Computing on-line the covering graph of the ideal lattice of posets

Claire Diehl, Claude Jard, Jean-Xavier Rampon

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Computing on-line the covering graph of the ideal lattice of posets

Claire DIEHL
Claude JARD
Jean-Xavier RAMPON

N° 1864
Mars 1993

PROGRAMME 1

Architectures parallèles,
Bases de données,
Réseaux et Systèmes distribués

Computing on-line the Covering Graph of the Ideal Lattice of Posets

Claire DIEHL, Claude JARD, Jean-Xavier RAMPON
e-mail: (jard,rampon)@irisa.fr

abstract

Partial ordered sets appear in a lot of branches of computer science. In some cases, people are interested in manipulating the associated set of all the linear extensions. This leads directly to the problem of building the ideal lattice of posets. Furthermore, we require that the poset is only known dynamically, element by element, in order to permit an on-line construction of the lattice. This has practical interest notably in the area of on-the-fly testing of distributed systems. We present an original algorithm to do that, with a time complexity comparable to the best known off-line algorithm. We also detail two particular cases of practical interest which have a better performance: the case in which the order elements are given as a linear extension, and the case in which the poset is partitioned into chains.

Calcul à la volée du graphe de couvertures du treillis des ideaux d'un ensemble ordonné

résumé

De nombreuses branches de l'informatique sont aujourd'hui concernées par la théorie des ensembles ordonnés. Lors de l'utilisation d'un ensemble ordonné comme outil de modélisation, un des paramètres fréquement considéré est l'ensemble de ses extensions linéaires et par là même le treillis de ses idéaux. Les objets ainsi modélisés ont bien souvent un caractère dynamique important comme dans le cadre de la vérification des exécutions réparties. Nous nous sommes donc interessés à la construction à la volée du treillis des idéaux d'un ensemble ordonné dont les éléments ne sont connus que les uns après les autres. Nous présentons un algorithme à la volée original permettant cette construction en une complexité comparable avec celle des meilleurs algorithmes connus qui eux, nécessitent la connaissance préalable de l'ensemble ordonné dans sa totalité. Lorsque les éléments de l'ensemble ordonné sont donnés suivant une de ses extensions linéaires ou suivant une décomposition en chaînes, de légères modifications permettent alors d'améliorer la complexité obtenue. Nous détaillons ces deux cas qui revêtent un interêt pratique.

Computing on–line the Covering Graph of the Ideal Lattice of Posets

Contents

1 Introduction

Designing on–line algorithms is a natural idea when one has to process dynamic and time–evolving information. In an on–line approach, data is only available piece by piece. At each new added part, the algorithm carries on its computation, accumulating the result dedicated to the new piece of data, to the already computed results. We present here how to compute on–line the covering graph of the ideal lattice of posets.

This research has been motivated by practical problems in the context of parallel programs debugging. For the goal of testing, the suspected misbehavior of the system under test is described by a global property (basically a global predicate on variables, or the possible occurrence orders of observable events). The problem is to verify whether this property is satisfied or not during the execution. In the context of asynchronous parallelism on distributed architectures that we considered, the correct evaluation of global properties requires a careful analysis of the causal structure of the execution. The causal structure, induced by the message exchanges between processes in the distributed system, forms a partial order, as Lamport remarked in 1978. As a consequence, numerous questions about distributed executions refer to the notions of linear extensions and order ideals (also called consistent cuts). Actually, one can based the testing method on a kind of reachability analysis [6] which exactly builds the covering graph of the ideal lattice of the causality relation. In that context, testing must be performed on-the-fly, i.e. in parallel with the execution of the application under test.

The problem of building the covering graph of the ideal lattice of a poset has already been considered by theoreticians. Up to our knowledge, the best algorithm was given by Bordat, two years ago [3]. Its time complexity is proportional to the size of the lattice that multiplies the width of the order. But it does not work on–line at all, since it is based on a Depth–First–Search of the whole poset. Nevertheless, Bordat uses an efficient labeling for the edges of the ideal lattice that we recover fruitfully. At the same time, the on–line approach to that problem has also been considered by Cooper and Marzullo [5] in the applied context of detecting global predicates in distributed computations. Their algorithm, which requires however to process the order elements level by level, is based on a naive (and costly) enumeration.

Our technical contribution is the design of the first algorithm that permits to build on–line the covering graph of the ideal lattice of a general partially ordered set. Its time complexity is asymptotically comparable to the Bordat's algorithm. We have also enlighten two particular cases of practical interest which have a better performance: the case in which the order elements are given as a linear extension, and tne case in which the poset is partitioned into chains.

The paper is organized as follows. After having introduced in Section 2 different notations and preliminaries, we characterize by Theorem 1 of Section 3 the exact relation between the covering graphs of the ideal lattices yielded by a poset and by one of its subposet when a unique vertex is missing. We then present the assumptions for computing on–line in Section 4, i.e. the exact nature of inputs and our model for complexity analysis. The general algorithm is structured into two parts. Since the lattice construction proceeds by successive granulations. We distinguish in Section 5.1, the "supremum algorithm" which have to find in the already built lattice, the starting point of granulation. In Section 5.2, the "granulation algorithm" duplicates a part of the lattice rooted by the element returned by the supremum algorithm and builds the expected lattice by creating some edges. We continue by the study of the two particular cases of linear extension input hypothesis in Section 6.1 and the chain partition hypothesis in Section 6.2. Some long proofs of complexity results are postponed in the annexes.

2 Definitions and preliminaries

A set P associated with a partial order relation (i.e. an antisymmetric, transitive and reflexive or irreflexive binary relation on P) is called a partially ordered set or a *poset*. If the relation is reflexive such a poset is written $\widetilde{P} = (P, \leq_{\widetilde{P}})$ else $\widetilde{P} = (P, <_{\widetilde{P}})$.

Let x and y be two elements of P:

We say that x and y are *comparable* in $\widetilde{P}$, when either $x \leq_{\widetilde{P}} y$ or $y \leq_{\widetilde{P}} x$. Otherwise, x and y are said to be *incomparable* in $\widetilde{P}$. If $x \leq_{\widetilde{P}} y$ holds, then

x is a *predecessor* of y in $\tilde{P}$ and y is a *successor* of x in $\tilde{P}$.

We say that x is *covered* by y in $\tilde{P}$, and we write $x \text{-}<_{\tilde{P}} y$, if $x <_{\tilde{P}} y$ and $\forall z \in P$, $(x <_{\tilde{P}} z \leq_{\tilde{P}} y) \Rightarrow (z = y)$; x is an *immediate predecessor* of y in $\tilde{P}$ and y is an *immediate successor* of x in $\tilde{P}$. The directed graph associated of this *covering relation* (i.e. the transitive reduction of $<_{\tilde{P}}$) is the *covering graph* of $\tilde{P}$ and is denoted by $Cov(\tilde{P}) = (P, E_P)$.

Given a graph $G = (P, E_P)$, for any subset A of P, $E_P(A)$ is the set of the edges corresponding to the subgraph of G on A.

Let A be a subset of P:

The *subposet* of $\tilde{P}$ on A, $\tilde{P}/_A = \tilde{A} = (A, \leq_P)$ is the poset induced by $\tilde{P}$ on A.

For $|A| \geq 2$, if all elements of A are pairwise comparable (resp. incomparable) in $\tilde{P}$, A is a *chain* (resp. an *antichain*) in $\tilde{P}$. The *height* of $\tilde{P}$, $h(P)$, is the maximum cardinality minus one of a chain in $\tilde{P}$. The *width* of $\tilde{P}$, $w(P)$, is the maximum cardinality of an antichain in $\tilde{P}$. A *chain decomposition* of $\tilde{P}$ is a partition $\{P_i\}_{i \in I}$ of P where each P_i is a chain in $\tilde{P}$.

$Max(A, \tilde{P}) = \{a \in A, \forall x \in A, (a \leq_{\tilde{P}} x) \Rightarrow (a = x)\}$ is the *maximal* elements set of A in $\tilde{P}$. Analogously, $Min(A, \tilde{P}) = \{a \in A, \forall x \in A, (x \leq_{\tilde{P}} a) \Rightarrow (a = x)\}$ is the *minimal* elements set of A in $\tilde{P}$.

$\downarrow_{\tilde{P}} A] = \{x \in P, \exists a \in A, x \leq_{\tilde{P}} a\}$ (resp. $\uparrow_{\tilde{P}} A] = \{x \in P, \exists a \in A, a \leq_{\tilde{P}} x\}$) is the *predecessor set* (resp. *successor set*) of A in $\tilde{P}$.
$\downarrow_{\tilde{P}} A[= \downarrow_{\tilde{P}} A] \setminus A$ (resp. $\uparrow_{\tilde{P}} A[= \uparrow_{\tilde{P}} A] \setminus A$) is the *strictly predecessor set* (resp. *strictly successor set*) of A in $\tilde{P}$.
$\downarrow^{im}_{\tilde{P}} A = Max(\downarrow_{\tilde{P}} A[, \tilde{P})$ (resp. $\uparrow^{im}_{\tilde{P}} A = Min(\uparrow_{\tilde{P}} A[, \tilde{P})$) is the *immediate predecessor set* (resp. *immediate successor set*) of A in $\tilde{P}$.
If A is a singleton $\{x\}$, we shall simply write $\downarrow_{\tilde{P}} x]$, $\uparrow_{\tilde{P}} x]$, $\downarrow_{\tilde{P}} x[$, $\uparrow_{\tilde{P}} x[$, $\downarrow^{im}_{\tilde{P}} x$ and $\uparrow^{im}_{\tilde{P}} x$.

