Efficient Formal Concept Analysis through Recursive Context Partitioning

Tim Pattison, Aaron Ceglar, and Derek Weber

{tim.pattison,aaron.ceglar,derek.weber}@dst.defence.gov.au
Defence Science & Technology Group
West Ave, Edinburgh
South Australia 5111

Abstract. Formal Concept Analysis takes as input a bigraph known as a formal context. It produces a partially-ordered set of formal concepts which constitutes a complete lattice. This lattice can be represented as a directed acyclic graph, whose vertices are formal concepts and whose arcs connect neighbours in the ordering relation between them. This paper describes a divide-and-conquer technique for discovering and exploiting hierarchical structure in a formal context. Simultaneous hierarchical partitioning of both the context bigraph and the resultant lattice digraph is used to achieve efficient computation and, elsewhere, interactive visualisation of the concept lattice.

1 Introduction

Formal Concept Analysis (FCA) derives a multiple-inheritance class hierarchy from a formal context. A formal context consists of a set of objects, a set of attributes, and a binary relation between them. The classes derived by FCA are known as formal concepts. Each consists of a set of objects, called its extent, and a set of attributes, called its intent, such that each object in its extent, and no others, has all attributes in its intent. The set of formal concepts, when partially ordered by set inclusion, forms a complete lattice. This lattice can be efficiently represented as a directed acyclic graph (DAG), whose vertices are formal concepts, and whose adjacency relation is the transitive reduction of the ordering relation.

The number of formal concepts is bounded above by an exponential function of the number of objects and attributes in the context. Scaling FCA to the interactive analysis of large data sets poses two fundamental challenges: the time required to compute the concepts and construct the large lattice digraph; and the difficulty of meaningful and responsive user interaction with this digraph.

The times required to enumerate all formal concepts of a formal context, and to calculate the transitive reduction of the ordering relation between them, are both bounded above by a polynomial function of the number of formal concepts. One class of "divide and conquer" techniques tackles this inherent computational complexity by partitioning the context, performing FCA on each resultant sub-context, and combining the results. FCA is thereby mapped onto multiple independent processors, each performing FCA on a sub-context which is significantly smaller than the overall context. Several FCA algorithms use this approach for the enumeration of concepts [7,2] and construction of the lattice digraph [3,11].

This paper describes the Carve technique for recursively partitioning a formal context, which produces a corresponding hierarchical partition of the lattice digraph. Carve exploits structure which we have identified in an empirical author-publication context for co-authorship analysis, and which Bhatti et al. [4] found in software systems. We explain how recognising this structure leads to a novel divide-and-conquer algorithm for efficient FCA. The resultant hierarchical partitioning of the lattice digraph is exploited elsewhere for both layout of, and interaction with, the Hasse diagram [10].

Carve recursively partitions a formal context for analysis by *any* FCA algorithm which, or algorithms which collectively: enumerates the formal concepts; calculates the transitive reduction of the ordering relation; and returns the corresponding labelled DAG. It further assembles the resultant digraphs into that for the original context. Carve does not compete with existing FCA algorithms; it accelerates their application to formal contexts exhibiting the requisite structure.

This paper is organised as follows. Section 2 introduces relevant aspects of the theories of partial orders, FCA, graphs and graph drawing. Section 3 then establishes the theoretical foundations for the Carve algorithm, after which the algorithm is detailed in Section 4. A brief discussion of related work is presented in Section 5, followed by a summary in Section 6.

2 Preliminaries

This section introduces relevant aspects of the theories of partial orders and FCA, as well as graphs and graph drawing. It assumes relevant knowledge found in such textbooks as [5] and [8] respectively.

Definition 1 *A* formal context (G, M, I) *is a triple consisting of a set G of objects, a set M of attributes and a binary relation $I \subseteq G \times M$ such that an object $g \in G$ has attribute $m \in M$ iff $(g, m) \in I$.*

Definition 2 *The context bigraph for the formal context (G, M, I) is a bipartite graph having object vertex set G, attribute vertex set M, and edge set I.*

Definition 3 *A* connected component *of a context bigraph is a maximal subgraph in which a path exists between all pairs of vertices.*

The example formal context of Figure 1a is represented in Figure 1b as a context bigraph. Object vertices are drawn with grey fill and attribute vertices white. The context bigraph has three connected components.

