XRM: AN XML-BASED LANGUAGE FOR RULE MINING SYSTEMS

B. Bouchou[1] , A. Cheriat[1] , M. Halfeld Ferrari Alves[1] , T. Jen[1] , D. Laurent[2]

[1]*Université François Rabelais Blois-Tours-Chinon,*
Antenne Universitaire de Blois, 3, Place Jean Jaurès 41000 Blois - France
[2]*Université de Cergy-Pontoise,*
2, avenue A. Chauvin - BP 222 95302 Cergy-Pontoise Cedex - France

Keywords: XML, XML Schema, association rule, data mining tools, logic formula.

Abstract: We present XRM, an XML-based language capable of promoting the collaboration among data mining systems. XRM is a general framework to express any system results and/or data as logic formulas. In this way, XRM offers flexibility to represent data, constraints and patterns, and allows mining systems to present their results in an exchangeable format. In this work, we concentrate on the use of XRM to represent different forms of association rules. XRM is built on XML Schema.

1 INTRODUCTION

We present XRM, an XML-based language capable of assuring the exchange of information among data mining systems. XRM allows the representation of first order formulas and, thus, is an efficient tool to represent data, constraints and patterns. In this paper, we focus on association rule mining systems. However, XRM can be used with every mining system whose data and results can be expressed by first order formulas. Figure 1 shows a rule extraction process whose result is presented as an XRM document. It illustrates that, in this context, a lot of different applications can interact with the rule mining system.

The aim of data mining systems is to efficiently extract knowledge from large data collections in order to identify relevant trends. Mining results are presented through different kinds of patterns (association rules, classification, clusters, etc). To improve the knowledge discovery process it is essential to define a model that allows (i) the integration of results coming from different data mining systems and (ii) the use by an application (from any different domain) of the results coming from a previous execution of a mining extraction task. The motivation for XRM comes from firstly the possibility of integrating results obtained by mining systems that use, in fact, a subset of logic formulas (such as association rules, conjunctive queries and so on) to express the extracted knowledge. We propose a standard exchange format among systems.

Association rules are undoubtedly one of the most popular type of patterns. The most famous application example of association rule mining is the market basket analysis: mine items sold together and then compute the association rules that indicate the probability of one set of items being in the basket given that another set is in. Association rule mining has evolved giving rise to more sophisticate approaches that propose to mine queries (Dehaspe and Toivenen, 1999; Diop et al., 2002). The problem of iterative query answering is becoming more and more important. The motivation of these methods comes from the observation that a first extraction can accelerate further ones: results already obtained during the extraction task of one user can improve the mining task of subsequent users (Diop et al., 2002).

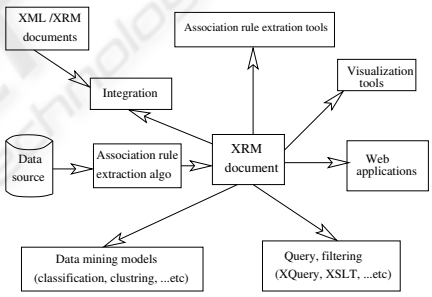

Figure 1: Utilization of XRM Model.

XML allows the representation of data by using tags that indicate the semantic structure of the data. Groups sharing data with similar meaning can agree

Bouchou B., Cheriat A., Halfeld Ferrari Alves M., Jen T. and Laurent D. (2004).
XRM: AN XML-BASED LANGUAGE FOR RULE MINING SYSTEMS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 441-446
DOI: 10.5220/0002614504410446

on different sets of tags and a schema for representing different kinds of data. In this way XML can be adopted as a framework for exchanging knowledge information among data mining systems.

Although some work have already been proposed in this context such as PMML (Data Mining Group, 2003), our approach is a contribution since we concentrate on the representation of logic formulas. In doing this, we allow the user working on data mining to express data (*i.e.*, facts over a database schema), database constraints (such as functional dependency) and extended association rules (*i.e.*, patterns more sophisticate than those allowed by the association rule module of PMML). Moreover, as XRM is built over XML Schema it can ensure a certain level of correctness of the data and patterns by defining integrity constraints. The following example shows that XRM can simplify the task of describing sophisticate association rules.

