A lattice-based method for structural analysis *

Gemma C. Garriga, Pablo Díaz-López, and José L. Balcázar

Departament de Llenguatges i Sistemes Informàtics
Laboratori d'Algorísmica Relacional, Complexitat i Aprenentatge
Universitat Politècnica de Catalunya, Barcelona
{gcasas,pdiaz,balqui}@lsi.upc.edu

Abstract. In this paper we revisit the foundations of formal concept analysis for ordered contexts of [4]. From the theoretical point of view, the obtained lattice has proved to be a proper unifying framework for reasoning about different sequential mining tasks: from the discovery of partial orders to the clustering of input sequences. Here we will show how these results on sequences can be naturally extended to the mining of any partial order structure. We empirically validate the approach by testing it on real world data. Our experimental evaluation shows that this lattice-based method is an intuitive tool for analyzing acyclic structures.

1 Introduction

Typically, research on pattern discovery has progressed from mining itemsets to mining sequences [1], and from sequences to mining other structures such as trees, lattices or graphs, e.g. [9, 10, 13, 18] among others. Indeed, the most basic structure that data can exhibit is the sequential order, and this has become a very popular topic with many applications in different real-world problems, such as web mining, DNA sequence analysis, intrusion detection and so on. Since this sequential mining task was first introduced most of the current work has focused on providing efficient algorithms for their analysis: from the mining of closed frequent sequential patterns [14, 15, 17], to the identification other complex structures such as partial orders [2, 11, 12], or also, methods for clustering the input sequences into meaningful subclasses [8].

Previous works show that we can unify all these sequential mining tasks under a natural lattice framework provided by formal concept analysis [5, 6]. This is a methodology for data analysis and knowledge representation that has potential to be applied to a variety of fields; as an important example, it is worth mentioning the theoretical foundations given in [16] for the binary data. In this paper we revisit the lattice theory for the sequential context and use it for reasoning about the partial order discovery task and the clustering of input sequences. The goals of the paper are twofold: first, we show that these previous

* This work is supported in part by MCYT TIC 2002-04019-C03-01 (MOISES) and by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

results on sequences can be naturally extended to the mining of any partial order structure; and second, we test the usability of our mathematical foundations by applying the method to student enrollment data and web browsing data.

This work is organized as follows: next section introduces the preliminaries of the sequential data mining problem and gives the formalization of the lattice theory within sequences. Section 3 analyzes the construction of such lattice in practice, and section 4 studies the combinatorial properties of this object. Section 5 extends the results on sequences to the mining of partial orders. Finally, section 6 presents the experimental evaluation of the approach.

2 Preliminaries

Let $\mathcal{I} = \{i_1, \ldots, i_n\}$ be a fixed set of items. A subset $I \subseteq \mathcal{I}$ is called an itemset. For the moment, we deal with sequential categorical data, described as a collection of ordered transactions $\mathcal{D} = \{d_1, d_2, \ldots d_n\}$, where each d_i is a sequence. The sequences $d_i \in \mathcal{D}$ will be called input sequences or transactions. A *sequence* is an ordered list of itemsets $s = \langle (I_1)(I_2)\ldots(I_n)\rangle$, meaning that itemset I_i occurs before itemset I_j if $i < j$. An example of such representation is shown in figure 1; e.g. in the first input sequence, items A and E occur simultaneously, but always before items C, D and A.

Seq id	Input sequences
d_1	$\langle (AE)(C)(D)(A)\rangle$
d_2	$\langle (D)(ABE)(F)(BCD)\rangle$
d_3	$\langle (D)(A)(B)(F)\rangle$

Fig. 1. Example of sequential data $\mathcal{D}$.

