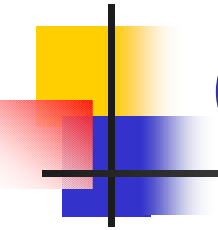
A linear delay algorithm for building concept lattices

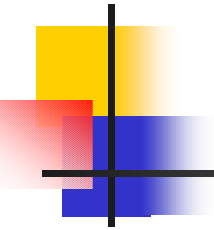
Yang Huang
National Center for Biotechnology Information, NLM, NIH
Joint work with Martin Farach-Colton

work done at Department of Computer Science, Rutgers University



Overview

- Introduction to concept lattice
- Related work on lattice construction
- Some characterization of lattices
- A linear delay algorithm for building concept lattices

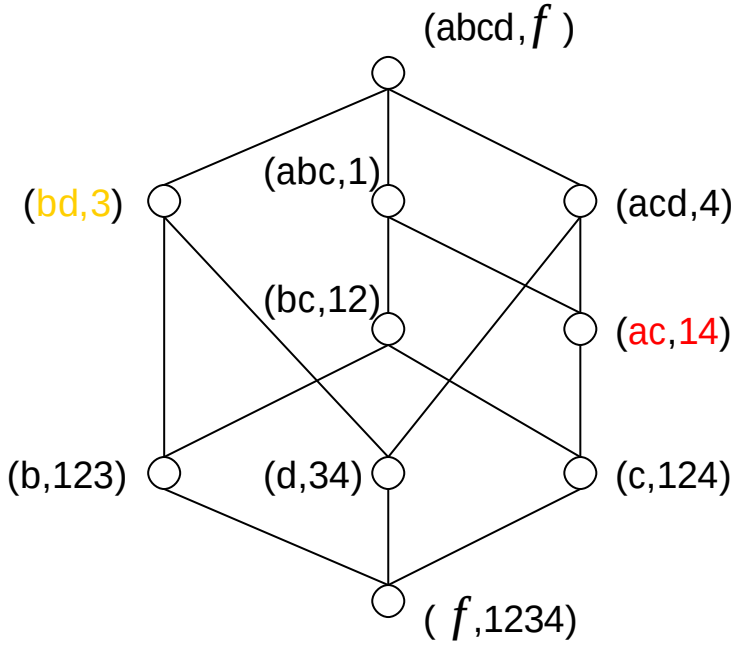


Motivation

- Concept lattices have proven useful in many applications:
  - Knowledge representation (Kalfoglou et al., 2004)
  - Information retrieval (Carpineto and Romano, 2004)
  - Software engineering (Snelting and Tip, 1998).
- Concept lattices have applications in gene-expression analysis. (Choi et al., 2007)
- It is important to design fast lattice construction algorithms.

Introduction to concept lattices

	1	2	3	4
a	1	0	0	1
b	1	1	1	0
c	1	1	0	1
d	0	0	1	1



{ac} ≐{14}. {14} ≐{ac}. {ac} ≐̸{ac} and {14} ≐̸{14}. Both of them are closed.

{ab} ≐{1}. {1} ≐{abc}. Hence {ab} is not closed.

Concepts

- Given a set of objects *G*, a set of attributes *M* and a binary relation $I \subseteq G \times M$, |*G*| >= |*M*|. for $A \subseteq G$ and $B \subseteq M$ we define

$$A' = \{t \mid (g,t) \in I, g \in A\}$$

$$B' = \{g \mid (g,t) \in I, t \in B\}$$

- A set *A* is called *closed* if *A* ″=*A*.
- A *concept* is a pair of closed sets $(A, B) \in 2^G \times 2$ For a concept C, $A = ext(C)$, $B = int(C)$.

Definition of concept lattice

- Define a partial order $\prec$ on the set of concepts B

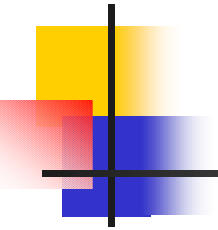
$$(A_1, B_1) \prec (A_2, B_2) \Leftrightarrow A_1 \subset A_2 (B_1 \supset B_2)$$

- The partially ordered set $L = < B, \prec >$ has the structure of complete lattice and is called a *concept lattice* or *Galois lattice*.

