Coproduct Transformations on Lattices of Closed Partial Orders ⋆

Gemma C. Garriga and José L. Balcázar

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
{gcasas,balqui}@lsi.upc.es

Abstract. In the field of Knowledge Discovery, graphs of concepts are an expressive and versatile modeling technique that provides ways to reason about information implicit in the data. Typically, nodes of these graphs represent unstructured closed patterns, such as sets of items, and edges represent the relationships of specificity among them. In this paper we want to consider the case where data keeps an order, and nodes of the concept graph represent complex structured patterns. We contribute by first characterizing a lattice of closed partial orders that precisely summarizes the original ordered data; and second, we show that this lattice can be obtained via coproduct transformations on a simpler graph of so-called stable sequences. In the practice, this graph transformation implies that algorithms for mining plain sequences can efficiently transform the discovered patterns into a lattice of closed partial orders, and so, avoiding the complexity of the mining operation for the partial orders directly from the data.

1 Introduction

Formal Concept Analysis, mainly developed by [5], is based on the mathematical theory of complete lattices (e.g. [4]). This area has been used in a large variety of fields in computer science, such as in Knowledge Discovery where graphs of concepts are an expressive modeling technique to show structural relations implicit in the given set of data ([9, 10, 13]).

The two basic notions of Formal Concept Analysis are those of *formal context* and *formal concept*. In the main case of interest for Knowledge Discovery, a formal context consists of a binary relation R that can be regarded as a set of objects associated with a set of items (attributes holding in each object), that is, $R \subseteq \mathcal{O} \times \mathcal{I}$. On the other hand, a formal concept is a pair of a closed set of objects and a closed set of items linked by a Galois connection. To characterize a formal concept we need to define appropiate closure operators on the universe of items and objects respectively.

⋆ This work is supported in part by MCYT TIC 2002-04019-C03-01 (MOISES)

A closure operator Γ on a lattice, such as the one formed by the subsets of any fixed universe, satisfies the three basic closure axioms: monotonicity, extensivity and idempotency. It follows from these properties that the intersection of closed sets, which are those sets that coincide with their closure, is also another closed set. One way of constructing closure operators is by composition of two derivation operators forming a Galois connection [5]. The standard Galois connection for a binary database R maps each family of objects to the set of the items that hold in all of them, and each set of items to the set of objects in which they hold. Then, the resulting closure operator Γ acts as follows: given R, the closure $\Gamma(Z)$ of a set of items $Z \subseteq \mathcal{I}$ includes all items that are present in all objects having all items in Z. The closed sets obtained are exactly the closed sets employed in closed set mining (see e.g. [9]), which in certain applications presents many advantages (see e.g. [3, 9, 10, 13]). Similarly, we can consider the dual operator Γ^{-1} operating on the universe of the set of objects $\mathcal{O}$ and giving rise to closed set of objects, that is, $\Gamma^{-1}(O) = O$ where $O \subseteq \mathcal{O}$. For any binary database R, the closure systems of Γ and Γ^{-1} are isomorphic.

The closed sets found in the database R can be graphically organized in a hierarchical order, called the concept lattice, i.e. a graph where each node is a pair (O, Z) formed by the closed itemset $\Gamma(Z) = Z$ and the maximal set of objects O where Z is contained (symetrically, for $\Gamma^{-1}(O) = O$); so, each node is labelled by the pair of closed sets that is joined by the Galois connection. This concept lattice provides a comprehensive graphical representation that shows the structural relations between the concepts and summarizes, at the same time, all the characteristics of the binary data.

In this paper we want to analyze the case where R is *not* a binary relation, but the items keep an order in each one of the objects where they hold. Here we study the construction of a lattice where formal concepts model more complex structures, such as partial orders. Our contribution is to prove that this closure system of partial orders can be obtained by coproduct transformations on a simpler graph of so-called stable sequences studied recently ([6, 11, 12]). Algorithmically, this transformation avoids the computation of partial orders directly from the data.

2 Preliminaries

Let $\mathcal{I} = \{i_1, \ldots, i_n\}$ be a finite set of items. *Sequences* are ordered lists of itemsets where we assume that no item occurs more than once in a sequence. The input data we are considering consists of a database of ordered transactions that we model as a set of sequences, $\mathcal{D} = \{s_1, s_2, \ldots s_n\}$. Our notation for the component itemsets of a given sequence will be $s = \langle (I_1)(I_2) \ldots (I_n) \rangle$, where each $I_i \subseteq \mathcal{I}$ and I_i occurs before itemset I_j if $i < j$. Note that each I_i may contain several items that occur simultaneously; e.g. $\langle (AC)(B) \rangle$, where items A and C are given at the same time but always before item B. However, in order to simplify the definitions of the next theoretical discussion we consider for the moment that each I_i contains only one single item. Later in the paper, we will show how the simultaneity condition can be incoroporated again into the theoretical framework. The set of *all* the possible sequences will be noted by $\mathcal{S}$.