Sequence $s = \langle (I_1)\ldots(I_n)\rangle$ is a *subsequence* of sequence $s' = \langle (I'_1)\ldots(I'_m)\rangle$ if there exist integers $j_1 < j_2 \cdots < j_n$ such that $I_1 \subseteq I'_{j_1}, \ldots, I_n \subseteq I'_{j_n}$. We note this case by $s \subseteq s'$. E.g., $\langle (D)(A)(B)\rangle$ is a subsequence of the second and third input sequences of figure 1. The *transaction identifier list* of a sequence s in $\mathcal{D}$ is the list of input sequence identifiers from $\mathcal{D}$ where s is contained. The *support* of a sequence is the number of input sequences where it is contained; a sequence is frequent if its support is over a minimum user-specified threshold. The *intersection* of a set of sequences $s_1, \ldots, s_n$ is the set of maximal subsequences contained in all the s_i. Note that the intersection of a set of sequences is not necessarily a single sequence, for example, the intersection of d_1 and d_2 of figure 1 is the set of sequences $\{\langle (AE)(C)\rangle, \langle (AE)(D)\rangle, \langle (D)(A)\rangle\}$: all of them are contained in d_1 and d_2, and among those having this property they are maximal.

2.1 Lattice theory for sequences revisited

The first step to define the lattice-theoretic framework for a set of sequences $\mathcal{D}$ is to make available a proper closure operator. A closure operator on a lattice,

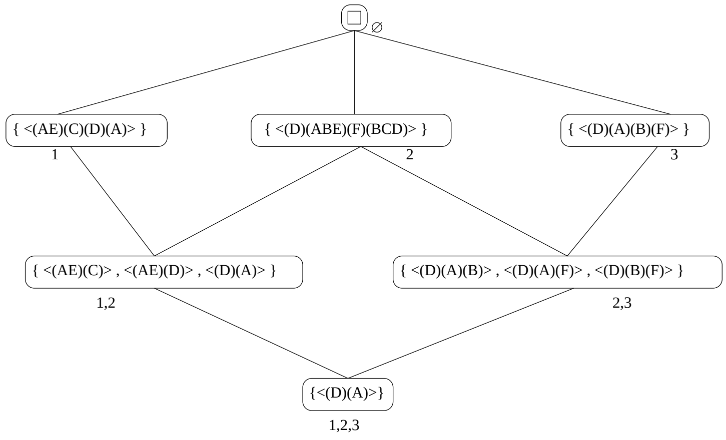


Fig. 2. Example of the closure system for data in figure 1. Each node in the lattice contains a closed set of sequences and a transaction list. This will be our lattice model.

such as the one formed for subsets of any fixed universe, is one that satisfies the three basic axioms: monotonicity, extensivity and idempotency. A way to construct such operator is by composition of two mappings forming a Galois connection (see e.g. [3]). In fact, we do have already our closure operator for sequences set in place through the Galois connection from [4]. Intuitively, the framework developed in [4] works with the universe of sets of sequences.

We will call $\varDelta$ this closure operator from [4] working on sets of sequences. It operates as follows: given $\mathcal{D}$, the closure of a set of sequences S, i.e. $\varDelta(S)$, includes all the maximal sequences that are present in all the transactions $d \in \mathcal{D}$ where sequences in S are contained; for example, for data from figure 1 we have that $\varDelta(\{\langle(A)(B)\rangle, \langle(B)(F)\rangle\}) = \{\langle(D)(A)(B)\rangle, \langle(D)(A)(F)\rangle, \langle(D)(B)(F)\rangle\}$. Another simple way to explain $\varDelta$ is to realize that $\varDelta(S)$ returns the intersection of those transactions from $\mathcal{D}$ where sequences in S are contained simultaneously. In last example, $\varDelta(\{\langle(A)(B)\rangle, \langle(B)(F)\rangle\})$ returns the intersection of d_2 and d_3 from $\mathcal{D}$.