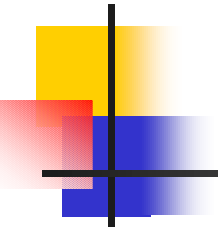
Hasse diagram of lattices

- A concept (A_1, B_1) is an *immediate successor (predecessor)* of another concept (A_2, B_2) if A_1 is a maximal subset (minimal superset) of A_2.
- In the *Hasse diagram* of lattices,
  - Each vertex represents a concept.
  - Each edge connects a concept with one of its immediate successors.

Complexity for enumeration problems



- Lattice construction is an enumeration problem.
- Total time complexity
  - *Polynomial total time*: The total running time is a polynomial in the size of the input and output.
- Delay time complexity
  - *Polynomial delay time*: the delay until the 1st entity is outputted and the delay between any two consecutive output entities is bound by a polynomial in the input size.



Polynomial delay preferred

- Guarantee when the next concept will be generated.
- Allow concept processing to follow immediately.
- Generate a subset of all concepts.

Related work on lattice construction

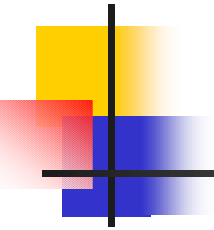
- Bordat's algorithm (Bordat, 1986)
  - polynomial delay $O(|G||M|^2)$
- Nourine and Raynaud's algorithm (Nourine and Raynaud, 1999)
  - total time complexity $O(|G||M||B|)$
  - Non-polynomial delay algorithm
- Incremental algorithm (Godin et al., 1995)

Related work on lattice construction

- Partition-based algorithm (Valtchev et al., 2002)
  - Exact complexity unknown
- Graph algorithm (Berry et al, 2003)
  - $O(|G||M|)$ per concept + $O(|G||M|^2)$ per traversed maximal chain
- Choi ś algorithm (Choi, 2006)

Related work (cont.)

- The problem is closely related to
  - generating all maximal bipartite cliques in a given bipartite graph
    - Makino and Uno ś algorithm (Makino and Uno, 2004)
  - generating all frequent closed itemsets in a transaction database.
    - Closet+ (Wang et al., 2004), CHARM (Zaki and Hsiao, 2005) and LCM2 (Uno et al., 2004)