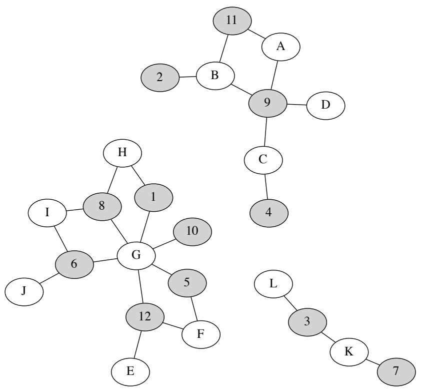
Definition 4 *The extent operator $^{\triangleleft} : \mathcal{P}(M) \to \mathcal{P}(G)$ and intent operator $^{\triangleright} : \mathcal{P}(G) \to \mathcal{P}(M)$ return the maximal sets*

$$B^{\triangleleft} = \{g \in G \mid (g, m) \in I \;\; \forall \; m \in B\} \tag{1a}$$

$$A^{\triangleright} = \{m \in M \mid (g, m) \in I \;\; \forall \; g \in A\} \tag{1b}$$

	A	B	C	D	E	F	G	H	I	J	K	L
1							✓	✓				
2		✓										
3											✓	✓
4			✓									
5						✓	✓					
6							✓		✓	✓		
7											✓	
8							✓	✓	✓			
9	✓	✓	✓	✓								
10							✓					
11	✓	✓										
12					✓	✓	✓					

(a) Formal context



(b) Context bigraph

Fig. 1: An example formal context and its corresponding bigraph.

of objects possessing all attributes in $B \subseteq M$ and attributes possessed by all objects in $A \subseteq G$.

Here, $\mathcal{P}(G)$ and $\mathcal{P}(M)$ denote the powersets – sets of all subsets – of G and M respectively. $m^{\triangleleft} \equiv \{m\}^{\triangleleft}$ is the set of objects which have attribute m and $g^{\triangleright} \equiv \{g\}^{\triangleright}$ is the set of attributes possessed by object g.

Proposition 1. *For $A \subseteq G$ and $B \subseteq M$*

$$B^{\triangleleft} = \{g \in G \mid g^{\triangleright} \supseteq B\} \tag{2a}$$
$$A^{\triangleright} = \{m \in M \mid m^{\triangleleft} \supseteq A\} \tag{2b}$$

Definition 5 *A* formal concept *of the context (G, M, I) is an ordered pair (A, B) with extent $\emptyset \subseteq A \subseteq G$ and intent $\emptyset \subseteq B \subseteq M$ satisfying*

$$A^{\triangleright} = B \tag{3a}$$
$$B^{\triangleleft} = A \tag{3b}$$

Definition 6 *A* biclique *(A, B) in the context bigraph (G, M, I) is a set $\emptyset \subset A \subseteq G$ of object vertices and a corresponding set $\emptyset \subset B \subseteq M$ of attribute vertices such that each vertex in A is adjacent to all vertices in B, and vice versa.*

Definition 7 *A biclique (A, B) is* maximal *if there is no biclique $(A', B') \neq (A, B)$ having $A' \supseteq A$ and $B' \supseteq B$.*

Proposition 2 (Gaume et el. [6]). *(A, B) is a formal concept in the formal context (G, M, I) having extent $A \neq \emptyset$ and intent $B \neq \emptyset$ iff (A, B) is a maximal biclique in the bigraph (G, M, I) having object vertex set A and attribute vertex set B.*

The elements of the set $\mathfrak{B}(G, M, I)$ of formal concepts are partially ordered by defining the relation $\leq$ between concepts $(A, B), (C, D) \in \mathfrak{B}(G, M, I)$ such that

$$A \subseteq C \iff (A, B) \leq (C, D) \iff B \supseteq D \tag{4}$$

Definition 8 *Let $\langle \mathfrak{P}; \leq \rangle$ be a finite partially-ordered set, and let $a, b \in \mathfrak{P}$ satisfy $a < b$ – i.e. $a \leq b$ and $a \neq b$. If $\nexists c \in \mathfrak{P}$ satisfying $a < c < b$, then we write $a \prec b$. Element a is then called a* lower neighbour *of b, and conversely b is an* upper neighbour *of a.*

Proposition 3. *FCA of the formal context (G, M, I) produces the lattice*

$$\langle \mathfrak{B}(G, M, I); \leq \rangle = \left\langle \mathfrak{C} \cup \bigvee_{c \in \mathfrak{C}} c \cup \bigwedge_{c \in \mathfrak{C}} c; \leq \right\rangle \tag{5}$$

where c ranges over the set $\mathfrak{C}$ of maximal bicliques in (G, M, I).

