \wedge 1 = a)$$

$$(iii) \forall a \in B (a \vee a' = 1, a \wedge a' = 0)$$

Un sottoinsieme A di un'algebra booleana B è una sottoalgebra se A è un sottoreticolo di B contenente 0 e 1 e tale che:

$$a \in A \Rightarrow a' \in A$$

Date due algebre booleane B e C :

$f: B \rightarrow C$ è un omomorfismo booleano se f è un reticolo omomorfo che preserva 0 , 1 e $'$.

$$f(0) = 0 \text{ e } f(1) = 1$$

$$f(a') = (f(a))' \quad \forall a \in B$$

1.10 Algebra di Heyting

Un'algebra di Heyting H è un reticolo limitato tale che $\forall a, b \in H$ vi è un elemento maggiore $x \in H$ tale che:

$$a \wedge x \leq b$$

Tale elemento è lo pseudo-complemento di a rispetto a b , e si denota con $a \rightarrow b$.

In ogni algebra di Heyting, si definisce lo pseudo-complemento $\neg a$ di un elemento a impostando $\neg a = (a \rightarrow 0)$.

Per definizione, $a \wedge \neg a = 0$, ed $\neg a$ è il più grande elemento che possiede questa proprietà.

Non è però generalmente vero che $a \vee \neg a = 1$, poiché \neg è solo uno pseudo-complemento, e non un complemento vero e proprio, come nel caso dell'algebra booleana.

1.11 Algebra di Kleene

1.11.1 Algebra di De Morgan

Un algebra di De Morgan è un reticolo distributivo chiuso $(A, \wedge, \vee, \neg, 0, 1)$ tale che:

$$\forall x, y \in A$$

$$(i) \neg(x \vee y) = \neg x \wedge \neg y$$

$$(ii) \neg\neg x = x$$

1.11.2 Algebra di Kleene

Un algebra di Kleene $(A, \wedge, \vee, \neg, 0, 1)$ è un algebra di De Morgan tale che:

$$\forall x, y \in A$$

$$x \wedge \neg x \leq y \vee \neg y$$

Un elemento c di un algebra di Kleene A viene detto centro di A se:

$$c = \neg c$$

A è un algebra di Kleene centrata se A ha un centro.

1.11.3 Proprietà di interpolazione di un algebra di Kleene

Sia $(A, \wedge, \vee, \neg, 0, 1)$ un algebra di Kleene centrata, e sia c il centro di A .

A possiede la proprietà di interpolazione se e solo se:

$$\forall x, y \geq c \mid x \wedge y \leq c$$

$$\exists z \mid z \vee c = x, \neg z \vee c = y$$

1.11.4 Costruzione di Kalman

Sia $(L, \wedge, \vee, 0, 1)$ un reticolo distributivo chiuso.

Si definisce costruzione di Kalman:

$$K(L) = \{ (x,y) \in L \times L \mid x \wedge y = 0 \}$$

Le coppie appartenenti a $K(L)$ formano un'algebra di Kleene centrata, con centro $(0,0)$.

Sull'insieme $K(L)$ definiamo la seguente relazione d'ordine:

$$(x,y) \leq (h,k) \text{ se e solo se } x \leq h \text{ e } k \leq y$$

1.12 IUML-algebra

Una IUML-algebra (IUML è acronimo di idempotent uniform mingle logic algebra) è un reticolo limitato commutativo distributivo

$$\langle A, \wedge, \vee, *, \rightarrow, \perp, T, e \rangle$$

che soddisfa le seguenti condizioni:

$$\forall x,y \in A$$

$$e \leq (x \rightarrow y) \vee (y \rightarrow x)$$

$$(x \rightarrow e) \rightarrow e = x$$

1.13 Teoria della rappresentazione: il caso finito

1.13.1 Atomo

Sia L un reticolo con minor elemento 0 .

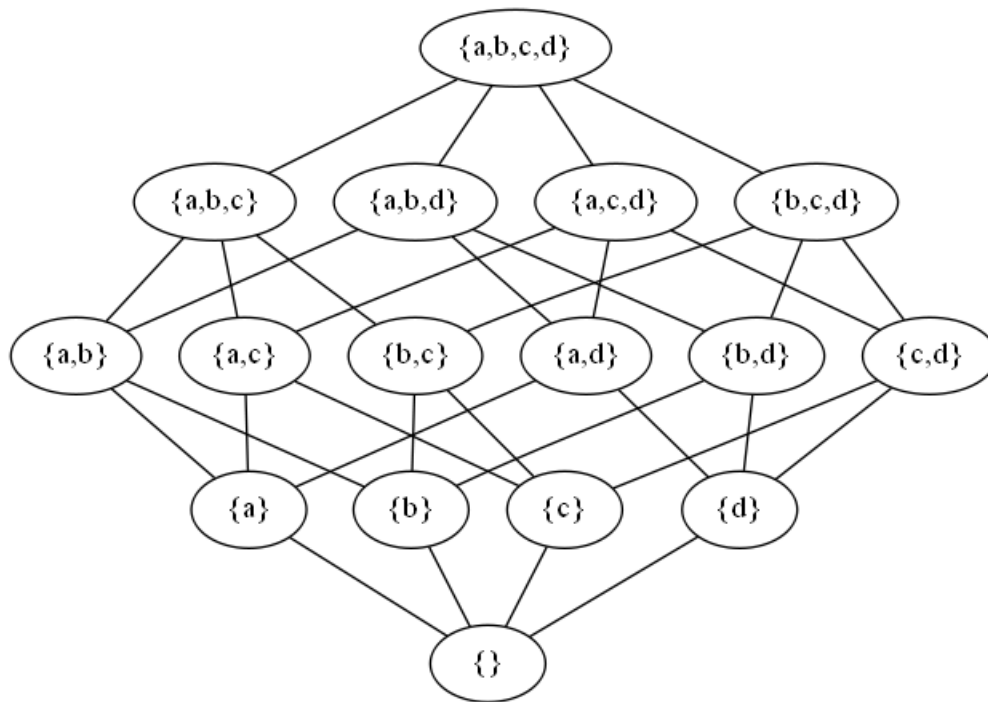
$a \in L$ viene detto atomo se:

$$0 < a$$

L'insieme degli atomi di L viene detto $A(L)$.

Esempio

X è un'algebra booleana



$A(X)$ è l'insieme degli atomi di X



1.13.2 Corollario del teorema di rappresentazione per algebre booleane finite

Sia B un reticolo finito. Le seguenti affermazioni sono equivalenti:

- (i) B è un reticolo booleano
- (ii) $B \cong P(A(B))$
- (iii) B è isomorfo a 2^n , per $n \geq 0$

Inoltre, qualsiasi reticolo booleano finito ha 2^n elementi, per $n \geq 0$

1.13.3 Elementi join-irreducibile

Sia L un reticolo.

Un elemento $x \in L$ viene detto join-irreducibile se:

- (i) $x \neq 0$
- (ii) $x = a \vee b \Rightarrow x = a$ oppure $x = b$

Un elemento meet-irreducibile viene definito in modo duale.

L'insieme degli elementi join-irreducibile di L si denota con $J(L)$, mentre l'insieme degli elementi meet-irreducibile con $M(L)$.

