Algorithmic Combinatorics based on Slicing Posets

Vijay K. Garg*
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712-1084, USA
garg@ece.utexas.edu

Abstract

A combinatorial problem usually requires enumerating, counting or ascertaining existence of structures that satisfy a given property B in a set of structures L. This paper describes a technique based on a generalization of Birkhoff's Theorem of representation of finite distributive lattices that can be used for solving such problems mechanically and efficiently. Specifically, we give an efficient (polynomial time) algorithm to enumerate, count or detect structures that satisfy B when the total set of structures is large but the set of structures satisfying B is small. We illustrate our techniques by analyzing problems in integer partitions, set families, and set of permutations.

1 Introduction

Consider the following combinatorial problems:

(Q1) Count the number of subsets of the set $[n]$ (the set $\{1 \dots n\}$) which have size m and do not contain any consecutive numbers.

(Q2) Enumerate all integer partitions less than or equal to $(\lambda_1, \lambda_2, \dots, \lambda_n)$ in which λ_1 equals λ_2.

(Q3) Give the number of permutations of $[n]$ in which i less than or equal to j implies that the number of inversions of i is less than or equal to the number of inversions of j.

Our goal in this paper is to show how such problems can be solved *mechanically* and *efficiently* for any fixed values of the parameters n and m.

It is important to note that someone trained in combinatorics may be able to solve all of these problems efficiently (and the reader is encouraged to solve these problems before reading further). Our emphasis is on techniques that can be applied mechanically. On the other hand, note that for the fixed values of n and m, all the sets above are finite and therefore all the problems can be solved mechanically. Our emphasis is on *efficiency*. To be more precise, let L be a large set of combinatorial structures (for example, all subsets of $\{1 \dots n\}$ of size m, all permutations of $[n]$ etc.) Each combinatorial problem requires enumerating, counting, or searching the subset of structures that satisfy a given property B. Call this set $L_B \subseteq L$. For example, in the problem (Q1), L is the set of all subsets of $[n]$ of size m and L_B is the set of all subsets of $[n]$ of size m that do not contain any consecutive numbers. For any fixed set of parameters m and n, the size of L is large but finite, enabling one to enumerate all possible structures and then to check each one of them for the property B. This approach results in an algorithm that requires time proportional to the set L which is exponential in n (or m). This paper proposes a technique that provides answers to some combinatorial problems in polynomial time and for others, such as those involving enumeration, in time proportional to the size of L_B (and not L).

*supported in part by the NSF Grants CNS-0509024, ECS-9907213, CCR-9988225, Texas Education Board Grant ARP-320, an Engineering Foundation Fellowship, and an IBM grant.

To explain our technique, we use the term *small* to mean polynomial in n and m, and *large* to mean exponential in n or m. Thus, the set L is large. We first build a *small* structure P such that all elements of L can be generated by P. Second, we compute a *slice* of P with respect to B, denoted by P_B, such that P_B generates all elements of L_B. P_B is a small structure and can be efficiently analyzed to answer questions about L_B or enumerate all elements of L_B.

Our approach is based on a slight generalization of Birkhoff's Theorem which establishes the duality between finite posets and finite distributive lattices. One can go from a finite poset to its dual finite distributive lattice by constructing the set of its order ideals and from the finite distributive lattice to the corresponding poset by restricting it to join-irreducible elements. The notion of order ideals of a poset can be easily extended to that for a directed graph. In our approach, we use a small directed graph P such that the set of all its ideals correspond to the large structure L. Now consider any predicate B defined on L, or equivalently, the subset L_B of L. B is called regular if L_B is a sublattice of L. From Birkhoff's theorem we know that there exists a poset that generates L_B. We show that every sublattice of L can be generated by a poset that can be obtained by adding edges to the poset P. Note that when edges are added to the graph of a poset, cycles may be formed. In this case we simply consider the poset of strongly connected components in the graph. We denote the small structure obtained after adding edges to P as P_B. Now P_B can be used to enumerate elements in L_B, or to analyze the number of elements in L_B. One of the main contributions of this paper is an algorithm to determine which edges to add in the graph P.

When B is not regular, we can still use this idea by considering the structure P_B which properly includes all ideals in L_B and as few other ideals as possible. In particular, a property that can be expressed as a conjunction, disjunction or negation of regular predicates [MG01] can also be analyzed in this manner. Furthermore, it can be shown that the technique is applicable to all predicates which can be efficiently detected. This class of predicates includes relational predicates and observer-independent predicates [Gar02, CBDGF95].

We apply these ideas to many traditional problems in combinatorics. It is easily shown that most combinatorial structures such as the set of permutations, set of all subsets, set of subsets of size k, all integer partitions less than a given partition, all tuples in product spaces etc. can be generated as the set of order ideals of small posets. We show that many interesting subsets of these structures can be *efficiently* analyzed by generating appropriate slices.

To enumerate elements of L_B, one can use an algorithm for enumeration of ideals of a poset. Many algorithms have been proposed; for example by Steiner [Ste86] and Squire [Squ95]. In distributed computing, the algorithms to explore the global state lattice address the identical problem (see [CM91, VD01, JMN95, Gar03]). Determining the count of the elements in L_B given P_B is #P-complete for general posets [PB83] but can be done efficiently for 2-dimensional posets [Ste84].

2 Notation and Definitions

We use the standard notation and terminology of lattice theory [DP90]. Let $P = (X, \leq)$ be a poset and let $G \subseteq X$. G is called an *order ideal* in P if for any e, f such that $e \leq f$, $f \in G$ implies that $e \in G$. We simply use *ideal* for order ideal in this paper. Let L denote the family of all ideals of P. Define a partial order on L by $G \leq H$ in L if and only if $G \subseteq H$. It is well known that L is a distributive lattice. Let $J(L)$ denote the set of all join-irreducible elements in L. Birkhoff's theorem states that any finite distributive lattice L is isomorphic to the set of ideals of the poset $J(L)$. This is illustrated in Figure 1. Figure 1(a) shows a poset whose lattice of ideals is shown in Figure 1(b). The join–irreducible elements of this lattice are

$$\{a\}, \{b\}, \{a, b, c\}, \{b, d\}$$

The poset formed by these elements is isomorphic to the poset in Figure 1(a).

In this paper, we are interested in producing structures that generate subsets of the finite distributive lattice. It is more convenient to use directed graphs instead of posets for this purpose because we can get sublattices by simply adding edges to the original directed graph (this will be shown later).

The notion of ideals can be extended to graphs in a straightforward manner. A subset of vertices, H, of a directed graph is an *ideal* if it satisfies the following condition: if H contains a vertex v and (u, v) is an edge in the graph, then H also contains u. We will continue to use P for the directed graph. Observe that an ideal of P either contains all vertices in a strongly connected component or none of them. Let $\mathcal{I}(P)$ denote the set of ideals of a directed graph P. Observe that the empty set and the set of all vertices trivially belong to $\mathcal{I}(P)$. We call them *trivial* ideals. The following theorem is a slight generalization of the result in lattice theory that the set of ideals of a partially ordered set forms a distributive lattice [DP90].

