Towards data mining without information on knowledge structure

Alexandre Vautier[1], Marie-Odile Cordier[1], and René Quiniou[2]

[1] Irisa - Université de Rennes 1
[2] Irisa - Inria
Campus de Beaulieu 35042 Rennes Cedex, France
{Alexandre.Vautier}@irisa.fr

Abstract. Most knowledge discovery processes are biased since some part of the knowledge structure must be given before extraction. We propose a framework that avoids this bias by supporting all major model structures e.g. clustering, sequences, etc., as well as specifications of data and DM (Data Mining) algorithms, in the same language. A unification operation is provided to match automatically the data to the relevant DM algorithms in order to extract models and their related structure. The MDL principle is used to evaluate and rank models. This evaluation is based on the covering relation that links the data to the models. The notion of schema, related to the category theory, is the key concept of our approach. Intuitively, a schema is an algebraic specification enhanced by the union of types, and the concepts of list and relation. An example based on network alarm mining illustrates the process.

1 Introduction

Frawley et al. [5] have introduced the well-known definition of Knowledge Discovery: "Knowledge discovery is the non trivial extraction of implicit, previously unknown, and potentially useful information from data. Given a set of facts (data) F, a language L, and some measure of certainty C, we define a *pattern* as a statement S in L that describes relationships among a subset F_S of F with a certainty c, such that S is simpler (in some sense) than the enumeration of all facts in F_S".

In most DM (Data Mining) tasks, the language drives the pattern search. This is the case in inductive databases [6], for instance: the user who mines the data has to query a database by using a language L. In other words, he has to define, to some extent, the structure of the "unknown" information. By structure, we mean a decision tree, a clustering, frequent itemsets, etc. So the information is not completely unknown, the structure is at least guessed by the user even if the data inside the structure are unknown.

It is usual to handle data without any idea on their underlying structure. It is the case when you have to mine alarms from a telecommunication network to detect intrusion without a priori knowledge on them. To choose the relevant

DM algorithm to run on a set of alarms is a challenge since even DM specialists are not familiar with the full range of DM algorithms.

The key idea is to propose a framework which provides a specification language called *Schema* in which the user can describe his data and which automatically builds models from various DM algorithms and evaluate them. This point of view is different from the data mining formalization by inductive databases [6] where the DM task is defined as an inductive query to a database. The query contains the “unknown” structure. This inductive database scheme has been instantiated in specific fields: association rules, frequent itemsets, decision trees, etc. However a common framework for inductive databases is still missing.

Our approach is closer to the 3W model [7] that proposes a language to unify DM processes. However, we focus on the automatic computation of models from data whereas the 3W model supports DM as a multi-step process correctly specified. The same authors have proposed later a method [9] to find constrained regions (sets of data cube) that summarize data. The evaluation of these regions relies on the Kolmogorov complexity that we use also to evaluate models.

Bernstein et al. [2] introduce the concept of Intelligent Discovery Assistants (IDAs) “which provide systematic enumeration of valid DM processes and an effective ranking of these valid processes by different criteria to facilitate the choice of DM processes to execute”. We use also data description in the form of a specification to find DM algorithms. For the time being, we do not rank DM processes and propose instead an execution of all of them to extract many models. However, we propose a generic rank of extracted models.

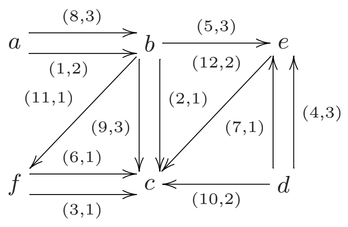
The evaluation scores the relevance of a model relatively to the data and a specification (a schema). We do not use classic model evaluation methods since they are not homogeneous and cannot be compared. We introduce a generic evaluation function based on the Kolmogorov complexity and more precisely on MDL [11] (Minimum Description Length). The more a model and a representation of the data in the model are short, the more the model is interesting. The covering relation between a model and the data is used to find different ways to encode data knowing a model.

Back to the example above, the system can extract different structures of models. The network alarms can be viewed as network links in which a DM algorithm looks for frequently used links. Another DM algorithm can extract frequent sequence patterns from an alarm sequence. Finally, alarms can be clustered on their severity and their date. The system has to rank these models in order to present them to the user.

In this paper, we focus on the power and the versatility of the proposed specification language. A prototype implementation exists and is currently applied on the detection of DDoS (Distributed Denial of Service) attacks from an alarm stream. Due to lack of space, we will not develop those computational aspects nor present experiments. The rest of the paper is organized as follows. Section 2 motivates our approach on an example on network alarms that will be used throughout the paper. Section 3 describes our DM framework. Section 4 gives details on the specification of data, models and DM algorithms. It shows how specifications can be unified to execute automatically DM algorithms on data and introduce schema foundations. Furthermore, it presents the specification of covering relations used in the Section 5 to rank models from Kolmogorov complexity. We end with concluding remarks and directions for future work.