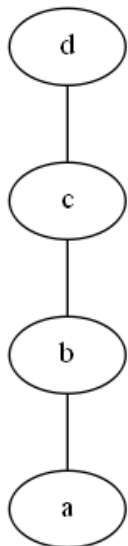
1.13.4 Lemma

Sia L un reticolo con elemento minimo 0 . Allora:

- (i) $0 < x \Rightarrow x \in J(L)$
- (ii) se L è un reticolo booleano, $x \in J(L) \Rightarrow 0 < x$

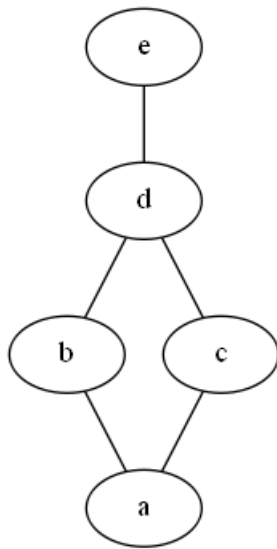
Esempio

(i)



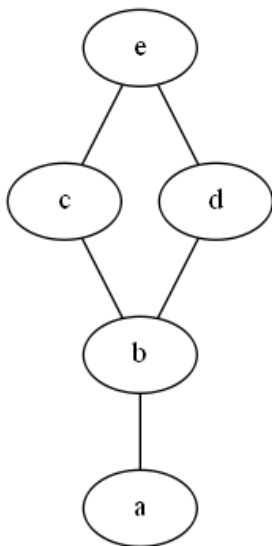
$$J(L) = \{b, c, d\}$$

(ii)



$$J(L) = \{b, c, e\}$$

(iii)



$$J(L) = \{b, c, d\}$$

1.13.5 Isomorfismo

Due reticoli $\langle L; \vee_L; \wedge_L \rangle$ e $\langle M; \vee_M; \wedge_M \rangle$ sono isomorfi se esiste una funzione biettiva $f: L \rightarrow M$ tale che per ogni x, y in L si ha che:

$$(i) f(x \vee_L y) = f(x) \vee_M f(y)$$

$$(ii) f(x \wedge_L y) = f(x) \wedge_M f(y).$$

Due algebre di Kleene $(A, \wedge, \vee, \neg, 0, 1)$ e $(B, \wedge, \vee, \neg, 0, 1)$ sono isomorfe se esiste una funzione biettiva $f: A \rightarrow B$ tale che:

$$(i) f(0) = 0, f(1) = 1$$

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

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

$$(iv) f(\neg x) = \neg f(x)$$

1.13.6 Teorema di rappresentazione di Birkhoff per reticoli finiti distributivi

Sia L un reticolo finito distributivo. Allora:

$\eta: L \rightarrow O(J(L))$ definita come

$$\eta(a) = \{x \in J(L) \mid x \leq a\}$$

è un isomorfismo di L su $O(J(L))$.

1.13.7 Corollario del teorema di rappresentazione di Birkhoff

Sia L un reticolo finito.

Allora le seguenti affermazioni sono equivalenti:

(i) L è distributivo

(ii) $L \cong O(J(L))$

(iii) L è isomorfo ad un reticolo di insiemi

(iv) L è isomorfo ad un sottoreticolo di 2^n per $n \geq 0$

1.13.8 Dualità fra reticoli finiti distributivi e insiemi finiti ordinati

Sia P un insieme finito ordinato.

Allora:

$$\varepsilon: x \rightarrow \downarrow x$$

è un isomorfismo:

$$P \cong J(O(P))$$

dove $(O(P), \subseteq)$ è un reticolo distributivo.

In altre parole, per ogni reticolo distributivo L esiste un poset (che poi non è altro che il poset $J(L)$ degli elementi join-irreducibile di L) tale che L è isomorfo all'insieme dei downset di $J(L)$, e viceversa, ogni poset P coincide con l'insieme degli elementi join-irreducibile di un reticolo distributivo.

1.13.9 Teorema

Sia $(A, \wedge, \vee, \neg, 0, 1)$ un algebra di Kleene.

Sia $(L, \wedge, \vee, 0, 1)$ un reticolo distributivo chiuso.

A è isomorfa a $K(L)$ se e solo se A è centrata e possiede la proprietà di interpolazione.

In altre parole, per ogni algebra di Kleene A centrata e con interpolazione esiste un reticolo distributivo L tale che A è isomorfo all'insieme delle coppie disgiunte di L . Mettendo insieme questo risultato con il precedente, che caratterizza i reticoli distributivi come insiemi di downset di poset, otteniamo il seguente teorema:

Per ogni algebra di Kleene A , centrata e con interpolazione, esiste un poset P tale che A è isomorfa all'algebra delle coppie disgiunte di downset di P .

GMaker

2.1 Panoramica del software

2.1.1 Panoramica

GMaker è un software che permette di visualizzare grafi (o equivalentemente, diagrammi di Hasse).

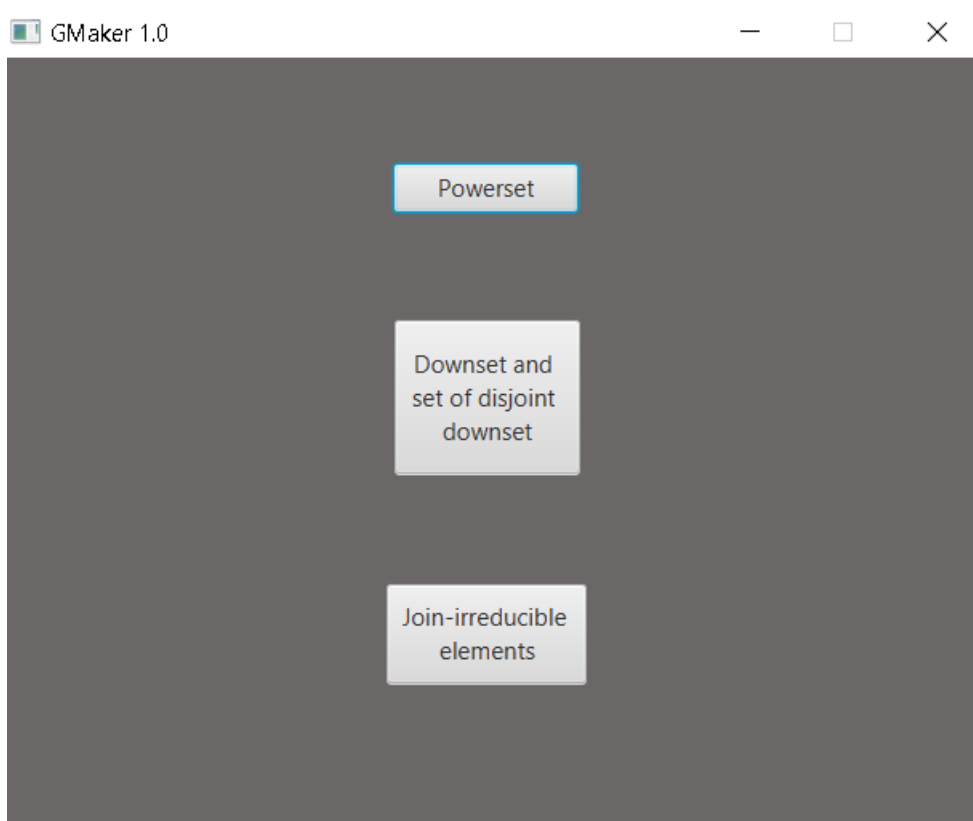
E' scritto interamente in Java, e per la generazione del grafo si appoggia ad una libreria software, Graphviz.

GMaker presenta un'interfaccia grafica realizzata con la tecnologia JavaFX.

Per poter utilizzare il software, è necessario scaricare Graphviz, e modificare la variabile di sistema PATH, ovvero aggiornando il suo valore col path corrente di Graphviz.