A *linear extension* of a poset $\widetilde{P}$ is a chain $\widetilde{C}$ that preserves $\widetilde{P}$, i.e. $x \leq_{\widetilde{P}} y \Rightarrow x \leq_{\widetilde{C}} y$.

A is an *ideal* of $\widetilde{P}$ iff it contains all its predecessors $\downarrow_{\widetilde{P}} A] = A$. $I(P)$ is the set of all ideals of $\widetilde{P}$ and $\widetilde{I(P)}$ is this set ordered by inclusion [1].

A has an *infimum* (resp. a *supremum*) in $\widetilde{P}$ if $|Max(\{x \in P, x \leq_{\widetilde{P}} y \ \ \forall y \in A\}, \widetilde{P})| = 1$ (resp. $|Min(\{x \in P, y \leq_{\widetilde{P}} x \ \ \forall y \in A\}, \widetilde{P})| = 1$). We denote by $\bigvee_{\widetilde{P}} A$ (resp. $\bigwedge_{\widetilde{P}} A$) the supremum (resp. the infimum) of A in $\widetilde{P}$.

$\widetilde{P}$ is a *lattice* iff any of its two elements subset has a supremum and an infimum in $\widetilde{P}$.
Studies of correlations between poset properties and lattice properties are always of interest since the well known result of Birkhoff [1] on finite distributive lattices and posets. In the infinite case, an extension of this result and interesting properties can be found in Bonnet and Pouzet [2]. In the finite case, one of the most recent studies with an algorithmic point of view has been done by Bordat [4].

In this paper, we are only concerned by finite posets.

3 Structure of the covering relation of $\widetilde{I(P)}$

Our goal is to compute on-line the covering graph of the ideal lattice of a poset. Thus, we study correlations between covering relations of the ideal lattices yielded by a poset and by one of its subposet when a unique vertex is missing. Theorem 1 gives a complete characterization of correlations between these two covering relations.

For the proof of Theorem 1, we need the following lemma, saying that the ideals grow by adding one element at a time:

[1] For any I_1, I_2 in $I(P)$, $I_1 \cup I_2$ and $I_1 \cap I_2$ belong to $I(P)$. Moreover, $I_1 \cup I_2$ (resp. $I_1 \cap I_2$) is the smallest (resp. greatest) element of $I(P)$ including (resp. included in) both I_1 and I_2. Thus $\widetilde{I(P)}$ is a lattice.

Lemma 1 *Let $\tilde{Q}$ be a poset,*
$\forall I, J \in I(Q)$, $I \prec_{\widetilde{I(Q)}} J \Longleftrightarrow I \subset J$ *and* $|J \setminus I| = 1$

Proof: (i) Assume that $I \subset J$, then $I <_{\widetilde{I(P)}} J$. The result follows directly from the fact that $\forall Z, K \in I(P)$, $Z <_{\widetilde{I(P)}} K \Longrightarrow Z \subset K$.
(ii) Assume that $I \prec_{\widetilde{I(Q)}} J$, then $I \subset J$. If $J \setminus I = \emptyset$ then $I = J$ which contradicts $I \prec_{\widetilde{I(Q)}} J$. If $|J \setminus I| \geq 2$, let $x, y \in J \setminus I$ with $x \neq y$. Without loss of generality, we can assume that $y \not<_{\tilde{Q}} x$, then $I <_{\widetilde{I(Q)}} I \cup \downarrow_{\tilde{Q}} x] <_{\widetilde{I(Q)}} J$ which contradicts again $I \prec_{\widetilde{I(Q)}} J$.
□

$J \setminus I$ is called the *label* of the edge $I \prec_{\widetilde{I(Q)}} J$.

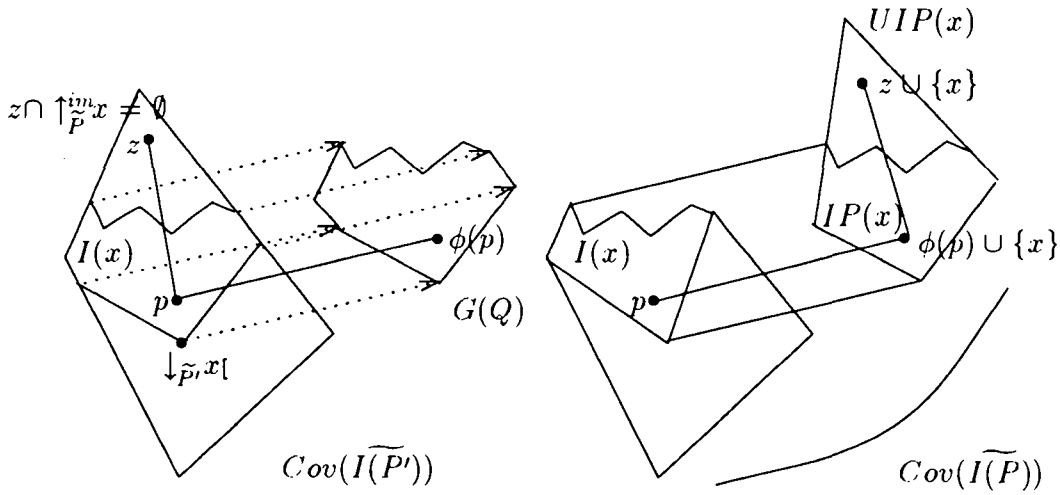

Figure 1: Illustration of Theorem 1

Let $\tilde{P}$ be a poset and x one of its element. As one can see in Figure 1, $\widetilde{I(P)}$ is structured in three different parts. The upper part $UIP(x)$ is formed with the ideals containing an immediat successor of x: $UIP(x) = \{I \in I(P) : I \cap \uparrow^{im}_{\tilde{P}} x \neq \emptyset\}$. The medium part $MI(x)$ is formed with the ideals containing the immediat predecessors of x and none of its successors: $MI(x) = \uparrow_{\widetilde{I(P)}} (\downarrow_{\tilde{P}} x[)] \setminus UIP(x)$. $MI(x)$ is partitioned in two isomorphic sublattices: $I(x) = MI(x) \setminus \{I \in MI(x) : x \in I\}$

and $IP(x) = MI(x) \setminus I(x)$. $I(x)$ is formed with the ideals of $MI(x)$ not containing x. The lower part is formed with the remaining ideals. This is formalized by Theorem 1.

Theorem 1 *Let $\widetilde{P}$ be a poset, let $x \in \widetilde{P}$, let $\widetilde{P'}$ be the subposet $\widetilde{P \setminus \{x\}}$ and $Cov(\widetilde{I(P')}) = (I(P'), E_{I(P')})$. Let $UI(x) = \{I \in I(P'), I \cap \uparrow_{\widetilde{P}}^{im} x \neq \emptyset\}$ and $I(x) = \uparrow_{\widetilde{I(P')}} (\downarrow_{\widetilde{P}} x[)] \setminus UI(x)$. Let $G(Q) = (Q, E_Q)$ be an acyclic directed graph isomorphic to $Cov(\widetilde{I(x)})$ by a map ϕ.*
Then, the acyclic directed graph $G(Z) = (Z, E_Z)$ where $Z = Q \uplus I(P')$ and E_Z is defined by:

1. $\forall q_1, q_2 \in Q \times Q : q_1 q_2 \in E_Z \iff \phi^{-1}(q_1)\phi^{-1}(q_2) \in E_{I(P')}$
2. $\forall p, q \in I(P') \times Q :$
   (a) $pq \in E_Z \iff q = \phi(p)$
   (b) $qp \in E_Z \iff (\phi^{-1}(q)p \in E_{I(P')}$ *and* $p \in UI(x))$
3. $\forall p_1, p_2 \in I(P') \times I(P') :$
   $p_1 p_2 \in E_Z \iff (p_1 p_2 \in E_{I(P')}$ *and* $p_1, p_2 \notin I(x) \times UI(x))$

is isomorphic to $Cov(\widetilde{I(P)})$ by the map φ:

$$\varphi : z \longmapsto \begin{cases} z \cup \{x\} & \text{if } z \in UI(x) \\ \phi^{-1}(z) \cup \{x\} & \text{if } z \in Q \\ z & \text{otherwise} \end{cases}$$