From the point of view of Formal Concept Analysis, we can consider each input sequence as an object. Formally, data $\mathcal{D}$ can be transformed into an ordered context (see [6]) where *objects* of the context are sequences, *attributes* of the context are items, and the database becomes a ternary relation, subset of $\mathcal{O} \times \mathcal{I} \times \mathbb{N}$, in which each tuple $\langle o, i, t \rangle$ indicates that item i appears in the t-th element of the object o representing an input sequence s. A simple example of the described data and the associated context can be found in figure 1, where each object o_i of the formal context represents the corresponding input sequence, $s_i \in \mathcal{D}$. The context for a set of data $\mathcal{D}$ is only relevant to this paper to see objects $o_i \in \mathcal{O}$ and input sequences $s_i \in \mathcal{D}$ as equivalent, which eases the definition of the subsequent closure operator on ordered data.

Seq id	**Sequence**
s_1	$\langle (A)(B)(C)(D) \rangle$
s_2	$\langle (B)(C)(D)(A) \rangle$
s_3	$\langle (B)(C)(A)(D) \rangle$

(a) Collection of data $\mathcal{D}$

	A	**B**	**C**	**D**
o_1	1	2	3	4
o_2	4	1	2	3
o_3	3	1	2	4

(b) Context for $\mathcal{D}$

Fig. 1. Example of ordered data $\mathcal{D}$ and its context

Sequence $s = \langle (I_1) \dots (I_n) \rangle$ is a *subsequence* of another $s' = \langle (I'_1) \dots (I'_m) \rangle$ if there exist integers $j_1 < j_2 \dots < j_n$ such that $I_1 \subseteq I'_{j_1}, \dots, I_n \subseteq I'_{j_n}$. We note it by $s \subseteq s'$. For example, the sequence $\langle (A)(D) \rangle$ is a subsequence of the first and third input sequences in figure 1.

The *intersection* of a set of sequences $s_1, \dots, s_n \in \mathcal{S}$ is the set of *maximal* subsequences contained in all the s_i. Note that the intersection of a set of sequences, or even the intersection of two sequences, is not necessarily a single sequence. For example, the intersection of the two sequences $s = \langle (A)(C)(B) \rangle$ and $s' = \langle (A)(B)(C) \rangle$ is the set of sequences $\{\langle (A)(C) \rangle, \langle (A)(B) \rangle\}$: both are contained in s and s', and among those having this property they are maximal; all other common subsequences are not maximal since they can be extended to one of these. The maximality condition discards redundant information since the presence of, e.g., $\langle (A)(B) \rangle$ in the intersection already informs of the presence of each of the itemsets (A) and (B).

2.1 Stable Sequences and Closure Operators

A sequence s is *stable* in input data $\mathcal{D}$ if s is maximal in the set of objects where it is contained, that is, it cannot be extended. More formally, we say that:

Definition 1. *A sequence $s \in \mathcal{S}$ is stable if there exists no sequence s' with $s \subset s'$ s.t. they are both subsequences of the same set of objects (equivalently, input sequences).*

For instance, taking data from figure 1, sequence $\langle(B)(D)\rangle$ is not stable since it can be extended to $\langle(B)(C)(D)\rangle$ in all the objects where it belongs. However, $\langle(B)(C)(D)\rangle$ or $\langle(A)(D)\rangle$ are stable sequences in $\mathcal{D}$. The set of all stable sequences of data from figure 1 are shown in figure 2. The most relevant existing contributions on the mining of stable sequences are given by two algorithms, CloSpan [11] and BIDE [12], which find in a resonably efficient way all the stable sequences for the input dataset $\mathcal{D}$. Stable sequences are called "closed" there; we prefer a different term to avoid confusion with the closure operator. Note however, that these mentioned algorithms do not impose our condition of avoiding repetition of items in the input sequences.

Objects	**Stable Sequences**
1,2,3	$\langle(B)(C)(D)\rangle$
1,2,3	$\langle(A)\rangle$
2,3	$\langle(B)(C)(A)\rangle$
1,3	$\langle(A)(D)\rangle$
1	$\langle(A)(B)(C)(D)\rangle$
2	$\langle(B)(C)(D)(A)\rangle$
3	$\langle(B)(C)(A)(D)\rangle$

Fig. 2. Set of stable sequences and objects where they belong

As it is formalized in [6], the set of stable sequences can be characterized in terms of a closure operator, named Δ, operating on the universe of sets of sequences. Briefly, the defined Galois connection maps each family of objects to the set of the maximal sequences that hold in all of them, and each set of sequences to the set of all objects in which they hold. It is proved there that these mappings indeed enjoy the properties of a Galois connection so that their composition provides the necessary closure operator. Again, a closed set of sequences are those coinciding with their closure, that is, $\Delta(\{s_1, \dots s_n\}) = \{s_1, \dots s_n\}$ where $\{s_1, \dots s_n\} \subseteq \mathcal{S}$; similarly, a closed set of objects in this ordered context is defined by the dual closure operator, i.e. $\Delta^{-1}(O) = O$. A main result in [6] is that stable sequences are exactly those that belong to a closed set of sequences.

As in any other Galois connections (see [5]), this characterization gives immediately a lattice of formal concepts, that is, a graph where each node is a pair $(O, \{s_1, \dots s_n\})$ where $\{s_1, \dots s_n\}$ are stable sequences belonging to the closed set of objects O, and viceversa. For instance, in data of figure 1 we have that the set of sequences $\{\langle(B)(C)(D)\rangle, \langle(A)(D)\rangle\}$ is stable for the first and third objects; reciprocally, the set of objects formed by first and third transaction is closed for the set of stable sequences $\{\langle(B)(C)(D)\rangle, \langle(A)(D)\rangle\}$. So, $(\{o_1, o_3\}, \{\langle(B)(C)(D)\rangle, \langle(A)(D)\rangle\})$ will be a formal concept of the lattice. It is proved also in [6] that all the stable sequences mined by CloSpan or BIDE can be organized in different formal concepts of the same lattice, and this graph of stable sequences characterizes the non-redundant sequential patterns of the ordered data.