E' possibile visualizzare vari grafi, che rappresentano determinate strutture algebriche:

- (i) $P(A)$: insieme delle parti di un insieme A
- (ii) $O(A)$: insieme dei down-sets di un insieme ordinato A
- (iii) $(U(A), \leq)$: insieme dei down-sets disgiunti
- (iv) $J(A)$: insieme degli elementi join-irreducible



2.1.2 $P(A)$: insieme delle parti di un insieme A

L'input richiesto per la generazione del diagramma di Hasse dell'insieme delle parti di un insieme (powerset) è:

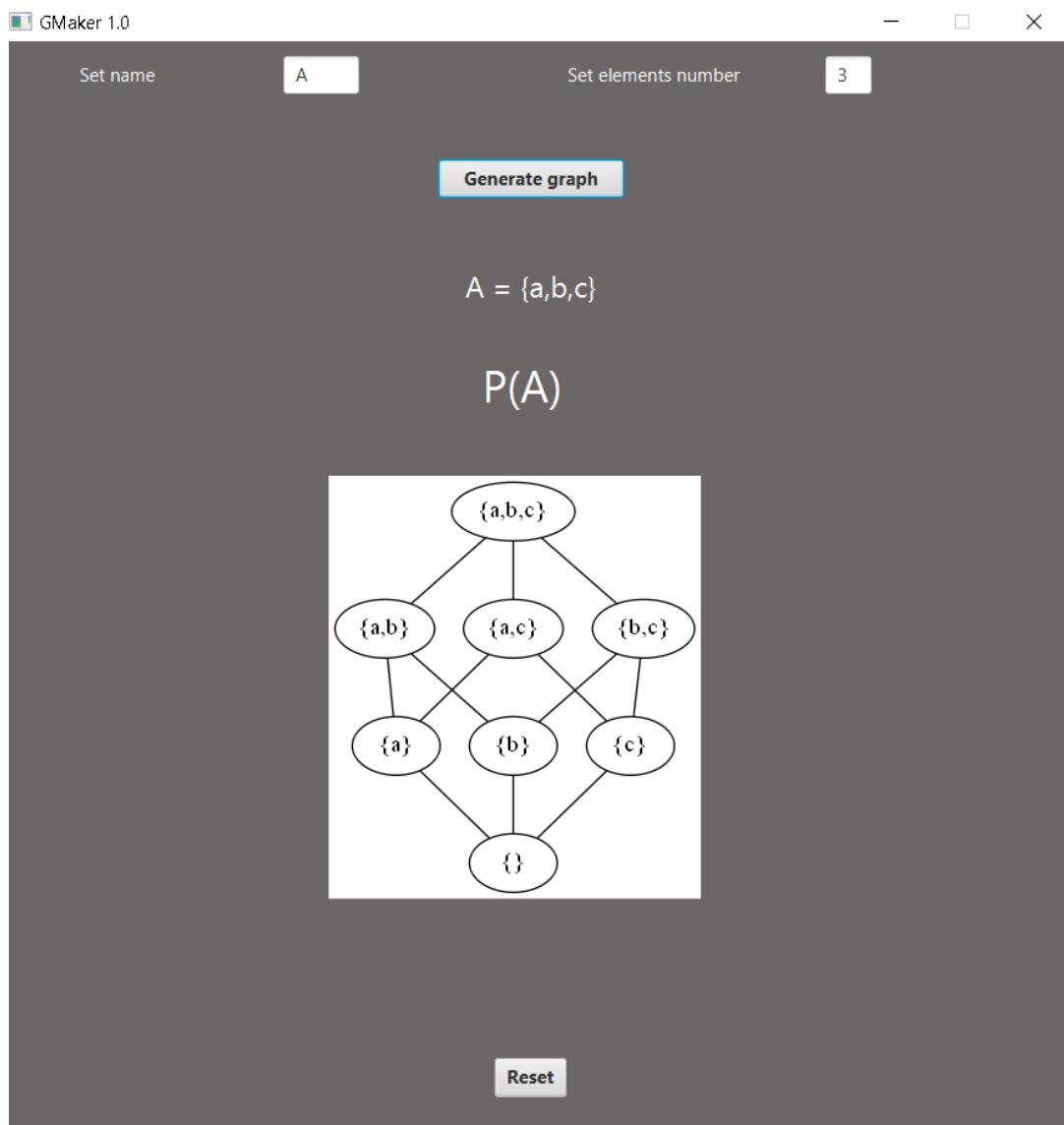
- (i) nome dell'insieme
- (ii) numero di elementi

Gli elementi dell'insieme sono lettere dell'alfabeto, a partire da “a”, “b”, e così via.

Dopo aver specificato nome dell'insieme e numero di elementi, viene generato il diagramma di Hasse corrispondente.

Esempio

Una volta inseriti i dati richiesti, dopo aver cliccato su “Generate graph”, il grafo viene visualizzato nella medesima pagina. Se si vuole generare un nuovo grafo, è sufficiente cliccare su “Reset”.



2.1.3 $O(A)$: insieme dei down-sets di un poset A

L'input richiesto per la generazione del diagramma di Hasse dell'insieme dei down-sets di un insieme ordinato A differisce dall'input richiesto per la generazione del precedente diagramma. Infatti, in input si avrà un grafo.

Nello specifico, in input è richiesto un insieme ordinato, pertanto è necessario specificare la relazione d'ordine (\leq) presente fra i vari elementi dell'insieme.

Dunque, ciò che viene richiesto è:

- (i) nome dell'insieme
- (ii) numero di elementi
- (iii) relazione d'ordine fra gli elementi dell'insieme

A questo punto, viene generato il grafo dell'insieme ordinato A . Questo grafo sarà il grafo in input richiesto per la generazione del grafo dell'insieme dei down-sets di A .

Esempio

Inseriti nome e numero di elementi dell'insieme, e dopo aver correttamente aggiunto le relazioni fra gli elementi, cliccando su "Generate input graph" viene visualizzato il grafo in input.

GMaker 1.0

Set name Set elements number

For each element in the set, establish the relation with other elements

Add

Relations

(a , b)

(b , c)