Example 1.1 Consider the association rule *R1*: 80% of the customers who buy *bread* and *butter*, also buy *milk* (*i.e.*, *confidence=0.80*). Moreover, this holds in 20% of the transactions (*i.e.*, *support=0.20*). Rule *R1* can be easily represented in PMML. Now consider the association rule *R2*, with *confidence=0.50* and *support=0.25*: Customers buying dairy products in June are professors doing local shopping. The representation of *R2* in PMML is extremely complicate, artificial and time consuming, since we need firstly to combine several tables into a universal table, then to transfer it to a binary table. The reason for this is that the DTD proposed by PMML cannot express predicates, variables or quantifiers. To express *R2* we need a tool capable of expressing the logic formula $(\forall x, \exists p, \exists v_1, \exists y_1, \exists s_1, \exists v_2, \exists y_2, \exists s_2)$ $(Cust(x, p, v_1) \wedge Sale(x, y_1, s_1, "June") \wedge Prod(y_1, "dairy") \Rightarrow (Cust(x, "professor", v_2) \wedge Sale(x, y_2, s_2, "June") \wedge Store(s_2, v_2))$. □

The main contributions of this paper are:

- The introduction of an XML-based language to represent logic formulas. This language allows the representation of data, constraints and patterns (as logical formulas). Thus, XRM offers a flexibility for systems that extend the association rule mining by considering not only simple rules over a unique database relation but also queries involving base relations and views. It is a general framework to express any system on logic formulas (not only association rules).
- The possibility given to the mining systems to present their results in an exchangeable format, making it easy for other tools to work on them. In fact, different tasks can be accomplished over the results of a mining process: their integration to the results coming from other mining systems, their use in other mining process, their graphical representation, their manipulation by XML query languages, etc.
- The use of XML Schema (W3C, 2001) in the definition of XRM that ensures a certain level of correctness of data to be mined and allows the automatic verification of patterns produced by mining systems.

This paper is organized as follows. Section 2 discuss some related work. Section 3 recalls the data mining concepts to be represented in our language. In Section 4, XRM is presented by a general schema and we discuss some of its details. Section 5 concludes with some further work.

2 RELATED WORK

The Predictive Model Markup Language (PMML) proposed by (Data Mining Group, 2003) is a format to exchange patterns among systems. It is an XML-based language, built over a DTD, for describing data mining models. Part of this DTD concerns rule models and focus on association rules. This specification is however restricted to relatively simple rule models, since only transactions with a single attribute can be represented. Variables, negation, quantifiers, connectives or multi-dimensional association rules cannot be expressed by PMML. The model Rule Model proposed by (Wettschereck and Müller, 2001) modifies the DTD of PMML for association rules. In this way, it can describe multi-relational association rules. In that approach, an association rule consists of a set of literals but quantifiers and connectives cannot be represented.

Contrary to the above approaches, XRM is built over XML Schema. Thus, XRM can impose constraints that are not possible to express when dealing with a DTD. Moreover, as XRM allows the representation of logic formulas, sophisticate multi-relational association rules can be treated.

XDM (Meo and Psaila, 2003) uses XML as a unifying framework for inductive databases (Imielinski and Mannila., 1996) and, more generally, for knowledge discovery systems. It is devised to capture the KDD process and allows the storage of the derivation process (described by statements). In fact, XRM can be seen as a complement for XDM: as XDM is independent of a specific format for data and pattern, one can consider that data and patterns represented by XRM documents might be stored in an inductive database based on XDM.