Interestingly enough for our subsequent contribution, the work in [6] also proves that if we rather deal with input sequences more than with objects, then the set of stable sequences of a formal concept is exactly the intersection of those input sequences renamed after the objects of the same concept. That is, renaming each object by its input sequence in $\mathcal{D}$, then a formal concept $(O, \{s_1, \ldots s_n\})$ becomes $(S, \{s_1, \ldots s_n\})$ where S are the input sequences of O, and we always have the following property: $\{s_1, \ldots s_n\} = \bigcap_{s \in S} s$.

3 Partial Orders on Sequences

Formally, we will model partial orders as a full subcategory of the set of directed graphs; so, as a starting point, we recall the most basic among the numerous definitions of graphs given in the literature.

Definition 2. *A directed graph is modeled as a triple $G = (V, E, l)$ where V is the set of vertices; $E \subseteq V \times V$ is the set of edges; and l is the injective labelling function mapping each vertex to an item, i.e. $l : V \to \mathcal{I}$.*

The set $\mathcal{I}$ in the labelling function is exactly the finite set of items defined in the preliminaries; this will be a fixed set in all the graphs belonging to our category. For our present work we consider that the labelling function l of a graph is *injective* (and not necessarily surjective). An edge between two vertices u and v will be denoted by $(u, v) \in E$, implying a direction on the edge from vertice u to v.

Definition 3. *A graph morphism $h : G \to G'$ between two graphs $G = (V, E, l)$ and $G' = (V', E', l')$ consists of an* injective *function $h_V : V \to V'$ that preserves labels (that is, $l' \circ h_V = l$), and $(u, v) \in E \Rightarrow (h(u), h(v)) \in E'$.*

Note here that the injectivity of the morphism h between any two graphs, whose labelling function must be also injective, forces the morphism h to be unique. So, if there are $h : G \to G'$ and $g : G' \to G$, it implies $h = g$ and $G = G'$. The composition of $h : G \to G'$ with a morphism $g : G' \to G''$ is the morphism $g \circ h : G \to G''$ consisting of the composed function $g_V \circ h_V$. It is well known that the good properties of graph morphisms turn the set of graphs into a category. From the category of the set of graphs, we will be specially interested in the following constructor operator.

Definition 4 (Coproduct). *The coproduct of a family of graphs* $\{G_i\}$ *is a graph* $\widehat{G} = \coprod G_i$ *in the same category and a set of morphisms* $\{h_i : G_i \rightarrow \widehat{G}\}$ *such that, for every graph* G' *and every family of morphisms* $\{h'_i : G_i \rightarrow G'\}$*, there is an unique morphism* $g : \widehat{G} \rightarrow G'$ *such that* $g \circ h_i = h'_i$*.*

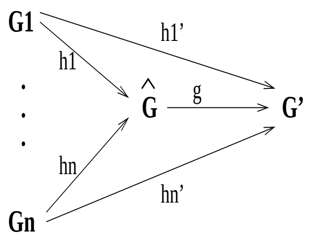

Fig. 3. Coproduct diagram

In category-theoretic terms, the result $\widehat{G}$ of a coproduct is the initial object among all those candidates G'. Moreover, we know that this $\widehat{G}$ is unique in any coproduct since the morphisms $\{h_i\}$, $\{h'_i\}$ and g are injective and the family of graphs considered here have an injective labelling function. Therefore, the coproduct of two graphs G_1 and G_2 in our category defines exactly a union of G_1 and G_2 where vertices in G_1 and G_2 with the same label are joined, and where the injectivity of morphisms ensures that all edges from both graphs are preserved.

From the set of all directed graphs, we will be interested in the full subcategory that models partial orders. A *partial order* (also called poset) is an acyclic directed graph $G_p = (V, E, l)$ such that the relation on V stablished by edges in E is reflexive, antisymmetric and transitive. The *sources* of a poset are those vertices that do not have any predecessor; similarly, the *sinks* are those vertices not preceded by any other vertex in the poset. Note that a poset may have different unconnected components, and so, it may have several source nodes.

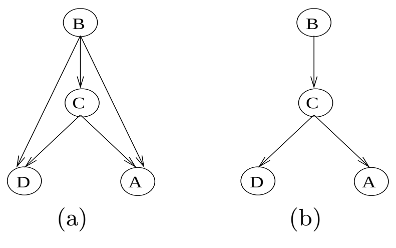

Fig. 4. Example of a partial order and its transitive reduction

The graphical representation of partial orders is particularly useful for displaying results: we will display a poset by using arrows between the connected labelled vertices, and the symbol $\|$ (parallel) to indicate trivial order among the different components of a poset. The *transitive reduction* of $G_p = (V, E, l)$ is the smallest relation resulting from deleting those edges in E that come from transitivity. Posets will be graphically depicted here by means of its transitive reduction to make them more understandable (as in figure 4(b)), but of course, all edges of the transitive closure are indeed included in E (figure 4(a)).