Generate input graph

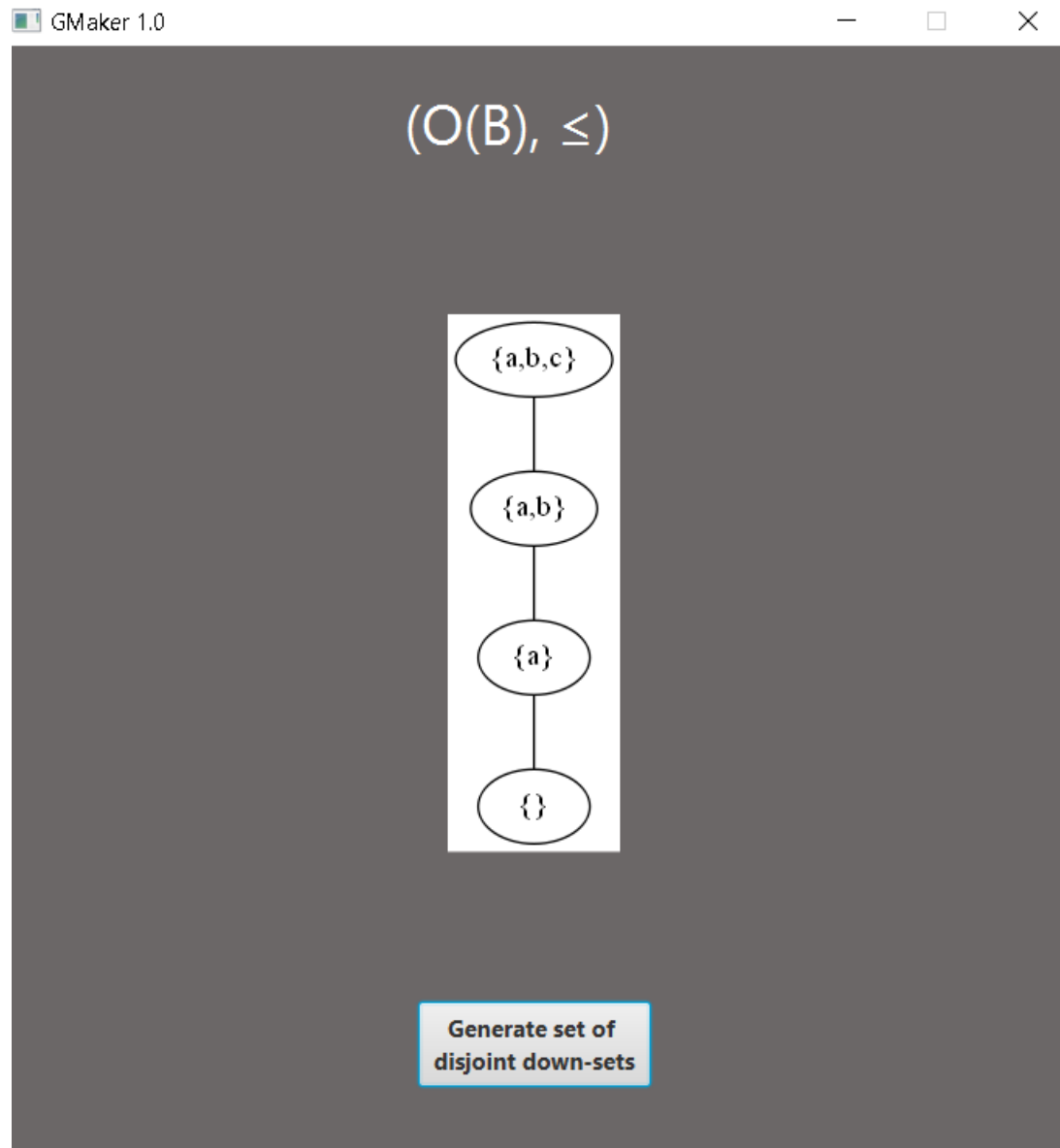
(B, \leq)

```

graph BT
    a((a)) --- b((b))
    b --- c((c))
  
```

Generate Downset

Successivamente, cliccando su “Generate Downset”, si apre una nuova finestra nella quale viene visualizzato il grafo che rappresenta l’insieme dei down-sets.



Se si vuole generare un nuovo grafo, è sufficiente cliccare su “Reset” (nella pagina di generazione del grafo in input).

2.1.3.1 Relazione fra elementi dell'insieme

Per ogni elemento dell'insieme, bisogna specificare la relazione d'ordine dell'elemento rispetto a uno o più elementi.

Le opzioni disponibili per un elemento sono:

- (i) \leq
- (ii) no relation

Esempio

Sia C l'insieme:

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

La relazione d'ordine fra gli elementi è la seguente:

$$b \leq c$$

Per generare il grafo che rappresenta il poset, si indica che a non ha nessuna relazione fra gli elementi dell'insieme. Cliccando su “Add”, la relazione viene aggiunta.

GMaker 1.0

Set name

C

Set elements number

3

For each element in the set, establish the relation with other elements

a

no relation

Add

Relations

a: no relation

Element "a" has no order relation. Add a relation or proceed with graph generation

Generate input graph

Successivamente, si va a specificare la relazione $b \leq c$.

The screenshot shows the GMaker 1.0 interface. At the top, there are two input fields: "Set name" with the value "C" and "Set elements number" with the value "3". Below these, there are two main panels. The left panel, titled "For each element in the set, establish the relation with other elements", contains three dropdown menus: the first is set to "b", the second to "<=", and the third to "c". To the right of these dropdowns is a blue "Add" button. The right panel, titled "Relations", displays the text "a: no relation" and "(b , c)". Below the "Add" button, a green message states "Relation (b , c) successfully added! Add another relation or proceed with graph generation". At the bottom center, there is a button labeled "Generate input graph".

A questo punto, per ogni elemento dell'insieme è stato specificata la relazione d'ordine (infatti $b \leq c$, e dunque $c > b$), o è stato specificato che l'elemento non è in relazione con nessun altro elemento. Si può dunque procedere con la generazione del grafo.

2.1.4 $U(A)$: insieme dei down-sets disgiunti di un poset A

L'input richiesto per la generazione dell'insieme dei down-sets disgiunti di un insieme ordinato A è l'insieme dei down-sets generato al passo precedente.

Infatti:

$$U(A) = \{ (X, Y) \in O(A) \times O(A) \mid X \cap Y = \emptyset \}$$

dove X, Y sono insiemi $\in O(A)$.

Esempio

Il procedimento è analogo a quanto descritto per la generazione dell'insieme dei down-sets.

Set name: A

Set elements number: 3

For each element in the set, establish the relation with other elements

Element	Relation	Element
a	<=	c

Relations

- (a, b)
- (a, c)

Buttons: Add, Reset

Generate input graph

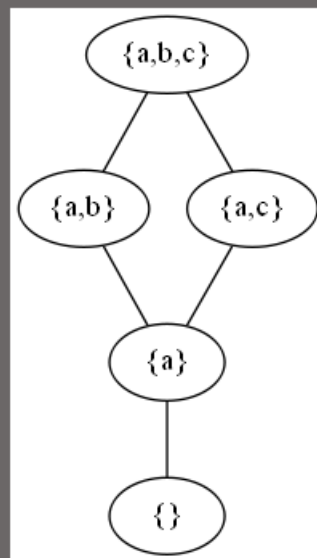
(A, \leq)