By definition, a set of sequences S is *closed* if and only if it coincides with its closure, i.e. $\varDelta(S) = S$. E.g., the set of sequences $\varDelta(\{\langle(D)(A)\rangle\}) = \{\langle(D)(A)\rangle\}$ is closed for data from figure 1. Note that the closure operator completes and extends a set of sequences with respect to $\mathcal{D}$; so, it is always true that the intersection of any subset of input sequences will be a closed set. Indeed, it is easy to verify that all sequences in a closed set are maximal w.r.t. $\subseteq$. From here, we also have that each one of the individual sequences in the closed sets correspond to a "closed sequential pattern" of [15] (this term denotes those patterns not extendable to others with the same support, i.e. s is a "closed sequential pattern" in $\mathcal{D}$ if there exists no sequence s' s.t. $s \subset s'$ with $supp(s) = supp(s')$; e.g. for data in figure 1, we have that $\langle(A)\rangle$ is not a "closed sequential pattern", since it can be extended to $\langle(D)(A)\rangle$ that has the same support).

This characterization of $\varDelta$ gives immediately a closure system for the sequential data: it is a Hasse diagram where each node is a closed set of sequences S

(i.e. each node corresponds to $\Delta(S) = S$), along with a list of transaction identifiers where all those sequences in S are contained simultaneously. Indeed, these lists of transaction identifiers in a node form the dual view of the lattice, which is not of our interest for this current contribution. A closed set of sequences S is *predecessor* of another S' in the lattice if $S \preceq S'$: i.e. for all $s \in S$ there exists $s' \in S'$ s.t. $s \subseteq s'$. In figure 2 we depict the lattice model for data in figure 1. An artificial top not belonging to any input sequence is added to the lattice, noted here by the unsatisfiable boolean constant $\square$.

3 Lattice construction

This lattice representation is a useful visual tool to explore the properties of our data. Indeed, this lattice system can be constructed by conveniently organizing the "closed sequential patterns" of [15] into nodes, i.e. into sets of sequences partially ordered by $\preceq$. To get such a organization, we just need to observe that each closed set of sequences in a node contains those "closed sequential patterns" with the same tid list, plus possibly other closed patterns that make the final groupings to be partially ordered by $\preceq$ (see the algorithmic scheme proposed in [5]). Yet, to guarantee that our lattice is a closure system, after this organization of "closed sequential patterns" we need to check a basic property of lattice theory: any two nodes must have one single supremum, i.e., that by following the ascending paths of two different nodes we will reach one single common node being their top. Forcing the fulfilment of this condition, we still may need to add some extra nodes to ensure the consistency of the system. Interestingly, the lattice theory guarantees that these extra nodes, added as the "artificial" supremum of two closed sets, will consist on the union of the maximal sequences contained by their immediate predecessors, so that here is no need to scan the input sequences $\mathcal{D}$ for completing the lattice. In some cases, as showed in [5], not having a consistent closure system still provides a good summarization of the data; but in the experimental section of this paper we will show that these extra nodes are able to better explain general tendencies of the data. Note that in practice, a minimum support condition is imposed on the mined "closed patterns", and the lattice is not completely constructed up to the top, but just the nodes with sequences over a minimum support threshold. Improvements of this algorithmic approach will be addressed in the future work.

4 Combinatorial properties of the lattice

This described lattice can be used to formalize some of the classical tasks of the analysis of sequences. First, we will show how to capture the task of describing the input sequences with local ordering relationships, that is, partial orders occurring in the data. This approach was initially proposed by [12] and other interesting works followed afterwards (e.g. [2, 11]). A partial order (poset) is a directed acyclic graph where the relation between nodes is reflexive, antisymmetric and transitive; moreover, there is a labelling function assigning names

