Interaction Challenges for the Dynamic Construction of Partially-Ordered Sets

Tim Pattison and Aaron Ceglar

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

Abstract. We describe a technique for user interaction with the interim results of Formal Concept Analysis which we hypothesise will expedite user comprehension of the resultant concept lattice. Given any algorithm which enumerates the concepts of a formal context, this technique incrementally updates the set of formal concepts generated so far, the transitive reduction of the ordering relation between them, and the corresponding labelled Hasse diagram. User interaction with this Hasse diagram should prioritise the generation of missing concepts relevant to the user's selection. We briefly describe a prototype implementation of this technique, including the modification of a concept enumeration algorithm to respond to such prioritisation, and the incremental updating of both the transitive reduction and labelled Hasse diagram.

1 Introduction

Formal Concept Analysis (FCA) takes as input a formal context consisting of a set of attributes, a set of objects, and a binary relation indicating which objects have which attributes. It produces a partially-ordered set, or *poset*, of formal concepts, the size of which is, in the worst case, exponential in the number of objects and attributes in the formal context [1]. The computational tasks of enumerating the set of formal concepts, and of calculating the transitive reduction of the ordering relation amongst them, therefore scale poorly with the size of the formal context. These steps are required to determine the vertices and arcs of the directed acyclic graph whose drawing is known as the Hasse diagram of the partial order. The layout of this layered graph prior to its presentation to the user is also computationally intensive [2]. For contexts of even moderate size, there is therefore considerable delay between user initiation of the process of FCA and presentation of its results to the user.

A number of algorithms exist which efficiently enumerate the formal concepts of a formal context [3–6]. In this paper, we describe an approach which incrementally updates and presents the partial order amongst the formal concepts generated so far. In particular, it: incrementally updates the transitive reduction of the interim partial order as each new concept is generated; incrementally updates the layout of the Hasse diagram; and animates the resultant changes to

the Hasse diagram to assist the user in maintaining their mental model. This approach enables user exploration and interrogation of the interim partial order in order to expedite their comprehension of the resultant complete lattice of concepts. It applies equally to any other partial order, the enumeration of whose elements is computationally intensive.

We also describe how this interaction can prioritise the generation and display of those missing concepts which are most relevant to the user's current exploratory focus. By addressing the scalability challenge of visual analytics [7], this user guidance of computationally intensive FCA algorithms [8] facilitates the required "human-information discourse".

1.1 Previous work

Incremental algorithms exist for updating the set of formal concepts and the transitive reduction of the ordering relation following the addition of a new object to the formal context [9–11]. A new object can give rise to multiple additional concepts which must be inserted in the existing complete lattice to produce an updated lattice which is also complete. In contrast, the technique described in this paper involves the addition of a single element at a time to a partially ordered set which is not in general a complete lattice.

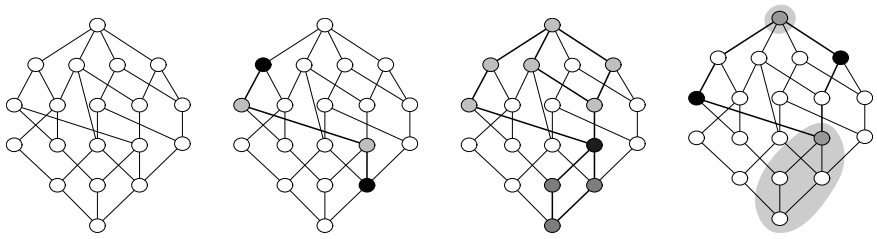
Ceglar and Pattison [8] have argued that user guidance of the FCA process could allow the satisfaction of the user's requirements with a smaller lattice, and consequently in less time, than standard FCA algorithms. They described a prototype tool which facilitates interactive user guidance and implements an efficient FCA algorithm which they have modified to respond to that user guidance. The user interaction challenges identified by that work are described and addressed in this paper.

2 Interacting with a Hasse diagram

2.1 The Hasse diagram

A finite poset $\langle P; < \rangle$ consists of a finite set P and an irreflexive, antisymmetric, transitive binary relation $<$ between its elements. Two elements $a, b \in P$ are said to be *comparable* if $a < b$ or $b < a$, and *incomparable* otherwise. $\langle P; < \rangle$ can be represented as an acyclic directed graph, or digraph, in which each element of the set P is a vertex and an arc connects each pair of comparable elements. The direction of each arc is "upward" in the sense of the relation $<$, from the lesser to the greater element.