Thus the global supremum $\vee_{c \in \mathfrak{C}} c$ and infimum $\wedge_{c \in \mathfrak{C}} c$ of the set $\mathfrak{C}$ of maximal bicliques complete the set $\mathfrak{B} \supseteq \mathfrak{C}$ of formal concepts in cases where they have empty intent or extent, and are therefore not already included in $\mathfrak{C}$. We refer to the supremum and infimum collectively as the extrema.

The complete lattice $\langle \mathfrak{B}(G, M, I); \leq \rangle$ can be efficiently represented as a labeled digraph whose vertices are concepts and whose arcs connect neighbours. Each arc is directed from the lower to the upper neighbour, and concepts are comparable iff there exists a directed path between the corresponding vertices. For each concept (A, B), the corresponding vertex is labelled with attribute set $\mu \subseteq B$ and object set $\gamma \subseteq A$ defined as follows:

$$\mu := \{m \in M | m^{\lhd} = A\} \tag{6a}$$

$$\gamma := \{g \in G | g^{\rhd} = B\} \tag{6b}$$

Definition 9 *The concept (A, B) is an* attribute concept *for each attribute $m \in \mu$ and an* object concept *for each object $g \in \gamma$.*

The resultant directed graph is a single-source, single-sink DAG. The source vertex `s`, corresponding to the infimum, has only outgoing arcs; the sink vertex `t`, corresponding to the supremum, has only incoming arcs. The extent [intent][1] of any concept is the set of all object [attribute] labels encountered on downward [upward] paths from that concept. Two lattices are order isomorphic iff their lattice digraphs, excluding labels, are isomorphic.

Proposition 4. *Let `t` and `s` be the sink and source vertices, respectively, of a lattice digraph. Denote by μ_t and γ_t the attribute and object label sets of `t`, and by μ_s and γ_s the attribute and object label sets of `s`. Then*

$$\begin{aligned} \mu_t &= G^{\rhd} & \mu_s &= \{m \in M | m^{\lhd} = M^{\lhd}\} \\ \gamma_t &= \{g \in G | g^{\rhd} = G^{\rhd}\} & \gamma_s &= M^{\lhd} \end{aligned} \tag{7}$$

[1] Square brackets indicate that a sentence is true both when read without the bracketed terms and when read with each bracketed term substituted for the preceding term.

Proposition 4 indicates that t is labelled with attributes possessed by all objects and with objects possessing only these universal attributes. Similarly, s is labelled with objects possessing all attributes and with attributes possessed only by these universal objects.

A Hasse diagram [8] is a two-dimensional spatial embedding of the lattice digraph in which the vertical component of each arc is upwards, and edge directions are accordingly omitted. Attribute [object] labels are placed above [below] the labelled concept. Figure 2b shows the Hasse diagram for the context bigraph in Figure 2a. The labelling indicates, inter alia, that the supremum is an attribute concept for attribute `G` and an object concept for object `10`. The vertex having attribute label set $\mu = \{\texttt{F}\}$ and object label set $\gamma = \{\texttt{5}\}$ in Figure 2b corresponds to the concept $(\{\texttt{5}, \texttt{12}\}, \{\texttt{F}, \texttt{G}\})$; it "inherits" attribute `G` from its upper neighbour and object `12` from its lower neighbour.

3 Foundations of Recursive Partitioning

3.1 Partitioning the Context Bigraph

Each biclique of the context bigraph must be entirely contained within a single connected component. The maximal bicliques of (G, M, I) can therefore be enumerated by dividing the bigraph into its connected components and enumerating the maximal bicliques of each component. Separate connected components, and hence also their bicliques, share neither object nor attribute vertices.