Graph visualization:

```
graph BT; a((a)) --- b((b)); a --- c((c));
```

Generate Downset

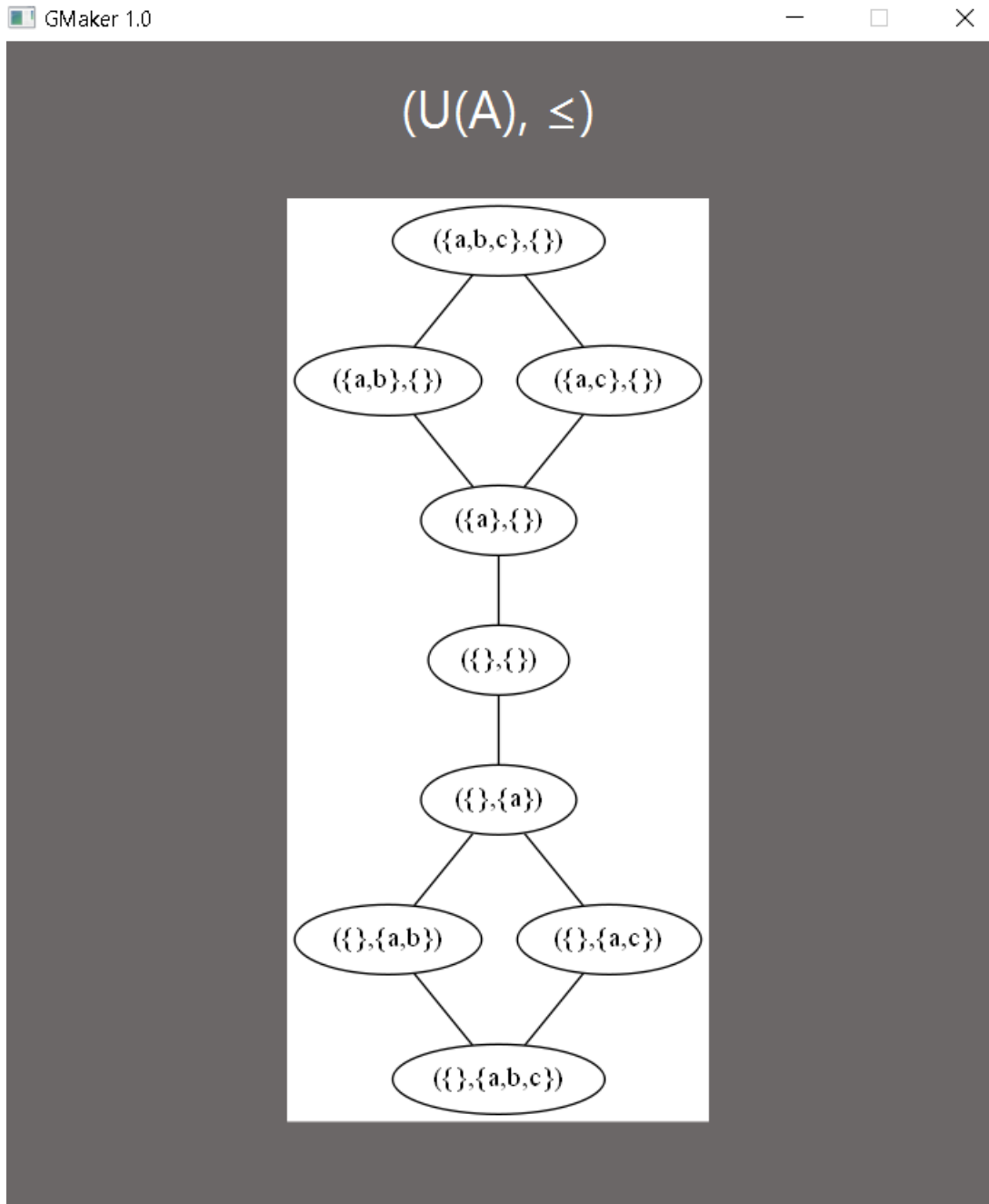
$(O(A), \leq)$



**Generate set of
disjoint down-sets**

**Generate set of
join-irreducible
elements**

A questo punto, cliccando su “Generate set of disjoint down-sets”, si aprirà una nuova finestra in cui verrà visualizzato il grafo.



2.1.5 $J(A)$: insieme degli elementi join-irreducible di un poset A

L'input richiesto per la generazione dell'insieme degli elementi join-irreducible di un insieme ordinato A è l'insieme $O(A)$.

Ciò che viene richiesto è:

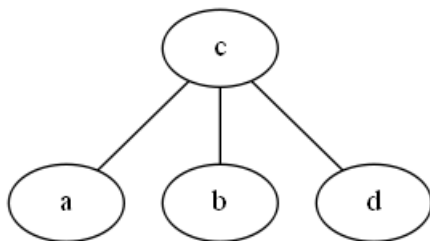
- (i) nome dell'insieme
- (ii) numero di elementi
- (iii) relazione d'ordine fra gli elementi dell'insieme

Dopo l'inserimento dei dati, viene generato l'insieme dei down-sets del poset A. Questo grafo è pertanto l'input per la generazione dell'insieme degli elementi join-irreducible.

Esempio

Dopo aver inserito nome dell'insieme, numero di elementi e relazione fra questi ultimi, cliccando su "Generate input graph" viene generato il grafo in input, che rappresenta l'insieme dei down-sets.

NOTA: il grafo dell'insieme ordinato è il seguente:



GMaker 1.0

Set name: **A** Set elements number: **4**

For each element in the set, establish the relation with other elements

d **<=** **c** **Add**

Relations

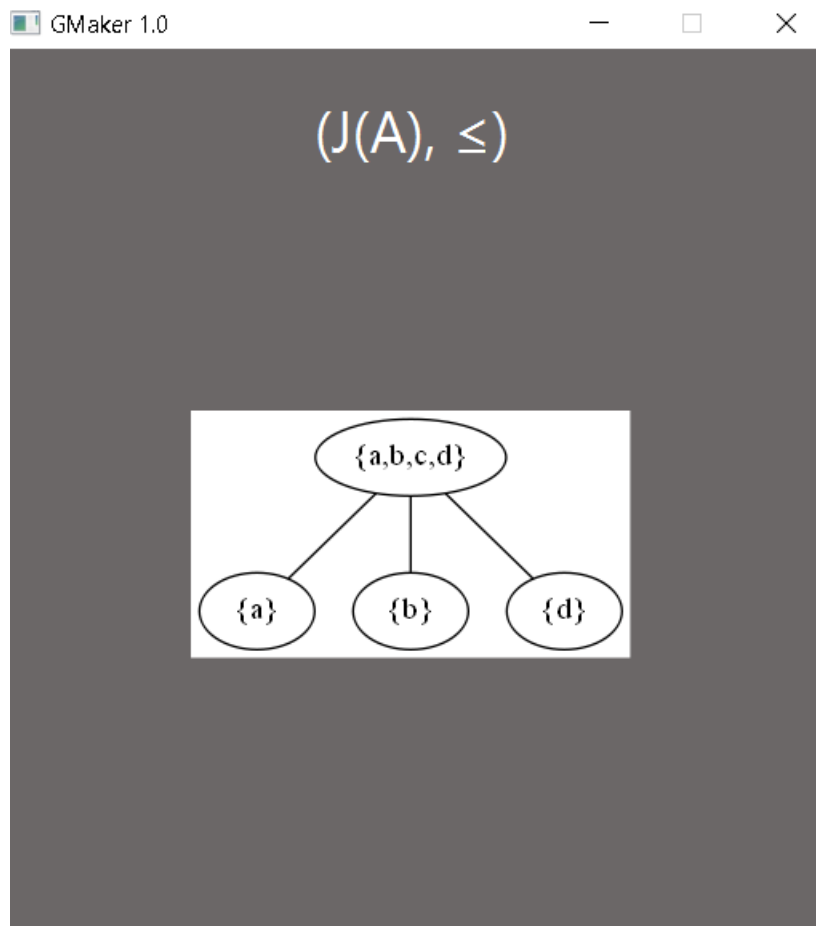
(a , c)
(b , c)
(d , c)

Generate input graph

Generate join-irreducible elements graph

A questo punto, cliccando su “Generate join-irreducible elements graph”, si aprirà una nuova finestra in cui verrà visualizzato il grafo.

Da notare che, alternativamente, è possibile generare il grafo dal menu di visualizzazione del down-set.



2.2 Descrizione degli algoritmi

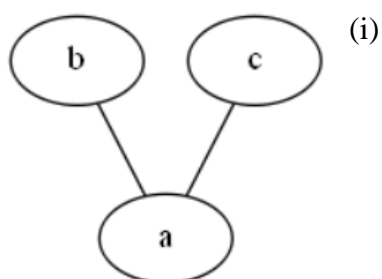
2.2.1 Algoritmo per la generazione dell'insieme dei down-sets

Per descrivere come opera l'algoritmo di generazione dell'insieme dei down-sets, si parte suddividendo logicamente l'input.

L'input è un insieme ordinato, visto anche come un grafo.

Si consideri, come esempio, il seguente grafo ordinato:

Esempio



Le relazioni d'ordine presenti sono le seguenti:

$a < b$

$a < c$

Il primo passo dell'algoritmo è la distinzione fra due tipi di elementi: gli elementi di livello 0 e gli elementi di livello 1.

