Volume 31 Number 3 October 2007 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Supervised Machine Learning: A Review of Classification Techniques

S. B. Kotsiantis
Department of Computer Science and Technology
University of Peloponnese, Greece
End of Karaiskaki, 22100 , Tripolis GR.
Tel: +30 2710 372164
Fax: +30 2710 372160
E-mail: sotos@math.upatras.gr

Overview paper

Keywords: classifiers, data mining techniques, intelligent data analysis, learning algorithms

Received: July 16, 2007

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown. This paper describes various supervised machine learning classification techniques. Of course, a single article cannot be a complete review of all supervised machine learning classification algorithms (also known induction classification algorithms), yet we hope that the references cited will cover the major theoretical issues, guiding the researcher in interesting research directions and suggesting possible bias combinations that have yet to be explored.

Povzetek: Podan je pregled metod strojnega učenja.

1 Introduction

There are several applications for Machine Learning (ML), the most significant of which is data mining. People are often prone to making mistakes during analyses or, possibly, when trying to establish relationships between multiple features. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines.

Every instance in any dataset used by machine learning algorithms is represented using the same set of features. The features may be continuous, categorical or binary. If instances are given with known labels (the corresponding correct outputs) then the learning is called supervised (see Table 1), in contrast to unsupervised learning, where instances are unlabeled. By applying these unsupervised (clustering) algorithms, researchers hope to discover unknown, but useful, classes of items (Jain et al., 1999). Another kind of machine learning is reinforcement learning (Barto & Sutton, 1997). The training information provided to the learning system by the environment (external trainer) is in the form of a scalar reinforcement signal that constitutes a measure of how well the system operates. The learner is not told which actions to take, but rather must discover which actions yield the best reward, by trying each action in turn.

Numerous ML applications involve tasks that can be set up as supervised. In the present paper, we have concentrated on the techniques necessary to do this. In particular, this work is concerned with classification problems in which the output of instances admits only discrete, unordered values.

Data in standard format					
case	Feature 1	Feature 2	...	Feature n	Class
1	xxx	x		xx	good
2	xxx	x		xx	good
3	xxx	x		xx	bad
...					...

Table 1. Instances with known labels (the corresponding correct outputs)

We have limited our references to recent refereed journals, published books and conferences. In addition, we have added some references regarding the original work that started the particular line of research under discussion. A brief review of what ML includes can be found in (Dutton & Conroy, 1996). De Mantaras and Armengol (1998) also presented a historical survey of logic and instance based learning classifiers. The reader should be cautioned that a single article cannot be a

comprehensive review of all classification learning algorithms. Instead, our goal has been to provide a representative sample of existing lines of research in each learning technique. In each of our listed areas, there are many other papers that more comprehensively detail relevant work.

Our next section covers wide-ranging issues of supervised machine learning such as data pre-processing and feature selection. Logical/Symbolic techniques are described in section 3, whereas perceptron-based techniques are analyzed in section 4. Statistical techniques for ML are covered in section 5. Section 6 deals with instance based learners, while Section 7 deals with the newest supervised ML technique—Support Vector Machines (SVMs). In section 8, some general directions are given about classifier selection. Finally, the last section concludes this work.

2 General issues of supervised learning algorithms

Inductive machine learning is the process of learning a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that can be used to generalize from new instances. The process of applying supervised ML to a real-world problem is described in Figure 1.

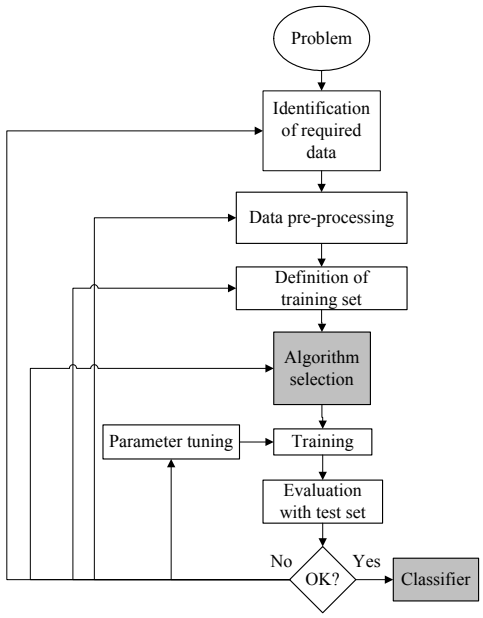

Figure 1. The process of supervised ML

The first step is collecting the dataset. If a requisite expert is available, then s/he could suggest which fields (attributes, features) are the most informative. If not, then the simplest method is that of "brute-force," which means measuring everything available in the hope that the right (informative, relevant) features can be isolated. However, a dataset collected by the "brute-force" method is not directly suitable for induction. It contains in most cases noise and missing feature values, and therefore requires significant pre-processing (Zhang et al., 2002).

The second step is the data preparation and data pre-processiong. Depending on the circumstances, researchers have a number of methods to choose from to handle missing data (Batista & Monard, 2003). Hodge & Austin (2004) have recently introduced a survey of contemporary techniques for outlier (noise) detection. These researchers have identified the techniques' advantages and disadvantages. Instance selection is not only used to handle noise but to cope with the infeasibility of learning from very large datasets. Instance selection in these datasets is an optimization problem that attempts to maintain the mining quality while minimizing the sample size (Liu and Motoda, 2001). It reduces data and enables a data mining algorithm to function and work effectively with very large datasets. There is a variety of procedures for sampling instances from a large dataset (Reinartz, 2002).

Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible (Yu & Liu, 2004). This reduces the dimensionality of the data and enables data mining algorithms to operate faster and more effectively. The fact that many features depend on one another often unduly influences the accuracy of supervised ML classification models. This problem can be addressed by constructing new features from the basic feature set (Markovitch & Rosenstein, 2002). This technique is called feature construction/transformation. These newly generated features may lead to the creation of more concise and accurate classifiers. In addition, the discovery of meaningful features contributes to better comprehensibility of the produced classifier, and a better understanding of the learned concept.