Theorem 1 *[MG01] Given a directed graph P, $\langle \mathcal{I}(P); \subseteq \rangle$ forms a distributive lattice.*

Observe that when the directed graph has no edges (i.e., the poset is an antichain), the set of ideals correspond to the boolean lattice on the set of vertices. At the other extreme, when the graph is strongly connected, there are only trivial ideals. Since trivial ideals are always part of $\mathcal{I}(P)$, it is more convenient to deal only with nontrivial ideals of a graph. It is easy to convert a graph P to P' such that there is one-to-one correspondence between all ideals of P and all nontrivial ideals of P'. We construct P' from P by adding two additional vertices $\bot$ and $\top$ such that $\bot$ is the smallest vertex and $\top$ is the largest vertex (i.e., there is a path from $\bot$ to every vertex and a path from every vertex to $\top$). It is easy to see that any nontrivial ideal will contain $\bot$ and not contain $\top$. As a result, every ideal of P is a nontrivial ideal of the graph P' and vice versa. We will deal with only nontrivial ideals from now on and an ideal would mean nontrivial ideal unless specified otherwise. The directed graph representation of Figure 1(a) is shown in Figure 1(c).

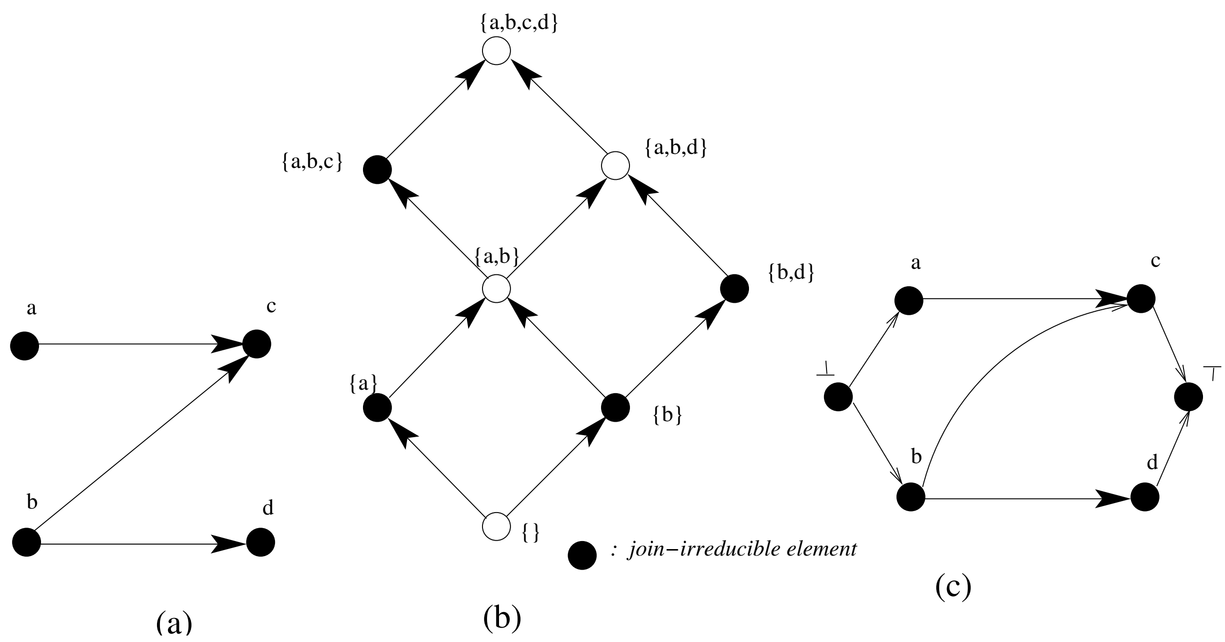


Figure 1: (a) a partial order (b) the lattice of ideals. (c) the directed graph

Figure 2 shows a directed graph and its nontrivial ideals. The directed graph in Figure 2(a) is derived from Figure 1(a) by adding an edge from c to b and adding two additional vertices $\bot$ and $\top$.

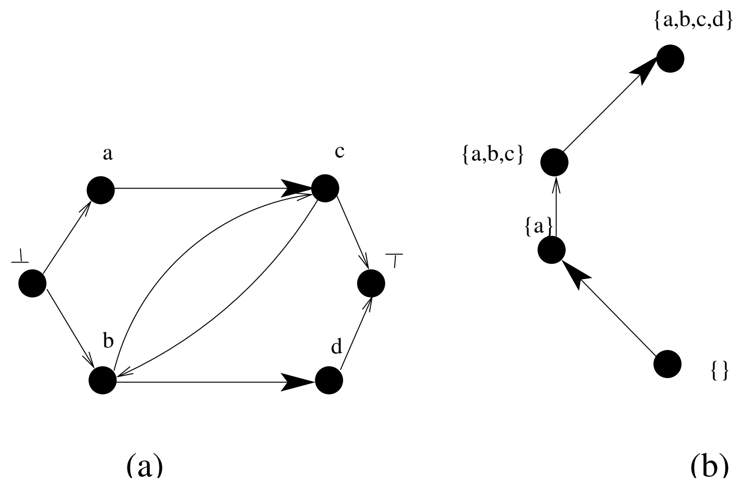


Figure 2: (a) A directed graph (b) the lattice of its nontrivial ideals.

The resulting set of nontrivial ideals is a sublattice of Figure 1(b). In the figure, we have not shown $\perp$ in the ideals because it is implicitly included in every nontrivial ideal.

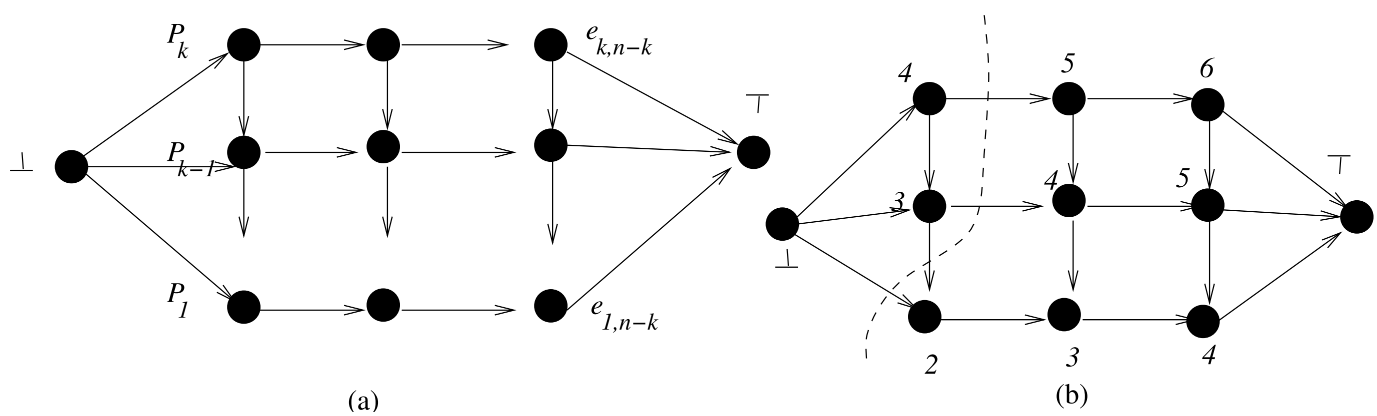


Figure 3: (a) The poset or directed graph $\mathcal{K}$ for generating subsets of X of size k. (b) The ideal denoting the subset $\{1, 3, 4\}$.