Un elemento x è un elemento di livello 0 se:

(i) x è minore di un elemento, oppure

(ii) x non ha nessuna relazione d'ordine con altri elementi

Nell'esempio (i), l'elemento a appartiene agli elementi di livello 0.

Un elemento y è un elemento di livello 1 se y è maggiore di un elemento.

Nell'esempio (i), b e c appartengono entrambi agli elementi di livello 1.

Da notare che un elemento può essere sia di livello 0 che di livello 1.

L'algoritmo riconosce gli elementi che appartengono ad entrambe le categorie, e li memorizza in una struttura dati, eliminandoli dalle strutture dati che contengono, rispettivamente, elementi di livello 0 ed elementi di livello 1.

L'algoritmo è suddiviso fondamentalmente in 6 passi.

Gli elementi vengono memorizzati in una struttura dati di tipo Set, in cui un elemento non viene aggiunto nel caso in cui sia già presente.

Esempio di dato di tipo Set:

(i) $\{a\}$

(ii) $\{a, b\}$

(iii) $\{\}$

Se si cerca di aggiungere l'elemento $\{b, a\}$ ad una struttura dati tipo Set contenente l'elemento $\{a, b\}$, l'elemento $\{b, a\}$ non viene aggiunto.

Di seguito, i passi in pseudolinguaggio:

(i) ADD \emptyset TO DOWNSET

```
(ii) FOR EACH  $X \in \text{LVL } 0 \text{ ELEMENTS}$ {  
    ADD X TO DOWNSET  
}  
FOR EACH  $Y \in (\text{LVL } 0 \text{ ELEMENTS AND LVL } 1 \text{ ELEMENTS})$ {  
    REMOVE Y FROM DOWNSET  
}
```

```
(iii) INT Z := 2  
WHILE(Z <= LVL 0 ELEMENTS NUMBER){  
    IF(Z = 2){  
        ADD COUPLE (X1, X2) TO DOWNSET  
        //gli elementi della coppia sono diversi  
        ADD COUPLE (X1, X3) TO DOWNSET  
        ...  
    }  
    IF(Z = 3){  
        ADD TRIPLE (X1, X2, X3) TO DOWNSET  
        //gli elementi della tripla sono diversi  
        ...  
    }  
    ...  
    IF(Z = n){  
        ADD (X1, X2, ..., Xn) TO DOWNSET  
        //gli elementi sono tutti distinti  
        ...  
    }  
    Z = Z + 1  
}
```

```

(iv) FOR EACH X ∈ LVL 1 ELEMENTS{
    IF( (X IS NOT UPPER ELEMENT OF MORE THAN 1 RELATION) AND
        (X IS NOT A LEVEL0 AND LEVEL1 ELEMENT) ){
        ADD (X AND ITS PREDECESSOR) TO DOWNSET
        //il predecessore è l'elemento minore dell'elemento x
    }
}

(v) IF(ALL RELATIONS HAVE NOT SAME UPPER ELEMENT){
    FOR EACH RELATION R{
        ADD (R1, R2, X) TO DOWNSET
        /* R1 e R2 sono primo e secondo termine della
relazione, X è un elemento di livello 0 */
    }
}

IF(ALL RELATIONS HAVE SAME LOWER ELEMENT L){
    FOR EACH LEVEL 1 ELEMENT X{
        ADD (L, X, Y) TO DOWNSET
        /* Y è un altro elemento di livello 1, tale che Y ≠ X */
    }
}

(vi) ADD (ALL SET ELEMENTS) TO DOWNSET

```

2.2.2 Algoritmo per la generazione del set of disjoint down-sets

In input si riceve una struttura dati contenente i down-sets.

Gli elementi disgiunti vengono memorizzati in una struttura dati di tipo vettore.

Di seguito, i passi in pseudolinguaggio:

```
FOR EACH ELEMENT X ∈ DOWNSET{
    FOR EACH ELEMENT Y ∈ DOWNSET{
        IF (X = ∅ AND Y = ∅) {
            ADD (X, Y) TO SET OF DISJOINT DOWNSET
        }
        ELSE{
            IF (X ∩ Y = ∅) {
                ADD (X, Y) TO SET OF DISJOINT DOWNSET
            }
        }
    }
}
```

2.2.3 Algoritmo per la generazione dell'insieme dei join-irreducible elements

L'algoritmo riceve in input l'insieme dei down-sets di un insieme ordinato.

Gli elementi join-irreducible vengono memorizzati in una struttura dati di tipo Set.

Di seguito, i passi in pseudolinguaggio:

```
FOR EACH ELEMENT  $X \in \text{DOWNSET}$ {  
    IF ( $X$  IS A SINGLETON) {  
        ADD  $X$  TO JOIN-IRREDUCIBLE ELEMENTS SET  
    }  
    ELSE {  
         $C = 0$   
         $C := \text{LOWER\_COVERS}(X)$   
        IF ( $C = 1$ ) {  
            ADD  $X$  TO JOIN-IRREDUCIBLE ELEMENTS SET  
        }  
    }  
}
```

La funzione LOWER_COVERS ha come parametro un insieme, diverso da singleton, e ritorna il numero di insiemi che il parametro copre.

2.3 Graphviz

2.3.1 Descrizione

Graphviz (Graph visualization software) è una libreria software che consente di disegnare grafi mediante una specifica del grafo in linguaggio DOT.

Il linguaggio DOT è un linguaggio che consente di creare grafi: ciò significa che, mediante tale linguaggio, è possibile specificare che tipo di grafo si vuole implementare (ad esempio grafo aciclico diretto oppure non diretto), numero di nodi, archi, layout di nodi e archi, e molto altro.

Un file DOT è un tipo di file con estensione gv oppure dot.

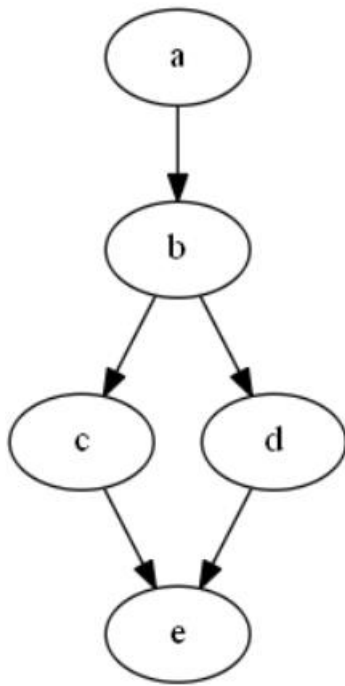
Un file DOT viene processato da un programma; tra questi, dot è il programma che GMaker richiama per processare il file DOT e renderizzarlo in forma grafica.

Esempio

Un esempio di file DOT è il seguente:

```
digraph graphname{  
a -> b;  
b -> c;  
b -> d;  
c -> e;  
d -> e;  
}
```

Il file DOT viene passato in input al programma dot, che lo processa e genera il seguente grafo diretto:



La prima parola che compare nel file indica il tipo di grafo che si vuole rappresentare; in questo caso, “digraph” sta per directed graph, ovvero un tipo di grafo in cui gli archi sono orientati, cioè hanno una direzione.