Proof: Since $\widetilde{P'}$ is a subposet of $\widetilde{P}$ it is clear that $\varphi(Z) \subseteq I(P)$, and that for $I \in I(P)$ if $x \notin I$ then $I \in I(P') \setminus UI(x)$, if $\exists y \in I$ such that $x \leq_{\widetilde{P}} y$ then $I \setminus \{x\} \in UI(x)$ otherwise $I \setminus \{x\} \in I(x)$. Thus $I(P) \subseteq \varphi(Z)$ and then φ is a bijection from Z to $I(P)$. It remains to show that φ is a morphism from $G(Z)$ to $Cov(\widetilde{I(P)})$.
Let us denote by $UIP(x)$ the set $\{I \cup \{x\},\ I \in UI(x)\}$ and by $IP(x)$ the set $\{I \cup \{x\},\ I \in I(x)\}$ (remark that $I(P) = (I(P') \setminus UI(x)) \uplus IP(x) \uplus UIP(x)$). Since $\forall K \in IP(x)$ we have $K \not\leq_{\widetilde{I(P)}} I$, $\forall I \in I(P') \setminus UI(x)$ and we have $J \not\leq_{\widetilde{I(P)}} K$, $\forall J \in UIP(x)$. It is then clear that the subgraphs of $G(Z)$ on $I(P') \setminus UI(x)$ respectively on $UI(x)$ are isomorphic by the corresponding restriction of φ to the subgraphs of $Cov(\widetilde{I(P)})$ on $I(P') \setminus UI(x)$ respectively on $UIP(x)$. Let ψ be a map from $I(x)$ to $IP(x)$ such that $\psi(I) = I \cup \{x\}$. Since ψ is bijective and since $\forall I, J \in I(x),\ I \subset J \iff \psi(I) \subset \psi(J)$, the

subgraphs of $Cov(\widetilde{I(P)})$ on $I(x)$ respectively on $IP(x)$ are isomorphic.
In order to conclude the proof it remains to study the edge connections between the three subgraphs of $Cov(\widetilde{I(P)})$ on $I(P') \setminus UI(x)$, $IP(x)$ and $UIP(x)$. First we are going to show that $\forall I \in IP(x)$, $\exists! I' \in I(P') \setminus UI(x)$ such that $I' -<_{\widetilde{I(P)}} I$ and that $I' = I \setminus \{x\}$. By definition of $IP(x)$, $\forall I \in IP(x)$, $I \setminus \{x\} \in I(P' \setminus UI(x)$ thus by Lemma 1 $I' -<_{\widetilde{I(P)}} I$. We are going to show the unicity of I': assume that for $I \in IP(x)$ there exists $I'_1, I'_2 \in I(P') \setminus UI(x)$ covered by I. Since $x \in I$ and x is neither in I'_1 nor in I'_2 then by Lemma 1 $I'_1 = I \setminus \{x\}$ and $I'_2 = I \setminus \{x\}$. Thus $I'_1 = I'_2$. As consequence, $\forall I \in I(x)$ and $\forall J \in UIP(x)$, I is not covered by J in $\widetilde{I(P)}$.
Since $\widetilde{P'}$ is a subposet of $\widetilde{P}$ it is clear that $\forall I \in I(P') \setminus (I(x) \cup UI(x))$, $\forall J \in UI(x)$, $\exists K \in I(x)$ such that $I <_{\widetilde{I(P')}} K <_{\widetilde{I(P')}} J$ and thus $I <_{\widetilde{I(P)}} K <_{\widetilde{I(P)}} K \cup \{x\} <_{\widetilde{I(P)}} J \cup \{x\}$. It remains to show that $\forall I, J \in IP(P) \times UIP(x)$ we have $I -<_{\widetilde{I(P)}} J \iff J \setminus I = \{y\} \in \uparrow^{im}_{\widetilde{P}} x$ which is immediate with Lemma 1 and that $I -<_{\widetilde{I(P)}} J \iff I \setminus \{x\} -<_{\widetilde{I(P')}} J \setminus \{x\}$ which is a consequence of Lemma 1 and also that $\widetilde{P'}$ is a subposet of $\widetilde{P}$.
□

To give an idea of the overall construction, let us take an example of poset $\widetilde{P}$, whose the covering graph is given by Figure 2. We also suppose that the elements are successively read in the order a, d, b, c, e, g, f. For each element, we know its immediat successor and predecessor sets (figured by local parts of the covering graph).

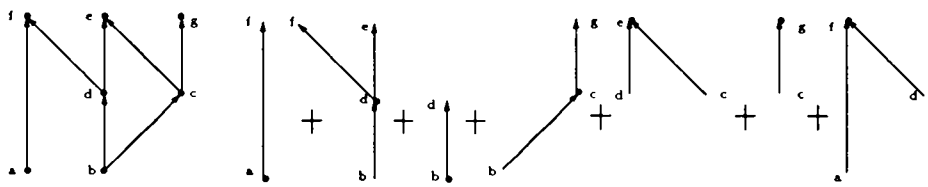

Figure 2: $Cov(\widetilde{P})$ and the input sequence

Figure 3 shows how the lattice grows when adding one element at a time. It illustrates the behavior of the covering algorithms that are described in detail in the following. The set $I(x)$ of the elements to be duplicated when reading a new vertex x is represented with black dots.

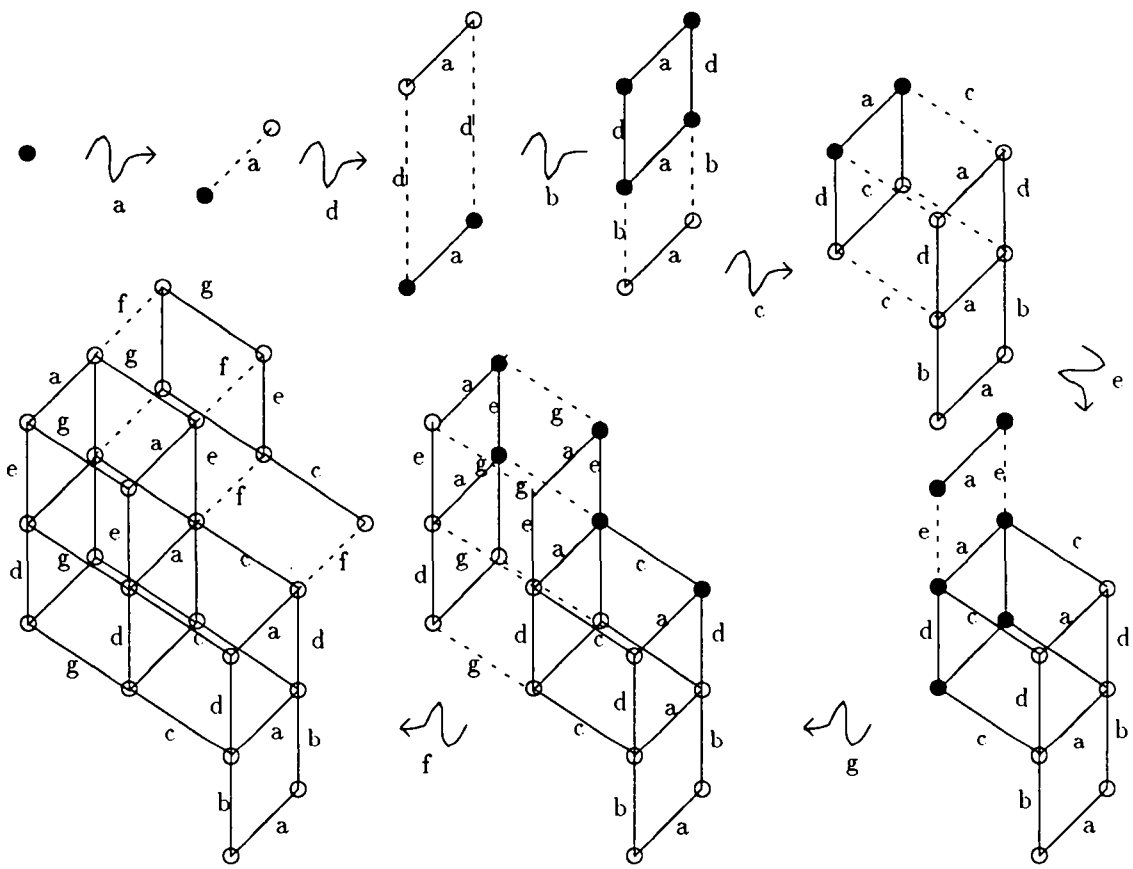

Figure 3: Illustration of the covering construction

4 General assumptions for computing on-line

Following Theorem 1, the body structure of each proposed algorithm computing the covering graph of the ideal lattice of a poset $\widetilde{P}$ can be decomposed in two main parts. The first one is devoted to find the supremum of the set $D(x)$ in $I(\widetilde{P'})$ (the "supremum algorithm") and the second one deals with the concrete building of the covering graph: creation of the isomorphic subgraph, addition and deletion of edges (the "granulation algorithm"). The combination of these two algorithms produces the so called "covering algorithm" which makes evolve the ideals lattice at each new input vertex.