We use the graph in Figure 3 as a running example in the paper and denote it by $\mathcal{K}$. Let $X = \{1..n\}$ and k be a number between 1 and n (inclusive). Ignoring the vertices $\top$ and $\perp$, $\mathcal{K}$ has k chains and each chain has $n - k$ elements. We call these chains $P_1, \ldots, P_k$. In the graph, there are two types of edges. For every chain there is an edge from j^{th} element to $(j + 1)^{th}$ element, and an edge from j^{th} element in P_{i+1} to j^{th} element in P_i. Therefore, if an ideal of P contains j elements of P_i, then it also contains at least j elements of P_{i+1}. We show that the ideals of $\mathcal{K}$ are in 1-1 correspondence with all the subsets of X of size k. The correspondence between subsets of X and ideals can be understood as follows. If chain P_i has t elements in the ideal, then the element $t + i$ is in the set Y. Thus chain P_1 chooses a number from $1 \ldots n - k + 1$ (because there are $n - k$ elements); chain P_2 chooses the next larger number and so on. Figure 3(b) gives an example of the graph for subsets of size 3 of the set [6]. The ideal, shown corresponds to the subset $\{1, 3, 4\}$. It can also be easily verified that the there are $\binom{n}{k}$ ideals of this poset.

2.1 Counting Ideals

For counting the number of elements in L and its sublattices, we use $N(P)$ to denote the number of ideals of the poset P. Since our interest is in efficient calculation of $N(P)$, we will restrict the *dimension* of the partial order generating the lattice. For any poset (X, P), the dimension of (X, P), denoted by $dim(X, P)$, is the least positive integer t for which there exists a family $\{C_1, C_2, \ldots, C_t\}$ of linear extensions of P (total orders compatible with P) such that $P = \cap_{i=1}^{t} C_i$. Determining whether a poset P with n points is 2-dimensional and isomorphism testing for 2-dimensional orders can be done in $O(n^2)$ time [Spi85]. All the posets used in this paper are 2-dimensional. The reader is referred to [Tro92] for the dimension theory of posets. The following lemma shows that the number of ideals of a poset can be calculated efficiently for series-parallel posets (a special case of 2-dimensional posets) [FLST86]. For generalization to counting ideals of two dimensional posets see [Ste84].

Lemma 1 (Counting Lemma)

(1) If Q is an extension of P then $N(Q) \leq N(P)$.

(2) (Parallel) Let $P+Q$ be the disjoint union (or direct sum) of posets P and Q (see [DP90]). Then, $N(P+Q) = N(P)N(Q)$.

(3) (Series) Let $P \oplus Q$ be the ordinal sum of posets P and Q[DP90]. Then, $N(P \oplus Q) = N(P) + N(Q) - 1$.

(4) Assume that P can be decomposed into the least number of chains $C_1, C_2, \ldots C_n$. Then

$$N(P) \leq \prod_{i=1}^{n} (|C_i| + 1).$$

When each chain is at most m in length, we get that $N(P) \leq (m+1)^n$.

For some examples, instead of enumerating all ideals of a poset we may be interested in enumerating or counting ideals in a certain *level set*. To define *level sets*, first define a poset to be *ranked* if for each element $e \in P$, one can assign a non-negative integer, $rank(e)$, such that if f covers e, then $rank(f) = rank(e) + 1$. The set of all elements in P with rank i are called its *level set* with rank i. Every distributive lattice is a ranked poset [Sta86].

3 Predicates on Ideals

Our technique is crucially based on the notion of predicates on the set of ideals. A predicate is simply a boolean function from the set of all ideals to $\{0, 1\}$. Equivalently, a predicate specifies a subset of the ideals in which the boolean function evaluates to 1. In the poset $\mathcal{K}$, "does not contain consecutive numbers" is a predicate which is either true or false for any ideal. For example, it is true for $\{1, 3, 5\}$ but false for $\{1, 3, 4\}$. In the boolean lattice of all subsets of $[n]$, $\mathcal{B}_n$, "has size k" is a predicate which is true if the subset has size k and false otherwise.

We now define various classes of predicates. The class of meet-closed predicates are useful because they allow us to compute the least ideal that satisfies a given predicate.

Definition 1 (meet-closed Predicates) *A predicate B is meet-closed for a poset or a graph P if*

$$\forall G, H \in \mathcal{I}(P) : B(G) \wedge B(H) \Rightarrow B(G \sqcap H)$$

The predicate "does not contain consecutive numbers" in the poset $\mathcal{K}$ is meet-closed whereas the predicate "has size k" in the poset $\mathcal{B}_n$ is not.

It follows from the definition that if there exists any ideal that satisfies a meet-closed predicate B, then there exists the least one. Note that the predicate *false* which corresponds to the empty subset and the predicate *true* which corresponds to the entire set of ideals are meet-closed predicates.

As another interesting example, define a n-size subset of $[2n]$ to satisfy the predicate Catalan if by putting left parentheses on the specified subset and right parentheses in remaining positions, one gets a well-balanced string of parentheses. For example, the subset $\{1, 3, 4\}$ corresponds to the string "()(())" where left parenthesis are at positions 1, 3 and 4. This subset satisfies Catalan predicate whereas the subset $\{1, 4, 5\}$ does not. We will use $G[i]$ to denote the ith largest number in the set. To formalize a well-balanced expression, note that if $G[i]$ equals k for any $i \in [n]$, then there are $i-1$ left parentheses and $k-i$ right parentheses to the left of position k. Therefore G satisfies Catalan predicate iff for all i

$$G[i] - i \leq i - 1$$

Or, equivalently

$$G[i] \leq 2 * i - 1$$

It is now clear that Catalan is a meet-closed (and join-closed) predicate. If $G[i] \leq 2 * i - 1$ and $H[i] \leq 2 * i - 1$, then $min(G[i], H[i]) \leq 2 * i - 1$.

We now give another characterization of such predicates that will be useful for computing the least ideal that satisfies the predicate. To that end, we first define the notion of a crucial event for an ideal.

Definition 2 (Crucial Element) *Let $(E, \leq)$ be a poset. For an ideal $G \subsetneq E$ and a predicate B, we define $e \in E - G$ to be crucial for G as:*

$$crucial(G, e) \stackrel{\text{def}}{=} \forall H \supseteq G : (e \in H) \vee \neg B(H).$$

Definition 3 (Linear Predicates) *A predicate B is linear iff for all ideals $G \subsetneq E$,*

$$\neg B(G) \Rightarrow \exists e \in E - G : crucial(G, e).$$

Intuitively, this means that any ideal H, that is at least G, cannot satisfy the predicate unless it contains e. Now, we have

Theorem 2 *A predicate B is linear iff it is meet-closed.*

Proof: First assume that B is not closed under meet. We show that B is not linear. Since B is not closed under meets, there exist two ideals H and K such that $B(H)$ and $B(K)$ but not $B(H \sqcap K)$. Define G to be $H \sqcap K$. G is a strict subset of $H \subseteq E$ because $B(H)$ but not $B(G)$. Therefore, G cannot be equal to E. We show that B is not linear by showing that there does not exist any crucial element for G. A crucial element e, if it exists, cannot be in $H - G$ because K does not contain e and still $B(K)$ holds. Similarly, it cannot be in $K - G$ because H does not contain e and still $B(H)$ holds. It also cannot be in $E - (H \cup K)$ because of the same reason. We conclude that there does not exist any crucial event for G.