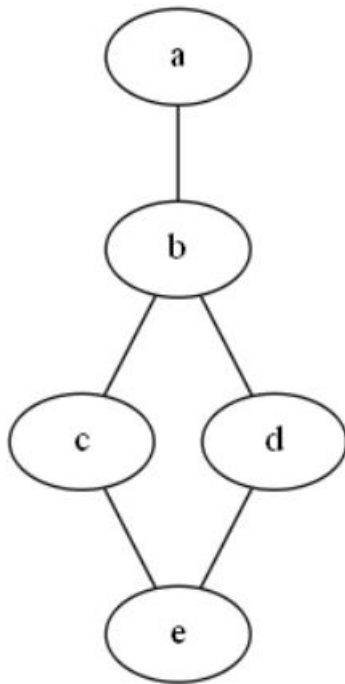
A seguire vi è il nome del grafo, e all’interno delle parentesi graffe si specificano i nodi (in questo caso a, b, c, d, e) e gli archi fra i vari nodi.

Dato che si è specificato di implementare un grafo diretto, gli archi fra due nodi si indicano con “->”.

Nel seguente esempio, si decide di implementare un grafo indiretto.

```
graph graphname{
a -- b;
b -- c;
b -- d;
c -- e;
d -- e;
}
```

L'output generato è il seguente:



In questo file, compare la parola chiave “graph”, che sta ad indicare un tipo di grafo indiretto.

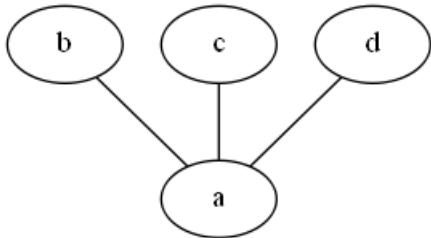
Gli archi fra i vari nodi vengono specificati mediante “--”.

E’ possibile invertire la direzione. Negli esempi precedenti, il grafo veniva costruito dall’alto verso il basso. Per convenzione, gli insiemi ordinati si disegnano dal basso verso l’alto. Con il linguaggio DOT, è possibile fare ciò specificando la direzione scelta. Sono possibili quattro opzioni:

- (i) TB (top to bottom)
- (ii) BT (bottom to top)
- (iii) LR (left to right)
- (iv) RL (right to left)

Di seguito un esempio dove `rankdir = BT`

```
graph relations_graph{ rankdir=BT{a}--{b}  
{a}--{c}  
{a}--{d}  
}
```



L'output che viene generato può essere di vari formati, tra cui png, svg e pdf.

I software che prendono in input file dot o file gv, tra cui dot, sono generalmente privi di interfaccia grafica.

Per renderizzare il file dot, si richiama il programma dot tramite una shell, specificando nome del file dot in input, e nome del file di output.

Esempio

Nel seguente esempio, viene creato il file `output.png` da riga di comando:

```
Command Prompt  
C:\Users\Ale\Desktop>dot -Tpng input.dot > output.png  
C:\Users\Ale\Desktop>
```

2.3.2 Algoritmi per la creazione dei file DOT

2.3.2.1 File DOT per down-sets

Il metodo Java utilizzato per la creazione del file DOT prende in input la struttura dati contenente gli elementi del down-set.

Come si è visto, la sintassi prevede che un nodo nel grafo sia racchiuso fra doppi apici, e per collegare due nodi fra loro si usano i caratteri "--" (il grafo è indiretto).

Pseudolinguaggio:

```
FOR EACH ELEMENT X IN THE DOWN-SET{
    IF (X IS A SINGLETON) {
        STRING S := "EMPTY SET" -- "X"
        ADD S TO FILE DOT
    }
}

FOR EACH SINGLETON X IN THE DOWN-SET{
    FOR EACH ELEMENT Y IN THE DOWN-SET{
        IF (Y IS A 2-ELEMENTS SET) {
            IF (X IS CONTAINED IN Y) {
                STRING S := "X" -- "Y"
                ADD S TO FILE DOT
            }
        }
    }
}
```

Il ciclo for qui sopra si ripete fino a che le variabili X ed Y non contengono, rispettivamente, un insieme con n-1 elementi, ed un insieme con n elementi (dove n è il numero di elementi contenuto nell'insieme più grande del down-set).

Esempio

Il seguente grafo viene dato in input:

GMaker 1.0

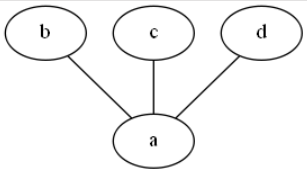
Set name Set elements number

For each element in the set, establish the relation with other elements

Relations

(a , b)
(a , c)
(a , d)

(A, \leq)



```
graph downsets_graph{ rankdir=BT
    "{a,b,c}" -- "{a,b,c,d}"
    "{a,b,d}" -- "{a,b,c,d}"
    "{a,b}" -- "{a,b,c}"
    "{a,b}" -- "{a,b,d}"
    "{a,c,d}" -- "{a,b,c,d}"
    "{a,c}" -- "{a,b,c}"
    "{a,c}" -- "{a,c,d}"
    "{a,d}" -- "{a,b,d}"
    "{a,d}" -- "{a,c,d}"
    "{a}" -- "{a,b}"
    "{a}" -- "{a,c}"
    "{a}" -- "{a,d}"
    "{}" -- "{a}"
}
```



2.3.2.2 File DOT per set of disjoint down-sets

Il metodo Java utilizzato per la creazione del file DOT prende in input la struttura dati contenente l'insieme dei down-set disgiunti.

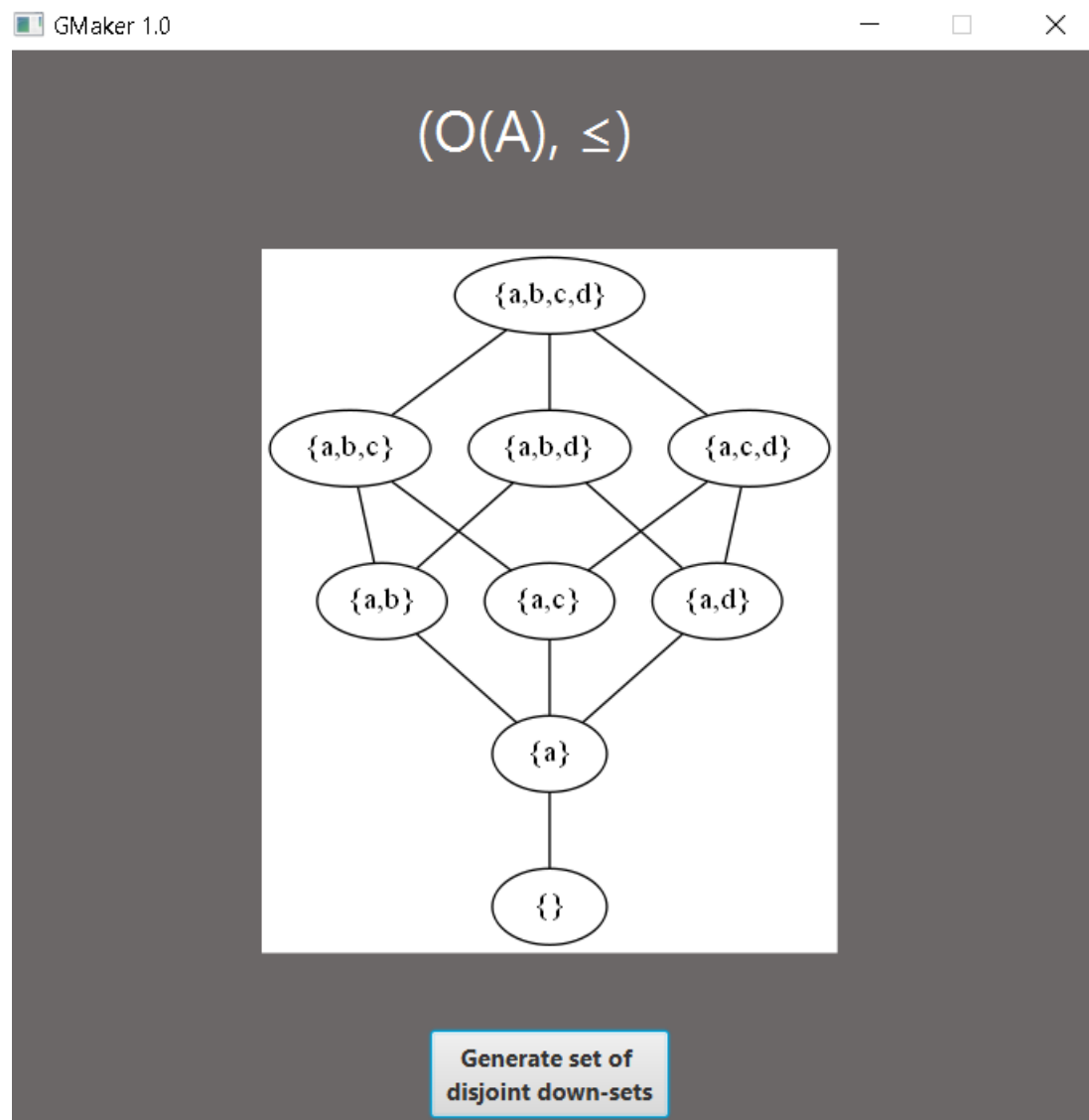
La struttura dati conterrà elementi del tipo (x, y) , dove x e y sono disgiunti fra di loro.