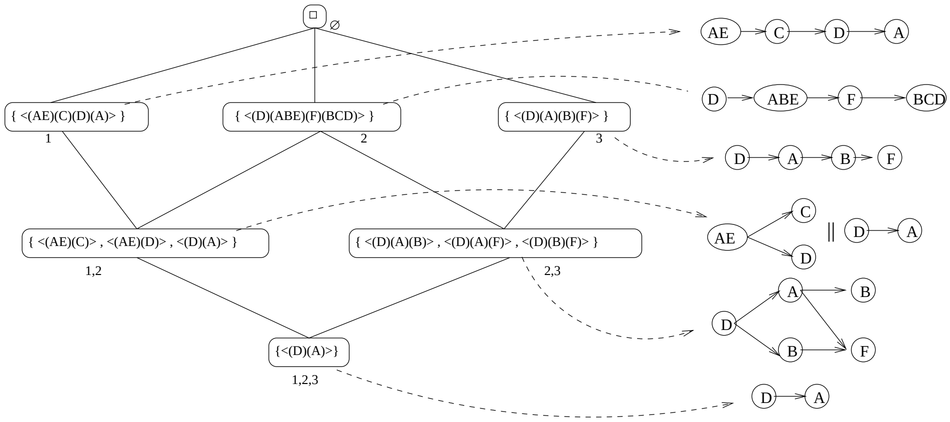


Fig. 3. Generation of the most specific partial orders for data in figure 1.

(items) to each one of the nodes. The graphical representation of a partial order is particularly useful for displaying results: we will display the poset by using arrows between the connected labelled vertices, and the symbol $\|$ (parallel) to indicate trivial order among the different components of a partial order.

Broadly speaking, we say that a partial order is *compatible* with a set of input sequences if the labels of the nodes occur in each one of the input sequences in the corresponding order given by the edges. The support of a partial order is the number of input sequences that the poset is compatible with, and it is frequent when its support is over a user-specified value. A poset p is said to be more *specific* than another poset p' when it is possible to transform p into p' by removing edges or vertices (formally, we are considering here that all the edges coming from the transitive closure are present in the graph representing the poset structure). This can be formalized also by defining an label-preserving function that maps edges and vertices of p' to edges and vertices of p respectively.

The goal now is to describe the input sequential data as a set of the most specific frequent partial orders. An interesting kickoff point to address this task is to see individual sequences in each node as total orders (i.e., serial episodes of [12]) occurring always together in the same set of input transactions. Because we deal with closures, we can overlap these different total orders of the same node and get the most specific poset compatible with the input sequences identified by the enclosed list of transactions. Formally, we have the following property (the original statement differs from this one, but their equivalence is readily seen):

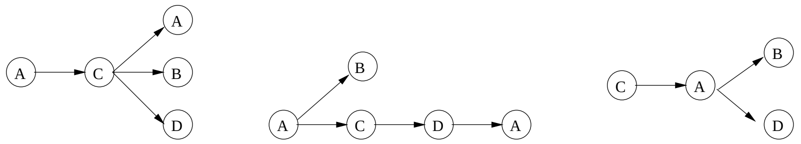
Theorem 1. *[6] Sequences in a closed set define the maximal paths of the most specific partial order compatible with the input sequences identified by the list of transactions of the node.*

We also have that this transformation procedure from the lattice is complete (see [6]). Figure 3 illustrates the theorem: each node is transformed into a partial order, which will be the most specific one compatible with the transaction identifier list of the same node.

From a different perspective, the lattice model can be also seen as a natural way of clustering input sequences. Former work in [8] considers as discriminating features to decide the final clusters, all the sequential patterns over a certain support with a certain restricted length. There, the critical step is to project the new incoming sequences into the new feature space, so that the space must be restricted to only a reduced set of features. One way of addressing this problem is to prune the number of discovered sequential patterns by selecting only a subset of them prior to projection. Taking the nodes of our lattice, we can consider the closed set of sequences of each node as our set of discriminating features for a class, and the dual list of identifiers as the corresponding input objects projected in that class. Then, each concept represents a subclass of projected objects. Generally speaking, the lattice will exhibit the tendencies of our data from bottom to top: nodes located in the bottom part of the lattice correspond to concepts with a higher support, thus the semantic split performed between the concepts of the same level is more significant; on the other hand, the nodes located in the top of the lattice represent the unifying concepts of the different tendencies coming from the bottom. Comparing this approach to [8], we have here that the final obtained clusters are organized in a lattice structure, so that the projection of new incoming input sequences into this space can be done by simply traversing those ascending paths where the new object can be included (similar in spirit to the incremental concept formation for binary data [7]).