2 Motivating example

To illustrate our method, we show an example where the goal is to extract knowledge from a set of network alarms. An alarm corresponds to a suspicious flow from a source actor to a target actor. An alarm is formally a triple (d, e, s) where d represents the date, e is the link $(\mathit{source}, \mathit{destination})$ associated to the alarm and s is the severity (1 - low to 3 - high). The graph of Figure 1(a) represents the dataset $\mathcal{A}$. A node represents an actor and an edge labeled by (d, s) represents an alarm occurring at date d with severity s.



$$\begin{tikzcd}
a \arrow[r, shift left, "(8{,}3)"] \arrow[r, shift right, "(1{,}2)"'] & b \arrow[r, "(5{,}3)"] \arrow[dl, "(11{,}1)"'] \arrow[d, shift right, "(9{,}3)"'] \arrow[d, shift left, "(2{,}1)"] & e \arrow[dl, "(12{,}2)"'] \\
f \arrow[r, shift left, "(6{,}1)"] \arrow[r, shift right, "(3{,}1)"'] & c & d \arrow[l, "(10{,}2)"] \arrow[u, shift right, "(7{,}1)"] \arrow[u, shift left, "(4{,}3)"']
\end{tikzcd}$$

(a)

Generalized links:

$m_1 = [b \to *, * \to c]$

$m_2 = [b \to *, d \to *, * \to c, * \to e]$

Sequences of two links:

$m_3 = [a \to b > b \to c, b \to c > d \to e]$

Clustering on date and severity:

$m_4 = [((1 \ldots 3), (1 \ldots 2)), ((8 \ldots 11), (2 \ldots 3))]$

$m_5 = [((1 \ldots 7), (1 \ldots 2)), ((8 \ldots 11), (2 \ldots 3))]$

(b)

Fig. 1. (a) Example set $\mathcal{A}$ of network alarms. (b) Models extracted from data $\mathcal{A}$.

Without any information on the structure of knowledge, knowing which DM algorithm to execute on data $\mathcal{A}$ is difficult. In the example of Figure 1, models in the form of generalized links, sequences and clusters could be extracted, as depicted in Figure 1(b). The models m_1 and m_2 are generalized links (links with the symbol $*$). They can be extracted from the alarms in $\mathcal{A}$ by searching nodes with high degree. The model m_3 is composed of 2 sequences of 2 events (an event is a link) and could be extracted by the algorithm of Srikant and Agrawal [12]. The models m_4 and m_5 are two partial clusterings of alarms from $\mathcal{A}$ on the date and the severity. They can be generated by algorithm k-means, for instance.

Firstly, the diversity of DM algorithms and the numerous different ways they can be executed on data make it very difficult for a user to choose DM computation on a given dataset. That is why we provide an automatic way to connect DM algorithms and data. Secondly, the many results of DM algorithm executions on data require a generic evaluation of the resulting models so they could be ranked before being displayed to the user. These two important points are developed in the sequel.

3 A general Data Mining Framework

The framework associated to Data Mining is illustrated in Figure 2. ① The first component is a database that contains the specification of DM algorithms. Each DM algorithm is described by a unique schema. The structure of the model that the algorithm outputs is also described by a schema. ② The user provides also the specification of data in a schema. ③ The system finds the relevant DM algorithms by matching data and algorithm specifications. The corresponding schemas are unified in a new operational schema. ④ Each valid DM algorithm is executed and outputs one or several models. ⑤ Each model is ranked on the MDL principle. This evaluation corresponds to minimize the size of the data according to a model and the covering relation between the model and the data. ⑥ The score of the model is the minimal size found at the previous step.

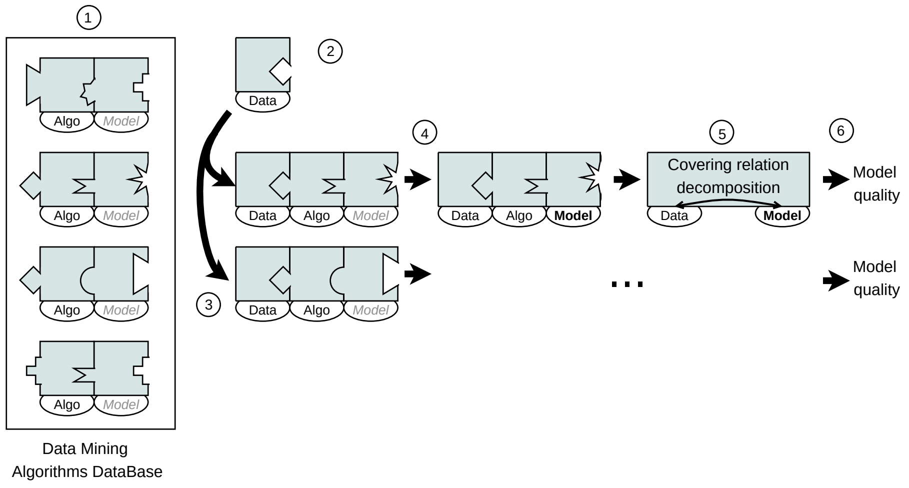

①
②
Data
Algo
Model
④
⑤
⑥
Covering relation decomposition
Model quality
③
...
Data Mining Algorithms DataBase

Fig. 2. Framework for mining data without information on knowledge structure

4 A specification language based on schemas

To begin with, the foundations of schemas are discussed. Instead of giving a formal description of schemas, the grounding elements of the specification language based on schemas are illustrated on the specification of data and the specification of three DM algorithms. Finally, we show how a data schema and a DM algorithm schema can be unified to yield an operational specification.