Now assume that B is not linear. This implies that there exists $G \subsetneq E$ such that $\neg B(G)$ and none of the elements in $E - G$ is crucial. We first claim that $E - G$ cannot be a singleton. Assume if possible $E - G$ contains only one element e. Then, any ideal H that contains G and does not contain e must be equal to G itself. This implies that $\neg B(H)$ because we assumed $\neg B(G)$. Therefore, e is crucial contradicting our assumption that none of the elements in $E - G$ is crucial. Let $W = E - G$. For each $e \in W$, we define H_e as the ideal that contains G, does not contain e and still satisfies B. It is easy to see that G is the meet of all H_e. Therefore, B is not meet-closed because all H_e satisfy B, but not their meets.

■

Example 1 Consider the poset of n elements $\{1..n\}$ in which all elements are incomparable. Figure 6(a) shows the graph for this poset. It is clear that the ideals of this graph correspond to subsets of $[n]$ and thus generate the boolean lattice. Now consider the predicate B defined to be true on G as "If G contains any odd $i < n$, then it also contains $i+1$." It is easy to verify that B is meet-closed. Given any G for which B does not hold, the crucial elements consist of

$$\{i | i \ even, 2 \le i \le n, i-1 \in G, i \notin G\}$$

Example 2 Consider the poset $\mathcal{K}$. Let the predicate B be "G does not contain any consecutive numbers." Consider any G for which B does not hold. This means that it has some consecutive numbers. Let i be the smallest number such that $i-1$ is also in G. If i is on chain P_j, then the next element in P_j is crucial. For example, the ideal $\{1,3,4\}$ does not satisfy B. The smallest element who predecessor is also in G is 4 which is on the chain P_3. Therefore, the second element in P_3 is crucial. In other words any ideal that satisfies B and is bigger than $\{1,3,4\}$ is at least $\{1,3,5\}$. If i is the last element in P_j, then any element in $E-G$ can serve as a crucial element because in this case for any H that contains G, $B(H)$ is false.

Our interest is in detecting whether there exists an ideal that satisfies given predicate B. We assume that given a ideal, G, it is efficient to determine whether B is true for G or not. On account of linearity of B, if B is evaluated to be false in some ideal G, then we know that there exists a crucial event in $E-G$. We now make an additional assumption called the efficient advancement property.

(Efficient Advancement Property) There exists an efficient (polynomial time) function to determine the crucial event.

We now have

Theorem 3 *If B is a linear predicate with the efficient advancement property, then there exists an efficient algorithm to determine the least ideal that satisfies B (if any).*

Proof: An efficient algorithm to find the *least* cut in which B is true is given in Figure 4. We search for the least ideal starting from the ideal $\{\perp\}$. If the predicate is false in the ideal, then we find the crucial element using the efficient advancement property and then repeat the procedure.

■

Assuming that $crucial(e, G, B)$ can be evaluated efficiently for a given graph, we can determine the least ideal that satisfies B efficiently even though the number of ideals may be exponentially larger than the size of the graph. As an example, to find the least ideal in $\mathcal{K}$ that satisfies "does not contain consecutive numbers," we start with the ideal $\{1,2,3\}$. Since 1 and 2 are consecutive, we advance along P_2 to the ideal $\{1,3,4\}$ which still does not satisfy the predicate. We now advance along P_3 to the ideal $\{1,3,5\}$ which is the smallest ideal that satisfies the given predicate.

So far we have focused on meet-closed predicates. All the definitions and ideas carry over to join-closed predicates. If the predicate B is join-closed, then one can search for the largest ideal that satisfies B in a fashion analogous to finding the least ideal when it is meet-closed.

Predicates that are both meet-closed and join-closed are called regular predicates.

Definition 4 (Regular Predicates [GM01]) *A predicate is regular if the set of ideals that satisfy the predicate forms a sublattice of the lattice of ideals.*

```
boolean function detect(B:boolean_predicate, P:graph)
var
      G: ideal initially G = {⊥};

while (¬B(G) ∧ (G ≠ P)) do
      Let e be such that crucial(e, G, B) in P;
      G := G ∪ {e}.
endwhile;
if B(G) return true;
else return false;
```

Figure 4: An efficient algorithm to detect a linear predicate

Equivalently, a predicate B is *regular* with respect to P if it is closed under $\sqcup$ and $\sqcap$, i.e.,

$$\forall G, H \in \mathcal{I}(P) : B(G) \wedge B(H) \Rightarrow B(G \sqcup H) \wedge B(G \sqcap H)$$

The set of ideals that satisfy a regular predicate forms a sublattice of the lattice of all ideals. Since a sublattice of a distributive lattice is also distributive, the set of ideals that satisfy a regular predicate forms a distributive lattice. From Birkhoff's theorem we know that a distributive lattice can be equivalently represented using the poset of its join-irreducible elements. The poset of join-irreducible elements provides a compact way of enumerating all ideals that satisfy the predicate and will be useful in efficient solving of combinatorial problems.

There are two important problems with this approach. First, what if the predicate is not regular? Is it still possible to represent the set of ideals satisfying B compactly? Second, we need to calculate the structure that captures the set of ideals satisfying B efficiently. These problems are addressed in the next section.

4 Slices

The notion of slice will be used to represent the subset of ideals that satisfy B in a concise manner. The slice of a directed graph P with respect to a predicate B (denoted by by $slice(P, B)$) is a graph derived from P such that all the ideals in $\mathcal{I}(P)$ that satisfy B are included in $\mathcal{I}(slice(P, B))$. Note that the slice may include some additional ideals which do not satisfy the predicate. Formally,

Definition 5 (Slice [MG01]) *A slice of a graph P with respect to a predicate B is the directed graph obtained from P by adding edges such that:*
(1) it contains all the ideals of P that satisfy B and
(2) of all the graphs that satisfy (1), it has the least number of ideals.

We first show that given a distributive lattice L generated by the graph (poset) P, every sublattice of L can be generated by a graph obtained by adding edges to P.

Theorem 4 *Let L be a finite distributive lattice generated by the graph P. Let L' be any sublattice of L. Then, there exists a graph P' that can be obtained by adding edges to P that generates L'.*

Proof: Our proof is constructive. We show an algorithm to compute P'. For every vertex $e \in P$, we first define $J(e, L')$ as the least ideal in L that includes e and is part of L'. If there is no ideal in L' that includes e, then $J(e, L')$ is defined to be trivial ideal of L that includes the $\top$ element.

Since L' is a sublattice, it is clear that if the set of ideals that include e are part of L', then their intersection (meet) also includes e and is part of L'. Thus, $J(e, L')$ is well defined. Note that $J(\bot, L')$ corresponds to the least element of L' and $J(\top, L')$ corresponds to the trivial ideal that includes $\top$.

Now we add the following edges to the graph P. For every pair of vertices e, f such that $J(e, L') \leq J(f, L')$, we add an edge from e to f. We now claim that the resulting graph P' generates L'.