4.1 Model for complexity analysis

The model used for complexity analysis is standard:

* Memory allocation is proportional to the size allocated.
* Memory access is performed in constant time.
* Data are stored as usually under binary digits. Thus addition or subtraction of two integers n and m are performed in $log(max(n,m))$.
* In order to be in accordance with the usual complexity analysis, the equality test on data is assumed to be performed in constant time [2].

4.2 Input and data assumptions

We make the assumption that each vertex of the incoming sets is directly related to corresponding vertex in the covering graph. This assumption is taken in order to simplify the algorithmic part and has no consequences on the announced complexities for whole computation of the covering graph of the ideal lattice. Indeed if we don't follow the previous assumption, we can

[2] However, if we don't follows this assumption a factor $log(max(n,m))$ must be multiplied when n and m are the size of the compared data

make the connection between a vertex of an incoming set and its corresponding vertex in the covering graph in $\mathcal{O}(log(|P'|))$ when $\widetilde{P'}$ is the known poset [3].

The incoming sets are assumed to be stored by increasing labels since such a sort can be achieved in $\mathcal{O}(|w(P')|\ log(|w(P')|)$ using for example a Quick-Sort algorithm.

Similarly, the incoming sets are assumed to be the immediat predecessors (resp. successors) set. Indeed, finding the maximal (resp. minimal) elements set of a given subset of a poset $\widetilde{P'}$ can be achieved in $\mathcal{O}(|P'| + |E_{P'}|)$ which belongs to $\mathcal{O}(|P'|\ w(P'))$.

By aim of simplicity, when dealing with vertices of the ideal lattice, we confuse vertices of $I(P)$ with the ideal they represent, knowing that such a vertex will be never related with the set of vertices corresponding with the ideal it represents. This amalgam is of importance to express with clarity properties of these vertices.

5 Computation of the covering graph of $I(\widetilde{P})$ The general case

5.1 The supremum algorithm

The problem of finding $\downarrow_{\tilde{P}} x[$ in $I(\widetilde{P'})$ comes to find in $I(\widetilde{P'})$ the supremum of a set of its join irreducible elements corresponding to an antichain in $\tilde{P}$.

For this purpose we introduce the following proposition which, given a poset $\tilde{Q}$, describes the structure of maximal paths joining any two elements subset of $I(Q)$ to they supremum in $I(\widetilde{Q})$. Moreover, Proposition 1 gives rise to an efficient mechanism to reach this supremum starting from any elements of the subset.

[3] In order to do this, it is sufficient to store the vertex set of $\widetilde{P'}$ through, for example, an A.V.L. tree (Adelson-Velskii-Landis 1962). In the same way updating this data structure with the new incoming vertex for the new poset can also be achieved in $\mathcal{O}(log(|P'|)$.

Proposition 1 *Let $\widetilde{Q}$ be a poset and assume that each edge yz of the covering graph of $\widetilde{I(Q)}$ is labeled by $I_z \setminus I_y$ where I_y (resp. I_z) is the ideal corresponding to y (resp. z) in $\widetilde{I(Q)}$. Let $I_1, I_2 \in I(Q)$, then any path of the covering graph of $\widetilde{I(Q)}$ from I_1 to $I_1 \cup I_2$ have the same length and the label set of its edges is exactly $\downarrow_{\widetilde{Q}} I_1] \setminus \downarrow_{\widetilde{Q}} I_2]$. Moreover, there exists at least one path.*

Proof: Since $I_1 \leq_{\widetilde{I(Q)}} (I_1 \cup I_2)$ there exists a path from from I_1 to $I_1 \cup I_2$. The equality of lengths of all paths from I_1 to $I_1 \cup I_2$ is a direct consequence of the fact that $\widetilde{I(Q)}$ is a distributive lattice and thus graded.
Let S be the label set of a given path from I_1 to $I_1 \cup I_2$: by definition of an ideal it is clear that $S \subseteq \downarrow_{\widetilde{Q}} I_1] \setminus \downarrow_{\widetilde{Q}} I_2]$, and that $\downarrow_{\widetilde{Q}} I_1] \setminus \downarrow_{\widetilde{Q}} I_2] \subseteq S$ is a direct consequence of Lemma 1.
□

The "supremum algorithm" described behind takes as inputs the covering graph of a poset $\widetilde{Q}$, the covering graph of its the ideal lattice $\widetilde{I(Q)}$ (with some additional conditions on these covering graphs) and $D(x)$ an antichain of $\widetilde{Q}$. It returns the infimum of $\widetilde{I(Q)}$ if $D(x)$ is the empty set and the supremum of $D(x)$ in $\widetilde{I(Q)}$ otherwise.

This algorithm makes a full use of Proposition 1. Indeed, following this proposition, in order to reach the supremum of $D(x)$ we have first to find the supremum in $\widetilde{I(Q)}$ of any two elements subset of $D(x)$. Then starting from this supremum and a new element of $D(x)$ we have to find they supremum in $\widetilde{I(Q)}$, it now suffices to repeat this last step until all elements of $D(x)$ have been chosen.

The "supremum algorithm"

Input:

1. $G = (Q, E_Q) = Cov(\tilde{Q})$ the covering graph of a poset $\tilde{Q}$.
2. $D(x)$ an antichain in $\tilde{Q}$. Such that each element of $D(x)$ is directly related to its corresponding vertex in $Cov(\tilde{Q})$.
3. The covering graph of $\widetilde{I(Q)}$ such that:
   (a) Each $y \in Q$ is directly related to its corresponding ideal in $\widetilde{I(Q)}$.
   (b) Each edge YZ is labeled by $Z \setminus Y$.
   (c) Outgoing edges of any vertices are stored ordered by increasing label.

Output:

The supremum of $D(x)$ in $\widetilde{I(P)}$: I.

Body:

Initialization:

10: If $|D(x)| = 0$ then return the infimum of $\widetilde{I(Q)}$; Exit;
20: Let $y \in D(x)$; Delete y from $D(x)$;
30: Let I be the ideal corresponding to y in $\widetilde{I(Q)}$;
40: $G := Cov(\tilde{Q})$ where $\tilde{Q} := Q \setminus \widetilde{\downarrow_{\tilde{Q}} y]}$;

Algorithm:

50: While $D(x) \neq \emptyset$ do
  60: Let $z \in D(x)$; Delete z from $D(x)$;
  70: Let S be the list of all elements of $\downarrow_{\tilde{Q}} z]$ sorted by increasing label;
  80: $G := Cov(\tilde{Q})$ where $\tilde{Q} := Q \setminus \widetilde{\downarrow_{\tilde{Q}} z]}$;
  90: While $S \neq \emptyset$ do
    100: Let $J \in \uparrow^{im}_{\widetilde{I(Q)}} I$ such that the label of the edge IJ is corresponding with an element of S;
    110: $I := J$;
    120: Delete from S the element corresponding with the label of the edge IJ;
  EndWhile of line 90
EndWhile of line 50
130: Return I;

Theorem 2 *Let y be the first vertex chosen in $D(x)$, then the "supremum algorithm" has a running time complexity of:*

$$\mathcal{O}(|E_G(\downarrow_{\widetilde{Q}} D(x))| + \underset{I \in I(Q)}{Max} |\uparrow^{im}_{\widetilde{I(Q)}} I||\downarrow_{\widetilde{Q}} D(x)| + \sum_{z_i \in D(x)\setminus\{y\}} |\downarrow_{\widetilde{Q}_i} z_i|^2)$$

which belongs to $\mathcal{O}(w(Q)\,|Q|^2)$. [4]

Proof: We are going to prove the termination and the correctness of the "supremum algorithm". The proof of the complexity of this algorithm is described in Annex 1.

Let $D_0(x)$ resp. $\widetilde{Q_0}$ be the set $D(x)$ resp. the poset $\tilde{Q}$ given as inputs. Let $D_i(x)$, resp. G_i and resp. I_i be the set $D(x)$, resp. the covering graph G and the current ideal at the i^{th} incomming in the While loop of Step 50. Assume that $I_i = \downarrow_{Q_0} D_0(x) \setminus D_i(x)]$, that $G_i = Cov(Q_0 \setminus \downarrow_{Q_0} (\widetilde{D_0(x)} \setminus D_i(x))])$ and that $|D_i(x)| = |D_0(x)| - i$. We are going to show that $I_{i+1} = \downarrow_{Q_0} D_0(x) \setminus D_{i+1}(x)]$, that $G_{i+1} = Cov(Q_0 \setminus \downarrow_{Q_0} (\widetilde{D_0(x)} \setminus D_{i+1}(x))])$ and that $|D_{i+1}(x)| = |D_0(x)| - (i+1)$.