2.1 Algorithm selection

The choice of which specific learning algorithm we should use is a critical step. Once preliminary testing is judged to be satisfactory, the classifier (mapping from unlabeled instances to classes) is available for routine use. The classifier's evaluation is most often based on prediction accuracy (the percentage of correct prediction divided by the total number of predictions). There are at least three techniques which are used to calculate a classifier's accuracy. One technique is to split the training set by using two-thirds for training and the other third for estimating performance. In another technique, known as cross-validation, the training set is divided into mutually exclusive and equal-sized subsets and for each subset the classifier is trained on the union of all the other subsets. The average of the error rate of each subset is therefore an estimate of the error rate of the classifier. Leave-one-out validation is a special case of cross validation. All test subsets consist of a single instance. This type of validation is, of course, more expensive computationally, but useful when the most accurate estimate of a classifier's error rate is required.

If the error rate evaluation is unsatisfactory, we must return to a previous stage of the supervised ML process (as detailed in Figure 1). A variety of factors must be examined: perhaps relevant features for the problem are

not being used, a larger training set is needed, the dimensionality of the problem is too high, the selected algorithm is inappropriate or parameter tuning is needed. Another problem could be that the dataset is imbalanced (Japkowicz & Stephen, 2002).

A common method for comparing supervised ML algorithms is to perform statistical comparisons of the accuracies of trained classifiers on specific datasets. If we have sufficient supply of data, we can sample a number of training sets of size N, run the two learning algorithms on each of them, and estimate the difference in accuracy for each pair of classifiers on a large test set. The average of these differences is an estimate of the expected difference in generalization error across all possible training sets of size N, and their variance is an estimate of the variance of the classifier in the total set. Our next step is to perform paired t-test to check the null hypothesis that the mean difference between the classifiers is zero. This test can produce two types of errors. Type I error is the probability that the test rejects the null hypothesis incorrectly (i.e. it finds a "significant" difference although there is none). Type II error is the probability that the null hypothesis is not rejected, when there actually is a difference. The test's Type I error will be close to the chosen significance level.

In practice, however, we often have only one dataset of size N and all estimates must be obtained from this sole dataset. Different training sets are obtained by sub-sampling, and the instances not sampled for training are used for testing. Unfortunately this violates the independence assumption necessary for proper significance testing. The consequence of this is that Type I errors exceed the significance level. This is problematic because it is important for the researcher to be able to control Type I errors and know the probability of incorrectly rejecting the null hypothesis. Several heuristic versions of the t-test have been developed to alleviate this problem (Dietterich, 1998), (Nadeau and Bengio, 2003).

Ideally, we would like the test's outcome to be independent of the particular partitioning resulting from the randomization process, because this would make it much easier to replicate experimental results published in the literature. However, in practice there is always certain sensitivity to the partitioning used. To measure replicability we need to repeat the same test several times on the same data with different random partitionings — usually ten repetitions— and count how often the outcome is the same (Bouckaert, 2003).

Supervised classification is one of the tasks most frequently carried out by so-called Intelligent Systems. Thus, a large number of techniques have been developed based on Artificial Intelligence (Logical/Symbolic techniques), Perceptron-based techniques and Statistics (Bayesian Networks, Instance-based techniques). In next sections, we will focus on the most important supervised machine learning techniques, starting with logical/symbolic algorithms.

3 Logic based algorithms

In this section we will concentrate on two groups of logical (symbolic) learning methods: decision trees and rule-based classifiers.

3.1 Decision trees

Murthy (1998) provided an overview of work in decision trees and a sample of their usefulness to newcomers as well as practitioners in the field of machine learning. Thus, in this work, apart from a brief description of decision trees, we will refer to some more recent works than those in Murthy's article as well as few very important articles that were published earlier. Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values. Figure 2 is an example of a decision tree for the training set of Table 2.

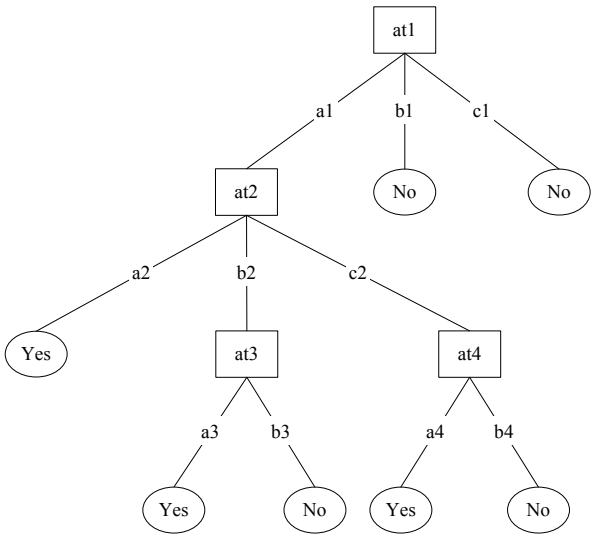

Figure 2. A decision tree

at1	at2	at3	at4	Class
a1	a2	a3	a4	Yes
a1	a2	a3	b4	Yes
a1	b2	a3	a4	Yes
a1	b2	b3	b4	No
a1	c2	a3	a4	Yes
a1	c2	a3	b4	No
b1	b2	b3	b4	No
c1	b2	b3	b4	No

Table 2. Training Set

Using the decision tree depicted in Figure 2 as an example, the instance ⟨at1 = a1, at2 = b2, at3 = a3, at4 = b4⟩ would sort to the nodes: at1, at2, and finally at3, which would classify the instance as being positive

(represented by the values "Yes"). The problem of constructing optimal binary decision trees is an NP-complete problem and thus theoreticians have searched for efficient *heuristics* for constructing near-optimal decision trees.