Pick any nontrivial ideal G of P', i.e., an ideal that includes $\bot$ and does not include $\top$. We show that $G = \cup_{e \in G} J(e, L')$. This will be sufficient to show that $G \in L'$ because G is a union of ideals in L' and L' is a lattice. Since $e \in J(e, L')$ it is clear that $G \subseteq \cup_{e \in G} J(e, L')$. We show that $G \supseteq \cup_{e \in G} J(e, L')$. Let $f \in J(e, L')$ for some e. This implies that $J(f, L') \subseteq J(e, L')$ because $J(f, L')$ is the least ideal containing f in L'. By our algorithm, there is an edge from f to e in P', and since G is an ideal of P' that includes e, it also includes f. Thus $G \supseteq \cup_{e \in G} J(e, L')$.

Conversely, pick any element G of L'. We show that G is a nontrivial ideal of P'. Since $L' \subseteq L$ and L corresponds to nontrivial ideals of P, it is clear that G is a nontrivial ideal of P. Our obligation is to show that it is a nontrivial ideal of P' as well. Assume, if possible, G is not a nontrivial ideal of P'. This implies that there exists vertices e, f in P' such that $f \in G$, $e \notin G$ and (e, f) is an edge in P'. The presence of this edge in P', but not in P implies that $J(e, L') \subseteq J(f, L')$. Since $f \in G$ and $G \in L'$, from definition of $J(f, L')$, we get that $J(f, L') \subseteq G$. But this implies $J(e, L') \subseteq G$, i.e., $e \in G$, a contradiction.

■

Now, the following result is easy.

Theorem 5 *For any P and B, slice(P, B) exists and is unique.*

Proof: First note that intersection of sublattices is also a sublattice. Now given any predicate B consider all the sublattices that contain all the ideals that satisfy B. The intersection of all these sublattices gives us the smallest sublattice that contains all the ideals. From Theorem 4, we get that there exists a graph that generates this sublattice.

■

The procedure outlined in the above proof is not efficient because it requires us to take intersection of all bigger sublattices. We now show how to efficiently compute slices for the predicates for which there exists an efficient detection algorithm. The slicing algorithm is shown in Figure 5. It takes as input a graph P and a boolean predicate B. The algorithm constructs the slice by adding edges to the graph P. For this purpose, it initializes in line (3) a graph R as P. In rest of the function, edges are added to R which is finally returned.

The addition of edges is done as follows. For every pair of vertices, e and f, in the graph P, the algorithm constructs Q from P by adding edges from f to $\bot$ and $\top$ to e. Due to these edges in Q, all nontrivial ideals of Q contain f and do not contain e. We now invoke the detection algorithm on Q. If the detection algorithm returns false, then we know that there is no nontrivial ideal of P that contains f but does not contain e. Therefore, all ideals that satisfy B have the property that if they include f, they also include e. Hence, we add an edge from e to f in the graph R. We continue this procedure for all pairs of vertices. Theorem 6 shows the correctness of this procedure.

Theorem 6 *Let P be a directed graph. Let R be the directed graph output by the algorithm in Figure 5 for a predicate B. Then R is the slice of P with respect to B.*

Proof: Let $\mathcal{I}(P, B)$ denote the set of ideals of P that satisfy B. We first show that $\mathcal{I}(R) \supseteq \mathcal{I}(P, B)$. Adding an edge (e, f) in R eliminates only those ideals of P that contain f but do not contain e. But

```
(1)  graph function computeSlice(B:boolean_predicate, P: graph)
(2)  var
(3)      R: graph initialized to P;
(4)  begin
(5)      for every pair of nodes e, f in P do
(6)          Q := P with the additional edges (f, ⊥) and (⊤, e);
(7)          if detect(B, Q) is false
(8)              add edge (e, f) to R;
(9)      endfor
(10)     return R;
(11) end;
```

Figure 5: An efficient algorithm to compute the slice for a predicate B

all those ideals do not satisfy B because the edge (e, f) is added only when $detect(B, Q)$ is false. Thus, all the ideals of P that satisfy B are also the ideals of R.

Next we show that the $\mathcal{I}(R)$ is the smallest sublattice of $\mathcal{I}(P)$ that includes $\mathcal{I}(P, B)$. Let M be a graph such that $\mathcal{I}(M) \supseteq \mathcal{I}(P, B)$. Assume if possible $\mathcal{I}(M)$ is strictly smaller than $\mathcal{I}(R)$. This implies that there exists two vertices e and f such that there is an edge from e to f in M but not in R. Since R is output by the algorithm, $detect(B, Q)$ is true in line (7); otherwise, an edge would have been added from e to f. But, this means that there exists an ideal in P which includes f, does not include e, and satisfies B. This ideal cannot be in $\mathcal{I}(M)$ due to the edge from e to f contradicting our assumption that $\mathcal{I}(P, B) \subseteq \mathcal{I}(M)$.

■

Theorem 6 allows us to compute slice for any predicate that can be detected efficiently. For example, relational predicates can be detected efficiently using max-flow techniques ([Gar96] Chapter 6). When the predicate is known to be linear, then we can use more efficient algorithms to compute the slice as shown in [MG01].

5 Application to Combinatorics

In this section we give several examples of combinatorial structures that can be viewed as the set of ideals generated by a slice and show that the slice can be generated mechanically.

5.1 Boolean Algebra and Set Families

Let X be a ground set on n elements. Assume that we are interested in the sets of subsets of X. By using $\subseteq$ as the order relation, we can view it as a distributive lattice L. This lattice has $n + 1$ levels and each level set of rank k in the boolean lattice corresponds to $\binom{n}{k}$ sets of size k. L is generated by the directed graph in Figure 6(a). It is easy to verify that there is a bijection between every nontrivial ideal of the graph and a subset of X.

Now consider all subsets of X such that if they include e_i then they also include e_j. To obtain the slice with respect to this predicate, we just need to add an edge from e_j to e_i. Figure 6(b) shows the slice with respect to the predicate "if e_3 is included then so is e_2." To ensure the condition that e_i is always included, we simply add an edge from e_i to $\perp$ and to ensure that e_i is never included in any

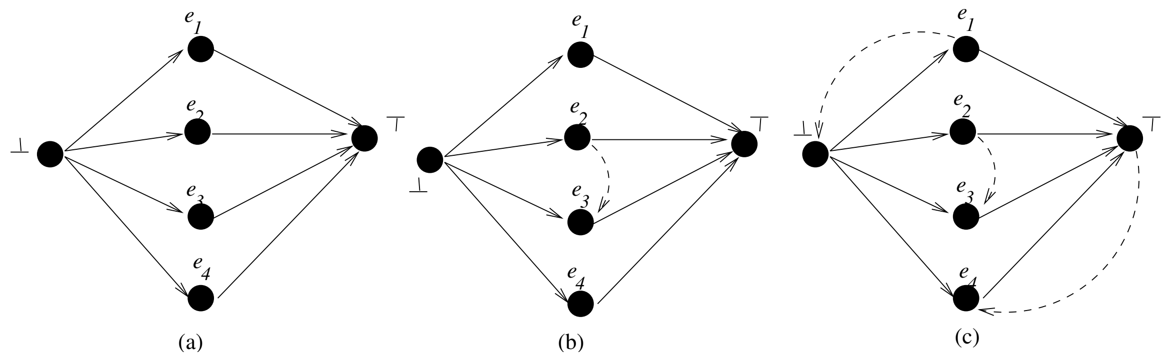


Figure 6: Graphs and slices for generating subsets of X when $|X| = 4$

subset, we add an edge from $\top$ to e_i. Figure 6(c) shows the slice which gives all subsets that always contain e_1, never contain e_4 and contain e_2 whenever they contain e_3.