By assumption on $|D_i(x)|$ it is clear that $|D_{i+1}(x)| = |D_0(x)| - (i+1)$ (Step 60). By assumption on G_i and I_i it is clear that $S = (\downarrow_{\widetilde{Q_0}} z]) \setminus (\downarrow_{\widetilde{Q_0}} I_i])$ and that $G_i = Cov(Q_0 \setminus \downarrow_{Q_0} (\widetilde{D_0(x)} \setminus D_{i+1}(x))])$ (Step 80).

Let J_{ij} rep. S_j be the current ideal resp. the set S at the j^{th} incomming in the While loop of Step 90 during the i^{th} incomming in the While loop of Step 50. Assume that $J_{ij} = \downarrow_{\widetilde{Q_0}} (D_0(x) \setminus D_i(x)) \cup (S_1 \setminus S_j)]$, that $S_j = (\downarrow_{\widetilde{Q_0}} z]) \setminus (\downarrow_{\widetilde{Q_0}} J_{ij}])$, and that $|S_j| = |S_1| - (j-1)$. We are going to show that $S_{j+1} = (\downarrow_{\widetilde{Q_0}} z]) \setminus (\downarrow_{\widetilde{Q_0}} J_{i(j+1)}])$, that $|S_{j+1}| = |S_1| - j$ and that $J_{i(j+1)} = \downarrow_{\widetilde{Q_0}} (D_0(x) \setminus D_i(x)) \cup (S_1 \setminus S_{j+1})]$.

---

[4] The potentially heaviest factor for the time complexity of the "supremum algorithm" is in $\mathcal{O}(\sum_{z_i \in D(x)\setminus\{y\}} |\downarrow_{\widetilde{Q_i}} z_i]|^2)$. Thus:

1. The choice of the first vertex to be deleted from $D(x)$ may be important for an effective utilization of the algorithm.
2. The width of the poset plays an important part for the time complexity. Indeed if $|D(x)| \in \Omega(|Q|)$ and there is an equipartition of the vertices of $\downarrow_{\widetilde{Q}} D(x)]$ over the set $\{\downarrow_{\widetilde{Q_i}} z_i],\ z_i \in D(x) \setminus \{y\}\}$, then this factor belongs to $\mathcal{O}(|Q|)$. So, the time complexity of the "supremum algorithm" belongs to $\mathcal{O}(|Q|^2)$. On the contrary, if the poset is of bounded width the time complexity of the "supremum algorithm" also belongs to $\mathcal{O}(|Q|^2)$.

By assumption on S_j and J_{ij} with the Proposition 1 we known that there exists an outgoing edge of J_{ij} with a label belonging to S_j. Thus, with Step 100 and Step 110 we have $J_{i(j+1)} = \downarrow_{\widetilde{Q_0}}(D_0(x) \setminus D_i(x)) \cup (S_1 \setminus S_{j+1})]$ and by Step 120 we have $|S_{j+1}|=|S_1| - j$ and $S_{j+1} = (\downarrow_{\widetilde{Q_0}} z]) \setminus (\downarrow_{\widetilde{Q_0}} J_{i(j+1)}])$. Since by assumption on I_i we have on one hand $J_{i1} = \downarrow_{\widetilde{Q_0}} D_0(x) \setminus D_i(x)]$ and with the expression given bove for S on other hand $S_1 = \downarrow_{\widetilde{Q_0}} z] \setminus \downarrow_{\widetilde{Q_0}} J_{i1}]$, it is then clear that $I_{i+1} = (\downarrow_{Q_0} D_0(x)) \setminus (D_{i+1}(x)])$ and that the While loop of Step 90 ended (since $| S | \leq | Q_0 |$). Then since it is clear that $I_1 = \downarrow_{Q_0} D_0(x) \setminus D_1(x)]$, that $G_i = Cov(Q_0 \setminus \downarrow_{Q_0} \widetilde{D_0(x)} \setminus D_i(x)])$ and that $| D_1(x) | = | D_0(x) | - 1$, thus the correctness and the terminaison (since $|D_0(x)| \leq w(Q)$) of the "supremum algorithm" are achieved.
□

5.2 The granulation algorithm

The "granulation algorithm" described behind takes as inputs the covering graph of the ideal lattice of a poset $\widetilde{Q}$, an ideal of $\widetilde{I(Q)}$ and an antichain of $\widetilde{Q}$ (with some additional conditions on these inputs). It returns a graph including the covering graph of $\widetilde{I(Q)}$ where some edges have been deleted and a new graph isomorphic to a subgraph of input graph. Moreover, some edge connections between these two graphs have been done.

This algorithm is very simple in its principle. Indeed, it based on a Breadth-First-Search algorithm where some tests have been added. During the search the duplication is performed and the additional tests are used in order to stop the search on the right vertices. Nevertheless, for the efficiency of the algorithm we must have a careful look on the data structures involved.

The "granulation algorithm"

Input:

1. A vertex $x \notin Q$ such that $\forall y \in Q,\ label(y) <_{\mathbf{N}} label(x)$.
2. $I \in I(Q)$ and $U(x)$ an antichain in $\widetilde{Q}$ such that $U(x) \subseteq \uparrow_{\widetilde{Q}} I[$ and $I \subseteq \downarrow_{\widetilde{Q}} U(x)[$. Moreover, $U(x)$ is given by a list sorted by increasing label;
3. The covering graph of $\widetilde{I(Q)}$ such that:
   (a) Each vertex of $I(Q)$ is marked free.
   (b) Each edge YZ is directly related to vertex Z.
   (c) Each edge YZ is labeled by $Z \setminus Y$.
   (d) Outgoing edges of any vertices are stored ordered by increasing label.

Output:

A directed graph $G = (V_G, E_G)$ such that:

(a) $V_G = I(Q) \cup I'(x)$ where $I'(x) = \{I', I \in I(x)\}$ and $I(x) = \{J \in I(Q),\ I \leq_{\widetilde{I(Q)}} J \text{ and } J \cap U(x) = \emptyset\}$.
(b) The subgraph of $Cov(I(Q))$ on $I(x)$ is isomorphic to the subgraph of G on $I'(x)$ by the map $\phi(Y) = Y'$. Moreover, edges $Y'Z'$ and YZ have the same label.
(c) $\forall Y \in I(x),\ YY' \in E_G$ and is labeled by x.
(d) $\forall Y \in I(x) \text{ and } Z \in UI(x), YZ \in E_{I(Q))} \Longrightarrow (YZ \notin E_G \text{ and } Y'Z \in E_G)$. Where $UI(x) = \{I \in I(Q),\ I \cap U(x) \neq \emptyset\}$.
(e) Each vertex of V_G is marked free.
(f) Each edge YZ in E_G is directly related to vertex Z.
(g) Each edge YZ in $E_G \cap I(Q) \times I(Q)$ is labeled by $Z \setminus Y$.
(h) Outgoing edges of any vertices are stored ordered by increasing label.

In order to have a more readable algorithm, we introduce the two following macro instructions. The first one gives the sequence of statements related to vertex which has not to be duplicated. The second one gives the sequence of instructions related to the creation of a new vertex.

Blocking–Vertex(X)

10 Add $X'Z$ labeled by $label(XZ)$ and directly related to vertex Z at the end of $outgoingedges(X')$;
20 Delete XZ from $outgoingedges(X)$;

New–Vertex

10 Create the vertex Z';
20 Links vertex Z directly to vertex Z';
30 Mark vertex Z to not free;
40 Add vertex Z at the end of A;

Body:

Initialization:

10: $A := \emptyset$; Create the vertex I'; Links x directly to I';
20: Let $IZ := First(outgoingedges(I))$;
30: While $IZ \neq Tail(outgoingedges(I))$ do
40: If $label(IZ)$ is corresponding with an element of $U(x)$
Then Blocking-Vertex(I);
Else 50: New-Vertex;
60: Add $I'Z'$ labeled by $label(IZ)$ and directly related to vertex Z' at the end of $outgoingedges(I')$;
End If of line 40
70: Let $IZ := next(outgoingedges(I))$;
End While of line 30
80: Add the edge II', directly related to vertex I' and labeled by $label(x)$, at the end of $outgoingedges(I)$;

Algorithm:

90: While A not empty do
100: Let $Y := First(A)$; Delete Y from A;
110: Let $YZ := First(outgoingedges(Y))$;
120: While $YZ \neq Tail(outgoingedges(Y))$ do
130: If $label(YZ)$ is corresponding with an element of $U(x)$
Then Blocking-Vertex(Y);
Else 140: If Z is marked free then New-Vertex;
150: Add $Y'Z'$ labeled by $label(YZ)$ and directly related to vertex Z' at the end of $outgoingedges(Y')$;
End If of line 130
160: Let $YZ := next(outgoingedges(Y))$;
End While of line 120
170: Add the edge YY', directly related to vertex Y' and labeled by $label(x)$, at the end of $outgoingedges(Y)$;
End While of line 90

Restoring:

180: Mark free all vertices Y and delete direct links of type YY' which have been respectively marked not free and created during the two previous parts.