The feature that best divides the training data would be the root node of the tree. There are numerous methods for finding the feature that best divides the training data such as information gain (Hunt et al., 1966) and gini index (Breiman et al., 1984). While myopic measures estimate each attribute independently, ReliefF algorithm (Kononenko, 1994) estimates them in the context of other attributes. However, a majority of studies have concluded that there is no single best method (Murthy, 1998). Comparison of individual methods may still be important when deciding which metric should be used in a particular dataset. The same procedure is then repeated on each partition of the divided data, creating sub-trees until the training data is divided into subsets of the same class.

Figure 3 presents a general pseudo-code for building decision trees.

```
Check for base cases
 For each attribute a
    Find the feature that best
    divides the training data such
    as information gain from
    splitting on a
Let a best be the attribute with the
highest normalized information gain
   Create a decision node node that
   splits on a_best
Recurse on the sub-lists obtained by
splitting on a best and add those
nodes as children of node
```

Figure 3. Pseudo-code for building a decision tree

A decision tree, or any learned hypothesis h, is said to overfit training data if another hypothesis h' exists that has a larger error than h when tested on the training data, but a smaller error than h when tested on the entire dataset. There are two common approaches that decision tree induction algorithms can use to avoid overfitting training data: i) Stop the training algorithm before it reaches a point at which it perfectly fits the training data, ii) Prune the induced decision tree. If the two trees employ the same kind of tests and have the same prediction accuracy, the one with fewer leaves is usually preferred. Breslow & Aha (1997) survey methods of tree simplification to improve their comprehensibility.

The most straightforward way of tackling overfitting is to pre-prune the decision tree by not allowing it to grow to its full size. Establishing a non-trivial termination criterion such as a threshold test for the feature quality metric can do that. Decision tree classifiers usually employ post-pruning techniques that evaluate the performance of decision trees, as they are pruned by using a validation set. Any node can be removed and assigned the most common class of the training instances that are sorted to it. A comparative study of well-known pruning methods is presented in (Elomaa, 1999). Elomaa (1999) concluded that there is no single best pruning method. More details, about not only postprocessing but also about preprocessing of decision tree algorithms can be fould in (Bruha, 2000).

Even though the divide-and-conquer algorithm is quick, efficiency can become important in tasks with hundreds of thousands of instances. The most time-consuming aspect is sorting the instances on a numeric feature to find the best threshold t. This can be expedited if possible thresholds for a numeric feature are determined just once, effectively converting the feature to discrete intervals, or if the threshold is determined from a subset of the instances. Elomaa & Rousu (1999) stated that the use of binary discretization with C4.5 needs about the half training time of using C4.5 multi-splitting. In addition, according to their experiments, multi-splitting of numerical features does not carry any advantage in prediction accuracy over binary splitting.

Decision trees are usually univariate since they use splits based on a single feature at each internal node. Most decision tree algorithms cannot perform well with problems that require diagonal partitioning. The division of the instance space is orthogonal to the axis of one variable and parallel to all other axes. Therefore, the resulting regions after partitioning are all hyper-rectangles. However, there are a few methods that construct multivariate trees. One example is Zheng's (1998), who improved the classification accuracy of the decision trees by constructing new binary features with logical operators such as conjunction, negation, and disjunction. In addition, Zheng (2000) created at-least M-of-N features. For a given instance, the value of an at-least M-of-N representation is true if at least M of its conditions is true of the instance, otherwise it is false. Gama and Brazdil (1999) combined a decision tree with a linear discriminant for constructing multivariate decision trees. In this model, new features are computed as linear combinations of the previous ones.

Decision trees can be significantly more complex representation for some concepts due to the replication problem. A solution is using an algorithm to implement complex features at nodes in order to avoid replication. Markovitch and Rosenstein (2002) presented the FICUS construction algorithm, which receives the standard input of supervised learning as well as a feature representation specification, and uses them to produce a set of generated features. While FICUS is similar in some aspects to other feature construction algorithms, its main strength is its generality and flexibility. FICUS was designed to perform feature generation given any feature representation specification complying with its general purpose grammar.

The most well-know algorithm in the literature for building decision trees is the C4.5 (Quinlan, 1993). C4.5 is an extension of Quinlan's earlier ID3 algorithm (Quinlan, 1979). One of the latest studies that compare decision trees and other learning algorithms has been done by (Tjen-Sien Lim et al. 2000). The study shows that C4.5 has a very good combination of error rate and speed. In 2001, Ruggieri presented an analytic evaluation of the runtime behavior of the C4.5 algorithm, which highlighted some efficiency improvements. Based on this

analytic evaluation, he implemented a more efficient version of the algorithm, called EC4.5. He argued that his implementation computed the same decision trees as C4.5 with a performance gain of up to five times.

C4.5 assumes that the training data fits in memory, thus, Gehrke et al. (2000) proposed Rainforest, a framework for developing fast and scalable algorithms to construct decision trees that gracefully adapt to the amount of main memory available. It is clear that in most decision tree algorithms; a substantial effort is "wasted" in the building phase on growing portions of the tree that are subsequently pruned in the pruning phase. Rastogi & Shim (2000) proposed PUBLIC, an improved decision tree classifier that integrates the second "pruning" phase with the initial "building" phase. In PUBLIC, a node is not expanded during the building phase, if it is determined that the node will be pruned during the subsequent pruning phase.

Olcay and Onur (2007) show how to parallelize C4.5 algorithm in three ways: (i) feature based, (ii) node based (iii) data based manner. Baik and Bala (2004) presented preliminary work on an agent-based approach for the distributed learning of decision trees.