3 MINING DATABASES

We assume that the reader is familiar with the bases of relational databases and first order logic. We just recall some definitions and notations used in this paper. A *relation schema* is a relation name R and a *database schema* is a nonempty finite set $\mathbf{R}$ of relational schemas. In the named perspective, names of attributes are considered and a tuple u (with sort

$U = \{A_1, \ldots, A_n\}$) is defined as a function and represented by $u = \langle A_1 : v_1, \ldots, A_n : v_n \rangle$, where $A_1, \ldots, A_n$ are attributes and each v_i is a constant in the underlying domain. In the unnamed perspective the order is important, and a tuple is seen as an element of the Cartesian product. A tuple u (with arity n) is denoted by $u = \langle v_1, \ldots, v_n \rangle$. In this paper we use the logic programming perspective and represent a *relation instance* over R as a set of facts over R. A fact is denoted by $R(A_1 : v_1, \ldots, A_n : v_n)$ or by $R(v_1, \ldots, v_n)$. A database instance over $\mathbf{R}$ is the union of relation instances over R, for all $R \in \mathbf{R}$.

Terms are built in the usual way from constants, variables and function symbols. An *atom* is either true, false or an expression of the form $R(t_1, \ldots, t_n)$ where R is a n-ary predicate and $t_1, \ldots, t_n$ are terms. *(Well-formed first order) formulas* (over $\mathbf{R}$) are defined recursively starting with atoms, using boolean connectives and the quantifiers ($\forall$ and $\exists$). Given a first order formula ϕ, we denote by $free(\phi)$ the set of of free variables in ϕ.

A datalog rule has the form $l : a_0 \leftarrow a_1, \ldots, a_n$, where the *head* a_0 is an *atom* and the *body*, $a_1, \ldots, a_n$ is composed by *atoms* or *negated atoms*. Each datalog rule is associated to a first order formula $\phi : (\neg a_1 \vee \ldots \vee \neg a_n \vee a_0)$ (quantifiers omitted).

We recall the notations and terminologies presented in (Diop et al., 2002), which will be used in the rest of this article. We start with the notion of referential that is defined as a *view* r over a database schema $\mathbf{R}$. Intuitively, r gives the "individuals" for which the support and the confidence of a rule are computed. For instance, the following referential r defines the customers that bought products in June: $r(x) \leftarrow Cust(x, p, v) \wedge Sale(x, y, s, "June")$.

Now, a *mining query* is an expression of the form: $(\exists Y)(r \wedge \phi)$, such that $Y = free(\phi) \setminus free(r)$ and ϕ is a logical formula. An *association rule* is an expression of the form: $(\forall K)(Q_1 \Rightarrow Q_2)$, where $K = free(r)$ and Q_1 and Q_2 are two mining queries containing the same referential r. For example, consider the two queries Q_1 and Q_2. Q_1 indicates customers buying dairy product in June: $Q_1 : (\exists y)(r(x) \wedge Sale(x, y, s, D) \wedge Store(s, "dairy"))$ and Q_2 indicates professors doing local shopping in June: $Q_2 : (\exists v)(r(x) \wedge Cust(x, "Professor", v) \wedge Sale(x, y, s, D) \wedge Store(s, v))$. Now, $L_2 : (\forall x)(Q_1 \Rightarrow Q_2)$ is an example of an association rule.

Given a logic formula ϕ (expressing a query or a view) and a database instance $\mathbf{I}$ over database schema $\mathbf{R}$, the expression $\phi(\mathbf{I})$ represents the relation resulting from the evaluation of ϕ over $\mathbf{I}$ and $|\phi(\mathbf{I})|$ is the number of tuples in this resulting relation. Now, for every instance $\mathbf{I}$ of $\mathbf{R}$, we have:

- The *support* of Q relatively to $\mathbf{I}$ and a referential r, denoted by $sup(Q/r, \mathbf{I})$ is the ratio $Sup(Q/r, \mathbf{I}) = \frac{|Q(\mathbf{I})|}{|r(\mathbf{I})|}$. A *frequent query* is a mining query Q for which $Sup(Q/r, \mathbf{I}) \geq minsup$, where *minsup* is a support threshold.
- The *support* of an *association rule* $L : (\forall K)(Q_1 \Rightarrow Q_2)$ relatively to r and $\mathbf{I}$, denoted by $Sup(L/r, \mathbf{I})$ is the ratio $Sup(L/r, \mathbf{I}) = \frac{|(Q_1 \wedge Q_2)(\mathbf{I})|}{|r(\mathbf{I})|}$.
- The *confidence* of an association rule $L : (\forall K)(Q_1 \Rightarrow Q_2)$ relatively to r and $\mathbf{I}$, denoted by $Conf(L/r, \mathbf{I})$ is the ratio: $Conf(L/r, \mathbf{I}) = \frac{|(Q_1 \wedge Q_2)(\mathbf{I})|}{|Q_1(\mathbf{I})|}$. We can also define the confidence of an association rule by the expression: $Conf(L/r, \mathbf{I}) = \frac{Sup(L/r, \mathbf{I})}{Sup(Q_1/r, \mathbf{I})}$.
- An association rule L is *interesting* iff: $Sup(L/r, \mathbf{I}) \geq minsup$ and $Conf(L/r, \mathbf{I}) \geq minconf$, where $minconf$ and $minsup$ are thresholds.

4 XRM

XRM specification is available in (Bouchou et al., 2004). This specification is built with XML Schema since it allows the implementation of integrity constraints over the model and the use of *name spaces*. The concept of name space brings the possibility of including in an XML document the reference of a schema previously defined. In other words, a document has an element <xmlns> having as attribute an URL that specifies the schema (or the content type) of the document. In this paper, this schema corresponds to the specification of XRM, designed to make possible the communication among data mining applications.

DTDs also allow the specification of schemas but they are less powerful than XML Schema. Indeed, a DTD offers very limited data types, it does not allow the use of XML name spaces and it does not support the concept of inheritance. Moreover, it is more difficult to extend a DTD than a schema proposed with XML Schema.

In this section we present the language XRM. We explain features concerning XML Schema when necessary. Figure 2 summarizes the elements specified by XRM. The specification of these elements takes into account the hierarchy of XML documents - trees where each node has a position, a label and a type (element or attribute). Most of XRM elements correspond to the concepts seen in Section 3. In order to give them a global scope (to be able to reference them anywhere), all components are listed under the root.

In what follows we first present the basic elements of XRM and then, we show how they can be used to describe association rules.

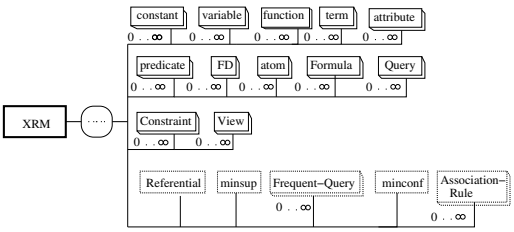

Figure 2: Components of XRM model.

4.1 Basic elements of XRM

The basic elements of XRM concern the presentation of first order formula. We show in Figure 3 how our model defines a *constant* (line 1), a *variable* (line 2) and a *function* (lines 3-16) . We recall that simple types and complex types are defined in XML Schema. Simple types are built by imposing some restrictions over predefined types (strings, positive integer, etc) or other simple types. Complex types are composed by a set of elements or attributes. From Figure 3, we notice that the elements constant and variable have predefined types, while function is a complex type element. Indeed, the complex type that defines a function has the following features: (i) two attributes: the first one specifying the function symbol (line 13) and the second one its arity (line 14); (ii) some sub-elements: represented by a list of constants, variables or functions (lines 5-12). These sub-elements represent the function parameters. Notice that we can refer to previously declared components, using the XML Schema option `ref` (lines 8-10).

XML Schema proposes different choices to define sub-elements. In Figure 3 we notice the use of `sequence` (line 5) and `choice` (line 6). The option `sequence` defines an ordered list of sub-elements, while `choice` specifies possible choices of sub-elements. We can also precise the minimum and maximum number of occurrences of each sub-element, with options `minOccurs` and `maxOccurs` (when these options are not specified they are considered to be 1). Thus, in our case, a function can have a sequence of 1 or n (due to the unbounded on line 5) parameters. Parameters[1] can be chosen to be constants, variables or functions.