Theorem 3 *Let I be the ideal given as input, and let $I(x) = \{J \in I(Q),\ I \leq_{\widetilde{I(Q)}} J$ and $J \cap U(x) = \emptyset\}$. Then the "granulation algorithm" has a running time complexity in $\mathcal{O}(|I(x)|\ w(Q))$.*

Proof: Since the underlying structure of the "granulation algorithm" is a classical one, we only give the sketch of the proof for the termination and the correctness. The proof of the complexity of this algorithm is described in Annex 2.

In order to achieve this proof we must assume that the operator Deletion position the current file element on the element just preceding the deleted one. We will not consider the "Restoring" part since it is clearly achieved through a Breadth-First-Search algorithm when a vertex is added to the set of the main While loop if it is marked.

The terminaison of the algorithm is achieved since for the While loop of Step 30 resp. Step 120 every element chosen at the Step 70 resp. Step 160 is a new one. And since every element is not marked before it is added to A, marked after and never unmarked in the While loop of Step 90.

In order to clarify the proof of correctness, we denote by $I(x)$ the set $\{J \in I(Q),\ I \leq_{\widetilde{I(Q)}} J$ *and* $J \cap U(x) = \emptyset\}$ and by A_i the set A at the i^{th} incomming in the While loop of Step 90. Since the algorithm is based on a Breadth-First-Search, then the only and all $Y \in I(P)$ such that $I \leq_{\widetilde{I(P)}} Y$ are potentially added in A. By definition of the edge labeling of $Cov(\widetilde{I(P)})$ it is clear that $label(YZ) \in U(x) \Longrightarrow Z \cap U(x) \neq \emptyset$. Then, since $Z_1 \cap U(x) \neq \emptyset$ *and* $Z_1 \leq_{\widetilde{I(P)}} Z_2 \Longrightarrow Z_2 \cap U(x) \neq \emptyset$, the macro instruction Blocking-Vertex assure that $\forall Y \in A,\ Y \cap U(x) = \emptyset$. With the macro instruction New-Vertex and the test on the vertex mark (a vertex is never unmarked before the "Restoring" part) it is clear that $\forall Y \in I(x)$ then there exists i such that Y belongs to A_i and that $|\{Y,\ Y$ *is added to* $A\}|=|I(x)|$. In order to get the isomorphism and the considering set of edges, it is sufficient to notice that:

A vertex is duplicated iff it belong to $\bigcup A_i$.
An edge is duplicated (with the same label) iff it joins two vertices of $\bigcup A_i$.
An edge is deleted iff its ending vertex does not belongs to $I(x)$.
An edge is created iff or it joins a vertice and its duplicated version nor it replace an edge deleted but form the duplicated vertex to the ending vertex of the delted one.

The remaining is immediate.

□

5.3 The covering algorithm

The "covering algorithm" given in this section, allows the on-line computation of covering graph of the ideal lattice when a new vertex is added to the poset. It is the algorithmic expression of Theorem 1.

The "covering algorithm"

Input: [5]

1. $Cov(\widetilde{P'}) = (P', E_{P'})$ the covering graph of a poset $\widetilde{P'}$.
2. $Cov(\widetilde{I(P')}) = (I(P'), E_{I(P')})$ the covering graph of $\widetilde{I(P')}$ such that:
   (a) Each $y \in P'$ is directly related to its corresponding ideal (i.e. $\downarrow_{\widetilde{P'}} y$]) in $\widetilde{I(P')}$.
   (b) Each vertex of $I(P')$ is marked free.
   (c) Each edge YZ is directly related to vertex Z.
   (d) Each edge YZ is labeled by $Z \setminus Y$.
   (e) Outgoing edges of any vertices are stored ordered by increasing label.
3. The vertex x of P missing in P', labeled such that $\forall y \in P'$, $label(y) <_{\mathbb{N}} label(x)$.
4. The set $D(x) = \downarrow \overset{im}{\widetilde{P}} x$, such that each element of $D(x)$ is directly related to its corresponding vertex in $Cov(\widetilde{P'})$.
5. The set $U(x) = \uparrow \overset{im}{\widetilde{P}} x$ given by a list sorted by increasing label and such that each element of $U(x)$ is directly related to its corresponding vertex in $Cov(\widetilde{P'})$.

Output:

1. $Cov(\widetilde{P}) = (P, E_P)$ the covering graph of the poset $\widetilde{P}$.
2. $Cov(\widetilde{I(P)}) = (I(P), E_{I(P)})$ the covering graph of $\widetilde{I(P)}$ such that:
   (a) Each vertex of $I(P')$ is marked free.
   (b) Each edge YZ is directly related to vertex Z.
   (c) Each $y \in P$ is directly related to its corresponding ideal in $\widetilde{I(P)}$.
   (d) Each edge YZ is labeled by $Z \setminus Y$.
   (e) Outgoing edges of any vertices are stored ordered by increasing label.

---

[5]

1. All the assumptions on $\widetilde{P'}$ and $\widetilde{I(P')}$ are fulfill when the set P' is the empty set.
2. We can assume that $\forall y \in P'$, $label(y) \leq_{\mathbb{N}} |P'|$ and that $label(x) = |P'| + 1$.

Body:

1. Build $Cov(\tilde{P})$.
2. Find $\downarrow_{\tilde{P}'} D(x)]$ in $I(\widetilde{P'})$.
3. Build the graph $G(Q) = (Q, E_Q)$ isomorphic to $Cov(\widetilde{I(x)})$, where $I(x) = \{I \in I(P'),\ \lfloor_{\tilde{P}} x[\leq_{\widetilde{I(P')}} I \ \ and \ \ I \cap \uparrow_{\tilde{P}} x[= \emptyset\}$ by a map ϕ. Such that $\forall \phi(Y)\phi(Z) \in E_Q,\ \ \phi(Y)\phi(Z)$ is labeled by $Z \setminus Y$.
4. For any $I \in I(x)$, create the edge $(I, \phi(I))$ with label x and store this edge according to the storage order.
5. For any $I \in I(x)$ and any $J \in IU(x)$ such that the edge IJ belongs to $Cov(\widetilde{I(P')})$: delete the edge IJ and create the edge $(\phi(I), J)$ with label $J \setminus I$.
6. Create a link between x and $\phi(\lfloor_{\tilde{P}'} D(x)])$.

Theorem 4 *The "covering algorithm" runs in time complexity:*

$$\mathcal{O}([(|I(P)| - |I(P')|) + |P'|^2]w(P')).$$

Proof: The correctness of the "covering algorithm" is clearly achieved through Theorem 1. The time complexity analysis can be performed step by step:

* By hypothesis on $D(x)$, Step 1 can be performed in $\mathcal{O}(|D(x)|)$.
* Step 2 can be performed using the "supremum algorithm". From Theorem 2, we can easily deduce that the "supremum algorithm" time complexity belongs to $\mathcal{O}(|E_{P'}(\downarrow_{\tilde{P}'} D(x)])| + [w(P') + |D(x)|| \downarrow_{\tilde{P}'} D(x)]|]\ |\downarrow_{\tilde{P}'} D(x)]|)$. Since this algorithm "destroys" the input poset, we must give it a duplication of $Cov(\widetilde{P'})$ which can be performed in $\mathcal{O}(|P'| + |E_{P'}|)$. Thus, we are able to performs Step 2 in $\mathcal{O}(|P'| + |E_{P'}| + [w(P') + |D(x)|| \downarrow_{\tilde{P}'} D(x)]]\ || \downarrow_{\tilde{P}'} D(x)]|)$.
* Following Theorem 2, Step 3 to Step 6 can be achieved using, the "granulation algorithm". Thus the time complexity of these steps belongs to $\mathcal{O}((|I(P)| - |I(P')|)w(P'))$

In order to conclude, it is enough to notice that $D(x)$ is an antichain in $\widetilde{P'}$ and that the number of outgoing edges of any vertices in both the covering graph of $I(\widetilde{P'})$ and the covering graph of $\widetilde{P'}$ is at most the width of $\widetilde{P'}$.
□

As consequence of this theorem, we are able to achieve the computation of covering graph of the ideal lattice of a poset given any sequence of vertices assuming that at each step the growing poset is a subposet of final one. Particularly, this condition is fulfilled if the sequence of incoming vertices is a linear extension of the final poset.

Corollary 1 *Let $\widetilde{P}$ be a poset, then $Cov(\widetilde{I(P)})$ can be computed on-line in time complexity:*

$$\mathcal{O}((|I(P)| + |P|^3)w(P)).$$

When $|I(P)|$ is in $\Omega(|P|^3)$, the computation of covering graph of the ideal lattice of a poset from any of its linear extensions can be performed with the same time complexity than with the algorithm given by Bordat [3]. This last algorithm, which is up to our knowledge the most efficient one, is not accurate for the on-line case (it is based on a Depth-First-Search of the all poset).

6 Computation of the covering graph of $\widetilde{I(P)}$ Some particular cases

6.1 Under linear extension hypothesis

We suppose here that the successive elements x of the input poset $\widetilde{P}$ are given as a linear extension, i.e. x is maximal. It is easy to see that the set $UI(x)$ defined in Theorem 1 as the upper part of the lattice w.r.t. x is now reduced to the empty set.