Proposition 5. *If* $\vee_{c \in \mathfrak{C}} c \in \mathfrak{C}$ *or* $\wedge_{c \in \mathfrak{C}} c \in \mathfrak{C}$ *then the bigraph* (G, M, I) *consists of a single connected component.*

3.2 Partitioning the Lattice Digraph

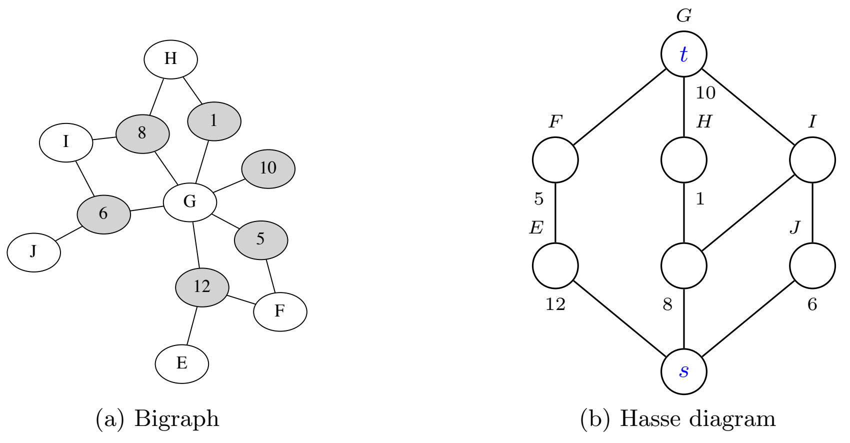
For any two concepts, the intent [extent] of their supremum [infimum] is the intersection of their intents [extents].

Proposition 6. *For any pair of maximal bicliques drawn from different connected components, their extrema are those of the set* $\mathfrak{C}$ *of maximal bicliques.*

Corollary 1 *No arcs exist between vertices of the lattice digraph corresponding to maximal bicliques from different connected components of* (G, M, I).

Corollary 1 indicates that partitioning the context bigraph into connected components partitions the vertices of the resultant lattice digraph, excepting the extrema, into disjoint sets between which there are no arcs. From the converse of Proposition 5, if (G, M, I) consists of more than one connected component, then the extrema are not maximal bicliques, and are not covered by Corollary 1.

Figure 3a uses rounded boxes as containers to illustrate this partitioning of the concepts for the example context in Figure 1. The concepts derived from the largest connected component of the context bigraph – with the exception of the



(a) Bigraph (b) Hasse diagram

Fig. 2: Bigraph and Hasse diagram for largest connected component of Figure 1b.

infimum which does not correspond to a biclique – are in the pink container. These concepts and the ordering relation between them are the same as those in Figure 2b, which resulted from FCA of the largest component in isolation. As expected, there are no edges (arcs) between concepts in different containers.

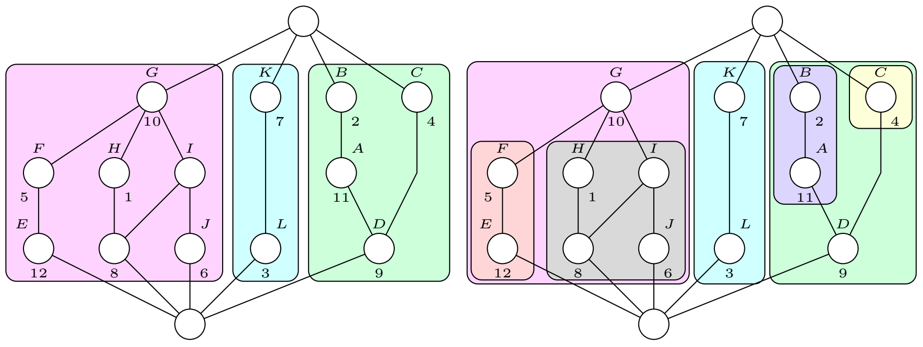
A context bigraph exhibiting two or more connected components can be partitioned into those components, those components analysed separately to identify their maximal bicliques and compute the transitive reduction of their partial ordering, and the sink [source] vertices of each resultant sub-graph of the lattice digraph connected to the global sink [source] to form the lattice digraph. FCA of the context (G, M, I) is thereby reduced to independent FCA for each of the sub-contexts $(G_i, M_i, I \cap (G_i \times M_i))$. Some care is required (see Section 3.5) to ensure the proper handling of the extrema of each sub-context.