The definition of a *term* uses the same XML Schema options mentioned above. Recall that a term is either a constant, a variable or a function. The set of attribute names that can be used in a XRM document is specified by the basic element *attribute*.

In the definition of a *predicate* (Figure 4) XRM gives the user the option of using the named (line 6) or the unnamed (line 7) perspective (Section 3).

[1]Notice that in the declaration of function, predicate, etc. XRM includes the notion of their parameters.

```
1)  <xrm:element name="constant" type="xrm:string"/>
2)  <xrm:element name="variable" type="xrm:string"/>
3)  <xrm:element name="function">
4)  <xrm:complexType>
5)  <xrm:sequence maxOccurs="unbounded">
7)  <xrm:choice>
8)  <xrm:element ref="constant"/>
9)  <xrm:element ref="variable"/>
10) <xrm:element ref="function"/>
11) </xrm:choice>
12) </xrm:sequence>
13) <xrm:attribute name="symbol" type="xrm:string"/>
14) <xrm:attribute name="arity" type="xrm:positiveInteger"/>
15) </xrm:complexType>
16) </xrm:element>
```

Figure 3: Constant, Variable and Function declarations.

```
1)  <xrm:element name="predicate">
2)  <xrm:complexType>
3)  <xrm:sequence>
4)  <xrm:element name="symbol" type="xrm:string"/>
5)  <xrm:choice>
6)  <xrm:element name="attribute-name"
      type="xrm:string" maxOccurs="unbounded"/>
7)  <xrm:element name="arity" type="xrm:string"/>
8)  </xrm:choice>
9)  </xrm:sequence>
10) </xrm:complexType>
11) </xrm:element>
```

Figure 4: Specification of a predicate in XRM.

In the description of a mining rule, we want the names of predicates to be unique. To this end, we use the notion of key provided by XML Schema, which is defined in two steps. In the first step we identify a set of context positions from the root on which the key is being defined. In the second step, we specify the set of values that distinguish each context position. Indeed, this notion of key corresponds to the absolute key presented in (Buneman et al., 2001). We use a subset of XPath expressions to specify context positions and to obtain the values that compose keys. In Figure 5, the key constraint ”predicate-PK” is presented. The first XPath expression (xpath=”XRM/predicate”) specifies the path to context positions (in this case, positions labeled predicate). The second Xpath expression (xpath=”symbol”) specifies that, in this context, the sub-element ”symbol” is the key.

```
1)  <xrm:key name="predicate-PK">
2)    <xrm:selector xpath="XRM/predicate"/>
3)    <xrm:field xpath="symbol"/>
4)  </xrm:key>
```

Figure 5: Key constraint for predicates.

An atomic formula (or an *atom*) is either true, false

or composed by a predicate symbol and terms. We want to constraint predicate symbols composing an atom to those already defined. To this end, we use the notion of foreign key (`keyref`). Similarly to the key definition, we use XPath expressions to specify foreign keys (lines 2-3 of Figure 6). We recall that foreign keys are always associated to a key. Now, to indicate the key constraint to which the foreign key is associated to, we add `refer`, as illustrated in Figure 6 (line 1). In this figure, we present the foreign key "atom-ref-RK". This foreign key indicates that the element "symbol", specified by Xpath expressions xpath=".//atom" and xpath="symbol", refers to a key defined by the key constraint *predicate-PK* (defined in Figure 5). Notice that we also use the notion of foreign key to assure that *attribute-name* (line 6 Figure 4) corresponds to an existing attribute.

```
1) <xrm:keyref name="atom-ref-RK" refer="predicate-PK">
2)  <xrm:selector xpath=".//atom"/>
3)  <xrm:field xpath="symbol"/>
4) </xrm:keyref>
```

Figure 6: Foreign key constraint for atoms.

In Figure 7 we show the recursive definition of a *formula*. In the option `choice` we present four ways of building a formula. The first choice corresponds to the atomic formula (line 4) while the second one defines negative formulas (lines 5-10). The third choice introduces quantified formula: a list of quantified variables precedes a formula (lines 11-19). The fourth choice builds a compound formula by using binary connectives (lines 20-26). Notice that *TQuantifier* (line 14) and *TBConnective* (line 23) are simple types corresponding to the quantifiers ($\exists$ and $\forall$) and to the binary connectives, respectively.