To sum up, one of the most useful characteristics of decision trees is their comprehensibility. People can easily understand why a decision tree classifies an instance as belonging to a specific class. Since a decision tree constitutes a hierarchy of tests, an unknown feature value during classification is usually dealt with by passing the example down all branches of the node where the unknown feature value was detected, and each branch outputs a class distribution. The output is a combination of the different class distributions that sum to 1. The assumption made in the decision trees is that instances belonging to different classes have different values in at least one of their features. Decision trees tend to perform better when dealing with discrete/categorical features.

3.2 Learning set of rules

Decision trees can be translated into a set of rules by creating a separate rule for each path from the root to a leaf in the tree (Quinlan, 1993). However, rules can also be directly induced from training data using a variety of rule-based algorithms. Furnkranz (1999) provided an excellent overview of existing work in rule-based methods.

Classification rules represent each class by disjunctive normal form (DNF). A k-DNF expression is of the form: $(X_1 \wedge X_2 \wedge \ldots \wedge X_n) \vee (X_{n+1} \wedge X_{n+2} \wedge \ldots X_{2n}) \vee \ldots \vee (X_{(k-1)n+1} \wedge X_{(k-1)n+2} \wedge \ldots \wedge X_{kn})$, where k is the number of disjunctions, n is the number of conjunctions in each disjunction, and X_n is defined over the alphabet $X_1, X_2, \ldots, X_j \cup \sim X_1, \sim X_2, \ldots, \sim X_j$. The goal is to construct the smallest rule-set that is consistent with the training data. A large number of learned rules is usually a sign that the learning algorithm is attempting to "remember" the training set, instead of discovering the assumptions that govern it. A separate-and-conquer algorithm (covering algorithms) search for a rule that explains a part of its training instances, separates these instances and recursively conquers the remaining instances by learning more rules, until no instances remain. In Figure 4, a general pseudo-code for rule learners is presented.

The difference between heuristics for rule learning and heuristics for decision trees is that the latter evaluate the average quality of a number of disjointed sets (one for each value of the feature that is tested), while rule learners only evaluate the quality of the set of instances that is covered by the candidate rule. More advanced rule learners differ from this simple pseudo-code mostly by adding additional mechanisms to prevent over-fitting of the training data, for instance by stopping the specialization process with the use of a quality measure or by generalizing overly specialized rules in a separate pruning phase (Furnkranz, 1997).

```
   On presentation of training examples
training examples:
   1. Initialise rule set to a default
(usually empty, or a rule assigning all
objects to the most common class).
   2. Initialise examples to either all
available examples or all examples not
correctly handled by rule set.
   3. Repeat
      (a) Find best, the best rule with
   respect to examples.
      (b) If such a rule can be found
         i. Add best to rule set.
         ii.  Set  examples  to  all
         examples          not  handled
         correctly by rule set.
       until no rule best can be found
       (for   instance,   because   no
       examples remain).
```

Figure 4. Pseudocode for rule learners

It is therefore important for a rule induction system to generate decision rules that have high predictability or reliability. These properties are commonly measured by a function called rule quality. A rule quality measure is needed in both the rule induction and classification processes such as J-measure (Smyth and Goodman, 1990). In rule induction, a rule quality measure can be used as a criterion in the rule specification and/or generalization process. In classification, a rule quality value can be associated with each rule to resolve conflicts when multiple rules are satisfied by the example to be classified. An and Cercone (2000) surveyed a number of statistical and empirical rule quality measures. Furnkranz and Flach (2005) provided an analysis of the behavior of separate-and-conquer or covering rule learning algorithms by visualizing their evaluation metrics. When using unordered rule sets, conflicts can arise between the rules, i.e., two or more rules cover the same example but predict different classes. Lindgren (2004) has recently given a survey of methods used to solve this type of conflict.

RIPPER is a well-known rule-based algorithm (Cohen, 1995). It forms rules through a process of repeated *growing* and *pruning*. During the growing phase the rules are made more restrictive in order to fit the training data as closely as possible. During the pruning phase, the rules are made less restrictive in order to avoid

overfitting, which can cause poor performance on unseen instances. RIPPER handles multiple classes by ordering them from least to most prevalent and then treating each in order as a distinct two-class problem. Other fundamental learning classifiers based on decision rules include the AQ family (Michalski and Chilausky, 1980) and CN2 (Clark and Niblett, 1989). Bonarini (2000) gave an overview of fuzzy rule-based classifiers. Fuzzy logic tries to improve classification and decision support systems by allowing the use of overlapping class definitions.

Furnkranz (2001) investigated the use of round robin binarization (or pairwise classification) as a technique for handling multi-class problems with separate and conquer rule learning algorithms. The round robin binarization transforms a c-class problem into c(c-1)/2 two-class problems <i,j>, one for each set of classes {i,j}, i= 1 ... c-1, j = i+1 ...c. The binary classifier for problem <i,j> is trained with examples of classes i and j, whereas examples of classes k ≠ i,j are ignored for this problem. A crucial point, of course, is determining how to decode the predictions of the pairwise classifiers for a final prediction. Furnkranz (2001) implemented a simple voting technique: when classifying a new example, each of the learned base classifiers determines to which of its two classes the example is more likely to belong to. The winner is assigned a point, and in the end, the algorithm predicts the class that has accumulated the most points. His experimental results show that, in comparison to conventional, ordered or unordered binarization, the round robin approach may yield significant gains in accuracy without risking a poor performance.

There are numerous other rule-based learning algorithms. Furnkranz (1999) referred to most of them. The PART algorithm infers rules by repeatedly generating partial decision trees, thus combining the two major paradigms for rule generation − creating rules from decision trees and the separate-and-conquer rule-learning technique. Once a partial tree has been build, a single rule is extracted from it and for this reason the PART algorithm avoids postprocessing (Frank and Witten, 1998).