3.3 Reducing the Context

Proposition 7. *Let (G', M', I') be formed from (G, M, I) by removing object set γ_s, attribute set μ_t, and all edges incident on either. Then $\langle \mathfrak{B}(G', M', I'); \leq \rangle$ is order isomorphic to $\langle \mathfrak{B}(G, M, I); \leq \rangle$.*

Proposition 8. *Let (G', M', I') be formed from (G, M, I) by removing object set γ_t, attribute set μ_s, and all edges incident on either. Then $\langle \mathfrak{B}(G', M', I'); \leq \rangle$ is order isomorphic to $\langle \mathfrak{B}(G, M, I); \leq \rangle$ iff the supremum and infimum of $\mathfrak{B}(G, M, I)$ have more than one lower and upper neighbour respectively.*

Propositions 7 and 8 involve the application of row and column reduction [5] to the objects and attributes for which the extrema are object and attribute concepts. They allow us to safely remove from the context all fully-connected attributes and objects, and in many cases any objects and attributes which are isolated after this removal.



(a) Initial partitioning of maximal bicliques (b) Recursive partitioning

Fig. 3: Hasse diagram for the context in Figure 1 showing partitioning of its maximal bicliques. Figure 3b adds an inclusion tree layout (nested containers) to illustrate the recursive execution of `Carve()`.

Figure 4a shows the connected components of the bigraph in Figure 2a after removal of the universal attribute `G` and its incident edges. Object `10` is now isolated, and cannot therefore participate in a biclique. Figure 4b shows the Hasse diagram for this sub-context after removal of both `G` and `10`. As indicated by the shaded containers, the maximal bicliques are partitioned into two sets, corresponding to the two remaining connected components of the bigraph, between which there are no connections. Comparison of Figures 4b and 2b confirms that the lattices before and after extremum reduction are order isomorphic. Since the supremum has more than one lower neighbour, removal of object `10` satisfies the condition of Proposition 8 for lattice isomorphism.

Let (G', M', I') denote the result of applying Propositions 7 and 8 to (G, M, I). The extrema of this *extremum-reduced* context are no longer object or attribute concepts, and accordingly have no labels. Whereas the concept lattices for (G, M, I) and (G', M', I') are order isomorphic, the corresponding lattice digraphs will therefore differ in the labelling of the supremum and infimum. Restoring these labels, which are now stored in $\gamma_s, \mu_s, \gamma_t$ and μ_t, converts the lattice digraph for (G', M', I') into that for (G, M, I). For example, restoring the label sets $\gamma_t = \{\texttt{10}\}$ and $\mu_t = \{\texttt{G}\}$ to the supremum converts the Hasse diagram in Figure 4b to that in Figure 2b.

3.4 Recursive Partitioning

We have seen that the removal of any universal attributes and objects, as per Proposition 7, can disconnect the context bigraph, and thereby pave the way for partitioning of (G, M, I). If (G, M, I) is instead a sub-context resulting from a previous partition, removal of these objects and attributes may facilitate partitioning of that sub-context, and hence recursive partitioning of the global context.

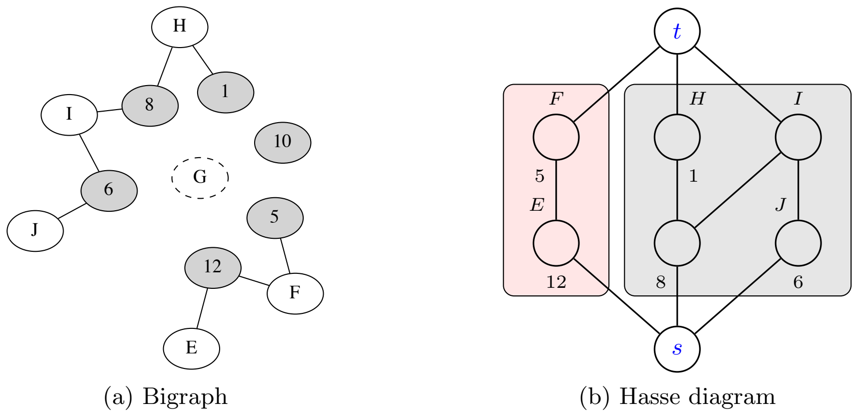


Fig. 4: Removal of attribute G from the bigraph in Figure 2a reveals new connected components and a corresponding partition of its maximal bicliques.