The greater the number of comparable pairs, the greater the number of arcs, and hence the harder it is for a user to interpret a drawing of this digraph. The transitive reduction $\prec$ [12] of the ordering relation $<$ results from removing each arc whose source and destination vertices are connected by one or more other directed paths through the digraph. The transitive reduction therefore has fewer arcs than there are pairs of comparable elements in P. In the resultant



(a) Hasse diagram (b) Comparability (c) Upset, downset (d) Bounds

Fig. 1: Determining comparability, upset and downset, and bounds from the Hasse diagram

digraph, $a, b \in P$ are comparable iff there exists a directed path between the corresponding vertices, and b is said to be an *upper neighbour* of a, and a a *lower neighbour* of b, iff there is an arc from a to b.

A layered drawing of the resultant digraph, in which the vertical component of each arc is upwards on the page, is known as a *Hasse diagram* of the partial order. This direction convention reduces clutter by avoiding the need to explicitly represent the direction of the arcs. An example Hasse diagram is shown in Fig. 1a. The black vertices in Fig. 1b are comparable because a monotonically upward path, via the light-grey vertices, exists between them. The more elements there are in P, and the more arcs there are in the transitive reduction of $<$, the harder the visual task of tracing paths in the Hasse diagram to determine comparability.

2.2 Analytical objectives

An upper [respectively lower][1] bound on a given set $S \subseteq P$ is an element $u \in P$ $[l \in P]$ satisfying $\forall s \in S$, $s < u$ $[l < s]$. If S consists of a single element $a \in P$, the set of upper [lower] bounds on S is called an upset [downset], and can be identified by tracing all upward [downward] paths from a in the Hasse diagram. In Fig. 1c, the vertices in the upset [downset] of the black vertex are light [dark] grey. The union of a with its upset and downset gives the set of elements of P which are comparable with a. If S consists of two or more elements of P, then the set $U_S \subset P$ $[D_S \subset P]$ of its upper [lower] bounds is the intersection of the upsets [downsets] of its elements. In Fig. 1d, the sets of upper and lower bounds on the pair of black vertices are encompassed by the shaded areas. The visual task of identifying and intersecting these upsets [downsets] is demanding for small partial orders, and rapidly becomes intractable as the size $|P|$ of P increases.

If there exists an $a^* \in U_S$ such that $a^* \leqslant a$ for all upper bounds $a \in U_S$ on S, then a^* is called the *least upper bound* (LUB); if there exists a $b^* \in D_S$

[1] Square brackets are used throughout this paper to indicate that a sentence is true both when read without the bracketed terms and when read with each bracketed term substituted for the term which precedes it.

such that $b \leqslant b^*$ for all lower bounds $b \in D_S$ on S, then b^* is called the *greatest lower bound* (GLB). In Fig. 1d, the least upper and greatest lower bounds on the pair of black vertices are the upper and lower shaded vertices respectively. By definition, the LUB and GLB are unique whenever they exist, since equality corresponds to identity. In addition to the challenge of identifying the set of upper [lower] bounds on S, the visual task of establishing the existence and identity of the LUB [GLB] from this set also becomes intractable for large $|P|$.

If the LUB and GLB exist for all pairs of elements in P, then $\langle P; < \rangle$ is called a *lattice*. If the LUB and GLB exist for all $S \subseteq P$, then $\langle P; < \rangle$ is called a *complete lattice*, and the LUB [GLB] on P is known as the *supremum* [*infimum*].

2.3 Computer-assisted interaction

In the previous section, we identified a number of operations on posets which a user can perform visually by tracing paths in the Hasse diagram, but which become intractable as $|P|$ increases. To support the user in these tasks, the computer should calculate the results of these operations and display them by highlighting elements of the Hasse diagram. In particular:

1. Comparability between selected elements of P can be determined computationally and all identified paths between them highlighted in the Hasse diagram.
2. The set of elements comparable with a selected element can be determined by calculating and highlighting the members of its upset and downset.
3. The set of upper [lower] bounds on a set S of elements can be determined by calculating the upsets [downsets] of each, and highlighting the members of their intersection.
4. The existence and identity of the LUB [GLB] on a selected subset S can be determined computationally from the result of 3. The identified element of P, if any, should be highlighted.