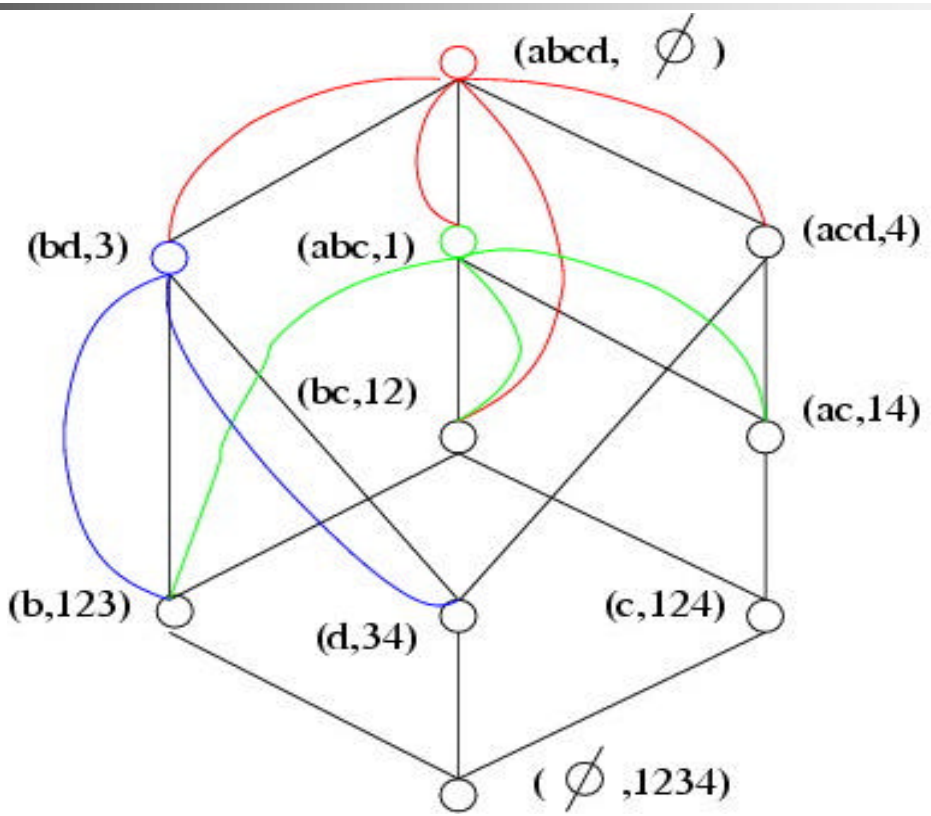
Overview

- Introduction to Galois lattice
- Related work on lattice construction
- Some characterization of lattices
- A linear delay algorithm

Irregular concepts

- Let the union of the attribute sets of immediate predecessors of D in the sublattice L^C be $int(IP_D^C)$.
- Definition: D is an *irregular concept* with regard to C if $int(D) \supset int(IP_D^C)$.
  - The set of irregular concepts with regard to C is denoted as IR^C.
  - All of C's immediate successors are in IR^C.

An example of irregular concepts



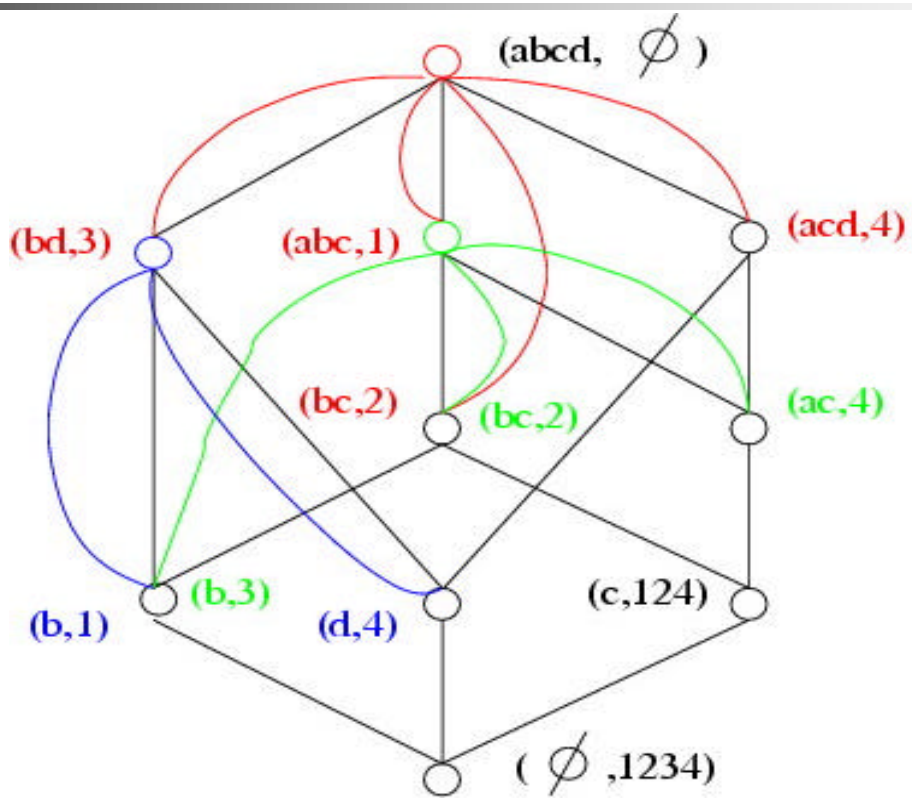
For example, $IR^{(abcd, f)} = \{(bd, 3), (abc, 1), (acd, 4), (bc, 12)\}$ and

$IR^{(abc, 1)} = \{(b, 123), (bc, 12), (ac, 14)\}$.

Property of irregular concepts

- Lemma:
  - The family of sets $int(D) \setminus int(IP_D^C)$, where $D \in IR^C$, constitutes a partition of the set of attributes, each of which belongs to one of elements of $ext(C)$ but not appears in $int(C)$.
- Corollary:
  - For any concept C, the size of IR^C is less than or equal to the size of M.