Some specific definitions we need on posets are the following ones.

Definition 5. *We define that a poset G_p is more general than another poset $G_{p'}$, noted by $G_p \trianglelefteq G_{p'}$, if there exists a morphism from G_p to $G_{p'}$, i.e. $h : G_p \rightarrow G_{p'}$. Then, we also say that $G_{p'}$ is more specific than G_p.*

For instance, the partial order represented in figure 4 is more specific than the trivial order $\| A, B, C, D \|$ (parallelization of all items), but more general than (the transitive closure of) the total order $B \rightarrow C \rightarrow D \rightarrow A$.

Definition 6. *A partial order $G_p = (V, E, l)$ is compatible with a sequence s if: $\forall u \in V$ we have that item $l(u)$ is in s; and, $\forall (u, v) \in E$ we have that $\langle (l(u))(l(v)) \rangle \subseteq s$.*

In other words, a poset G_p is compatible with a sequence s if there exists a morphism from G_p to the poset obtained as the transitive closure of the sequence s. We will see a sequence as a partial order containing all the orders added by transitive closure, so we can express it for our convenience as $G_p \trianglelefteq s$. For instance, the partial order of figure 4 is compatible with the second and third input sequences of the data in example 1. The trivial order is compatible with any sequence having all the items of the poset; so, at least there will be one poset (the trivial one) compatible with a given set of sequences. In case that sequences in S do not have any item in common, then we assume the existence of an empty poset compatible with them.

Definition 7. *We define a path from a poset $G_p = (V, E, l)$ as a sequence of items $\langle (i_1)(i_2) \dots (i_n) \rangle$ such that for all consecutive i_j and i_{j+1} in the sequence, there exists $(u, v) \in E$ s.t. $l(u) = i_j$ and $l(v) = i_{j+1}$.*

For instance, sequences $\langle (B)(C)(D) \rangle$, or $\langle (B)(A) \rangle$, or $\langle (B)(C)(A) \rangle$ define paths of the poset shown in figure 4(a). We define a path to be *maximal* with respect to the inclusion of sequences. E.g., path $\langle (B)(A) \rangle$ is not maximal since it is a subsequence of the path $\langle (B)(C)(A) \rangle$. Note that posets are acyclic, so, the maximal paths in a poset G_p will always be defined between sources and sinks of G_p (of course, avoiding the shortcuts of the transitive closure). Note also that since a poset is actually a graph, we are still able to operate coproducts on them;

although this does not necesarily imply that the coproduct of two partial orders is another partial order.

Next, we will consider the goal of summarizing the input sequences as the most specific partial orders compatible with them.

3.1 A Closure System of Partial Orders

This section presents our first contribution: the construction and visual display of a concept lattice where nodes contain partial orders, and the relationships among them will be representative of the relationships in the input ordered data. We will construct this lattice by avoiding the formalization of a closure operator; however, we will show that the constructed family of formal concepts is indeed a closure system.

We say that partial order G_p is *closed for a set of sequences* S if G_p is the most specific poset from all those posets compatible with all $s \in S$. For instance, given the set of sequences $S = \{\langle(B)(C)(A)\rangle, \langle(B)(C)(D)\rangle\}$ we have two maximal posets compatible with them: the trivial order $G_{p_1} = \parallel B, C \parallel$ and the total order $G_{p_2} = B \rightarrow C$; but only G_{p_2} is closed for S since $G_{p_1} \trianglelefteq G_{p_2}$. Next proposition 1 ensures unicity of closed posets for a set of sequences S.

Proposition 1. *Given a set of sequences S there is exactly one single closed poset for S.*

Proof. As mentioned after definition 6, there is at least one compatible poset with all the sequences in S, which is the trivial order with the shared items of all sequences in S. Now, assume to the contrary that we had two most specific posets compatible with all $s \in S$, named G_{p_1} and G_{p_2} with corresponding morphisms into s. If so, we can construct a third poset $\widehat{G}$ from the union of G_{p_1} and G_{p_2}, i.e. their coproduct $\widehat{G} = \coprod G_{p_i}$ for $i = 1, 2$ (as we mentioned, the union of two graphs in this category can be formalized as their coproduct). The properties of the coproduct ensure that $\widehat{G}$ is unique, compatible with all $s \in S$, and more specific than G_{p_1} and G_{p_2}. $\square$

We must point out an important remark after this proof: the coproduct is an operator defined on graphs, so, the coproduct of two partial orders may give a graph that is not another partial order. For instance, the coproduct of $G_{p_1} = A \rightarrow B$ and $G_{p_1} = B \rightarrow A$ leads to a graph with a cycle (then, not antisymmetric). However, in the proof of proposition 1 we are operating the coproduct on two posets simultaneously compatible with the same set of sequences S. Given that the sequences in S do not have repeated items by definition, it is not possible to get a cycle from the coproduct of two posets compatible with S; in other words, two partial orders whose union leads to a cycle cannot be both compatible with the same set S.