5 Mining acyclic structured data using the lattice

Next, we consider the case of mining data without cycles, that is having a set of trees, lattices, or general acyclic directed graphs. Indeed, any of these objects can be formalized as a partial order itself, and we use this term to generalize any acyclic directed graph. An example of three partial orders is shown in figure 4. The mining task now is to identify those common substructures in $\mathcal{D}$ over a certain user-specified threshold.



(a) Poset p_1 (b) Poset p_2 (c) Poset p_3

Fig. 4. An example of three input partial orders in our database.

The algorithmic advantage of dealing with partial orders as our input data is that we do not need to mine directly these objects. Indeed, as suggested after theorem 1, each partial order can be represented by a set of sequences corresponding to its maximal paths. Hence, to deal with this mining problem, we propose to consider each partial order in our data as its set of maximal paths, i.e. each poset p in the database will be transformed into a set of sequences $S = \{s_1, s_2, \ldots, s_n\}$ where each s_i is a sequence of labels defining a maximal path of the partial order. E.g., the first object of data in figure 4 corresponds to the set of sequences $S_1 = \{\langle(A)(C)(B)\rangle, \langle(A)(C)(D)\rangle, \langle(A)(C)(A)\rangle\}$. An example for the transformation applied to the three objects in figure 4 can be followed from the table in figure 5. The transformed data is noted by $\mathcal{D}$. Note that the correspondence of a partial order with their paths is not necessarily unique. This does not represent a problem for our formalizations, and indeed, just a nonredundant set of maximal sequences suffice to profit the theoretical results stemming from the lattice.

Poset Id	**Sets of Maximal Paths**
S_1	$\{\langle(A)(C)(B)\rangle, \langle(A)(C)(D)\rangle, \langle(A)(C)(A)\rangle\}$
S_2	$\{\langle(A)(B)\rangle, \langle(A)(C)(D)(A)\rangle\}$
S_3	$\{\langle(C)(A)(B)\rangle, \langle(C)(A)(D)\rangle\}$

Fig. 5. Example of a transformed structured database $\mathcal{D}$, corresponding each transaction to a partial order in figure 4.

The new problem now corresponds to mining a set of sets of sequences, that is the sets of maximal paths. We say that a sequence s' *matches* with the set of sequences $S \in \mathcal{D}$ if there exists at least one sequence $s \in S$ s.t. $s' \subseteq s$. Then, we also say that the *support* of a sequential pattern s is the number of $S \in \mathcal{D}$ where s matches. The mining problem can be described as that of finding sequences s whose support is over a minimum threshold in the new transformed data $\mathcal{D}$. Correspondingly to the sequential mining problem, some of these frequent sequences in the transformed data $\mathcal{D}$ may be redundant, so that we may restrict our search to the "closed" ones, that is, we will parallel the same definition of the "closed sequential patterns" of CloSpan [15]. The only semantic difference w.r.t. CloSpan is simply the notion of support of the sequences.

Interestingly enough, reducing the partial order mining problem to sequences allows to apply all the results obtained with sequences. Once the newly defined "closed sequential patterns" are found, they can be organized in a lattice, as in section 3; from here, all the combinatorial properties showed in section 4. A follow up of the lattice for data in figure 5 is shown in figure 6. Note that the nodes located just below the top of the lattice generate exactly the same partial orders corresponding directly to our input data in figure 4, that is, corresponding to the most specific structures with support exactly 1.