The execution of a procedure which removes from the context bigraph the objects and attributes identified in Propositions 7 and 8, along with their adjacent edges, identifies the connected components in the remaining context bi-graph, and calls itself to process each of the identified components, is described by its recursion tree. Figure 5 shows this tree superimposed on the context bigraph from Figure 1 using an inclusion tree layout. Each rounded rectangular container corresponds to a vertex in the recursion tree. A container includes another iff the former corresponds to an ancestor of the latter in the recursion tree.

This tree also constitutes a partition tree for the context bigraph, in which the non-leaf nodes are labelled with the objects $\gamma_t \cup \gamma_s$ and attributes $\mu_t \cup \mu_s$ associated with the extrema of the corresponding sub-context. In Figure 5, these objects and attributes appear only in non-leaf containers.

The leaf nodes of this tree correspond to sub-contexts whose lattice digraphs are either trivial – consisting of one or two vertices – or otherwise not amenable to further partitioning. Each non-trivial leaf-node sub-context must be processed by a suitable FCA algorithm which returns the corresponding lattice digraph labelled as per (6). The resultant single-source, single-sink digraphs form the

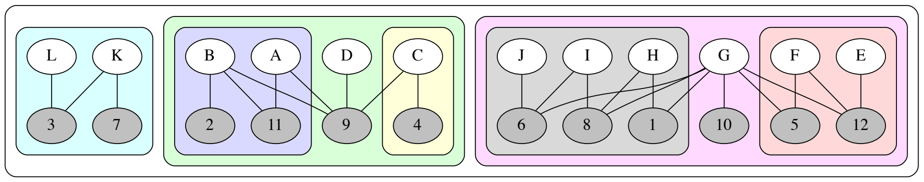


Fig. 5: Context bigraph from Figure 1 with recursion tree superimposed using inclusion tree layout.

building blocks which must then be appropriately assembled and interconnected to produce the lattice digraph for the overall context. Importantly, this assembly can be performed progressively in a single pass back up the recursion tree.

3.5 Assembling the Lattice Digraph

Assembling the lattice digraph for a given sub-context involves either connecting or merging the source (infimum) and sink (supremum) vertices $\mathtt{s}', \mathtt{t}'$ of the digraphs for each of its immediate sub-contexts with their counterparts $\mathtt{s}, \mathtt{t}$ for the current sub-context. In this section we examine the circumstances under which merging and connection, respectively, are appropriate.

Proposition 9. *Let (G', M', I') be an extremum-reduced context consisting of two or more connected components. Let (A, B) be the supremum [infimum] of one of these connected components, (G_i, M_i, I_i), and let $\mathtt{t}'$ [$\mathtt{s}'$] be the corresponding vertex of the lattice digraph for that sub-context. Then $(A, B) \in \mathfrak{B}(G', M', I')$ iff $\mu_{\mathtt{t}'} \neq \emptyset$ [$\gamma_{\mathtt{s}'} \neq \emptyset$].*

If the supremum [infimum] of (G_i, M_i, I_i) has an attribute [object] label, then an arc is added to connect it to its counterpart in the parent context (G', M', I'). This ensures that each maximal biclique in (G_i, M_i, I_i) has a directed path to the lattice supremum, and from the infimum, of the parent context. If the supremum [infimum] of (G_i, M_i, I_i) does not have an attribute [object] label, it should instead be merged with its counterpart in the parent context. The merge operation replaces two digraph vertices with a single vertex having the unions of their upper neighbours, lower neighbours, attribute labels and object labels.

4 Carve

Algorithm 1 lists the Carve algorithm for simultaneous recursive partitioning of a formal context and its corresponding lattice digraph. `Carve()` takes as input a context bigraph (G, M, I) and returns, by reference to its source and sink vertices, the corresponding lattice digraph, labelled as per (6). The Boolean parameter ϕ represents the level of recursion, and should be set to zero for the initial call. `Carve()` invokes the following functions:

`FindComponents`(G', M', I'): Takes as input a context bigraph (G', M', I') and returns an unordered set of its connected component bigraphs.

`FCA`(G', M', I'): Takes as input a formal context (G', M', I') which cannot be further decomposed by `Carve()` and returns the resultant lattice digraph. This function can be implemented using *any* FCA algorithm (see e.g. [9,1]) which enumerates the concepts, calculates the transitive reduction of the ordering relation between them, and calculates the labelling defined in (6).