As an application, we now solve some combinatorial problems. Let n be even. We are required to calculate the total number of subsets of $[n]$ which satisfy the property that if they contain any odd integer i, then they also contain $i+1$ (or equivalently, compute the number of ways to select groups from $n/2$ couples such that a wife is always accompanied by her husband in the group although a husband may not be accompanied by his wife). Although this problem can be solved directly by a combinatorial argument, we will show how our method can be applied. We first construct the poset which generates all the subsets of $[n]$. It is Figure 6(a) in the case of 4. We now define the subset of interest by a predicate B. For any subset G of $[n]$, we let $B(G)$ to be true if G contains $i+1$ whenever it contains any odd integer i. From our discussion of regular predicates, it is clear that B is regular. To compute the slice, it is sufficient to have a predicate detection algorithm. Detecting the least ideal that satisfies B can be done efficiently because it satisfies efficient advancement property. If an ideal does not satisfy B, then there is some unaccompanied wife and therefore the element corresponding to the husband is crucial. By applying predicate detection algorithm repeatedly, we can determine all the edges that need to be added to get the slice. In this example, it is sufficient to add an edge from e_{i+1} to e_i for odd i. The slice consists of $n/2$ chains each with exactly 2 elements (ignoring $\perp$ and $\top$). From the counting lemma (Lemma 1), it follows that the total number of ideals is $(2+1)^{n/2} = 3^{n/2}$. The reader should note that for any fixed value of n, the problem can be solved by a computer automatically and efficiently (because the slice results in a series-parallel poset).

5.2 Set families of Size k

It is important to note that regularity of B is dependent upon the lattice structure of L. For example, in many applications of set families, we are interested in sets of a fixed size k. The predicate B that the ideal is of size k is not regular. However, by considering alternative posets, this set family can still be analyzed.

Now let us apply our theory to the first combinatorial problem (Q1) mentioned in the introduction. Assume that we are interested in counting all subsets of n of size k which do not have any consecutive numbers. In this example, G satisfies B if whenever P_i has t elements in G, P_{i+1} has at least $t+1$ elements in G. This condition is linear (in fact regular) and we can use Figure 5 to compute the slice. Figure 3 shows the original graph with k processes, each with $n-k$ events. Let us call this graph $\mathcal{K}$ $(k, n-k)$. Figure 7 shows the slice which includes precisely those subsets that do not contain consecutive numbers. By collapsing all strongly connected components and by removing the transitively implied edges we get a graph which is isomorphic to $\mathcal{K}$ $(k, n-k-(k-1))$ (i.e., the original graph when there are k processes and each process executes only $n-k-(k-1)$ elements). Therefore, the total number of order ideals is $\binom{n-k+1}{k}$. Again one can come up with a combinatorial argument to solve the problem (for example, see Theorem 13.1 and Example 13.1 in [vLW92]), but the slicing approach is completely mechanical.

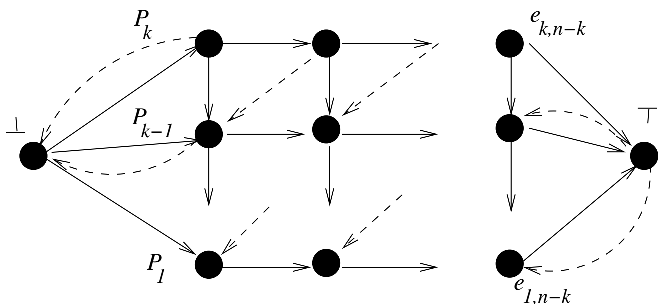


Figure 7: Slice for the predicate "does not contain consecutive numbers"

As another example, Catalan predicate is regular and we can compute the slice automatically for all n size subsets of $2n$ that are Catalan. The slice for n equal to 5 after simplification is shown in Figure 8.

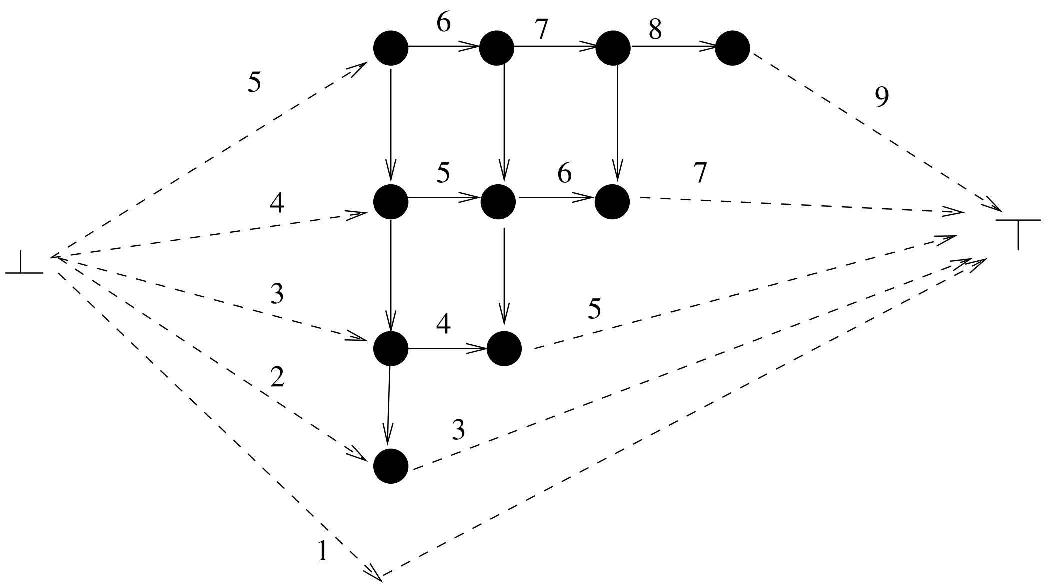


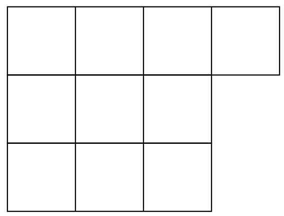
Figure 8: Slice for the predicate "Catalan numbers"

All the above constructions can be generalized to multidimensional grids to obtain results on multinomials instead of binomials.

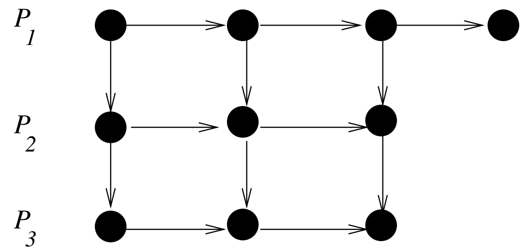
5.3 Integer Partitions and Young's lattice

A k-tuple of positive integers $\lambda = (\lambda_1, \ldots, \lambda_k)$ is an integer partition of N if $\lambda_1 + \ldots + \lambda_k = N$ and for all i, $\lambda_i \geq \lambda_{i+1}$. The Ferrers diagram of the partition $(4, 3, 3)$ of 10 is shown in Figure 9(a). A partition λ is contained in another partition δ if the number of parts of λ is at most that of δ and λ_i

is less than or equal to δ_i for any i between 1 and the number of parts of λ. For example, $(3,3,1)$ is less than $(4,3,3)$. Fix any partition λ. The set of all partitions that are less than or equal to λ form the *Young's lattice* denoted by Y_λ.