6 Experimental evaluation

We implemented this lattice-based method in C++ using STL libraries. For the visualization of the results, such as the final partial orders and also the lattice,

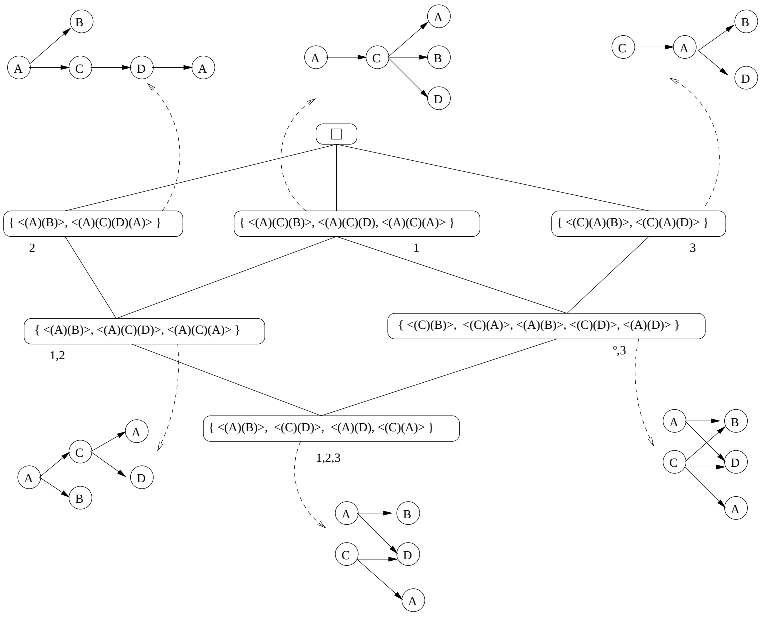


Fig. 6. Lattice of sets of sequences for data in figure 5, and the derivation of partial order structures from each one of the nodes.

our command line program outputs the necessary files in a standard GraphML format [1]. These files can be visualized through any tool for graph visualizations, such as Yed-Java Graph Editor [2]. Currently, we simply run our system in three phases: (1) identifying the frequent "closed sequential patterns" from the transformed data; (2) constructing the lattice by post-processing frequent "closed sequential patterns" and ensuring the closure system; and finally (3) visualizing the results. So, in the practice, we are constructing just the part of the lattice bounded by the "closed sequential patterns" over a minimum support; if this threshold was set to zero, then, the full lattice would be depicted. The first phase is the most I/O expensive, since it requires to examine a combinatorial number of patterns. The second phase only requires standard operations of sets, such as inclusion or intersection, so it is not time consuming; however, when dealing with very large datasets the lists of object identifiers to be compared are long enough to make these operations more expensive.

We evaluate our approach by performing experiments on a data set that describes the partial order in which about 12000 students of the Faculty of Informatics (FIB), at the Universitat Politècnica de Catalunya, registered to the courses during the last ten years. We run our three-step method for this data by imposing a minimum support threshold of 30%. Obtaining the final

[1] See http://graphml.graphdrawing.org/

[2] See http://www.yworks.com/en/products_yfiles_about.htm

lattice with this support takes about five minutes; of course, with larger datasets the algorithmic efficiency would tend to degrade (below, we point out a way to address this situation). At a glance, we observe important combinations of partial orders such as the one presented in figure 7. These partial orders show that the introductory courses corresponding to the two first semesters exhibit a trivial order among them, but they occur in the data before the courses corresponding to the third and fourth semester. This behaviour can be explained by filtering system that the university imposes between the first and second year: students are forced to succeed in all the introductory courses before stepping forward the second-year courses. On the other hand, the trivial order exhibited among the courses of the first and second semester, is justified by the high rate of failure that we have during the first year. So, eventually students end up mixing courses of these two first semesters and breaking any pre-specified order.

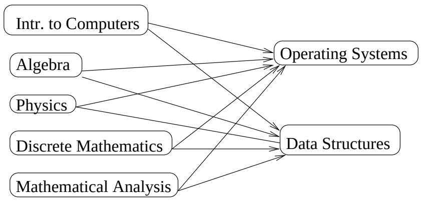
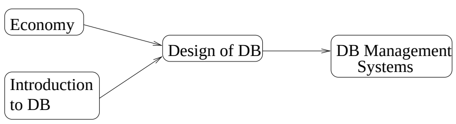
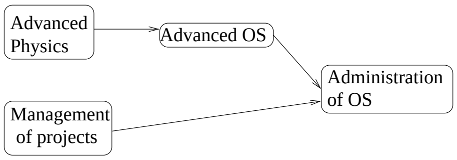


Fig. 7. A partial order derived from student enrollment data.