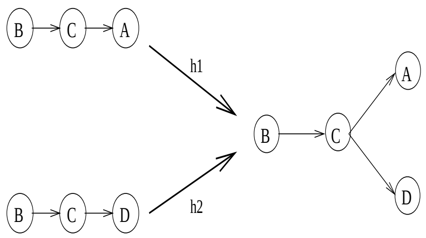

Fig. 5. Example of a coproduct

We say that *set of sequences* S is *closed* for a partial order G_p if S contains *all* the input sequences $s_i \in \mathcal{D}$ with which G_p is compatible. For example, the set of input sequences from figure 1 that are closed with respect to poset in figure 4 is $S = \{s_2, s_3\} = \{\langle(B)(C)(D)(A)\rangle, \langle(B)(C)(A)(D)\rangle\}$.

Now we are ready to define the notion of formal concept.

Definition 8. *A formal concept is a pair (S, G_p) where G_p is a closed partial order for the set of sequences S, and the set of sequences S is closed for the partial order G_p.*

Formal concepts (S, G_p) will be nodes of the concept lattice of partial orders. In practice, we will visualize these nodes principally by the closed poset G_p of the concept, and the dual S will be added as a list of object identifiers (thus, as it happens in general in Galois connections, these lists form a dual view of the same lattice that, in our case, is ordered by set-theoretic inclusion downwards); proposition 1 ensures that each node of the lattice can be represented by one single closed partial order. Edges in the lattice will be the specificity relationships among the concepts, named $\leq$, such that $(S_1, G_{p_1}) \leq (S_2, G_{p_2})$ if $G_{p_1} \trianglelefteq G_{p_2}$. The set of all concepts ordered by $\leq$ is called the concept lattice of partial orders of our context. Eventually, an artificial top representing a poset not belonging to any sequence is also added to the lattice and we note it by the unsatisfiable boolean constant $\Box$. In figure 6 we show the lattice of closed concepts for the data of figure 1.

Although the lattice of closed partial orders has been characterized without defining any specific closure operator that ensures $\cap$-stability on concepts, we can indeed prove that our system is closed under intersection. Semantically speaking, intersection must preserve the maximal common substructure from the intersected objects. Again, because of the injectivity of the labelling function along with the uniqueness of morphisms, the intersection of two posets can be easily formalized by the product operator of category theory (dual to the coproduct). However, we introduce now a completely different characterization of intersection of posets in terms of the coproduct of maximal paths.

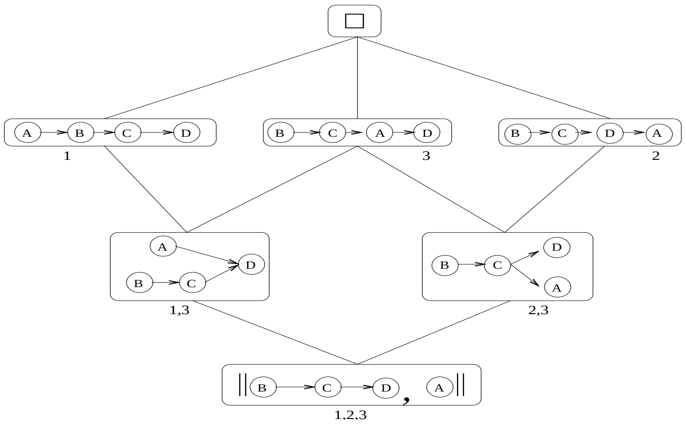

Fig. 6. Concept Lattice of partial orders for data in figure 1

Definition 9. *The intersection of two partial orders $G_p \cap G_{p'}$ is $G_p \cap G_{p'} = \coprod t_i$ where $\{t_i\}$ is the family of maximal paths contained in both G_p and $G_{p'}$.*

Next lemma proves that the intersection as it is defined here preserves indeed the maximal common substructure from the intersected partial orders.

Lemma 1. *The result of $G_{p_1} \cap G_{p_2}$ is the most specific poset G_p from those where $G_p \trianglelefteq G_{p_1}$ and $G_p \trianglelefteq G_{p_2}$.*

Proof sketch. If $G_p = \coprod t_i$ where $\{t_i\}$ is the family of maximal paths contained in both G_{p_1} and G_{p_2}, then obviously there exists by construction a morphism $G_p \trianglelefteq G_{p_1}$ and $G_p \trianglelefteq G_{p_2}$. Any other poset $G_{p'}$ having as well morphisms to G_{p_1} and G_{p_2}, must have a morphism to G_p, i.e. $G_{p'} \trianglelefteq G_p$, otherwise we would contradict the fact that some path t_i used in the coproduct is not a maximal path. Thus, we exactly get a product diagram; so, G_p defines the result of a product operation on partial orders from category theory. $\square$

Note that since intersection of posets is then associative. Next proposition 2 finally shows that our lattice is a system closed under intersection.

Proposition 2. *The intersection of closed posets is another closed poset.*

Proof sketch. We will reject the following false hypothesis: let (G_{p_1}, S_1) and (G_{p_2}, S_2) be two different closed partial orders for S_1 and S_2 respectively, and we suppose that $G_{p_1} \cap G_{p_2} = G_p$ is not another closed poset, i.e. G_p is not a part of any concept. In this case, let $G_{p'}$ be the closed poset immediately over G_p; that is, $G_p \trianglelefteq G_{p'}$ and it does not exist another closed $G_{p''}$ s.t. $G_{p''} \trianglelefteq G_{p'}$ and $G_p \trianglelefteq G_{p''}$. Then, it must be true that $G_{p'} \trianglelefteq G_{p_1}$ and $G_{p'} \trianglelefteq G_{p_2}$ (because both G_{p_1} and G_{p_2} are closed and different from G_p). But then, by lemma 1 we get a contradiction since G_p is not the most specific poset s.t. $G_p \trianglelefteq G_{p_1}$ and $G_p \trianglelefteq G_{p_2}$. □