4.1 Schema theory

Algebraic specification [4] is based on notions from universal algebras in pure mathematics and on concepts of abstract data types and software specification in computer science. An algebraic specification is composed of a set of *sorts* (also called *type*), a set of *operations* on sorts and a set of *equations* on operations. The domain of an operation is a cartesian product of sorts and its codomain is a unique sort. Three additional concepts are needed to specify a DM problem: powerset, relation and union of types. Powersets are included in the form of lists, relation and union of types will be introduced in the next sections.

The union of types enables a very precise type specification. The concept of *sketch*, later introduced by Ehresmann [3], includes a definition of union of types. Further, this notion of sketch was particularly well described by Barr and Wells [1]. Intuitively a sketch (precisely a finite discrete sketch) is an algebraic specification using the union of types in the form of a graph. The nodes represent the type and the edges represent the operations. To add the notions of relation and powerset, we have extended the concept of sketch to the concept of schema. Intuitively, a schema is an algebraic specification where the concepts of relation, list of types and union of types can be expressed.

4.2 Data specification

The specification of network alarms is depicted in the schema $\mathcal{S}_d$ of Figure 3. This specification corresponds to the step ② of the general framework. In such a schema, a node represents a type (in the sequel, a type is viewed as a set) and an edge represents a function ($\rightarrow$) or a relation ($\vert\!\!\leftrightarrow$). The symbol ($\leftrightarrow$) that is traditionally used to describe a relation is not employed since the relations are represented by lists instead of sets. The green dotted lines ········ represent projections and the red dashed lines – – represent inclusions. Functions and relations on some path in the graph can be composed by the operator $\circ$. To each node $\mathbf{T}$ is associated an edge, named *identity*, written $id_T : \mathbf{T} \rightarrow \mathbf{T}$. This edge represents the identity function and is not drawn on a schema, it is implicit.

The type **alarm** is the cartesian product of the types **date**, **link** and **severity**. The edges d, e and s represent projection functions from the type **alarm** to the types **date**, **link** and **severity**, respectively. In the same way, the type **link** is the cartesian product **actor** $\times$ **actor** associated to the two projection functions $source :$ **link** $\rightarrow$ **actor** and $target :$ **link** $\rightarrow$ **actor**.

The type **1** (named terminal object in category theory) is used to define constants. An edge e from the type **1** to the type **T** represents a set of constants of type **T**. For example, the edge $\Sigma_{actor} : \mathbf{1} \vert\!\!\leftrightarrow actor$ represents the six constants: a, b, c, d, e and f of the type **actor**. We also assume that each type **T** can be enumerated, i.e. for all type **T**, there is a unique edge $\Sigma_T : \mathbf{1} \vert\!\!\leftrightarrow T$. For example, the edge $\Sigma_{link} : \mathbf{1} \vert\!\!\leftrightarrow$ **link** enumerates the elements $\{(x, y) | x, y \in \textbf{actor}\}$.

From each node there is an edge, named $\varnothing$, to the terminal object. Thus from every node, one can access any constant by the composition $\circ$. For example,

$$\begin{tikzcd}
 & \mathcal{L}(\textbf{alarm}) \arrow[d, leftrightarrow, "ex_a"] & \\
 & \textbf{alarm} \arrow[dl, dotted, "d"'] \arrow[d, dotted, "e"] \arrow[dr, dotted, "s"] & \\
\textbf{date} & \textbf{link} \arrow[d, dotted, bend right, "source"'] \arrow[d, dotted, bend left, "target"] & \textbf{severity} \\
 & \textbf{actor} & \\
 & 1 \arrow[uul, dashed, "\Sigma_{date}"] \arrow[u, dashed, "\Sigma_{actor}"'] \arrow[uur, dashed, "\Sigma_{severity}"'] & \\
 & \mathcal{S}_d &
\end{tikzcd}$$

$$\Sigma_{date} = \{\mathbf{1} \to 1, \mathbf{1} \to 2, \ldots, \mathbf{1} \to 12\}$$
$$\Sigma_{actor} = \{\mathbf{1} \to a, \mathbf{1} \to b, \ldots, \mathbf{1} \to f\}$$
$$\Sigma_{severity} = \{\mathbf{1} \to 1, \mathbf{1} \to 2, \mathbf{1} \to 3\}$$

Fig. 3. The schema $\mathcal{S}_d$ of the network alarms

the relation $\Sigma_{actor} \circ \varnothing$: **severity** $\leftrightarrow$ **actor** gives an access from a severity to any actor. The edges $\varnothing$ are not represented in a schema, they are implicit.

The edge Σ_{actor} represents an inclusion relation. It means that the type **actor** represents a set that contains only the constants defined by the inclusions that arrives to **actor**: the type **actor** contains only the elements a, b, c, d, e and f. In the same manner, the type **date** is composed of integers from 1 to 12 and the type **severity** is composed of the integers $1, 2$ and 3. The type $\mathcal{L}$**(alarm)** represents the set of lists composed of elements of type **alarm**. The relation ex_a : $\mathcal{L}$**(alarm)** $\leftrightarrow$ **alarm** associates a list of alarms with the alarms of the list.