The assumption of maximality of x has no algorithmic consequences on the search of $\downarrow_{\widetilde{P}} x[$ in $\widetilde{I(P')}$. The "granulation algorithm" can be simplified: it is easy to see that step 5 of the general algorithm performs no operation. The resulting "covering algorithm" thus remains with the same time complexity. The only interesting consequence is the reinforcement of the on-line property. In effect, it becomes no more necessary to know the set $\uparrow^{im}_{\widetilde{P}} x$ of the immediat successors of a vertex x. This means that the only information needed to build the lattice concerns the past. This allows the covering algorithm to make irrevocable decisions (e.g. drawing or emitting diagnosis). The linear extension assumption is thus of a great practical interest in the application of distributed trace checking we considered.

6.2 Under chain partition hypothesis

Let $\widetilde{P}$ be an order, let $\{P_i\}_{1\leq i\leq k}$ be a chain decomposition of $\widetilde{P}$ and let Δ be the map defined by:

$$\begin{aligned}\Delta \ : \ & I(P) \longrightarrow I\!N^k \\ & I \longmapsto (\mid I \cap P_i \mid)_{1\leq i\leq k}\end{aligned}$$

This map embeds the lattice into the $(I\!N^k, \leq_{I\!N^k})$ [6] lattice [7].

Proposition 2 *$\forall I, J \in I(P)$, the following properties are equivalent:*

(i) $I \leq_{\widetilde{I(P)}} J$

(ii) $\Delta(I) \leq_{I\!N^k} \Delta(J)$

(iii) *All maximal chains from I to J in $\widetilde{I(P)}$ have length $lg(I,J) = \sum_{i=1}^{k}(\Delta(J)[i] - \Delta(I)[i])$. $\forall i \in \{1,\ldots,k\}$, $\Delta(J)[i] - \Delta(I)[i] \geq 0$. And there is at least one maximal chain.*

Proof: Obviously $(iii) \Longrightarrow (ii)$.
$(ii) \Longrightarrow (i)$: For any ideal $A \in I(P)$ and for any $i \in \{1,\ldots,k\}$, if $\mid A \cap P_i \mid = \alpha_i \neq \emptyset$ then $A \cap P_i$ is a maximal subchain in $\widetilde{P_i}$ with α_i elements and containing the smallest element of $\widetilde{P_i}$. Thus, for any $i \in \{1,\ldots,k\}$ we have: $\Delta(I)[i] \leq_{I\!N} \Delta(J)[i] \Longrightarrow I \cap P_i \subseteq J \cap P_i$. Consequently, $I = (I \cap (\bigcup_{1\leq i\leq k} P_i)) \subseteq (J \cap (\bigcup_{1\leq i\leq k} P_i)) = J$.
$(i) \Longrightarrow (iii)$: Since $I \leq_{\widetilde{I(P)}} J$, there exists a maximal chain from I to J in $\widetilde{I(P)}$. Let $(x_0 = I, x_1, \ldots, x_{\alpha-1}, x_\alpha = J)$ be such a chain. From Lemma 1 we have $\mid J \mid = \mid I \mid + \alpha$. Since I and J are ideals, $\forall i \in \{1,\ldots,k\}$ we have $I \cap P_i \subseteq J \cap P_i$ and then, $\forall i \in \{1,\ldots,k\}$ we have $\Delta(I)[i] \leq_{I\!N} \Delta(J)[i]$ and thus $\Delta(J)[i] - \Delta(I)[i] \geq 0$. Then since $\{P_i\}_{1\leq i\leq k}$ is a partition of P, it is clear that $lg(I,J) = \alpha$. Moreover, since $\widetilde{I(P)}$ is a distributive lattice, it is modular and thus graded. So all maximal chains have the same length.
□

In order to take part of the the chain decomposition, we have to bring the following additional modifications on the inputs:

---

[6] $\leq_{I\!N^k}$ is the canonical order on $I\!N^k$: $\forall x, y \in I\!N^k$, $x \leq_{I\!N^k} \Longleftrightarrow \forall i \in \{1..k\}$, $x[i] \leq y[i]$

[7] This coding is quite popular for the message causality relation in distributed systems. When reduced to the joint irreducible elements of the causality order, this map is called the *vector clock* timestamping [7]

1. The transitive reduction of $\widetilde{P}$ is given with a chain decomposition $\{P_i\}_{1\leq i\leq k}$.
2. For any $y \in P$, $\Delta(y) = (|\downarrow_{\widetilde{P}} y] \cap P_i|)_{1\leq i\leq k}$ and $i(y)$ such that $y \in P_{i(y)}$.
3. The label of an edge YZ in the covering graph of an ideal lattice contain the usual label $Z \setminus Y$ the index of the chain whose $Z \setminus Y$ belongs.
4. Outgoing edges of any vertices in the covering graph of $\widetilde{I(P')}$ are stored ordered by increasing index of the chain their label belong to.

The two first conditions conditions imply that the chain decomposition never change (chain can only growing) and that the incomming vertex x is given with its coding vector $\Delta(y)$ and the index of the chain it belong to. In the context of parallel debugging, the Mattern's algorithm [7] used for observing the causality relation allows to fulfill these new assumptions.

The "supremum2 algorithm" under chain partition hypothesis

The "supremum2 algorithm" takes as input and return as output these of the "supremum algorithm" with the above modifications. And also need the vector $\Delta(D(x))$ with can be directly deduce from $\Delta(x)$ ($\Delta(D(x)) = \Delta(x) - (0, \ldots, 1_{i(x)}, \ldots, 0)$).

Body:
Initialization:

10: Choose $y \in D(x)$;
20: Let $I := \downarrow^{im}_{\widetilde{Q}} y$;
30: Let $V := \Delta(D(x)) - \Delta(y)$;

Algorithm:

10: While $V \neq O_{\mathbf{N}^k}$ do
20: Choose J in *outgoingedges*(I) with index label j such that $V[j] \neq O$;
30: $V := V - (0, \ldots, 1_j, \ldots, 0)$;
40: $I := J$
End of While

Theorem 5 *The "supremum algorithm" runs in time complexity:*

$$\mathcal{O}(|Q|\ (k + log(|Q|))).$$

Proof: Since the proof of the correctness and of the terminaison of this algorithm follow exactly (using Proposition 2) those of Theorem 2, we only detail the time complexity:

* The "Initialisation" part can be performed in $\mathcal{O}(klog(|Q|))$ since this is the time complexity of Step 30 and all steps can be performed in $\mathcal{O}(1)$ with the input assumptions.
* The "Algorithm" part can be performed in $\mathcal{O}(|Q|\ (k + log(|Q|)))$ since Step 20 can be performed in $\mathcal{O}(w(Q))$ and $w(Q) \leq k$, Step 30 can be performed in $\mathcal{O}(k + log(|Q|))$, Step 40 can be done in constant time and $\sum_{i \in \{1,\ldots,k\}} V[i] \leq |Q|$.

□

As consequence of this theorem we obtain the folowing corollary[8].

Corollary 2 *The computation of the ideal lattice* $Cov(\widetilde{I(P)})$ *of a poset* $\tilde{P}$ *provided a chain decomposition, can be achieved in time complexity:*

$$\mathcal{O}((|I(P)| + |P|^2\ (k + log(|P|)))w(P)).$$

7 Conclusion

Motivated by the practical problem of checking global predicates in distributed systems, we were interested to build the covering graph of the ideal lattice of finite posets. In that context of on-line testing, we went naturaly to on-line algorithms.

Up to our knowledge, we present the first algorithm dealing with this problem. We obtain a general algorithm of time complexity that is very close to the best off-line algorithm: actually $\mathcal{O}((|I(P)| + |P|^3)w(P))$ where P is the input poset, $w(P)$ its width, and $I(P)$ the ideal lattice. We have also enligthen two particular cases of practical interest: the case in which the order elements are given as a linear extension, and the case in which the poset is partitioned into chains.

A preliminary version of the algorithm has been already implemented. A complete integration into a software verification tool for analysing traces of distributed systems is in progress, as a part of a French national project.

---

[8]

1. If we assume that vector substractions can be done in $\mathcal{O}(1)$, then the time complexity of the "supremum2 algorithm" is in $\mathcal{O}(|Q|\ w(Q))$.
2. Since the size of chain partition is known in advance it can be reasonably assumed that in each part the Step 30 can be performed in $\mathcal{O}(log(|Q|))$, and then the time complexity of the "supremum2 algorithm" is in $\mathcal{O}(|Q|\ (w(Q) + log(|Q|))$.

References

[1] G. Birkhoff. Rings of sets. *Duke Math J-3*, 311–316, 1937.

[2] R. Bonnet and M. Pouzet. Linear extension of ordered sets. In I.Rival, editor, *Ordered Sets*, pages 125–170, D.Reidel Publishing Company, 1982.