`Connect`(a, b): Creates an arc from vertex a to vertex b of the lattice digraph.

`Merge`(a, b): Applied to lattice digraph vertices a and b, it modifies vertex b to have label sets and adjacent arcs, both incoming and outgoing, which are the unions of those of a and b. Thus vertex a is merged with vertex b.

Algorithm 1 Build lattice digraph through recursive partitioning of context.

```
Require: $I \subseteq G \times M$
Ensure: $(s,t)$ are (source,sink) vertices of labelled lattice digraph for $(G,M,I)$
 1: Function $(s,t) =$ Carve$(G,M,I,\phi)$
 2: $\mu_t \leftarrow G^{\triangleright}$
 3: if $\mu_t = M$ then
 4:     $\gamma_t \leftarrow G$
 5:     return $(t,t)$
 6: end if
 7: if $\mu_t \neq \emptyset$ or $\phi = 0$ then
 8:     $\gamma_t \leftarrow \{g \in G | g^{\triangleright} = \mu_t\}$
 9: else
10:     $\gamma_t \leftarrow \emptyset$
11: end if
12: $\gamma_s \leftarrow M^{\triangleleft}$
13: if $\gamma_s \neq \emptyset$ or $\phi = 0$ then
14:     $\mu_s \leftarrow \{m \in M | m^{\triangleleft} = \gamma_s\}$
15: else
16:     $\mu_s \leftarrow \emptyset$
17: end if
18: $M' \leftarrow M \setminus (\mu_t \cup \mu_s)$
19: $G' \leftarrow G \setminus (\gamma_t \cup \gamma_s)$
20: if $M' = \emptyset$ or $G' = \emptyset$ then
21:     Connect$(s,t)$
22:     return $(s,t)$
23: end if
24: $I' \leftarrow I \cap (G' \times M')$
25: if $\mu_t \neq \emptyset$ or $\gamma_s \neq \emptyset$ or $\phi = 0$ then
26:     $Q \leftarrow$ FindComponents$(G',M',I')$
27: else
28:     $Q \leftarrow \{(G,M,I)\}$
29: end if
30: if $|Q| = 1$ and $\gamma_t = \emptyset$ and $\mu_s = \emptyset$ then
31:     $(s',t') \leftarrow$ FCA$(G',M',I')$
32:     Merge$(t',t)$
33:     Merge$(s',s)$
34: else
35:     for all $(G_i,M_i,I_i) \in Q$ do
36:         $(s',t') \leftarrow$ Carve$(G_i,M_i,I_i,1)$
37:         if $\mu_{t'} \neq \emptyset$ then
38:             Connect$(t',t)$
39:         else
40:             Merge$(t',t)$
41:         end if
42:         if $\gamma_{s'} \neq \emptyset$ then
43:             Connect$(s,s')$
44:         else
45:             Merge$(s',s)$
46:         end if
47:     end for
48: end if
49: return $(s,t)$
```

Lines 2 to 24 of `Carve()` create and label the source and sink vertices of the sub-lattice digraph, return the lattice digraph in cases where it is trivial, and if not, convert the context (G, M, I) to its extremum-reduced form (G', M', I'). Lines 25 to 29 partition (G', M', I') into its connected component bigraphs. Lines 30 to 48 calculate the sub-lattice digraph for each of these components and connect or merge it into the source and sink vertices of the lattice digraph for (G, M, I). From the perspective of `FCA()`, `Carve()` is a pre-processor which reduces the context to an extremum-reduced, connected bigraph, and a post-processor which connects the resultant sub-lattice digraph into that for the original context.

Figure 3b uses an inclusion tree layout to illustrates the recursive execution of `Carve()` when applied to the context of Figure 1. The containers are filled with the same colours as their counterparts in Figure 5, in which the same recursion tree is overlaid on the context bigraph. The single vertex in the yellow container is returned by line 5 of Algorithm 1. Those in the light grey container are generated by a call to `FCA()` at line 31, and those pairs in the remaining three leaf-node containers are returned by line 22.

5 Related Work

Berry et al. [3] proposed a divide-and-conquer approach to FCA based on the identification of a vertex separator known as a clique minimal separator. A clique minimal separator is an attribute or an object, or an attribute-object pair, whose removal disconnects the context bigraph. The vertices of a bigraph having one or more such separators are "partitioned" by notionally removing each separator in turn, and replicating it across each of the adjacent connected components of the resultant bigraph. Berry et al. [3] nominated an algorithm which efficiently identifies these separators and described a method for combining the concept lattices produced by applying FCA to each resultant sub-context.