4.3 DM algorithm and model specification

The DM algorithms stored in the database ① are specified in the same specification language as data. In order to show the versatility of schemas, we give the description of three DM algorithms that can extract the five models in the Figure 1(b): the schemas $\mathcal{S}_g, \mathcal{S}_c$ and $\mathcal{S}_s$ of Figures 4, 6 and 8. The specification of covering relations in schemas is particularly emphasized: it explicits the relationship between models and data, and it is useful for model evaluation.

In the schema $\mathcal{S}_g$, $\mathcal{L}$**(edge)** is the type of graphs which is represented by edge lists. The function *mine_graph* extracts a list of generalized edges (type $\mathcal{L}$**(edgeG)**) from an element of $\mathcal{L}$**(edge)**. The type **node** is considered as *abstract* since the relation Σ_{node} can be only partly specified. An element of **nodeG** is either an element of type **node** or the constant $*$ which stands for any element of **node**. This is expressed by the two inclusions i : **node** $\leftrightarrow$ **nodeG** and *gen* : $\mathbf{1} \to$ **nodeG**. These inclusions enable the construction of the relation $c_n = \langle \Sigma_{node}; id_{node} \rangle$. c_n is the covering relation between a generalized node and a node. Precisely, c_n is a *cofactorisation* of the relations Σ_{node} : $\mathbf{1} \leftrightarrow$ **node** and id_{node} : **node** $\leftrightarrow$ **node**. It corresponds to the construction "if-then-else": for all $x \in$ **nodeG**, if $x \in$ **node** then $c_n(x) = id_{node}(x)$ else $(x = *)$ $c_n(x) = \Sigma_{node}(x)$.

edgeG is the cartesian product **nodeG** $\times$ **nodeG** and **edge** is the cartesian product **node** $\times$ **node**. The relation c_e : **edgeG** $\leftarrowtail\!\!\!\rightarrow$ **edge** is the covering relation between generalized edges and edges. c_e is a *factorisation* of the relation $c_n \circ sourceG$: **edgeG** $\leftarrowtail\!\!\!\rightarrow$ **node** and the relation $c_n \circ targetG$: **edgeG** $\leftarrowtail\!\!\!\rightarrow$ **node**. Since an element of **edge** is defined by two elements of **node**, c_e "creates" several elements of **edge** from an element of **edgeG**. Finally the relation c_{le} is the covering relation between a list of generalized edges and the edges.

$$\begin{tikzcd}
\mathcal{L}(\mathbf{edge}) \arrow[r, "mine\_graph"] \arrow[d, leftrightarrow, "ex_e"] & \mathcal{L}(\mathbf{edgeG}) \arrow[dl, leftrightarrow, "c_{le}"'] \arrow[d, leftrightarrow, "ex_{eG}"] & \\
\mathbf{edge} \arrow[r, leftrightarrow, "c_e"] \arrow[d, dotted, bend right, "source"'] \arrow[d, dotted, bend left, "target"] & \mathbf{edgeG} \arrow[d, dotted, bend right, "sourceG"'] \arrow[d, dotted, bend left, "targetG"] & \\
\mathbf{node} \arrow[r, dashed, "i"'] \arrow[r, leftrightarrow, bend left, "c_n"] & \mathbf{nodeG} & \mathbf{1} \arrow[l, dashed, "gen"']
\end{tikzcd}$$

$$\begin{aligned}
gen &= \{\mathbf{1} \rightarrow *\} \\
c_n &= \langle \Sigma_{node}; id_{node} \rangle \\
c_e &= \langle c_n \circ sourceG, c_n \circ targetG \rangle \\
c_{le} &= c_e \circ ex_{eG}
\end{aligned}$$

Fig. 4. The schema $\mathcal{S}_g$ corresponding to the *mine_graph* algorithm

$$\begin{tikzcd}
\mathcal{L}(alarm) \arrow[r, "mine\_graph"] \arrow[d, leftrightarrow, "ex_a"] & \mathcal{L}(\mathbf{edgeG}) \arrow[dl, leftrightarrow, "c_{le}"'] \arrow[dd, leftrightarrow, "ex_{eG}"] & \\
alarm \arrow[dr, "c_{ge}"] \arrow[d, dotted, "e"'] & & \\
link \arrow[r, leftrightarrow, "c_e"] \arrow[d, dotted, bend right, "source"'] \arrow[d, dotted, bend left, "target"] & \mathbf{edgeG} \arrow[d, dotted, bend right, "sourceG"'] \arrow[d, dotted, bend left, "targetG"] & \\
actor \arrow[r, dashed, "i"'] \arrow[r, leftrightarrow, bend left, "c_n"] & \mathbf{nodeG} & \mathbf{1} \arrow[l, dashed, "gen"']
\end{tikzcd}$$

$$\begin{aligned}
gen &= \{\mathbf{1} \rightarrow *\} \qquad c_n = \langle \Sigma_{actor}; id_{actor} \rangle \\
c_e &= \langle c_n \circ sourceG, c_n \circ targetG \rangle \\
c_{ge} &= \langle \Sigma_{date} \circ \varnothing, c_e, \Sigma_{severity} \circ \varnothing \rangle \\
c_{le} &= c_{ge} \circ ex_{eG}
\end{aligned}$$

Fig. 5. The schema $\mathcal{S}'_g$ corresponding to a unification of $\mathcal{S}_g$ with $\mathcal{S}_d$.