An example of the Lemma

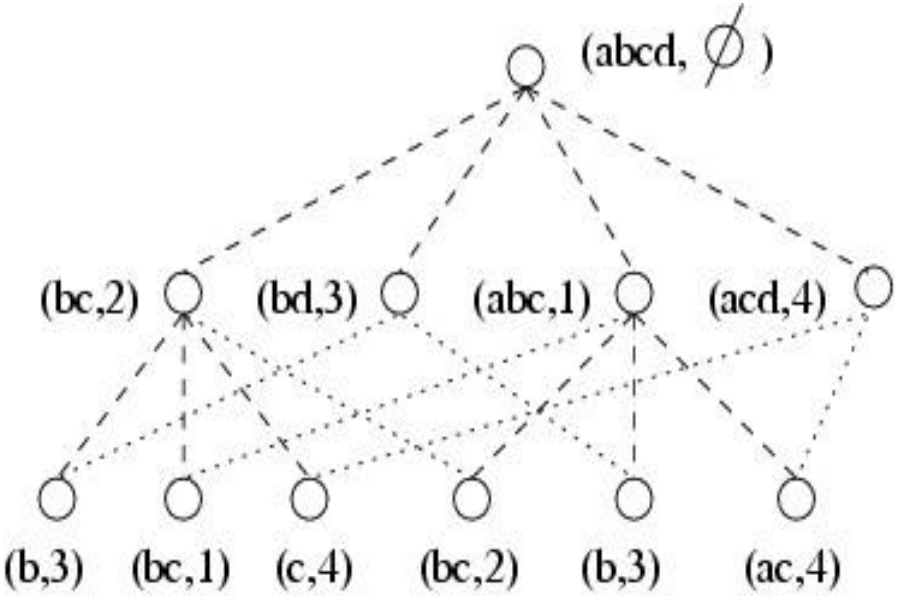


For example, every element of {234} belongs to some object in {abc} but not in the attribute set of (abc, 1).

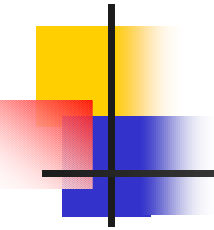
Generate irregular concepts

- Let $PIR^{C} = \{(A_i, B_i) \mid D = (A_i, A_i') \in IR^{C}\}$, where $B_i = A_i' \backslash int(IP_D^C)$.
- Theorem:
  - For $IR^{C} = \{C_i, i \in T\}$, we say $i,k \in [j_h]$ if $ext(C_i) \cap ext(C_j) = ext(C_k) \cap ext(C_j) \neq f$, where $[j_h]$ refers to an equivalent class.
    Let $A_{j_h} = ext(C_i) \cap ext(C_j)$ and $B_{j_h} = \cup_{i \in [j_h]} B_i$, then $PIR^{C_j} = \{(A_{j_h}, B_{j_h})\}$ if C_j is an immediate successor of C.

An example of the Theorem

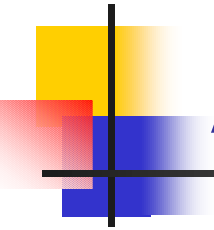


Generation of $PIR^{(bc,2)}$ and $PIR^{(abc,1)}$ by intersection.



Overview

- Introduction to Galois lattice
- Related work on lattice construction
- Some characterization of lattices
- A linear delay algorithm