(a)



(b)

Figure 9: (a) A Ferrer diagram (b) A graph for generating Young's lattice

We now apply our approach to Y_λ. Let the number of parts and the largest part in the partition λ be m and n respectively. Then we have a graph with n chains, and at most m elements per chain as shown in Figure 9(b) for m equal to 4 and n equal to 3. It is clear that for any ideal, the number of elements it includes from P_i is at least as many as it included from P_{i+1}. Clearly, the set of ideals of the graph as in Figure 9(b) is isomorphic to Young's lattice for the corresponding partition.

It follows that Young's lattice is distributive. One can see that the lattice of subsets of size k from the set of size n is a special case of Young's lattice when all λ_i's are equal. Therefore, the number of integer partitions whose Ferrers diagrams fit in a box of size k by $n-k$ is equal to $\binom{n}{k}$ (providing an alternate proof of Theorem 3.2 in [SW86]). Let $q(N,k,m)$ denote the number of partitions of N whose Ferrer's diagram fit in a box of size k by m. By summing up the sizes of all level sets, it also follows that

$$\binom{n}{k} = \sum_{l=0}^{k(n-k)} q(l,k,n-k)$$

Since the poset that generates corresponding Young's lattice is symmetric with respect to k and m, we get that $q(N,k,m)$ equals $q(N,m,k)$; and since the poset is dual of itself (i.e. we get back the same poset when all arcs are reversed) we also get that $q(N,k,m)$ equals $q(mk-N,k,m)$. All these results are well known and generally derived using Gaussian polynomials (see [vLW92]).

We now focus on subsets of partitions. Assume that we are interested in all those partitions such that their second component is some fixed value say b. It is easy to verify that partitions $\delta \in Y_\lambda$ such that $\delta_2 = b$ form a sublattice and therefore the condition $\delta_2 = b$ is a regular predicate.

Figure 10(a) gives the slice of partitions in which $\delta_2 = 2$. Since the second part must be 2, additional edges ensure that the ideal of the graph has exactly two elements from P_2. On collapsing the strongly connected components and transitively reducing the graph we get Figure 10(b). By applying counting lemma, we get that there are $(2+1)(2+1) = 9$ such partitions which can all be enumerated automatically using Figure 10(b). They are:

$$\{220, 221, 222, 320, 321, 322, 420, 421, 422\}$$

As another example assume that we are interested in all partitions less than λ which have distinct parts. Figure 11(a) gives the slice and Figure 11(b) gives the graph after simplification. The graph is equivalent to that of subsets of size 3 from [5]. Hence, there are $\binom{5}{3}$ such partitions. These partitions

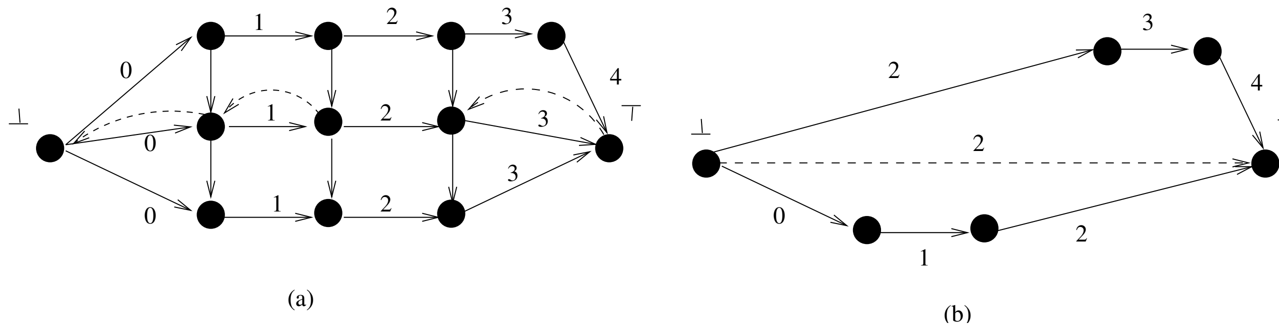


Figure 10: Slice for $\delta_2 = 2$

can also be enumerated from the figure. They are:

$$\{210, 310, 410, 320, 420, 430, 321, 421, 431, 432\}.$$

Some other subsets of partitions discussed in the literature are "partitions with odd number of parts," "partitions with distinct odd parts," "partitions with even number of parts" etc. These are also regular predicates.

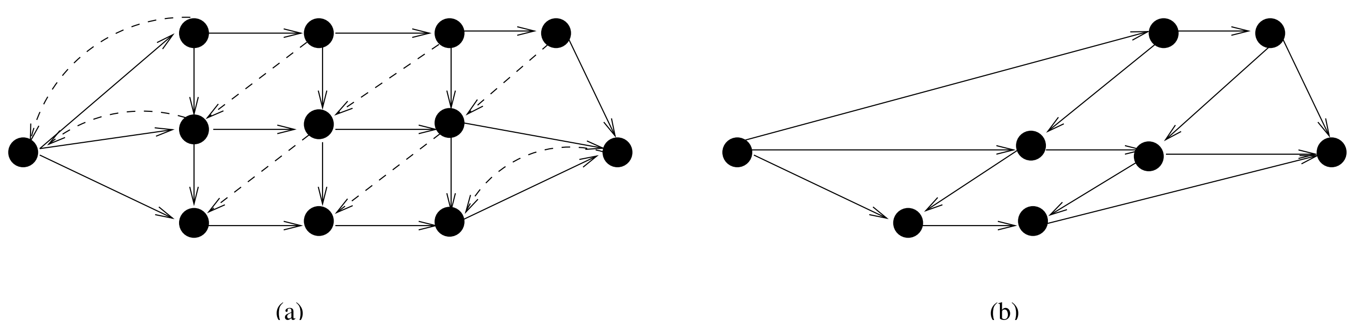


Figure 11: Slice for "distinct parts"

Now the reader may also see the solution for the second problem (Q2) mentioned in the introduction—enumerating all partitions in the Young's lattice Y_λ with first part equal to the second part. We simply define the predicate B on a partition δ to be true when δ_1 equals δ_2. It is clear that the predicate is closed under joins and meets and is therefore a regular predicate. One can draw the slice and conclude that the number of partitions δ in Y_λ satisfying $\delta_1 = \delta_2$ is equal to the number of partitions in Y_δ where $\delta = (\lambda_2, \lambda_3, \ldots, \lambda_k)$. The slice can also be used to enumerate all required partitions.

Note that the level set of rank N of Y_λ (where $\lambda = (\lambda_1, \lambda_2 \ldots, \lambda_t)$) corresponds to all partitions of N with at most t parts and the largest part at most λ_1. It follows that all partitions of N can be enumerated as the elements in level set of rank N of $Y_{(N,N,..N)}$.

5.4 Permutations

We first show a small graph that generates all permutations of n symbols. The graph consists of $n-1$ chains. Chain P_i has $i-1$ elements. There are multiple ways to interpret a given ideal G. We use $G[i]$ to denote the number of elements from chain i included in the ideal G.

(1) The simplest way is to view the permutation as a problem of putting n symbols into n places. $G[i]$ indicates the place for the symbol $i+1$. We start with placing n and then go backwards to 1.