The schema $\mathcal{S}_c$ (Figure 6) describes a 2-dimensional clustering. In the 2-dimensional space, a cluster can be approximated to a rectangle which can be represented by an horizontal and a vertical interval. A clustering is a list of clusters and the clustering algorithm is represented by the function $mine\_cluster$: $\mathcal{L}(\mathbf{point}) \rightarrow \mathcal{L}(\mathbf{clusterP})$. The considered clustering algorithms are parameter free since we assume that we have no knowledge about the data.

The schema $\mathcal{S}_s$ (Figure 8) describes an algorithm that extract 2-event sequences. An event (**event**) is composed of a time (**time**) and a type (**eventType**). A sequence (**seq**) is composed of two event types. In order to specify the covering relation between a list of events ($\mathcal{L}(\mathbf{event})$) and a list of sequences ($\mathcal{L}(\mathbf{seq})$) we need to convert an event into a 2-event (two successive events). This encoding expresses the event succession relation by chained 2-events. For example, the event list $(1, A), (2, B), (3, A), (4, A)$ is converted into the 2-event list $(1, A, B), (2, B, A), (3, A, A)$. This way, the covering relation c_s between a sequence and a 2-event list can be specified.

$$\begin{tikzcd}
 & & \mathcal{L}(\mathbf{point}) \arrow[d, leftrightarrow, "ex_p"] & & \\
 & & \mathbf{point} \arrow[dll, dotted, "x"'] \arrow[drr, dotted, "y"] & & \\
\mathbf{tX} \arrow[d, leftrightarrow, "c_{ix}"] & & & & \mathbf{tY} \arrow[d, leftrightarrow, "c_{iy}"] \\
\mathbf{intervalX} \arrow[u, dotted, bend left, "beginX"] \arrow[u, dotted, bend right, "endX"'] & & & & \mathbf{intervalY} \arrow[u, dotted, bend left, "beginY"] \arrow[u, dotted, bend right, "endY"'] \\
 & & \mathbf{clusterP} \arrow[uuu, "c_c"] \arrow[ull, dotted, "dX"] \arrow[urr, dotted, "dY"'] \arrow[d, leftrightarrow, "ex_c"'] & & \\
 & & \mathcal{L}(\mathbf{clusterP}) & &
\end{tikzcd}$$

$c_c = \langle c_{ix} \circ dX, c_{iy} \circ dY \rangle$

$c_{lc} : \mathcal{L}(\mathbf{clusterP}) \leftarrowtail\!\!\rightarrow \mathbf{point} = c_c \circ ex_c$

$mine\_cluster : \mathcal{L}(\mathbf{point}) \rightarrow \mathcal{L}(\mathbf{clusterP})$

Fig. 6. The schema $\mathcal{S}_c$ corresponding to the *mine_cluster* algorithm

$$\begin{tikzcd}
 & & \mathcal{L}(alarm) \arrow[d, leftrightarrow, "ex_a"] & & \\
 & & alarm \arrow[dll, dotted, "d"'] \arrow[drr, dotted, "s"] \arrow[r, dotted, "e"] & link & \\
date \arrow[d, leftrightarrow, "c_{id}"] & & & & severity \arrow[d, leftrightarrow, "c_{is}"] \\
\mathbf{intervalD} \arrow[u, dotted, bend left, "beginD"] \arrow[u, dotted, bend right, "endD"'] & & & & \mathbf{intervalS} \arrow[u, dotted, bend left, "beginS"] \arrow[u, dotted, bend right, "endS"'] \\
 & & \mathbf{clusterA} \arrow[uuu, "c_c"] \arrow[ull, dotted, "dD"] \arrow[urr, dotted, "dS"'] \arrow[d, leftrightarrow, "ex_c"'] & & \\
 & & \mathcal{L}(\mathbf{clusterA}) & &
\end{tikzcd}$$

$c_c = \langle c_{id} \circ dD, c_{is} \circ dS, \Sigma_{link} \circ \varnothing \rangle$

$c_{lc} : \mathcal{L}(\mathbf{clusterA}) \leftarrowtail\!\!\rightarrow alarm = c_c \circ ex_c$

$mine\_cluster : \mathcal{L}(\mathbf{alarm}) \rightarrow \mathcal{L}(\mathbf{clusterA})$

Fig. 7. The schema $\mathcal{S}'_c$ corresponding to a unification of $\mathcal{S}_c$ with $\mathcal{S}_d$.