This section shows that the input sequential data can be summarized by means of a concept lattice that presents a balance between generality and specificity for all the input sequences. As an example, observe lattice of figure 6 that gets an overview of the ordering relationships in the data. We also see that partial orders such as $\parallel A, B, C, D \parallel$ or $B \rightarrow D$ or $\parallel A, B \rightarrow D \parallel$ do not create a node in the concept lattice, i.e. they are not closed: this is because these posets turn to be redundant in describing our data, they are compatible with all the input sequences, but they are not specific enough to be closed.

The problem is now how to construct this useful concept lattice of closed partial orders. Currently, it is still a challenge how to deal with poset structures in the field of Knowledge Discovery: mining the partial orders directly from the data is a complex task incurring a substantial runtime overhead. For example, the work in [7] presents a method based on viewing a partial order as a generative model for a set of sequences and it applies different mixture model techniques. The final posets are not necessarily closed and so, they could be redundant; besides, they restrict the attention to a subset of partial orders called series-parallel partial orders (such as series-parallel digraphs) to avoid computational problems. Note that here we do not restrict in any sense the form of the final closed posets. Another work worth mentioning here is [8], where the authors present the mining of episodes (whose hybrid structures are indeed partial orders). However, again these structures are not closed in any lattice and computational problems still persist.

4 Coproduct Transformations to Closed Posets

In this section we show that our lattice of closed posets can be indeed obtained via coproduct transformations on the simpler lattice of stable sequences; thus, providing an efficient way to derive those closed posets. We present, next, one of the main results that will lead to our characterization.

Theorem 1. *Let G_p be the most specific partial order compatible with a set of sequences S; then the set of maximal paths of poset G_p defines exactly the intersection of all sequences in S, that is* $\bigcap_{\forall s \in S} s$.

Proof. We will prove both directions: maximal paths from G_p come from the intersection of all $s \in S$; and that intersections of all $s \in S$ define the maximal paths of G_p.

$\Rightarrow$) Let $t = \langle (i) \dots (j) \rangle$ define a maximal path between a source node, labelled with item i, and a sink node, labelled with item j, of the poset G_p, and suppose that t does not belong to the intersection of sequences in S. This implies that either 1/ t does not belong to some $s \in S$; or 2/ t belongs to all $s \in S$ but t is not maximal (i.e, there exists another t' belonging to the intersection of S s.t. $t \subset t'$). In the first case, it would mean that we started from a poset G_p not compatible with the set of sequences S; in the second case, it would mean that we started from a poset G_p which is not the most specific, since with $G_p \cup t'$ (formalized as the coproduct) we would get a more specific poset. In any case we are contradicting the original formulation of G_p in the theorem, so it is true that all maximal paths in G_p come from the intersection of sequences in S.

$\Leftarrow$) Let t be a sequence belonging to the intersection of all $s \in S$, such that it does not define a maximal path between a source and a sink in G_p. This implies that G_p is not the most specific for S, since we could add the path defined by t to the poset G_p and get a more specific poset still compatible with all S. Again, we reach a contradiction that makes the original statement true. Again, we must insist on the fact that sequences in S do not have repeated items by definition, so the sequence t considered here always leads to a path for G_p with no cycles.

□

To illustrate this theorem, let us consider any formal concept (S, G_p) of our lattice of partial orders: the maximal paths of the closed poset G_p are defined exactly by the intersection of all sequences in the closed set of sequences S. For instance, taking the lattice in figure 6: the intersection of the closed set $S = \{s_2, s_3\} = \{\langle(B)(C)(D)(A)\rangle, \langle(B)(C)(A)(D)\rangle\}$, is the set of sequences $\{\langle(B)(C)(D)\rangle, \langle(B)(C)(A)\rangle\}$, which coincides exactly with the maximal paths of the closed poset G_p in the same node. The next corollary follows immediately from the main theorem.

Corollary 1. *Let G_p be the most specific partial order compatible with a set of sequences S; then the set of all paths from G_p is defined exactly by subsequences of some sequence in the intersection of S, that is $t' \subseteq \bigcap_{\forall s \in S} s$.*

Proof. It immediately follows from theorem 1: if the maximal paths from G_p are defined by the intersection of all $s \in S$, then any subsequence of a maximal path defines a shorter path in G_p. □

Clearly after corollary 1, it is possible to reconstruct the transitive closure of a closed poset G_p with the intersections of the closed set of sequences S where G_p is maximally contained. The coproduct transformation will help in this procedure, as it is shown in next theorem 2. Again, for our notational purposes we consider that a sequence is indeed a poset with all the proper edges added by transitive closure.