3 Dynamic presentation of the partial order

So far we have assumed that the elements of $\langle P; < \rangle$ are known *a priori*, so that the transitive reduction can be computed and the Hasse diagram laid out before being presented to the user. Consider now the case where the user's request for the Hasse diagram triggers the on-demand enumeration of the elements of P. If this enumeration is computationally intensive, the user may experience excessive delay before the results are presented. Rather than waiting for the generation of all elements, the user may wish to commence interaction with the Hasse diagram for the elements generated so far, and have this diagram evolve to incorporate each new element as it is added. We hypothesise that user exploration of, and familiarisation with, the evolving $\langle Q \subseteq P; < \rangle$ will facilitate and expedite comprehension of $\langle P; < \rangle$.

In order to examine the feasibility of user interaction with the Hasse diagram of $\langle Q; < \rangle$ as a proxy for that of $\langle P; < \rangle$, the subsequent sections address the following four key questions:

1. What information about $\langle P; < \rangle$ can and cannot be ascertained through inspection of $\langle Q; < \rangle$?
2. Should the response to user interrogation of $\langle Q; < \rangle$ be expressed in terms of $\langle P; < \rangle$?
3. Can $\langle Q; < \rangle$ be incrementally updated to incorporate each new element of P as it is generated?
4. How should these updates be presented so as to minimise disruption to the user's mental model of $\langle Q; < \rangle$?

3.1 Comparing the interim and final partial orders

Two elements which are comparable in $\langle P; < \rangle$ are comparable in any $\langle Q \subset P; < \rangle$ in which they both exist. The upset [downset] of an element in $\langle P; < \rangle$ is a (possibly improper) superset of its counterpart in $\langle Q; < \rangle$. Accordingly, the set of upper [lower] bounds on $S \subseteq Q$ in $\langle P; < \rangle$ is also a superset of its counterpart in $\langle Q; < \rangle$. Importantly, the LUB [GLB] on $S \subseteq Q$ may not be present in $\langle Q; < \rangle$, even if it is in $\langle P; < \rangle$.

3.2 Interacting with the interim partial order

User interaction could be defined to implement the same computer-assisted operations on $\langle Q; < \rangle$ as were defined above on $\langle P; < \rangle$. However, since the user's objective is to find out about $\langle P; < \rangle$, interaction with $\langle Q; < \rangle$ should also prioritise the generation of the requisite elements of P.

The selection of two elements a and b in order to determine their comparability might confirm comparability by not only highlighting the elements $x \in Q$ which lie between a and b, but also prioritise the generation of all such $x \in P$. Selection of an element might not only display its upset and downset in $\langle Q; < \rangle$, but also prioritise the completion of these sets by the process which generates the elements of P. Similarly, the selection of a set $S \subset Q$ of elements in the interim partial order may not only result in the display of the corresponding upper and lower bound sets, but also prioritise the completion of these sets. And finally, if the user requests the LUB [GLB] on a set $S \subset Q$ of elements, then the computer could prioritise the generation of the corresponding result in $\langle P; < \rangle$. If the requested bound exists, the corresponding element of P could be added, if not already present, and highlighted in the Hasse diagram; otherwise a null result should be signalled to the user.

3.3 Updating the interim partial order

As each new element e of P is generated and added to $\langle Q; < \rangle$ to form $\langle Q \cup \{e\}; < \rangle$, both the element set and the transitive reduction $\prec$ of $<$ must be updated. Updating the transitive reduction involves identifying the upper and lower neighbours of e, adding the requisite arcs, and deleting any arcs from the lower to the upper neighbours.

The set $\mathcal{N}^u := \min\{u \in Q : e < u\}$ $\left[\mathcal{N}^l := \max\{l \in Q : l < e\}\right]$ of upper [lower] neighbours can be identified through a top-down [bottom-up] search of $\langle Q; < \rangle$ starting from each of the maximal [minimal] elements of Q. Each downward [upward] path from each maximal [minimal] element can be pruned from the point at which an element is encountered which is not greater [less] than e. An element in $\{u \in Q : e < u\}$ $[\{l \in Q : l < e\}]$ is also in $\mathcal{N}^u$ $[\mathcal{N}^l]$ iff it has no lower [upper] neighbour greater [less] than e.