[3] J.P. Bordat. Calcul des idéaux d'un ordonné fini. *Recherche opérationnelle/Operations Research*, 25(3):265–275, 1991.

[4] J.P. Bordat. Sur l'algorithmique combinatoire d'ordres finis. Thèse de doctorat d'état, USTL Montpellier, 1992.

[5] R. Cooper and K. Marzullo. Consistent detection of global predicates. In *Proc. ACM/ONR Workshop on Parallel and Distributed Debugging*, pages 163–173, Santa Cruz, California, May 1991.

[6] C. Diehl, C. Jard, and J.X. Rampon. *Reachability Analysis on Distributed Executions.* Publication interne 661, IRISA, Juin 1992.

[7] F. Mattern. Virtual time and global states of distributed systems. In Cosnard, Quinton, Raynal, and Robert, editors, *Proc. Int. Workshop on Parallel and Distributed Algorithms Bonas, France, Oct. 1988*, North Holland, 1989.

Annex 1 : complexity proof of theorem 2.

Let y be the first vertex chosen in $D(x)$, then the "supremum algorithm" has a running time complexity of:

$$\mathcal{O}(|E_G(\downarrow_{\widetilde{Q}} D(x))| + \max_{I \in I(Q)} |\uparrow^{im}_{I(Q)} I|| \downarrow_{\widetilde{Q}} D(x)| + \sum_{z_i \in D(x) \setminus \{y\}} |\downarrow_{\widetilde{Q}_i} z_i|^2)$$

which belongs to $\mathcal{O}(w(Q)\,|Q|^2)$.

Proof: In order to perform the time complexity analysis of the "supremum algorithm", let us denote by $G_i = (Q_i, E_{G_i})$ the current covering graph at the beginning of the i^{th} incoming in the While loop of *Step* 50. In the same way, we will denote by $\widetilde{Q_i}$ the poset having G_i as covering graph and by z_i the chosen vertex of $D(x)$ during this i^{th} incoming. By aim of simplicity, when we are dealing with the initial covering graph, we extend these notations to $G_0 = (Q_0, E_{G_0})$ and $\widetilde{Q_0}$.
The time complexity of the initialization part is in $\mathcal{O}(|\downarrow_{\widetilde{Q_0}} y| + |E_{G_0}(\downarrow_{\widetilde{Q_0}} y)|)$). Indeed following the input hypothesis, *Step* 40 can performed, using a Breadth–First–Search algorithm, in this time complexity and all the other steps of the initialization part can be performed in constant time.
The time complexity of the algorithm part is in:

$$\mathcal{O}(\sum_{z_i \in D(x) \setminus \{y\}} [|E_{G_i}(\downarrow_{\widetilde{Q}_i} z_i)| + |\downarrow_{\widetilde{Q}_i} z_i|^2 + (\max_{I \in I(Q)} |\uparrow^{im}_{I(Q)} I|)\, |\downarrow_{\widetilde{Q}_i} z_i|])$$

Indeed:

Step 60 can be performed in constant time by hypothesis on $D(x)$.

Step 80 and building list S can be performed in the same round, using a Breadth–First–Search algorithm, in $\mathcal{O}(|\downarrow_{\widetilde{Q}_i} z_i| + |E_{G_i}(\downarrow_{\widetilde{Q}_i} z_i)|)$. Then, *Step* 70 can be completed in $\mathcal{O}(|\downarrow_{\widetilde{Q}_i} z_i|\, log(|\downarrow_{\widetilde{Q}_i} z_i|)$ using a Quick–Sort algorithm.

Step 90 has for time complexity: $\mathcal{O}([\max_{I \in I(Q)} |\uparrow^{im}_{I(Q)} I| + |\downarrow_{\widetilde{Q}_i} z_i|]\, |\downarrow_{\widetilde{Q}_i} z_i|)$. Indeed, *Step* 110 and *Step* 120 can be performed in constant time and since the two lists are sorted, *Step* 100 can be performed in $\mathcal{O}(|\uparrow^{im}_{I(Q)} I_j| + |\downarrow_{\widetilde{Q}_i} z_i| - j)$ where I_j is the ideal at the beginning of the j^{th} incoming in the While loop of *Step* 90.

Thus, the announced time complexity for the "supremum algorithm" is obtained since for $i, j \in \{0, \ldots, |D(x)| - 1\}$ and $i \neq j$, we get on one hand that

$\downarrow_{\widetilde{Q_i}} z_i]$ and $\downarrow_{\widetilde{Q_j}} z_j]$ are disjoint sets of vertices in Q_0 and on the other hand that $E_{G_i}(\downarrow_{\widetilde{Q_i}} z_i])$ and $E_{G_j}(\downarrow_{\widetilde{Q_j}} z_j])$ are disjoint sets of edges in G_0. For the remaining it is sufficient to notice that $D(x)$ is an antichain in $\widetilde{Q_0}$ and that the number of outgoing edges of any vertices in both the covering graph of $I(\widetilde{Q_0})$ and the covering graph of $\widetilde{Q_0}$ is at most the width of $\widetilde{Q_0}$.
□

Annex 2 : complexity proof of theorem 3.

Let I be the input ideal, and let $I(x) = \{J \in I(Q),\ I \leq_{\widetilde{I(Q)}} J$ and $J \cap U(x) = \emptyset\}$. Then the "granulation algorithm" has a running time complexity of:

$$\mathcal{O}(|I(x)|\ w(Q))$$

Proof: In order to simplify the time complexity analysis of this algorithm, let us enounce some instructions which can be easily be performed in $\mathcal{O}(1)$.

* Adding an element at the end of a queue.
* Taking the first and the next element in a queue.
* Marking a vertex or testing if it is marked.
* Labeling an edge.
* Linking two current elements.
* Deleting a current edge.
* The creation of any vertex or edge, since their associated data structure are of bounded length.

It follows directly from above that the two macro instructions can be performed in $\mathcal{O}(1)$ and the only tricky points are the instructions of Step 40 and of Step 130. W.l.o.g. we can only look to the Step 40: since the two list are sorted it easy to see that the amortized complexity of the test instruction of Step 40 during a complete execution of the while loop of Step 30 is in $\mathcal{O}(|outgoingedge(I)| + |U(x)|)$ which belong to $\mathcal{O}(w(Q))$. Thus, the "Initialisation" part can be performed in $\mathcal{O}(w(Q))$ and since any vertex appears only one time in A and the only vertices added to A are elements of $I(x)$ then the "Algorithm" part can be performed in $\mathcal{O}(|I(x)|\ w(Q))$. Since the "Restoring" part can be clearly achieved in the same complexity than the "Algorithm" part (it is sufficient to add to A only maked vertices and doing this to perform the deletion and the unmark), we got the anounced time complexity.
□

Liste des publications internes Irisa 1993

PI 694 MECANISMES D'ABSTRACTION DANS UNE REPRESENTATION DE LA CONNAISSANCE CENTREE OBJET (R.C.O.)
Stéphane LE PEUTREC, Sophie ROBIN
Janvier 1993, 40 pages.

PI 695 DETECTION DE SEQUENCES ATOMIQUES DE PREDICATS LOCAUX DANS LES EXECUTIONS REPARTIES
Michel HURFIN, Noël PLOUZEAU, Michel RAYNAL
Janvier 1993, 16 pages

PI 696 SEMI-UNIFIED CACHES
Nathalie DRACH, André SEZNEC
Janvier 1993, 18 pages.

PI 697 CONCEPTS ET PROBLEMES DE L'ALGORITHMIQUE REPARTIE
Michel RAYNAL
Janvier 1993, 18 pages.

PI 698 TEMPORAL PLANNER = NONLINEAR PLANNER + TIME MAP MANAGER
Eric RUTTEN, Joachim HERTZBERG
Janvier 1993, 18 pages.

PI 699 EVALUATION COMPARATIVE D'ALGORITHMES DE NORMALISATION DES SCHEMAS RELATIONNELS
Annie FORET, Placide FRESNAIS
Février 1993, 22 pages.

PI 700 SEGMENTATION ET RECONNAISSANCE "EN LIGNE" DE MOTS MANUSCRITS
Sophie BERCU, Bernard DELYON
Février 1993, 32 pages.

PI 701 ON-THE-FLY VERIFICATION OF FINITE TRANSITION SYSTEMS
Claude JARD, Thierry JERON, Jean-Claude FERNANDEZ, Laurent MOUNIER
Février 1993, 26 pages.

PI 702 FORTRAN-S : A FORTRAN INTERFACE FOR SHARED VIRTUAL MEMORY ARCHITECTURES
Thierry PRIOL
Février 1993, 22 pages.

PI 703 COMPUTING ON-LINE THE COVERING GRAPH OF THE IDEAL LATTICE OF POSETS
Claire DIEHL, Claude JARD, Jean-Xavier RAMPON
Février 1993, 30 pages.

Unité de Recherche INRIA Rennes
IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399