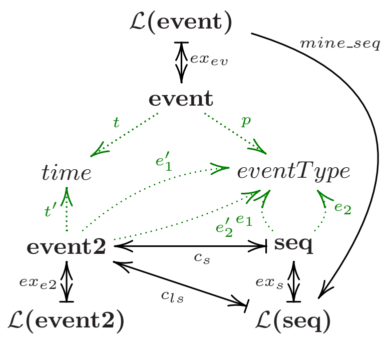
4.4 Schema unification

In this section, we detail the mechanism of schema unification ③. It is illustrated by unifying each of the three DM algorithm schemas $\mathcal{S}_g, \mathcal{S}_c$ and $\mathcal{S}_s$ to the data schema $\mathcal{S}_d$. The resulting unified schemas $\mathcal{S}'_g, \mathcal{S}'_c$ and $\mathcal{S}'_s$ are presented in Figures 5, 7 and 9. They are used to extract models automatically.

In order to simplify the presentation of unified schemas, all the types are not represented in Figures. For example in the schema $\mathcal{S}'_g$, the type **actor** is not completely defined since the relation Σ_{actor} is not written. By convention, a type that is not written in bold is defined in the schema $\mathcal{S}_d$. This is formally supported by the notion of *morphism of schemas* close to the *morphism of sketches* in category theory. This corresponds intuitively to type inheritance.

The unification $\mathcal{S}_U$ of two schemas $\mathcal{S}_A$ and $\mathcal{S}_B$ is obtained by matching edges and rewriting composed relations (composition, factorisation and cofactorisation). The unification of two schemas is not unique. However, in the DM context of this proposed framework, several constraints have to be respected which decrease the number of potential unifications. Firstly, the type corresponding to the inputs of the DM algorithm and the type corresponding to the type of data specified by the user must be unified. Secondly, the unified schema $\mathcal{S}_U$ should not contain any abstract type. This reflects the fact that mining algorithms works on completely defined inputs and outputs.

The schema $\mathcal{S}'_g$ (Figure 5) is constructed from schemas $\mathcal{S}_d$ and $\mathcal{S}_g$ according to these constraints. *mine_graph*, the graph DM algorithm, is instantiated by the type of data, network alarms in this case. This way, the type **nodeG** represents

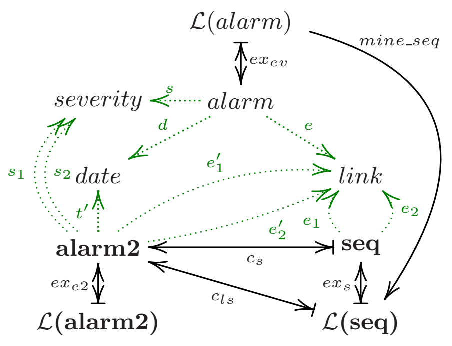


$c_s = \langle \Sigma_{time} \circ \varnothing, e_1, e_2 \rangle$

$c_{ls} = c_s \circ ex_s$

$convert : \mathcal{L}(\textbf{event}) \rightarrow \mathcal{L}(\textbf{event2})$

Fig. 8. The schema $\mathcal{S}_s$ related to the *mine_sequence* algorithm.



$c_s = \langle \Sigma_{date} \circ \varnothing, \Sigma_{severity} \circ \varnothing, \Sigma_{severity} \circ \varnothing, e_1, e_2 \rangle$

$c_{ls} = c_s \circ ex_s$

$convert : \mathcal{L}(\textbf{alarm}) \rightarrow \mathcal{L}(\textbf{alarm2})$

Fig. 9. The schema $\mathcal{S}'_s$ corresponding to a unification of $\mathcal{S}_s$ with $\mathcal{S}_d$.

the set $\{a, b, c, d, e, f, *\}$. The rewriting of the relation c_{le} is a little bit more complex. The types **edge** and $\mathcal{L}$(**edge**) are unified with types **link** and $\mathcal{L}$(**alarm**) respectively. This way, the *mine_graph* views an alarm as a link. The forgotten attributes date and severity must, however, be taken into account in the covering relation. A new covering relation c_{ge} between **edgeG** and **alarm** is added where c_{ge} is defined by the factorisation $\langle \Sigma_{date} \circ \varnothing, c_e, \Sigma_{severity} \circ \varnothing \rangle$. The old covering relation $c_{le} = c_e \circ ex_{eG}$ is rewritten by replacing c_e by c_{ge}.

The schema $\mathcal{S}'_c$ (Figure 7) is constructed from schemas $\mathcal{S}_d$ and $\mathcal{S}_c$. The function *mine_cluster* views a list of alarms as a list of pairs $(date, severity)$. The relation c_c is rewritten in order to include the forgotten information represented by the link element of an alarm. Other unifications could be proposed for clustering network alarms, the only constraint being that the types **Tx** and **Ty** have to be unified with completely ordered types.

The schema $\mathcal{S}'_s$ (Figure 9) is constructed from schemas $\mathcal{S}_d$ and $\mathcal{S}_s$. The event type is the link element of an alarm. The type **alarm2** needs two projections s_1 and s_2 to represent two consecutive alarms.