The next step consists in defining queries and views. In XRM, these definitions are done by referring to types previously defined: a *query* refers to a formula and a *view* refers to a query.

4.2 Association rules in XRM

In this part we specify association rules with the basic elements introduced in the previous section. Firstly, the specification of a referential is done just by referring to a view defined over a database schema. The thresholds *minsup* and *minconf* are specified by a simple type called "Prob-number", which represents values between 0 and 1.

To define a frequent query we use the concept of inheritance, implemented by the option `extension` in XML Schema. In Figure 8 (line 4), a *Frequent-Query* inherits all the properties of a query (due to the declaration <xrm:extension base= "TQuery">). Besides, it has its own properties: an element called "ref-referential" (line 6) and two attributes (lines 8-9).

```
1)  <xrm:element name="Formula">
2)  <xrm:complexType>
3)  <xrm:choice>
4)  <xrm:element ref="atom"/>
5)  <xrm:sequence>
6)  <xrm:element name="unary-connective"
          type="xrm:string" fixed="not"/>
7)  <xrm:element name="open-parenthesis"
        type="xrm:string" fixed="("/>
8)  <xrm:element ref="Formula"/>
9)  <xrm:element name="close-parenthesis"
        type="xrm:string" fixed=")"/>
10)  </xrm:sequence>
11)  <xrm:sequence>
12)  <xrm:element name="open-parenthesis"
        type="xrm:string" fixed="("/>
13)  <xrm:sequence maxOccurs="unbounded">
14)  <xrm:element name="quantifier"
       type="TQuantifier" default="Exist"/>
15)  <xrm:element name="variable" type="xrm:string"/>
16)  </xrm:sequence>
17)  <xrm:element name="close-parenthesis"
          type="xrm:string" fixed=")"/>
18)  <xrm:element ref="Formula"/>
19)  </xrm:sequence>
20)  <xrm:sequence>
21)  <xrm:element name="open-parenthesis"
        type="xrm:string" fixed="("/>
22)  <xrm:element ref="Formula"/>
23)  <xrm:element name="binary-connective"
           type="TBConnective"/>
24)  <xrm:element ref="Formula"/>
25)  <xrm:element name="close-parenthesis"
        type="xrm:string" fixed=")"/>
26)  </xrm:sequence>
27)  </xrm:choice>
28)  </xrm:complexType>
29)  </xrm:element>
```

Figure 7: Specification of a well-formed formula.

A frequent query is associated to the key constraint "Frequent-Query-PK" (line 13-16) and we define a foreign key constraint over "ref-referential" since we want this element to be an association to a referential. In Figure 9 we show the declaration of this foreign key constraint. Notice that the associated key constraint, denoted by "Referential-PK", is defined over a referential (the declarations of referential and its key constraint are not shown here).

Finally, XRM defines an *Association-Rule* as an element composed by the following features:
(i) A list of quantified variables. (ii) Elements corresponding to the antecedent and the consequent of an association rule. We define foreign key constraints over these elements, since they should be references to frequent queries. (iii) Attributes "support" and "confidence" of type Prob-number. We refer to (Bouchou et al., 2004) for an example of an XRM document that represents an association rule.

```
1)  <xrm:element name="Frequent-Query">
2)  <xrm:complexType>
3)  <xrm:complexContent>
4)  <xrm:extension base="TQuery">
5)  <xrm:sequence>
6)  <xrm:element name="ref-referential"/>
7)  </xrm:sequence>
8)  <xrm:attribute name="id" type="xrm:string"/>
9)  <xrm:attribute name="support" type="Prob-number"/>
10) </xrm:extension>
11) </xrm:complexContent>
12) </xrm:complexType>
13) <xrm:key name="Frequent-Query-PK">
14) <xrm:selector xpath="XLogic/Frequent-Query"/>
15) <xrm:field xpath="@id"/>
16) </xrm:key>
```

Figure 8: Specification of a frequent query and its associated key constraint.

```
1)  <xrm:keyref name="Frequent-referential-RK"
            refer="Referential-PK">
2)  <xrm:selector xpath="XRM/Frequent-Query"/>
3)  <xrm:field xpath="ref-referential"/>
4)  </xrm:keyref>
```

Figure 9: Foreign key constraint for frequent queries.