Theorem 2. *Let G_p be the most specific partial order compatible with a set of sequences S, then G_p is the result of the coproduct transformation on $\bigcap_{\forall s \in S} s$.*

Proof. We know by theorem 1 that the intersection of sequences in S are exactly those maximal paths of the most specific partial order G_p compatible with S. Then, it is possible to prove that a poset G_p comes from the coproduct transformations of its maximal paths. This can be easily proved by induction on the number of paths of the poset G_p, and taking into account that the coproduct operator is associative. Moreover, because these paths come from intersections of S and S is restricted to not having repetition of items, then the coproduct transformations do not lead to any cycle here. $\square$

This last theorem concludes that the closed poset G_p from formal concept, (S, G_p), can be generated by the coproduct transformations on the intersections of S. Besides, for the same reason given after proposition 1, we can be sure that coproduct transformations always return a valid poset in theorem 2: two posets G_{p_1} and G_{p_2} whose coproduct transformation has a cycle cannot be both compatible with the same set of sequences S at the same time. To exemplify theorem 2, let us take the closed set of sequences $S = \{s_2, s_3\} = \{\langle(B)(C)(D)(A)\rangle, \langle(B)(C)(A)(D)\rangle\}$, whose intersection gives the set of sequences $\{\langle(B)(C)(D)\rangle, \langle(B)(C)(A)\rangle\}$. The coproduct transformation on these intersections is given in figure 5, and it exactly returns the closed poset in that formal concept.

Next section will explain the relation of these intersections of S with the stable sequences; thus, reaching a final lattice transformation of stable sequences to closed posets by means of coproduct operations on its nodes.

4.1 Lattice Transformation and Algorithmic Consequences

As introduced in the preliminaries, the work in [6] shows that stable sequences can be characterized by the closure operator Δ. Therefore, the set of all stable sequences from a database $\mathcal{D}$ can be organized in formal concepts $(O, \{s_1, \ldots, s_n\})$, where $\{s_1, \ldots, s_n\}$ are a set of stable sequences for the closed set of objects O. Since each object in the ordered context represents indeed an input sequence from $\mathcal{D}$, we can rewrite the formal concepts as $(S, \{s_1, \ldots, s_n\})$ where S is the set of input sequences renamed after O. From this point of view it is possible to prove as in [6] that stable sequences are the intersection of S, that is, $\{s_1, \ldots, s_n\} = \bigcap_{s \in S} s$.

Actually, this last observation leads naturally to the following theorem.

Theorem 3. *A lattice of stable sequences can be transformed into a lattice of closed partial orders by rewriting each node via coproduct transformations.*

Proof. Let $(O, \{s_1, \ldots, s_n\})$ be a formal concept of stable sequences, that we can rewrite as $(S, \{s_1, \ldots, s_n\})$. We have that $\{s_1, \ldots, s_n\}$ are stable sequences (maximal) in S, and S is the set of *all* the input sequences in which they appear.

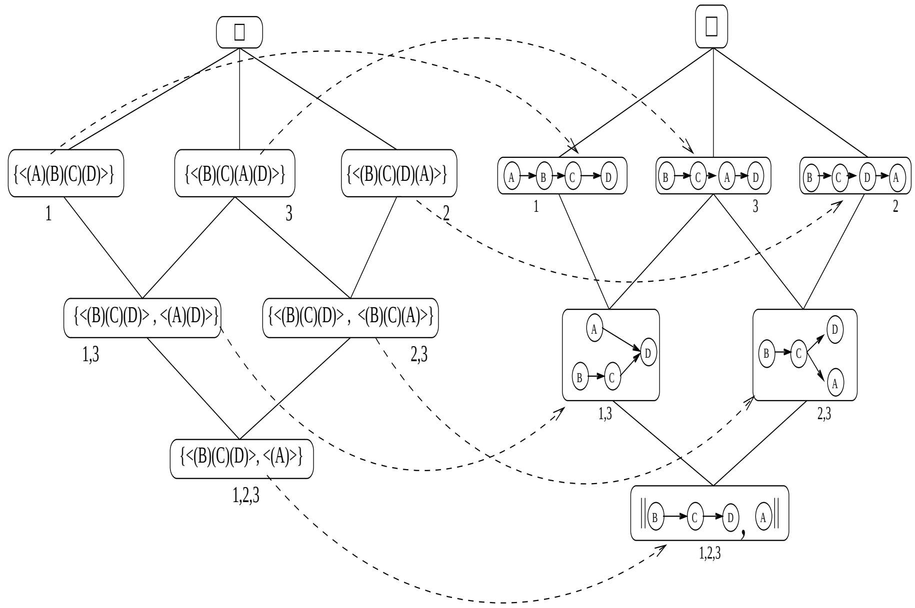

Fig. 7. Lattice transformation to closed partial orders

Let us construct a partial order G_p via coproduct transformations on these the stable sequences $\{s_1, \ldots, s_n\}$. We have that G_p is the most specific partial order for S: because $\{s_1, \ldots, s_n\}$ is equivalent to $\bigcap\{s | s \in S\}$ (as shown in [6]), and then, it immediately follows from theorem 1 and theorem 2 that G_p is the most specific poset for S. Thus, G_p is a closed poset for S. At the same time, S must be the maximal set of input sequences for the poset G_p to not contradict the fact of being O a maximal set of objects for the stable sequences $\{s_1, \ldots, s_n\}$; so, S is a closed set of input sequences for G_p. Therefore, (S, G_p) can be derived from from $(S, \{s_1, \ldots, s_n\})$ via coproduct transformations. $\square$

Figure 7 shows the set of all stable sequences from data in figure 1 organized in formal concepts. Each node of this lattice represents a set of stable sequences together with the list of object identifiers (input sequences) where they are maximally contained. We partially order sets of stable sequences in the lattice as in [6]: $\{s_1, \ldots, s_n\} \preceq \{s'_1, \ldots, s'_m\}$ if and only if $\forall s_i \exists s'_j,\ s_i \subseteq s'_j$. It can be graphically seen that each node of the lattice of stable sequences can be transformed into a node of the lattice of partial orders by means of a coproduct. Dashed lines of figure 7 indicate a node rewriting by means of a coproduct transformation on the stable sequences.