Once the sets $\mathcal{N}^u$ and $\mathcal{N}^l$ have been identified, the neighbour relation $\prec$ should be updated as follows. In $\langle Q \cup \{e\}; < \rangle$, $e \prec \alpha$ for each $\alpha \in \mathcal{N}^u$ and $\zeta \prec e$ for each $\zeta \in \mathcal{N}^l$. To maintain $\prec$ as the transitive reduction of $<$, any arcs in $\mathcal{N}^l \times \mathcal{N}^u$ must also be removed. Note that either or both of the sets $\mathcal{N}^u$ and $\mathcal{N}^l$ can be empty; the element e is maximal in $\langle Q \cup \{e\}; < \rangle$ iff $\mathcal{N}^u = \emptyset$, and minimal iff $\mathcal{N}^l = \emptyset$.

3.4 Presenting Hasse diagram updates

The layout of the Hasse diagram for $\langle Q \cup \{e\}; < \rangle$ will necessarily differ from that of $\langle Q; < \rangle$. The addition of the new vertex e will require either its accommodation within an existing layer or the creation of a new layer. The deletion of some existing arcs and the creation of new ones may also worsen the aesthetic criteria, such as number of edge crossings, which the chosen layout algorithm seeks to optimise [2].

The addition of the new vertex and its incident arcs, the deletion of superseded arcs, and any changes in layout, must be presented to the user in a way which minimises disruption of the user's mental model of $\langle Q; < \rangle$. The sequential animation of each step in this process is a logical solution to this problem. By also minimising and localising any changes to the layout necessitated by the addition of a new vertex, the number of vertices and edges whose movement the user must visually track can be minimised.

Figure 2 illustrates the process of adding a new element to $\langle Q; < \rangle$ in Fig. 2a. The upper (light grey) and lower (dark grey) neighbours of the new element are identified, and the edges between them (bold) slated for deletion. The vertical separation between the top and bottom layers is increased to make room for a new layer, which is required to accommodate the new vertex. Finally, the new vertex is inserted and attached to its upper and lower neighbours. Placement of the new vertex near its upper and lower neighbours helps localise the resultant changes to the Hasse diagram.

4 Application to Formal Concept Analysis

4.1 Introduction to FCA

FCA takes as input a bigraph consisting of a set G of objects, a set M of attributes, and a binary relation $R \subseteq G \times M$ between them. FCA produces a complete lattice of formal concepts. Each formal concept consists of a set $E \subseteq G$,

(a) Identify neighbours and superceded edges

(b) Make room for, insert and connect new vertex

Fig. 2: Inserting a new element into the poset $\langle Q; < \rangle$.

known as the *extent*, and a set $I \subseteq M$, known as the *intent*, such that $E \times I \subseteq R$. Each object in E has all attributes in I and each attribute in I is possessed by all objects in E. The extent and intent are maximal in the sense that adding elements to either set necessarily entails removing elements from the other. The relation $<$ between concepts is the subset relation $\subset$ between their extents. The upset [downset] of a concept consists of concepts whose intents [extents] are subsets of those of the nominated concept.

4.2 Labelling

The concept whose extent is the set of objects possessing attribute i is referred to as the *attribute concept* for i; the concept whose intent is the set of attributes possessed by object j is known as the *object concept* for j. In the Hasse diagram for the complete lattice of formal concepts, an attribute [object] concept is labelled with the corresponding attribute [object]. The intent of a concept can be inferred from the set of attribute labels on vertices in its upset, while its extent can be inferred from the set of object labels on vertices in its downset.

This reduced labelling scheme works for the interim partial order $\langle Q; < \rangle$ provided that all object and attribute concepts are present *ab initio*. The algorithm which generates the formal concepts should be chosen or modified so as to produce these first, and rendering of the Hasse diagram deferred until they are included in Q. It will be seen in Sect. 4.6 that this deferral also has benefits for the layout of the Hasse diagram.

4.3 Least upper and greatest lower bounds

In FCA, the intent of the LUB on a set of concepts is the intersection of their intents, while the extent of the GLB is the intersection of their extents. Thus, the LUB [GLB] indicates the set of attributes [objects] which they have in common. Since the partial order amongst the full set of formal concepts for a formal context is a complete lattice, the LUB and GLB are guaranteed to exist; they may however be absent from $\langle Q; < \rangle$, and will in general require computation on demand.