The unification can result in many schemas. Some of these schemas would turn to be meaningless if the DM algorithm schema is not well specified. In practice, the number of unifications can be reduced by the use of heuristics and constraints. For example, every abstract relations of a DM algorithm schema must be unified with implemented relations of a data schema. Furthermore equations between relations can be expressed in a schema. These equations constrain also the unification. As a result, after stating such constraints, the number of unifications is reduced to two in the case of the generalized edges example.

5 Generic Evaluation

This section shows how to compute a generic evaluation of models relatively to data. First we describe the principles on which this evaluation is grounded. Next, we take as example how to evaluate the model m_1 (Figure 1(b)).

The Kolmogorov complexity [10] measures the complexity of an object from the "absolute" length of a program executed on a "universal" machine (an equivalent of the Turing machine) that outputs this object. In our context of data mining, we use the MDL principle [11]. Some approaches [8] use compression to minimize the length of data description. In our case, given a unified schema, the data complexity relatively to a model is the cost of the model plus the cost of accessing the data from the model by the covering relation.

Definition 1 (Generic measure for model evaluation). *Let $\mathcal{S}$ be a schema, $c : \boldsymbol{M} \leftarrow\!\!\!\mapsto \boldsymbol{D}$ the covering relation of $\mathcal{S}$, $d \subseteq \boldsymbol{D}$ a set of data, $m \in \boldsymbol{M}$ a model, and $k(x)$ the cost in bits of representing the input*[3] *x. The complexity of the data d in the schema $\mathcal{S}$ relatively to the model m is:*

$$K(d, m, \mathcal{S}) = k(m|\boldsymbol{M}) + k(d|m, c, \boldsymbol{D})$$

For each model m extracted from the data d, the system has to compute $K(d, m, \mathcal{S})$ where $\mathcal{S}$ contains the specification of m. We illustrate the computation of $K(\mathcal{A}, m_1, \mathcal{S}'_g)$ where $m_1 = [e_{g1}, e_{g2}]$ (Figure 1), $e_{g1} = (b, *)$ and $e_{g2} = (*, c)$. Firstly, $k(m_1|\mathcal{L}(\mathbf{edgeG}))$ is computed. A generalized edge e_{gi} is composed of two generalized actors. The type **actorG** is composed of 7 constants. Thus the size (a size is a number of bits) of an **edgeG** is $2 \times log_2(7)$ and the size of m_1 is $k(m_1|\mathcal{L}(edgeG)) = 4 \times log_2(7) = 11.2$.

Secondly, $k(\mathcal{A}|m_1, c_{le}, \mathcal{L}(\mathbf{alarm}))$ is computed. There are many different ways to retrieve the data from the model. Each corresponds to some data encodings according to the model. The covering relation is composed of relations, factorisations and cofactorisations. To each *decomposition* of the covering relation corresponds a data encoding. The evaluation leads to finding the decomposition that minimizes the size of the data encoding. Three among many possible decompositions of c_{le} decompositions are shown. To each decomposition i of c_{le} corresponds a cost k_i. Recall that $c_{le} = c_{ge} \circ ex_{eG}$ where $c_{ge} = \langle \Sigma_{date} \circ \varnothing, c_e, \Sigma_{severity} \circ \varnothing \rangle$ and where $c_e = \langle c_n \circ sourceG, c_n \circ targetG \rangle$.

1. c_{le} is not decomposed. The set $c_{le}(m_1)$ is computed and the twelve alarms of $\mathcal{A}$ are found in this set: $log_2(C^{|\mathcal{A}|}_{|c_{le}(m_1)|})$, C^k_n is the number of subsets of size k in a subset of size n. The size of $|c_{le}(m_1)|$ is easily computed by enumerating all the edges covered by e_{g1} and e_{g2}. The alarms not covered by m_1 needs also to be represented: $k(\mathcal{A} \setminus c_{le}(m_1)|\mathcal{L}(\mathbf{alarm}))$ noted k' in the sequel.
$\Rightarrow k_1 = log_2(C^{|\mathcal{A}|}_{|c_{le}(m_1)|}) + k' = log_2(C^{12}_{432}) + k' = 76.0 + k'$.
2. c_{le} is decomposed into $c_{ge} \circ ex_{eG}$. The cost to find e_{g1} and e_{g2} in m_1 is $2 \times log_2(2)$. The size to find the alarms of $\mathcal{A}$ in $c_{ge}(e_{g1})$ and $c_{ge}(e_{g2})$ is:

[3] $k(d|m, c, \mathbf{D})$ means the cost to represent d knowing m, c and $\mathbf{D}$

  - $\mathcal{A}_{e_{g1}} = [(9,(b,c),3),\ (2,(b,c),1),\ (5,(b,e),3),\ (11,(b,f),1)] \subseteq c_{ge}(e_{g1})$
    $\Rightarrow log_2(C^{|\mathcal{A}_{e_{g1}}|}_{|c_{ge}(e_{g1})|})$
  - $\mathcal{A}_{e_{g2}} = [(6,(f,c),1),\ (3,(f,c),1),\ (10,(d,c),2),\ (12,(e,c),2),(9,(b,c),3),\ (2,(b,c),1)] \subseteq c_{ge}(e_{g2})$
    $\Rightarrow log_2(C^{|\mathcal{A}_{e_{g2}}|}_{|c_{ge}(e_{g2})|})$