For the task of learning binary problems, rules are more comprehensible than decision trees because typical rule-based approaches learn a set of rules for only the positive class. On the other hand, if definitions for multiple classes are to be learned, the rule-based learner must be run separately for each class separately. For each individual class a separate rule set is obtained and these sets may be inconsistent (a particular instance might be assigned multiple classes) or incomplete (no class might be assigned to a particular instance). These problems can be solved with decision lists (the rules in a rule set are supposed to be ordered, a rule is only applicable when none of the preceding rules are applicable) but with the decision tree approach, they simply do not occur. Moreover, the divide and conquer approach (used by decision trees) is usually more efficient than the separate and conquer approach (used by rule-based algorithms). Separate-and-conquer algorithms look at one class at a time, and try to produce rules that uniquely identify the class. They do this independent of all the other classes in the training set. For this reason, for small datasets, it may be better to use a divide-and-conquer algorithm that considers the entire set at once.

To sum up, the most useful characteristic of rule-based classifiers is their comprehensibility. In addition, even though some rule-based classifiers can deal with numerical features, some experts propose these features should be discretized before induction, so as to reduce training time and increase classification accuracy (An and Cercone, 1999). Classification accuracy of rule learning algorithms can be improved by combining features (such as in decision trees) using the background knowledge of the user (Flach and Lavrac, 2000) or automatic feature construction algorithms (Markovitch and Rosenstein, 2002).

4 Perceptron-based techniques

Other well-known algorithms are based on the notion of perceptron (Rosenblatt, 1962).

4.1 Single layered perceptrons

A single layered perceptron can be briefly described as follows:

If x_1 through x_n are input feature values and w_1 through w_n are connection weights/prediction vector (typically real numbers in the interval [-1, 1]), then perceptron computes the sum of weighted inputs: $\sum_i x_i w_i$ and output goes through an adjustable threshold: if the sum is above threshold, output is 1; else it is 0.

The most common way that the perceptron algorithm is used for learning from a batch of training instances is to run the algorithm repeatedly through the training set until it finds a prediction vector which is correct on all of the training set. This prediction rule is then used for predicting the labels on the test set.

WINNOW (Littlestone & Warmuth, 1994) is based on the perceptron idea and updates its weights as follows. If prediction value $y'=0$ and actual value $y=1$, then the weights are too low; so, for each feature such that $x_i=1$, $w_i=w_i \cdot \alpha$, where α is a number greater than 1, called the *promotion parameter*. If prediction value $y'= 1$ and actual value $y=0$, then the weights were too high; so, for each feature $x_i = 1$, it decreases the corresponding weight by setting $w_i=w_i \cdot \beta$, where $0<\beta<1$, called the *demotion parameter*. Generally, WINNOW is an example of an *exponential update algorithm*. The weights of the relevant features grow exponentially but the weights of the irrelevant features shrink exponentially. For this reason, it was experimentally proved (Blum, 1997) that WINNOW can adapt rapidly to changes in the target function (concept drift). A target function (such as user preferences) is not static in time. In order to enable, for example, a decision tree algorithm to respond to changes, it is necessary to decide which old training instances could be deleted. A number of algorithms similar to

WINNOW have been developed, such as those by Auer & Warmuth (1998).

Freund & Schapire (1999) created a newer algorithm, called *voted-perceptron*, which stores more information during training and then uses this elaborate information to generate better predictions about the test data. The information it maintains during training is the list of *all* prediction vectors that were generated after each and every mistake. For each such vector, it counts the number of iterations it "survives" until the next mistake is made; Freund & Schapire refer to this count as the "weight" of the prediction vector. To calculate a prediction the algorithm computes the binary prediction of each one of the prediction vectors and combines all these predictions by means of a weighted majority vote. The weights used are the survival times described above.

To sum up, we have discussed perceptron-like linear algorithms with emphasis on their superior time complexity when dealing with irrelevant features. This can be a considerable advantage when there are many features, but only a few relevant ones. Generally, all perceptron-like linear algorithms are *anytime online algorithms* that can produce a useful answer regardless of how long they run (Kivinen, 2002). The longer they run, the better the result they produce. Finally, perceptron-like methods are binary, and therefore in the case of multi-class problem one must reduce the problem to a set of multiple binary classification problems.

4.2 Multilayered perceptrons

Perceptrons can only classify linearly separable sets of instances. If a straight line or plane can be drawn to seperate the input instances into their correct categories, input instances are linearly separable and the perceptron will find the solution. If the instances are not linearly separable learning will never reach a point where all instances are classified properly. Multilayered Perceptrons (Artificial Neural Networks) have been created to try to solve this problem (Rumelhart et al., 1986). Zhang (2000) provided an overview of existing work in Artificial Neural Networks (ANNs). Thus, in this study, apart from a brief description of the ANNs we will mainly refer to some more recent articles. A multi-layer neural network consists of large number of units (neurons) joined together in a pattern of connections (Figure 5). Units in a net are usually segregated into three classes: input units, which receive information to be processed; output units, where the results of the processing are found; and units in between known as hidden units. Feed-forward ANNs (Figure 5) allow signals to travel one way only, from input to output.

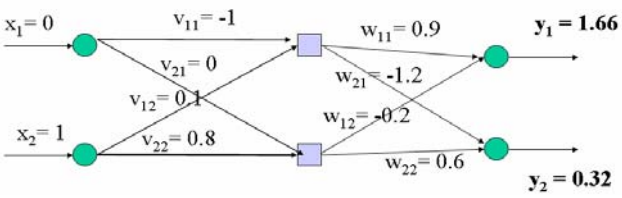

Figure 5. Feed-forward ANN

First, the network is trained on a set of paired data to determine input-output mapping. The weights of the connections between neurons are then fixed and the network is used to determine the classifications of a new set of data.