From the above presentation, we can notice that the use of XML Schema helps a lot in the specification of XRM. Key and foreign keys constraints are extremely useful to guarantee the consistency of XRM documents (avoiding, for instance, association rules that use frequent queries that do not exist). The inheritance property helps in generalizing some concepts. Moreover, XRM offers a natural way of describing patterns since, for instance, it keeps track of the meaning of each element in an association rule (*i.e.*, a rule is composed by quantified variables, frequent queries, and so on). The verbose aspect (an XML inheritance) of this approach is an advantage, since the aim here is to propose a tool that allows the communication among different types of application programs. Notice from Figure 2 that XRM also specifies *functional dependencies* (*FD*) and *constraints* (not discussed in this paper).

5 CONCLUSION

In this paper we present XRM, an XML-based language that allows the representation of first order formulas and, thus, is an efficient tool to represent data, constraints and patterns. Although in this paper we concentrate on association rule mining systems, XRM can be used with every mining system whose data and results can be expressed by first order formulas.

We are interested in the following directions for further research. First, the application of our approach to other data mining tasks, such as classification and clustering in first order logic, as introduced in (Džeroski and Lavrač, 2001). Second, the specification of a general framework to exploit data mining results. To this end, we should firstly extend the tree automata validation process presented in (Bouchou and Halfeld Ferrari Alves, 2003) to deal with XML Schema instead of DTDs. Our goal is to develop an update language allowing changes on valid XRM documents by preserving validity (we intend to adapt the method proposed in (Bouchou et al., 2003) to XRM).

REFERENCES

Bouchou, B., Cheriat, A., Halfeld Ferrari Alves, M., Jen, T., and Laurent, D. (2004). An XML approach for rule mining systems. Technical Report (To appear), LI, Université de Tours.

Bouchou, B., Duarte, D., Halfeld Ferrari Alves, M., and Laurent, D. (2003). Extending tree automata to model XML validation under element and attribute constraints. In *ICEIS*.

Bouchou, B. and Halfeld Ferrari Alves, M. (2003). Updates and incremental validation of XML documents. In *DBPL*. LNCS 2921, Springer Verlag.

Buneman, P., Davidson, S., Fan, W., Hara, C., and Tan, W. C. (2001). Keys for XML. In *World Wide Web*, pages 201–210.

Data Mining Group (2003). PMML. Technical report, http://www.dmg.org/pmml-v2-0.htm.

Dehaspe, L. and Toivenen, H. (1999). Discovery of frequent datalog patterns. Data mining and knowledge discovery. *Kluwer Academic Publishers*, 3:7–36.

Diop, C., Giacometti, A., Laurent, D., and Spyratos, N. (2002). Composition of mining contexts for efficient extraction of association rules. In *EDBT*. LNCS 2287, Springer Verlag.

Džeroski, S. and Lavrač, N., editors (2001). *Relational Data Mining*, chapter 6, 9. Springer-Verlag.

Imielinski, T. and Mannila., H. (1996). A database perspective on knowledge discovery. *Communication of the ACM*, 39:58–64.

Meo, R. and Psaila, G. (2003). An XML-Based definition of a database for knowledge discovery. Technical Report RT74-2003-04, Dipartimento di Informatica, Universit di Torino.

W3C (2001). XML Schema Part 1: Structures. Technical report, http://www.w3.org/TR/xmlschema-1/.

Wettschereck, D. and Müller, S. (2001). Exchanging data mining models with the predictive modelling markup language. In *Proc. of the ECML/PKDD-01 Workshop on Integration Aspects of Data Mining, Decision Support and Meta-Learning*, pages 55–66.