  $\Rightarrow k_2 = 2 \times log_2(2) + log_2(C^{|\mathcal{A}_{e_{g1}}|}_{|c_{ge}(e_{g1})|}) + log_2(C^{|\mathcal{A}_{e_{g2}}|}_{|c_{ge}(e_{g2})|}) + k'$
  $\Rightarrow k_2 = 2 \times log_2(2) + log_2(C^4_{216}) + log_2(C^6_{216}) + k' = 65.3 + k'$

3. c_{le} is decomposed as the second decomposition plus the relation $\Sigma_{severity} \circ \varnothing$. It means that only the elements of $c_{ge}(e_{g1})$ and $c_{ge}(e_{g2})$ that have a severity of at least one element of $\mathcal{A}_{e_{g1}}$ and $\mathcal{A}_{e_{g2}}$ respectively are selected. In other words, the elements $c'_{ge}(e_{g1}) \subseteq c_{ge}(e_{g1})$ of severities 1 and 3 and the elements $c'_{ge}(e_{g2}) \subseteq c_{ge}(e_{g2})$ of severities 1 and 2 are retained. $2 \times 2 \times log_2(3)$ bits are required to represent these severities.
   $\Rightarrow k_3 = 2 \times log_2(2) + 4 \times log_2(3) + log_2(C^{|\mathcal{A}_{e_{g1}}|}_{|c'_{ge}(e_{g1})|}) + log_2(C^{|\mathcal{A}_{e_{g2}}|}_{|c'_{ge}(e_{g2})|}) + k'$
   $\Rightarrow k_3 = 2 \times log_2(2) + 4 \times log_2(3) + log_2(C^4_{144}) + log_2(C^6_{144}) + k'$
   $\Rightarrow k_3 = 2 + 6.3 + 24 + 33.4 + k' = 65.7 + k'$

All the decompositions of the covering relation c_{le} can be enumerated by analyzing its formal expression in the unified schema. Then, the system chooses the decomposition that minimizes the cost of the access to data $\mathcal{A}$. This minimal cost plus the cost of the model gives the evaluation of the model.

6 Conclusion and future work

We have presented a framework for mining data without any information on the knowledge structure present in the data. Instead of seeing DM as several ad hoc processes, we have shown that DM algorithms, DM models and data can be unified by the same specification language, based on schemas. The schema expressiveness, due to its grounds in category theory, is the major contribution of this paper. Moreover, several kinds of computations can be performed on specifications, such as unification or evaluation of models. The unification formalizes how to interface data and DM algorithms. This step releases the user from the burden of describing exhaustively this interface in order to run manually DM algorithms on specific data. The formalization of the covering relation is the key concept to evaluate models by using the Kolmogorov complexity.

We chose to illustrate the major aspects through an example. The approach and the language of schema are very general. Until now, we have used schemas to represent two mining processes concerning two very different datasets related to network alarms: DDoS attacks (the example of the paper) and intrusion detection in a virtual private network (VPN). However, the approach should be applied to other domains to fully assess its generality.

The presented framework is being developed in Prolog and Java. Experiments on real network alarms provided by France-Telecom have been undertaken. Furthermore, as DM algorithms are automatically executed in the proposed framework, we are working also on how resulting models can be displayed to the user and, furthermore, how they could be visualized directly on data. Finally, the user should be able to explicit what is the "useful information" of the Frawley et al. Knowledge Discovery definition. Thus, the user should be able to adapt the evaluation method, for instance, by restricting the enumeration of covering relation decompositions or by suggesting other ways to encode data from a model.

Acknowledgements
Special thanks to Dominique Duval from LMC-Grenoble for numerous discussions on Category Theory. This work is part of the CURAR Project (CRE #171938) supported by France-Telecom R&D. Thanks to the other project members: Mireille Ducassé from INSA-Rennes and Christophe Dousson and Pierre Le Maigat from France Telecom-Lannion. Thanks also to the anonymous referees.

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice-Hall, 1990.

[2] A. Bernstein, F. Provost, and S. Hill. Towards intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):503–518, 2005.

[3] C. Ehresmann. *Catégories et structures*. Dunod, Paris, 1965.

[4] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification I*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985.

[5] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: an overview. *AI Mag.*, 13(3):57–70, 1992.

[6] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39:58–64, 1996.

[7] T. Johnson, L. V. S. Lakshmanan, and R. T. Ng. The 3w model and algebra for unified data mining. In *VLDB'00: Proc. 26th International Conference on Very Large Data Bases*, pages 21–32, USA, 2000. Morgan Kaufmann Publishers Inc.

[8] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD'04: Proc. 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215, USA, 2004. ACM Press.

[9] L. V.S. Lakshmanan, R. T. Ng, C. Xing Wang, X. Zhou, and T. Johnson. The generalized MDL approach for summarization. In *VLDB'02 : Proc. 28th International Conference on Very Large Data Bases*, pages 766–777, China, 2002.

[10] M. Li and P. Vitanyi. *Introduction to Kolmogorov complexity and its applications*. Springer, 1997.

[11] J. Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989.

[12] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 1996.