Decomposition using clique minimal separators produces a set of atomic bigraphs which by definition contain no clique minimal separators. FCA of each atomic bigraph produces corresponding lattice digraphs from which the original lattice digraph is reconstructed. In contrast with Carve, the resultant decomposition of the lattice digraph is not hierarchical, and reconstruction is complicated, inter alia, by the fact that edges can be required between constituent digraphs.

Valtchev et al. [11] described a procedure for recursive binary partitioning of an arbitrary formal context and the assembly of the overall lattice digraph from the digraphs arising from the sub-contexts. Only the attribute [object] set is partitioned, so that each sub-context contains all objects [attributes]. Whilst postulating that an "optimal" partition of the context would be one which minimised the size of the component digraphs, Valtchev et al. [11] left open the question of how to choose such a partition. In contrast, Carve discovers and exploits the structure of amenable contexts to recursively partition both the object and attribute sets, and significantly simplifies digraph assembly for this special case.

6 Summary

This paper has established the theoretical foundations of, and then detailed, the Carve algorithm, a divide-and-conquer technique for discovering and exploiting hierarchical structure in a formal context. Hierarchical partitioning of both the formal context and the resultant concept lattice have been used to achieve efficient computation of the lattice digraph. The discovered structure can be exploited for improved layout of, and interaction with, the Hasse diagram [10].

References

1. Andrews, S.: A 'best-of-breed' approach for designing a fast algorithm for computing fixpoints of Galois connections. Information Sciences **295**, 633–649 (2015). https://doi.org/10.1016/j.ins.2014.10.011
2. Baklouti, F., Levy, G.: Parallel algorithms for general Galois lattices building. In: Proc. Workshop on Distributed Data and Structures (WDAS 2003). Proceedings in Informatics, Carleton Scientific (2003)
3. Berry, A., Pogorelcnik, R., Sigayret, A.: Vertical decomposition of a lattice using clique separators. In: Napoli, A., Vychodil, V. (eds.) Concept Lattices and their Applications (CLA 2011). CEUR Workshop Proceedings, vol. 959, pp. 15–29 (2011), `http://ceur-ws.org/Vol-959/`
4. Bhatti, M.U., Anquetil, N., Huchard, M., Ducasse, S.: A catalog of patterns for concept lattice interpretation in software reengineering. In: Zhang, D., Reformat, M., Gokhale, S., Maldonado, J.C. (eds.) Proc. 24th Intl. Conf. Software Engineering & Knowledge Engineering (SEKE'2012). pp. 118–124. Knowledge Systems Institute Graduate School (2012)
5. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/978-3-642-59830-2
6. Gaume, B., Navarro, E., Prade, H.: A parallel between extended Formal Concept Analysis and bipartite graphs analysis. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) Computational Intelligence for Knowledge-Based System Design. LNCS, vol. 6178, pp. 270–280. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-14049-5_28
7. Gély, A.: A generic algorithm for generating closed sets of a binary relation. In: Ganter, B., Godin, R. (eds.) Formal Concept Analysis. LNCS, vol. 3403, pp. 223–234. Springer, Berlin (2005). https://doi.org/10.1007/978-3-540-32262-7_15
8. Gross, J.L., Yellen, J., Zhang, P. (eds.): Handbook of Graph Theory. Chapman & Hall/CRC Press, New York, second edn. (2013)
9. Kuznetsov, S.O., Poelmans, J.: Knowledge representation and processing with Formal Concept Analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **3**(3), 200–215 (2013). https://doi.org/10.1002/widm.1088
10. Pattison, T., Weber, D., Ceglar, A.: Enhancing layout and interaction in Formal Concept Analysis. In: Proc. 2014 IEEE Pacific Visualization Symposium. pp. 248–252. IEEE (2014). https://doi.org/10.1109/PacificVis.2014.21
11. Valtchev, P., Missaoui, R., Lebrun, P.: A partition-based approach towards constructing Galois (concept) lattices. Discrete Mathematics **256**(3), 801–829 (2002). https://doi.org/10.1016/S0012-365X(02)00349-7