Notably, the LUB and GLB on the full set P of formal concepts may not be present in $\langle Q; < \rangle$. A concept enumeration algorithm should be chosen or modified to produce these first. In addition to improving the resemblance between $\langle Q; < \rangle$ and the resultant complete lattice $\langle P; < \rangle$, the existence of the LUB [GLB] *ab initio* provides a single, consistent starting point for the top-down [bottom-up] search of the transitive reduction of $\langle Q; < \rangle$ for the upper [lower] neighbours of each new concept. This process is described in Sect. 4.4.

Selection of a set $\mathcal{A}$ of concepts could trigger the generation, display and highlighting of all concepts $x \in P$ satisfying $\mathcal{A}_* < x < \mathcal{A}^*$. Here $\mathcal{A}_*$ and $\mathcal{A}^*$ denote the GLB and LUB on the set $\mathcal{A}$ in the complete lattice $\langle P; < \rangle$ of formal concepts. $\mathcal{A}_*$ and $\mathcal{A}^*$ constitute the GLB and LUB on the complete lattice of elements whose generation is to be prioritised. Their generation is given the highest priority in order to bound this poset. The elements of $\mathcal{A}$ are all mutually comparable iff at least one directed path from $\mathcal{A}_*$ to $\mathcal{A}^*$ passes through all of these elements. In this case, $\mathcal{A}_* \in \mathcal{A}$ and $\mathcal{A}^* \in \mathcal{A}$; in the special case of $|\mathcal{A}| = 2$, this is a necessary and sufficient condition for comparability.

4.4 Insertion point

As described in Sect. 3.3, the upper [lower] neighbours of the new concept can be identified in $\langle Q; < \rangle$ using a top-down [bottom-up] directed search. The search commences from the LUB [GLB] on P, which by design is present in Q *ab initio*. The current vertex is marked as having been visited, and any lower [upper] neighbour whose intent [extent] is a subset of that of the new concept is queued for subsequent traversal. If the current concept has no such neighbours, then it is an upper [lower] neighbour of the new concept.

4.5 Layer assignment

The poset $\langle Q; < \rangle$ is presented to the user as a layered drawing of the corresponding directed acyclic graph. In order to maintain the direction convention of upward arcs in this Hasse diagram, each new concept must be assigned to a layer which is separated from the infimum [supremum] by at least the maximum length of all directed paths between them in the transitive reduction of $\langle Q; < \rangle$. If no existing layer satisfies both constraints, one must be added. If only one such layer exists, the new concept is assigned to that layer. If more than one existing layer satisfies both constraints, the choice amongst them is arbitrary.

Even if at least one layer satisfies both constraints on path length, it is still possible for the new concept to have upper and lower neighbours in adjacent layers. Adding an intervening layer in this case would create room for the new concept, and would not require the revision of any previous layer assignments. However, given that there are already sufficient layers to accommodate the new concept, a more space-efficient strategy would be to instead promote the lowest upper neighbours to the next highest layer or demote the highest lower neighbours to the next lowest layer. To make room for either change, it may also

be necessary to promote additional members of the upset or demote additional members of the downset of the new concept.

In our preliminary implementation of layer assignment, we: assign a vertex to the uppermost layer, if any, satisfying the dual path length constraints; attempt to demote elements of the downset in order to make room to insert a new concept between upper and lower neighbours which are in adjacent layers; and insert a new layer, if required, which has path length two from the infimum to make room for this demotion. The reason for the choice of path length two will become apparent in Sect. 4.6. This layering strategy is illustrated later in Sect. 4.8.

The chosen strategy requires that concepts be allowed to migrate between layers following initial presentation to the user, which has the potential to disrupt the user's mental model. On the other hand, creating more layers than are required results in inefficient use of the vertical space available for drawing the Hasse diagram, making it harder for the user to maintain simultaneous focus and context as $|Q|$ increases.

4.6 Layout

For the reason given in Sect. 4.3, the concept enumeration algorithm should first generate the LUB and GLB of the poset $\langle P; < \rangle$ of concepts. These should be placed at the centre top and centre bottom of the canvas. For the reason given in Sect. 4.2, all (remaining) attribute and object concepts should be generated next. In this section, we prioritise from amongst these the generation of the lower neighbours of the supremum, which are called *atoms*, and the upper neighbours of the infimum, which are called *co-atoms*. We describe a scheme whereby their relative ordering within their respective layers is chosen so as to improve the aesthetics of the resultant Hasse diagram. The horizontal placement of all subsequent concepts, including the remaining attribute and object concepts, is dependent on this ordering.