By generating the concept lattice with the optional courses only (i.e. those chosen by the students, where we expect a lower failure rate), we observe that the lattice structure shows three clear tendencies separating students into three main groups: a group of students interested in databases, another group interested in operating systems, and finally, a group interested in graphics. Of course there are other students in our data that took a wide variety of courses and that mixed other different disciplines; yet, these students would get projected in the top part of the lattice, thus with a support not high enough in the data to be considered significant for the frequent pattern mining algorithm of the first phase. Figure 8 shows two partial orders for two of these tendencies given by the lattice.



(a) Database courses



(b) Operating systems courses

Fig. 8. Partial orders derived from the student enrollment data.

In general, the orders reflect the curriculum imposed by the university: e.g. students cannot register to “DB Management” until they do not pass “DB Design”. However, we also observe that some subjects, such as “Economy” or “Management of Projects”, go beyond this preestablished order imposed by the curriculum. Both “Economy” and “Management of Projects” belong to the discipline of management and business, so, not related to databases or operating systems; however, the partial orders indicate that these are good complementary courses for any discipline. Another interesting observation stemming from the lattice corresponds to those students working for the master thesis (i.e. they are studying for a 5-year degree, in comparison to those other students that go for a 3-year degree), mainly belong to the group of students choosing databases as a discipline. This can be detected by including the master thesis as a subject in the database, and checking the partial orders containing it, e.g. as figure 9.

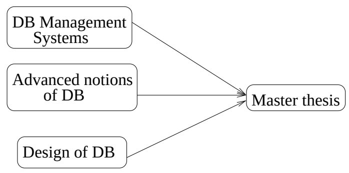


Fig. 9. A partial order derived from student enrollment data.

Finally, we also decided to keep track of those nodes that make the system stable under intersection, that is, those extra nodes that the algorithm added to ensure a closure system. We find many practical cases where those nodes summarize perfectly a tendency that would not be visualized by the immediate predecessors in the lattice. For example, the partial order from figure 9 stems from the union of three immediate predecessors in the lattice containing a total order each. This part of the lattice is depicted in figure 10. The unifying node from where we derive the final order, contains new information not included individually in the predecessors.

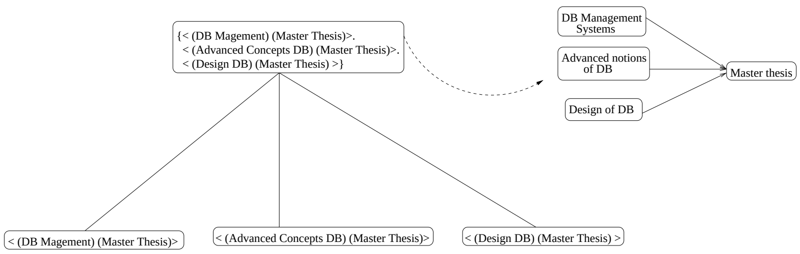


Fig. 10. A portion of the lattice for the student enrollment data.

To complete our evaluation, we also performed experiments with the msnbc.com anonymous web data provided UCI KDD Archive. This data describes the page visits of users at msnbc.com in a sequential order: each sequence in the dataset corresponds to the age views of a user during a twenty-four hour period, and each event in the sequence corresponds to a user's request for a page. Requests are recorded by category, such as “frontpage”, “news”, “tech” ... We have a total of 989818 users recorded in the data. In this lattice, we initially observe many recurrent partial orders with “frontpage” and “news” as a main events, that is, posets containing always the same event or the combinations of these two events. Broadly, the recurring posets reflect the revisiting of the front page and the news continuously during the twenty-four hour period of the same session: e.g. users would visit news page to check the last updates, or would unavoidably pass by the front page before clicking into a new category. Interestingly enough, one of the most significant partial orders that we find in the lattice is depicted in figure 11. It is supported by 80500 users and it suggests that those users visit almost all the categories during their session: they always starting from the frontpage and then, there exists a clear split between going to see the “news” or returning to the “frontpage” again to click into a new page.