During classification the signal at the input units propagates all the way through the net to determine the activation values at all the output units. Each input unit has an activation value that represents some feature external to the net. Then, every input unit sends its activation value to each of the hidden units to which it is connected. Each of these hidden units calculates its own activation value and this signal are then passed on to output units. The activation value for each receiving unit is calculated according to a simple activation function. The function sums together the contributions of all sending units, where the contribution of a unit is defined as the weight of the connection between the sending and receiving units multiplied by the sending unit's activation value. This sum is usually then further modified, for example, by adjusting the activation sum to a value between 0 and 1 and/or by setting the activation value to zero unless a threshold level for that sum is reached.

Generally, properly determining the size of the hidden layer is a problem, because an underestimate of the number of neurons can lead to poor approximation and generalization capabilities, while excessive nodes can result in overfitting and eventually make the search for the global optimum more difficult. An excellent argument regarding this topic can be found in (Camargo & Yoneyama, 2001). Kon & Plaskota (2000) also studied the minimum amount of neurons and the number of instances necessary to program a given task into feed-forward neural networks.

ANN depends upon three fundamental aspects, input and activation functions of the unit, network architecture and the weight of each input connection. Given that the first two aspects are fixed, the behavior of the ANN is defined by the current values of the weights. The weights of the net to be trained are initially set to random values, and then instances of the training set are repeatedly exposed to the net. The values for the input of an instance are placed on the input units and the output of the net is compared with the desired output for this instance. Then, all the weights in the net are adjusted slightly in the direction that would bring the output values of the net closer to the values for the desired output. There are several algorithms with which a network can be trained (Neocleous & Schizas, 2002). However, the most well-known and widely used learning algorithm to estimate the values of the weights is the Back Propagation (BP) algorithm. Generally, BP algorithm includes the following six steps:

1. Present a training sample to the neural network.
2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error.

4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat the steps above on the neurons at the previous level, using each one's "blame" as its error.

With more details, the general rule for updating weights is: $\Delta W_{ji} = \eta \delta_j O_i$ where:

- η is a positive number (called learning rate), which determines the step size in the gradient descent search. A large value enables back propagation to move faster to the target weight configuration but it also increases the chance of its never reaching this target.
- O_i is the output computed by neuron i
- $\delta_j = O_j(1-O_j)(T_j - O_j)$ for the output neurons, where T_j the wanted output for the neuron j and
- $\delta_j = O_j(1-O_j)\sum_k \delta_k W_{kj}$ for the internal (hidden) neurons

The back propagation algorithm will have to perform a number of weight modifications before it reaches a good weight configuration. For n training instances and W weights, each repetition/epoch in the learning process takes O(nW) time; but in the worst case, the number of epochs can be exponential to the number of inputs. For this reason, neural nets use a number of different *stopping rules* to control when training ends. The four most common stopping rules are: i) Stop after a specified number of epochs, ii) Stop when an error measure reaches a threshold, iii) Stop when the error measure has seen no improvement over a certain number of epochs, iv) Stop when the error measure on some of the data that has been sampled from the training data (hold-out set, validation set) is more than a certain amount than the error measure on the training set (overfitting).

Feed-forward neural networks are usually trained by the original back propagation algorithm or by some variant. Their greatest problem is that they are too slow for most applications. One of the approaches to speed up the training rate is to estimate optimal initial weights (Yam & Chow, 2001). Another method for training multilayered feedforward ANNs is Weight-elimination algorithm that automatically derives the appropriate topology and therefore avoids also the problems with overfitting (Weigend et al., 1991). Genetic algorithms have been used to train the weights of neural networks (Siddique and Tokhi, 2001) and to find the architecture of neural networks (Yen and Lu, 2000). There are also Bayesian methods in existence which attempt to train neural networks. Vivarelli & Williams (2001) compare two Bayesian methods for training neural networks. A number of other techniques have emerged recently which attempt to improve ANNs training algorithms by changing the architecture of the networks as training proceeds. These techniques include *pruning* useless nodes or weights (Castellano et al. 1997), and *constructive algorithms,* where extra nodes are added as required (Parekh et al. 2000).

4.3 Radial Basis Function (RBF) networks

ANN learning can be achieved, among others, through i) synaptic weight modification, ii) network structure modifications (creating or deleting neurons or synaptic connections), iii) use of suitable attractors or other suitable stable state points, iv) appropriate choice of activation functions. Since back-propagation training is a gradient descending process, it may get stuck in local minima in this weight-space. It is because of this possibility that neural network models are characterized by high variance and unsteadiness.

Radial Basis Function (RBF) networks have been also widely applied in many science and engineering fields (Robert and Howlett, 2001). An RBF network is a three-layer feedback network, in which each hidden unit implements a radial activation function and each output unit implements a weighted sum of hidden units outputs. Its training procedure is usually divided into two stages. First, the centers and widths of the hidden layer are determined by clustering algorithms. Second, the weights connecting the hidden layer with the output layer are determined by Singular Value Decomposition (SVD) or Least Mean Squared (LMS) algorithms. The problem of selecting the appropriate number of basis functions remains a critical issue for RBF networks. The number of basis functions controls the complexity and the generalization ability of RBF networks. RBF networks with too few basis functions cannot fit the training data adequately due to limited flexibility. On the other hand, those with too many basis functions yield poor generalization abilities since they are too flexible and erroneously fit the noise in the training data.