The supremum and infimum, along with all atoms and co-atoms, are generated and laid out before the Hasse diagram is first presented to the user. The atoms and co-atoms are ordered within their respective layers so as to minimise edge crossings in the relation R. As subsequent concepts are discovered and added, each edge morphs into one or more directed paths in the transitive reduction of $\langle Q; < \rangle$. Provided care is taken in the placement of the remaining concepts on these paths, the effort invested in minimising edge crossings in R might therefore be repaid with fewer arc crossings, and potentially also shorter paths, in the Hasse diagram.

For each newly-discovered concept, the horizontal barycentre [13] of the atoms and co-atoms with which it is comparable is calculated, and a total order on these barycentres is used to order the concepts within a layer. The barycentre calculation assumes that the co-atom and atom layers are assigned equal width and that atoms and co-atoms are equally spaced within their respective layers. The assigned order, which is designed to place each concept in reasonable horizontal proximity to the corresponding atoms and co-atoms, is only dependent on the fixed horizontal placement of the atoms and co-atoms, rather than the entire

upset and downset. This choice also preserves the order amongst the concepts already placed within the layer, since each new concept is simply inserted into that existing order.

The creation of a new layer may be required to separate vertically, by at least one layer, the upper and lower neighbours of a new concept. That new layer is created immediately above the layer of co-atoms, and elements of the downset of the new concept are demoted as required. Note that demotion beyond this layer would violate the constraint on path length from the infimum. Since each new concept is placed in the uppermost layer satisfying the constraint on path length to the supremum, and new layers are only ever added with path length two from the infimum, concepts will either remain in the layer to which they are originally assigned, or migrate downward.

4.7 Guided concept enumeration

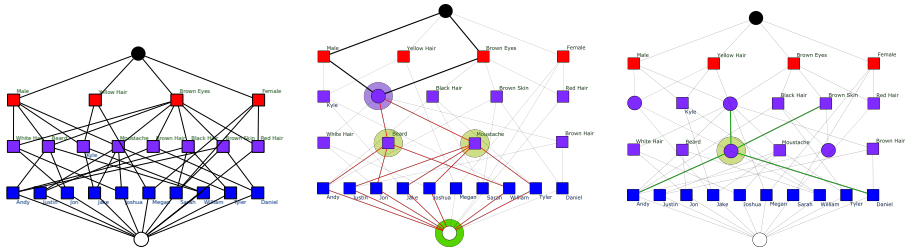
We have so far assumed that algorithms for the enumeration of formal concepts could be modified to prioritise the generation of concepts relevant to the user's current selection. To demonstrate that this is possible, we briefly describe a modification of the algorithm of Choi and Huang [14] to prioritise the generation of the downset of a selected concept.

Choi and Huang enumerate the concepts of a formal context in top-down, breadth-first order, extending the intent of each concept generated so far using attributes not currently in its intent. Ceglar and Pattison [8] have modified this algorithm to: hand over each novel concept, as it is generated, for subsequent processing; respond to user input, if any, after the generation of each novel concept; and switch to depth-first processing at the user's request to enumerate the downset of a nominated concept, resuming breadth-first processing upon completion. The upset of a nominated concept could be generated similarly by switching to bottom-up, depth-first processing, adding objects not in the extent of the nominated concept.

Efficient FCA algorithms employ various strategies for preventing the repeated generation of the same concept, typically by traversing a trie structure [15] superimposed on the concept lattice. Since user guidance of the FCA algorithm can interfere with this strategy, an efficient test for concept novelty, based for example on hash tables [15], is essential. In the proposed interactive approach, the overall efficiency of concept enumeration is less important than responsiveness to user interrogation of the interim partial order.

4.8 Prototype implementation

Figure 3 shows mockups of a prototype interface for the incremental construction of a poset of formal concepts. The example formal context consists of people and their physical attributes. Figure 3a depicts the state of the Hasse diagram first presented to the user. By this stage, all of the attribute and object concepts have been generated by the concept enumeration algorithm, labelled, and laid out to establish the framework for insertion of subsequent concepts.



(a) Initial diagram (b) Multiple selection (c) Insertion

Fig. 3: The result of multiple selection in the initial Hasse diagram, and of the subsequent insertion of multiple concepts.