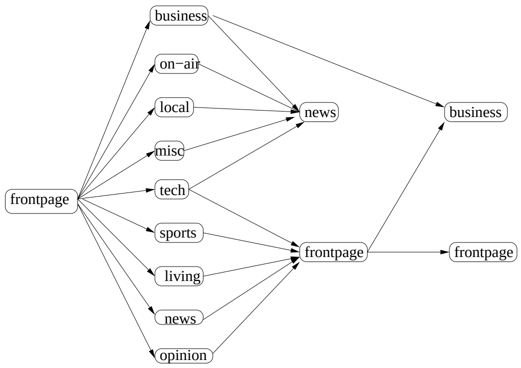


Fig. 11. A partial order derived from the msnbc.com anonymous web data.

7 Conclusions and further research

We revisited the foundations of lattice theory on sequences to prove that the characterized lattice exhibits interesting combinatorial properties useful for the analysis of sequences. We show that these results can be extended to the mining of acyclic data, represented as partial orders. Here we chose to construct this lattice by postprocessing the “closed sequential patterns” derived by current algorithms. We tested this approach with student enrollment data and web browsing data. Our experiments show that the lattice-based method not only combines sound mathematical foundations, but it also generates very attractive results that summarize perfectly the analyzed data. New open lines related to this framework remain to be studied. The most important one corresponds to semantic issues: how the obtained partial orders semantically compare to former works of mining trees and graphs (e.g. [9, 10, 13, 18]). Another issue corresponds to the efficiency of the lattice construction: instead of post-processing "closed sequential patterns", it would be more desirable to incorporate this procedure directly in the mining algorithms.

References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th Int. Conference on Data Engineering*, pages 3–14, 1995.
2. M.J. Atallah, R. Gwadera, and W. Szpankowski. Detection of significant sets of episodes in event sequences. In *Proceedings of the 4th International Conference on Data Mining*, pages 3–10, 2004.
3. B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations.* Springer, 1998.
4. G.C. Garriga. Towards a formal framework for mining general patterns from structured data. In *2nd Int. KDD Workshop on MRDM'03*, pages 215–229, 2003.
5. G.C. Garriga. Summarizing sequential data with closed partial orders. In *Proceedings of the SIAM Int. Conference on Data Mining*, pages 380–391, 2005.
6. G.C. Garriga and J.L. Balcázar. Coproduct transformations on lattices of closed partial orders. In *Proceedings of 2nd. Int. Conference on Graph Transformation*, pages 336–351, 2004.
7. R. Godin and R. Missaoui. An incremental concept formation approach for learning from databasese. *Theoretical Computer Science*, 133:387–419, 1994.
8. V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *Proceedings of the 1st. Int. Conference on Data Mining*, pages 179 – 186, 2001.
9. A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Mach. Learn.*, 50(3):321–354, 2003.
10. M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. In *SIAM International Conference on Data Mining (SDM'04)*, 2004.
11. H. Mannila and C. Meek. Global partial orders from sequential data. In *Proceedings of the 6th Int. Conf. on Knowledge Discovery in Databases*, pages 161–168, 2000.
12. H. Mannila, H. Toivonen, and A.I. Verkamo. Discovering frequent episodes in sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.
13. S. Nijssen and J.N. Kok. Efficient discovery of frequent unordered trees. In *First Int. Workshop on Mining Graphs, Trees and Sequences*, 2003.
14. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: mining sequential patterns by prefixprojected growth. In *Proceedings of the 17th Int. Conference on Data Engineering*, pages 215–224, 2001.
15. X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of the Int. Conference SIAM Data Mining*, pages 166–177, 2003.
16. M. Zaki. Generating non-redundant association rules. In *Proceedings of the 6th Int. Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2000.
17. M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal, special issue on Unsupervised Learning*, 42(1/2):31–60, 2001.
18. M. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2002.