Even though multilayer neural networks and decision trees are two very different techniques for the purpose of classification, some researchers (Eklund & Hoang, 2002), (Tjen-Sien Lim et al. 2000) have performed some empirical comparative studies. Some of the general conclusions drawn in that work are:

i) neural networks are usually more able to easily provide incremental learning than decision trees (Saad, 1998), even though there are some algorithms for incremental learning of decision trees such as (Utgoff et al, 1997) and (McSherry, 1999). Incremental decision tree induction techniques result in frequent tree restructuring when the amount of training data is small, with the tree structure maturing as the data pool becomes larger.
ii) training time for a neural network is usually much longer than training time for decision trees.
iii) neural networks usually perform as well as decision trees, but seldom better.

To sum up, ANNs have been applied to many real-world problems but still, their most striking disadvantage is their lack of ability to reason about their output in a

way that can be effectively communicated. For this reason many researchers have tried to address the issue of improving the comprehensibility of neural networks, where the most attractive solution is to extract symbolic rules from trained neural networks. Setiono and Leow (2000) divided the activation values of relevant hidden units into two subintervals and then found the set of relevant connections of those relevant units to construct rules. More references can be found in (Zhou, 2004), an excellent survey. However, it is also worth mentioning that Roy (2000) identified the conflict between the idea of rule extraction and traditional connectionism. In detail, the idea of rule extraction from a neural network involves certain procedures, specifically the reading of parameters from a network, which is not allowed by the traditional connectionist framework that these neural networks are based on.

5 Statistical learning algorithms

Conversely to ANNs, statistical approaches are characterized by having an explicit underlying probability model, which provides a probability that an instance belongs in each class, rather than simply a classification. Linear discriminant analysis (LDA) and the related Fisher's linear discriminant are simple methods used in statistics and machine learning to find the linear combination of features which best separate two or more classes of object (Friedman, 1989). LDA works when the measurements made on each observation are continuous quantities. When dealing with categorical variables, the equivalent technique is Discriminant Correspondence Analysis (Mika et al., 1999).

Maximum entropy is another general technique for estimating probability distributions from data. The over-riding principle in maximum entropy is that when nothing is known, the distribution should be as uniform as possible, that is, have maximal entropy. Labeled training data is used to derive a set of constraints for the model that characterize the class-specific expectations for the distribution. Csiszar (1996) provides a good tutorial introduction to maximum entropy techniques.

Bayesian networks are the most well known representative of statistical learning algorithms. A comprehensive book on Bayesian networks is Jensen's (1996). Thus, in this study, apart from our brief description of Bayesian networks, we mainly refer to more recent works.

5.1.1 Naive Bayes classifiers

Naive Bayesian networks (NB) are very simple Bayesian networks which are composed of directed acyclic graphs with only one parent (representing the unobserved node) and several children (corresponding to observed nodes) with a strong assumption of independence among child nodes in the context of their parent (Good, 1950).Thus, the independence model (Naive Bayes) is based on estimating (Nilsson, 1965):

$$R=\frac{P(i\mid X)}{P(j\mid X)}=\frac{P(i)P(X\mid i)}{P(j)P(X\mid j)}=\frac{P(i)\prod P(X_r\mid i)}{P(j)\prod P(X_r\mid j)}$$

Comparing these two probabilities, the larger probability indicates that the class label value that is more likely to be the actual label (if R>1: predict i, else predict j). Cestnik et al (1987) first used the Naive Bayes in ML community. Since the Bayes classification algorithm uses a product operation to compute the probabilities P(X, i), it is especially prone to being unduly impacted by probabilities of 0. This can be avoided by using Laplace estimator or m-esimate, by adding one to all numerators and adding the number of added ones to the denominator (Cestnik, 1990).

The assumption of independence among child nodes is clearly almost always wrong and for this reason naive Bayes classifiers are usually less accurate that other more sophisticated learning algorithms (such ANNs). However, Domingos & Pazzani (1997) performed a large-scale comparison of the naive Bayes classifier with state-of-the-art algorithms for decision tree induction, instance-based learning, and rule induction on standard benchmark datasets, and found it to be sometimes superior to the other learning schemes, even on datasets with substantial feature dependencies.

The basic independent Bayes model has been modified in various ways in attempts to improve its performance. Attempts to overcome the independence assumption are mainly based on adding extra edges to include some of the dependencies between the features, for example (Friedman et al. 1997). In this case, the network has the limitation that each feature can be related to only one other feature. Semi-naive Bayesian classifier is another important attempt to avoid the independence assumption. (Kononenko, 1991), in which attributes are partitioned into groups and it is assumed that x_i is conditionally independent of x_j if and only if they are in different groups.

The major advantage of the naive Bayes classifier is its short computational time for training. In addition, since the model has the form of a product, it can be converted into a sum through the use of logarithms - with significant consequent computational advantages. If a feature is numerical, the usual procedure is to discretize it during data pre-processing (Yang & Webb, 2003), although a researcher can use the normal distribution to calculate probabilities (Bouckaert, 2004).

5.2 Bayesian Networks

A Bayesian Network (BN) is a graphical model for probability relationships among a set of variables (features) (see Figure 6). The Bayesian network structure S is a directed acyclic graph (DAG) and the nodes in S are in one-to-one correspondence with the features X. The arcs represent casual influences among the features while the *lack* of possible arcs in S encodes conditional independencies. Moreover, a feature (node) is conditionally independent from its non-descendants given its parents (X_1 is conditionally independent from X_2

given X_3 if $P(X_1|X_2,X_3)=P(X_1|X_3)$ for all possible values of X_1, X_2, X_3).

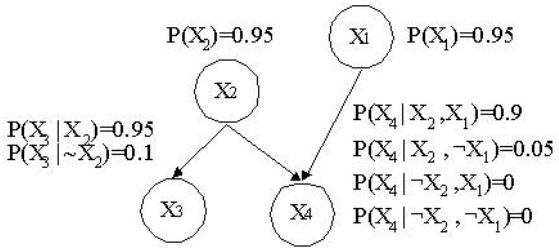

Figure 6. The structure of a Bayes Network

Typically, the task of learning a Bayesian network can be divided into two subtasks: initially, the learning of the DAG structure of the network, and then the determination of its parameters. Probabilistic parameters are encoded into a set of tables, one for each variable, in the form of local conditional distributions of a variable given its parents. Given the independences encoded into the network, the joint distribution can be reconstructed by simply multiplying these tables. Within the general framework of inducing Bayesian networks, there are two scenarios: known structure and unknown structure.