Figure 3b shows the result of the user selecting in this diagram the attribute concepts for "Beard" and "Moustache", which are highlighted in response with light green halos. This multiple selection triggers the calculation of the extent and intent intersections for the selected concepts. The former corresponds to the infimum, which is accordingly highlighted with a bright green halo; the latter corresponds to a new concept, which is consequently inserted into the Hasse diagram and highlighted with a purple halo. Since this least upper bound has path length 2 from the supremum, a new row has been inserted to accommodate concepts with path length 3, and the selected concepts demoted to it. The least upper bound is inserted into row 2 at ordinal position 2 of 5; this position is based on the horizontal barycentre of its associated atoms and co-atoms, which are predominantly to the left of the centreline. Figure 3c shows the state of the Hasse diagram following the subsequent addition of a number of concepts; the most recently added concept is highlighted and its adjacent arcs shown green.

The prototype interface does not yet implement all of the recommendations in this paper. For example, while multiple selection triggers the calculation and display of the LUB and GLB, it does not prioritise the highlighting and completion of the set of concepts between them. Change animation is also currently lacking. If the context bigraph consists of more than one connected component, these should be dealt with separately (albeit connected to the same supremum and infimum) and the results horizontally juxtaposed. This reduces the computational complexity, and ensures a better result, of the layout of the atoms and co-atoms. It also allows correct handling of the exceptional case where the same concept is both an atom and co-atom.

5 Summary and Future Work

This paper has described a technique for the incremental construction of the Hasse diagram of a poset. We hypothesise that user interaction with this evolving diagram will expedite user comprehension of the partial order. When applied to FCA, this technique incrementally updates the transitive reduction, and the

labelled Hasse diagram, of the partial order amongst the set of concepts generated so far. User interaction with the Hasse diagram prioritises the generation of missing concepts relevant to the user's selection. A prototype implementation, including the modification of a concept enumeration algorithm to respond to downset prioritisation, has also been described through which the hypothesis will be tested in future work. Scalable mechanisms for interactive concept deletion should also be explored to allow the exclusion from the interim partial order of concepts which are no longer of interest to the user.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag New York, Inc. (1997)
2. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.: Graph Drawing. Prentice Hall, Upper Saddle River, NJ (1999)
3. Kuznetsov, S.O., Obiedkov, S.A.: Algorithms for the construction of concept lattices and their diagram graphs. In: PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, Springer (2001) 289–300
4. Kuznetsov, S., Obiedkov, S.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental and Theoretical Artificial Intelligence **14**(2-3) (2002) 189–216
5. Priss, U.: Formal Concept Analysis in Information Science. Annual Review of Information Science and Technology **40** (2006) 521–543
6. Strok, F., Neznanov, A.: Comparing and analyzing the computational complexity of FCA algorithms. In: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. SAICSIT '10, New York, NY, USA, ACM (2010) 417–420
7. Thomas, J.J., Cook, K.A., eds.: Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE Press (2005)
8. Ceglar, A., Pattison, T.: Guided Formal Concept Analysis. Technical report, Defence Science and Technology Organisation (2014) To appear.
9. Norris, E.: An algorithm for computing the maximal rectangles in a binary relation. Revue Roumaine de Mathématiques Pures et Appliquées **23**(2) (1978) 243–250
10. Missikoff, M., Scholl, M.: An algorithm for insertion into a lattice: Application to type classification. In Litwin, W., Schek, H.J., eds.: Foundations of Data Organization and Algorithms. Volume 367 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1989) 64–82
11. Godin, R., Missaoui, R., Alaoui, H.: Incremental concept formation algorithms based on Galois (concept) lattices. Computational Intelligence **11**(2) (1995) 246–267
12. Aho, A., Garey, M., Ullman, J.: The transitive reduction of a directed graph. SIAM Journal on Computing **1**(2) (1972) 131–137
13. Mäkinen, E., Siirtola, H.: The barycenter heuristic and the reorderable matrix. Informatica **29** (2005) 357–363
14. Choi, V., Huang, Y.: Faster algorithms for constructing a Galois lattice, enumerating all maximal bipartite cliques and closed frequent sets. In: SIAM Conference on Discrete Mathematics, University of Victoria, Canada. (2006)
15. Sedgewick, R.: Algorithms. second edn. Computer Science. Addison-Wesley (1988)