In practice, this transformation carries several consequences. Mining complex structured patterns directly from the data, such as partial orders, graphs, molecules, etc, is a complex task due to the overhead incurred by the algorithms (see

[7, 8]). However, the presented lattice transformation allows the construction of closed posets in the data by transforming the graph of stable sequences. So, the algorithmic contributions to the mining of stable sequences, such as CloSpan ([11]) and still more efficient, BIDE ([12]), can be used to mine partial orders by performing coproduct transformations on the discovered stable sequences.

4.2 Simultaneity Condition Revisited

To consider input sequences $s = \langle (I_1)(I_2)\dots(I_n)\rangle$ where each I_i may contain several simultaneous items, we must redefine the kind of posets we deal with. In particular, the labelling function l must be a function mapping each vertex to a set of items, i.e. $l : V \to 2^{\mathcal{I}}$. Again, this function is required to avoid repetitions of items in the nodes of the partial order (since we are not allowing repetitions in the input sequences either); so, the injectivity condition of l implies here that the items in each node are unique in the poset. With the new labelling function, the path from a poset is a sequence of itemsets and not single items. All the presented results hold with the new definitions.

5 Conclusions and Further Work

This paper shows that a simple lattice of stable sequences can be transformed into a lattice of closed partial orders: the first step proves that the maximal paths of each closed poset G_p are characterized by the intersections of those input sequences where G_p is contained. Then, the convenient coproduct operator is used to naturally allow the formalization of G_p as the transformation on those intersections. The last step is to prove that the intersections of input sequences are indeed stable sequences, so that it is possible to rewrite each node of the lattice of stable sequences into a closed partial order.

This transformation performed on lattices is of great importance in the field of Knowledge Discovery, where the mining of partial orders directly from the data is a complex task. In particular, it implies that algorithms for mining plain stable sequences can efficiently transform the discovered patterns into a lattice of closed partial orders that best summarizes the data.

The next step to complete our transformation framework would be to consider repetition of items in the input sequences, and so, to allow a non-injective labelling function in graphs. In this case, the coproduct operator does not work because the result of the transformation might not be unique. However, the pushout operator seems to fit in the new conditions: the pushout constructor can be used to formalize exactly the union and intersection of partial orders, and to allow the transformation process on the intersections of sequences. The problem with the pushout operator is to define a convenient target graph so that the pushout diagram commutes; it is still unclear at the moment how to achieve this. We also plan to study as a future work other complex structures, such as concept lattices of graphs or molecules: there we hope to find out which substitution mechanisms are required to obtain closed structures for more complex data.

Acknowledgements:

The authors thank Elvira Pino for her useful explanations on category theory.

References

1. Adamek J., Herrlich H., and Strecker G. Abstract and Concrete Categories. John Wiley, New York, 1990.
2. Agrawal R., and Srikant S. *Mining Sequential Patterns*. In Proc. of the Int'l Conf. on Data Engineering, pages 3-14. 1995.
3. Balcázar J.L., and Baixeries J. *Discrete Deterministic Datamining as Knowledge Compilation*. Workshop on Discrete Mathematics and Data Mining, in SIAM Int. Conf, 2003.
4. Davey B.A., and Priestly H.A. *Introduction to Lattices and Order*. Cambridge, 2002.
5. Ganter B., and Wille R. *Formal Concept Analysis. Mathematical Foundations*. Springer, 1998.
6. Garriga G.C. *Towards a Formal Framework for Mining General Patterns from Structured Data*. Proc. of the 2nd. Multi-relational Datamining, in KDD Int. Conf, 2003.
7. Mannila H., and Meek C. *Global Partial Orders from Sequential Data*. Int. Conf. on Knowledge Discovery in Databases. 2000.
8. Mannila H., Toivonen H., and Verkamo A.I. *Discovery of frequent episodes in event sequences*. Data Mining and Knowledge Discovery, 1(3):259-289. 1997.
9. Pasquier N., Bastide Y., Taouil R., and Lakhal L. *Closed set based discovery of small covers for association rules*. In Proc. of the 15th. Conference on Advanced Databases, pages 361-381. 1999.
10. Pfaltz J.L., and Taylor C.M. *Scientific Knowledge Discovery through Iterative transformations of Concept Lattices*. Workshop on Discrete Mathematics and Data Mining, in SIAM Int. Conf., pages 65-74. 2002.
11. Yan X., Han J., and Afshar R. *CloSpan: Mining Closed Sequential Patterns in Large Datasets*. Int. Conference SIAM Data Mining. 2003.
12. Wang J., and Han J. *BIDE: Efficient Mining of Frequent Closed Sequences*. Int. Conf. on Data Engineering. 2004.
13. Zaki M. *Generating non-redundant Association Rules*. In Proc. of the Sixth Int. Conference on Knowledge Discovery and Data Mining, pages 34-43. 2000.