Pseudolinguaggio:

```
FOR EACH ELEMENT X IN THE DISJOINT SET{
    FOR EACH ELEMENT Y IN THE DISJOINT SET{
        X1 := X.GET_FIRST_ELEMENT
        X2 := X.GET_SECOND_ELEMENT
        Y1 := Y.GET_FIRST_ELEMENT
        Y2 := Y.GET_SECOND_ELEMENT
        IF( (X1 IS INCLUDED IN Y1) AND (X2 INCLUDES Y2) ){
            STRING S := "X" - "Y"
        }
    }
}
```


Esempio

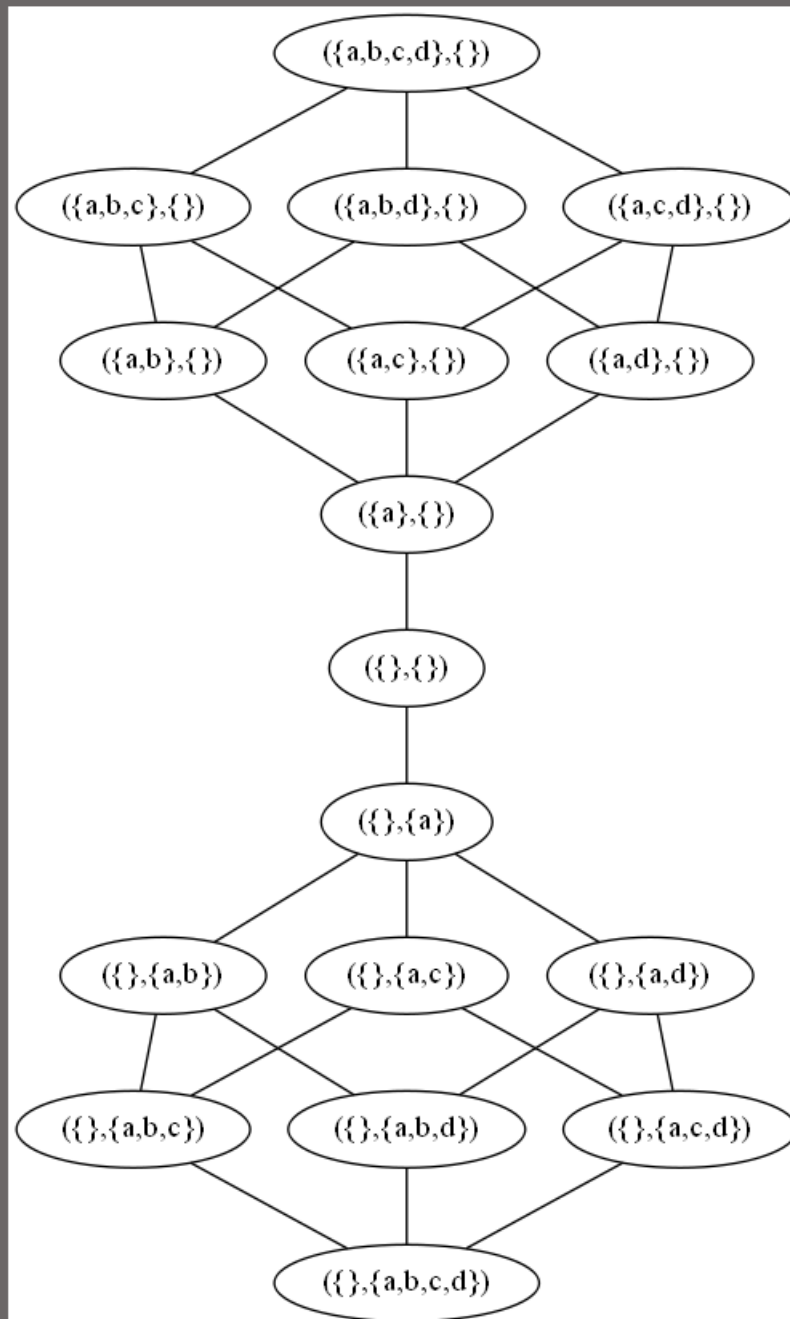
Il seguente grafo viene dato in input:



Di seguito un esempio di file DOT di un insieme di down-sets disgiunti, e successiva renderizzazione.

```
graph set_of_disjoint_downsets_graph{ rankdir=BT"({a,b,c},{})" --
"({a,b,c,d},{})"
"({a,b,d},{})" -- "({a,b,c,d},{})"
"({a,b},{})" -- "({a,b,c},{})"
"({a,b},{})" -- "({a,b,d},{})"
"({a,c,d},{})" -- "({a,b,c,d},{})"
"({a,c},{})" -- "({a,b,c},{})"
"({a,c},{})" -- "({a,c,d},{})"
"({a,d},{})" -- "({a,b,d},{})"
"({a,d},{})" -- "({a,c,d},{})"
"({a},{})" -- "({a,b},{})"
"({a},{})" -- "({a,c},{})"
"({a},{})" -- "({a,d},{})"
"({},{a,b,c,d})" -- "({},{a,b,c})"
"({},{a,b,c,d})" -- "({},{a,b,d})"
"({},{a,b,c,d})" -- "({},{a,c,d})"
"({},{a,b,c})" -- "({},{a,b})"
"({},{a,b,c})" -- "({},{a,c})"
"({},{a,b,d})" -- "({},{a,b})"
"({},{a,b,d})" -- "({},{a,d})"
"({},{a,b})" -- "({},{a})"
"({},{a,c,d})" -- "({},{a,c})"
"({},{a,c,d})" -- "({},{a,d})"
"({},{a,c})" -- "({},{a})"
"({},{a,d})" -- "({},{a})"
"({},{a})" -- "({},{})"
"({},{})" -- "({a},{})"
}
```

$(U(A), \leq)$



3 Bibliografia

- ROBERTO CIGNOLI, The class of Kleene algebras satisfying an interpolation property and Nelson algebras, *Algebra Universalis*, 23 (1986).
- Davey B.A., Priestley H.A, *Introduction to Lattices and Order*, Cambridge University Press, 1990,
- Graphviz: <https://www.graphviz.org/>
- Stackoverflow: <https://stackoverflow.com/>