The symbol n has n choices for places. This we can determine from $G[n-1]$ which is a number in $\{0..n-1\}$. Given the place for symbol n, the symbol $n-1$ has $n-1$ choices of places which is given by $G[n-2]$ because one place is already occupied. This process is repeated till symbol 2 is placed. Finally, symbol 1 has no choice.

(2) Another method is to use the inversion table[Knu98]. The number of inversions of i in a permutation π is the number of symbols less than i that appear to the right of i in π. The way a permutation is generated from a ideal is as follows. We begin the permutation by writing 1. $G[1]$ decides where to insert the symbol 2. There are two choices. These choices correspond to number of *inversions* introduced by 2. If we place 2 after 1, then we introduce zero inversions; otherwise we introduce one inversion. Proceeding in this manner we get that there is a bijection between the set of permutations and the set of ideals.

We will focus on the interpretation based on inversions. It is easy to show that the following predicates are regular. Further by computing the slice, we can also calculate the number of permutations satisfying B.

Lemma 2 *All the following properties of permutations are regular.*

(1) The symbol $m < n$ has at most *j inversions (for $j < m$). The total number of such permutations is $\frac{n!(j+1)}{m}$.*

(2) $i \le j$ implies that i has at most as many inversions as j. The total number of such permutations is same as the number of integer partitions less than $(n-1, n-2, ..., 1)$ which is equal to nth Catalan number.

Proof: Lemma 2 For the first part, note that it is sufficient to add an edge from $\top$ to event e_j in process P_{m+1}. This ensures that symbol m cannot have more than j inversions. Figure 12(a) shows the slice for set of permutations on [5] such that the symbol 4 has at most 1 inversion.

For the second part, we simple add an edge from event e_j on P_{i+1} to P_i for all j and i. Thus P_i can execute j elements only if P_{i+1} has executed j or more elements. The claim then follows by comparing the poset with that corresponding to Young's lattice.

■

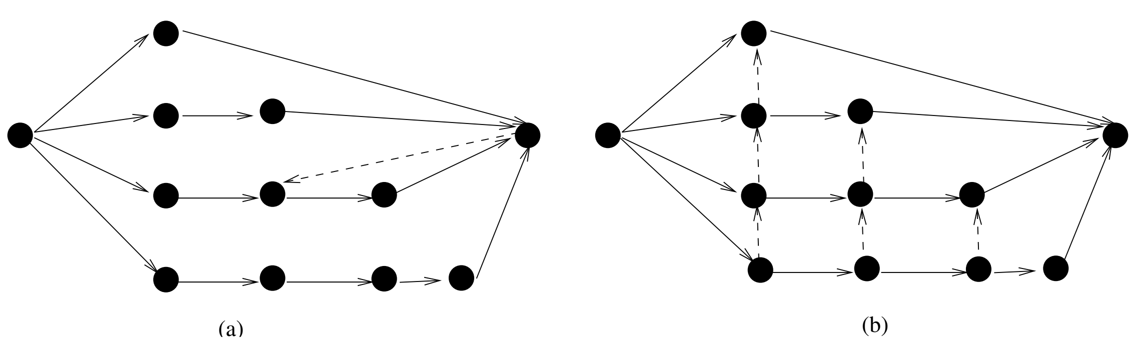


Figure 12: Slice for subsets of permutations

The level set at rank k of the permutation lattice consists of all permutations with total number of inversions equal to k and therefore such permutations can be efficiently enumerated [Knu98, ER02].

As another example, consider the problem of enumerating or counting all permutations such that no symbol has more that $n/2$ inversions and the total number of inversions is less than c. The predicate "no symbol has more than $n/2$ inversions" is a regular predicate that can be detected efficiently. The predicate "the total number of inversions is less than c," is a relational predicate and can also be detected efficiently. Therefore, by applying results of this paper, we can mechanically construct slices for such permutations.

6 Conclusions and Future Work

We have shown that slicing is useful in mechanical analysis of combinatorial problems and that one can efficiently compute slice for any predicate which can be efficiently detected.

In this paper, we have focused on the generators based on elements in the lattices. These ideas can be applied to other ways of generating large sets from small sets. For example, instead of using the lattice of ideals, we can use the set of paths in the ideal lattice (lattice paths) as the large set.

References

[CBDGF95] B. Charron-Bost, C. Delporte-Gallet, and H. Fauconnier. Local and temporal predicates in distributed systems. *ACM Transactions on Programming Languages and Systems*, 17(1):157–179, January 1995.

[CM91] R. Cooper and K. Marzullo. Consistent detection of global predicates. In *Proc. of the Workshop on Parallel and Distributed Debugging*, pages 163–173, Santa Cruz, CA, May 1991. ACM/ONR.

[DP90] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK, 1990.

[ER02] S. Effler and F. Ruskey. A CAT algorithm for listing permutations with a given number of inversions. *Information Processing Letters*, 2002.

[FLST86] U. Faigle, L. Lovász, R. Schrader, and Gy. Turán. Searching in trees, series-parallel and interval orders. *SIAM Journal on Computing*, 15(4):1075–1084, 1986.

[Gar96] V. K. Garg. *Principles of Distributed Systems*. Kluwer Academic Publishers, Boston, MA, 1996.

[Gar02] V. K. Garg. *Elements of Distributed Computing*. Wiley & Sons, New York, NY, 2002.

[Gar03] V. K. Garg. Enumerating global states of a distributed computation in lexicographic and breadth-first manner. In *International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, pages 134–139, November 2003.

[GM01] V. K. Garg and N. Mittal. On slicing a distributed computation. In *21st International Conference on Distributed Computing Systems (ICDCS' 01)*, pages 322–329, Washington - Brussels - Tokyo, April 2001. IEEE.

[JMN95] Roland Jégou, Raoul Medina, and Lhouari Nourine. Linear space algorithm for on-line detection of global predicates. In Jörg Desel, editor, *Proceedings of the International Workshop on Structures in Concurrency Theory (STRICT)*, Workshops in Computing, pages 175–189. Springer-Verlag, 1995.

[Knu98] Donald E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, second edition, 1998.

[MG01] N. Mittal and V. K. Garg. Slicing a distributed computation: Techniques and theory. In *5th International Symposium on DIStributed Computing (DISC'01)*, pages 78 – 92, October 2001.

[PB83] J.S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

[Spi85] Jeremy Spinrad. On comparability and permutation graphs. *SIAM Journal on Computing*, 14(3):658–670, 1985.

[Squ95] M. Squire. *Gray Codes and Efficient Generation of Combinatorial Structures*. PhD Dissertation, Department of Computer Science, North Carolina State University, 1995.

[Sta86] R. Stanley. *Enumerative Combinatorics Volumn 1*. Wadsworth and Brookes/Cole, Monterey, California, 1986.

[Ste84] G. Steiner. Single machine scheduling with precedence constraints of dimension 2. *Math. Operations Research*, 9:248 – 259, 1984.

[Ste86] G. Steiner. An algorithm to generate the ideals of a partial order. *Operations Research Letters*, 5(6):317 – 320, 1986.

[SW86] D. Stanton and D. White. *Constructive Combinatorics*. Springer-Verlag, 1986.

[Tro92] W.T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory*. The Johns Hopkins University Press, 1992.

[VD01] S. Venkatesan and Brahma Dathan. Techniques to tackle state explosion in global predicate detection. *IEEE Transactions on Software Engineering*, 27(8):704 – 714, August 2001.

[vLW92] J.H. van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, 1992.