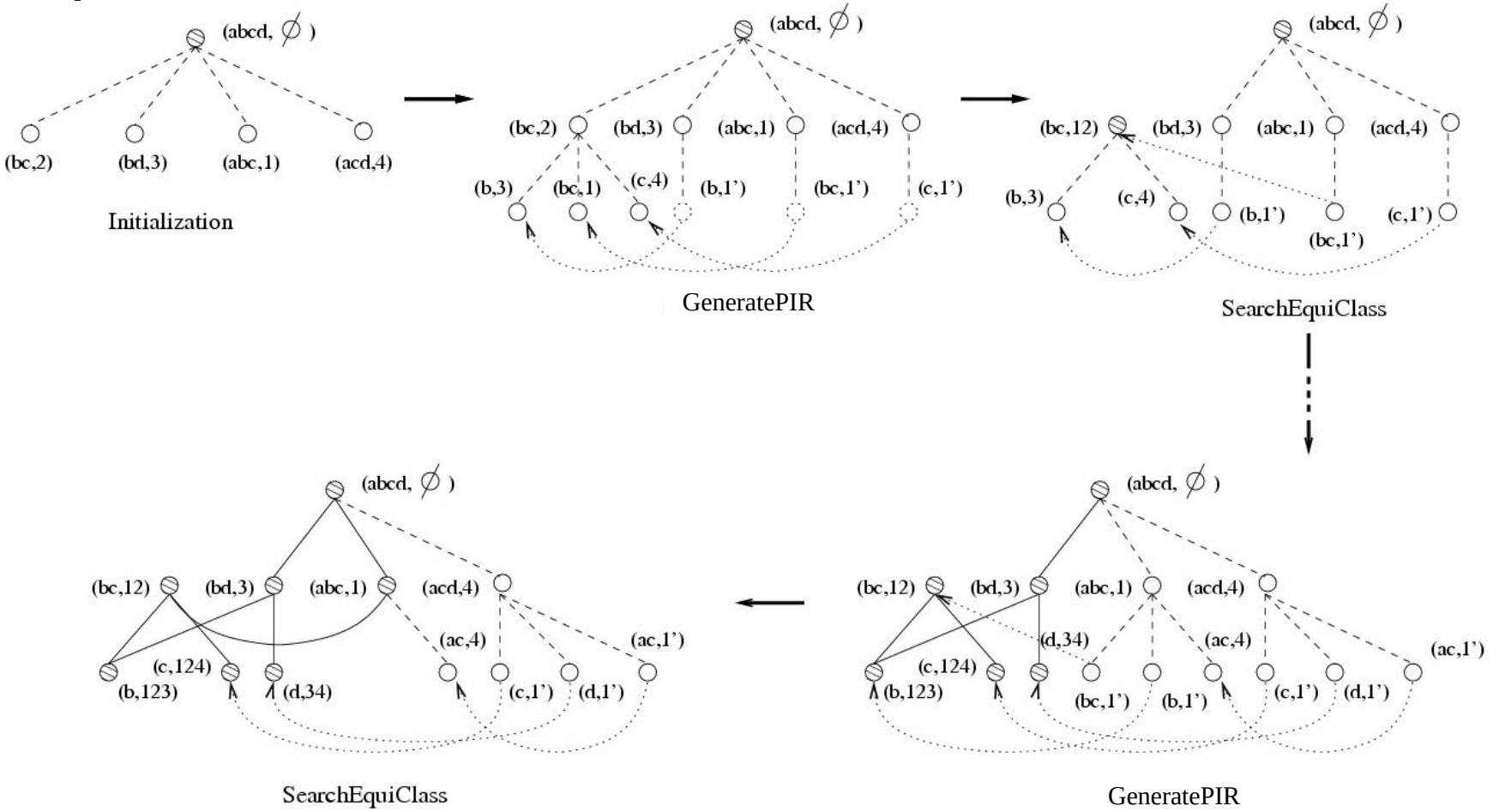
Algorithm overview

- Initialization
  - Generate the supremum of the lattice and its *PIR*.
- GeneratePIR
  - Perform intersection with object sets of other nodes in *PIR*.
- SearchEquiClass
  - Search equivalent classes using a trie for object sets.
  - Build concepts and identify immediate successors.

Algorithm overview (cont.)

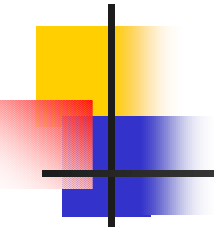
- Construct the lattice in a DFS way.
  - First initialization, then perform GeneratePIR and SearchEquiClass for each concept, and then move to its child nodes. If child nodes are all done, move to its siblings.
- Every concept is generated once and only once.
  - Shadow nodes prevent generating a concept multiple times.

A running example of the algorithm



Algorithm analysis

- Initialization
  - Scan the input matrix and construct a trie. Complexity $O(|G||M|)$
- GeneratePIR
  - Perform intersection on object sets. Complexity $O(|G||M|)$
- SearchEquiClass
  - Construct a trie for searching equivalent classes. Complexity $O(|G||M|)$



Future direction

- Find the lower bound for constructing a concept lattice given the input binary matrix.
- Design algorithms to match the lower bound
- More applications of concept lattices in various areas.