In the first scenario, the structure of the network is given (e.g. by an expert) and assumed to be correct. Once the network structure is fixed, learning the parameters in the Conditional Probability Tables (CPT) is usually solved by estimating a locally exponential number of parameters from the data provided (Jensen, 1996). Each node in the network has an associated CPT that describes the conditional probability distribution of that node given the different values of its parents.

In spite of the remarkable power of Bayesian Networks, they have an inherent limitation. This is the computational difficulty of exploring a previously unknown network. Given a problem described by n features, the number of possible structure hypotheses is more than exponential in n. If the structure is unknown, one approach is to introduce a scoring function (or a score) that evaluates the "fitness" of networks with respect to the training data, and then to search for the best network according to this score. Several researchers have shown experimentally that the selection of a single good hypothesis using greedy search often yields accurate predictions (Heckerman et al. 1999), (Chickering, 2002). In Figure 7 there is a pseudo-code for training BNs.

Within the score & search paradigm, another approach uses local search methods in the space of directed acyclic graphs, where the usual choices for defining the elementary modifications (local changes) that can be applied are arc addition, arc deletion, and arc reversal. Acid and de Campos (2003) proposed a new local search method, restricted acyclic partially directed graphs, which uses a different search space and takes account of the concept of equivalence between network structures. In this way, the number of different configurations of the search space is reduced, thus improving efficiency.

```
   Initialize an empty Bayesian network
G containing n nodes (i.e., a BN with n
nodes but no edges)
1. Evaluate the score of G: Score(G)
2. G′ = G
3. for i = 1 to n do
4. for j = 1 to n do
5. if i • j then
6. if there is no edge between the
   nodes i and j in G• then
7. Modify G′ by adding an edge between
   the nodes i and j in G• such that i
   is a parent of j: (i • j)
8. if the resulting G′ is a DAG then
9. if (Score(G′) > Score(G)) then
10. G = G′
11. end if
12. end if
13. end if
14. end if
15. G′ = G
16. end for
17. end for
```

Figure 7. Pseudo-code for training BN

A BN structure can be also found by learning the conditional independence relationships among the features of a dataset. Using a few statistical tests (such as the Chi-squared and mutual information test), one can find the conditional independence relationships among the features and use these relationships as constraints to construct a BN. These algorithms are called *CI-based* algorithms or constraint-based algorithms. Cowell (2001) has shown that for any structure search procedure based on CI tests, an equivalent procedure based on maximizing a score can be specified.

A comparison of scoring-based methods and CI-based methods is presented in (Heckerman et al., 1999). Both of these approaches have their advantages and disadvantages. Generally speaking, the dependency analysis approach is more efficient than the search & scoring approach for sparse networks (networks that are not densely connected). It can also deduce the correct structure when the probability distribution of the data satisfies certain assumptions. However, many of these algorithms require an exponential number of CI tests and many high order CI tests (CI tests with large condition-sets). Yet although the search & scoring approach may not find the best structure due to its heuristic nature, it works with a wider range of probabilistic models than the dependency analysis approach. Madden (2003) compared the performance of a number of Bayesian Network Classifiers. His experiments demonstrated that very similar classification performance can be achieved by classifiers constructed using the different approaches described above.

The most generic learning scenario is when the structure of the network is unknown and there is missing data. Friedman & Koller (2003) proposed a new approach for this task and showed how to efficiently compute a sum over the exponential number of networks that are consistent with a fixed order over networks.

Using a suitable version of any of the model types mentioned in this review, one can induce a Bayesian Network from a given training set. A classifier based on the network and on the given set of features X_1,X_2, ... X_n,

returns the label *c,* which maximizes the posterior probability p($c \mid X_1, X_2, \ldots X_n$).

Bayesian multi-nets allow different probabilistic dependencies for different values of the class node (Jordan, 1998). This suggests that simple BN classifiers should work better when there is a single underlying model of the dataset and multi-net classifier should work better when the underlying relationships among the features are very different for different classes (Cheng and Greiner, 2001).

The most interesting feature of BNs, compared to decision trees or neural networks, is most certainly the possibility of taking into account prior information about a given problem, in terms of structural relationships among its features. This prior expertise, or domain knowledge, about the structure of a Bayesian network can take the following forms:

1. Declaring that a node is a root node, i.e., it has no parents.
2. Declaring that a node is a leaf node, i.e., it has no children.
3. Declaring that a node is a direct cause or direct effect of another node.
4. Declaring that a node is not directly connected to another node.
5. Declaring that two nodes are independent, given a condition-set.
6. Providing partial nodes ordering, that is, declare that a node appears earlier than another node in the ordering.
7. Providing a complete node ordering.

A problem of BN classifiers is that they are not suitable for datasets with many features (Cheng et al., 2002). The reason for this is that trying to construct a very large network is simply not feasible in terms of time and space. A final problem is that before the induction, the numerical features need to be discretized in most cases.

6 Instance-based learning

Another category under the header of statistical methods is Instance-based learning. Instance-based learning algorithms are lazy-learning algorithms (Mitchell, 1997), as they delay the induction or generalization process until classification is performed. Lazy-learning algorithms require less computation time during the training phase than eager-learning algorithms (such as decision trees, neural and Bayes nets) but more computation time during the classification process. One of the most straightforward instance-based learning algorithms is the *nearest neighbour* algorithm. Aha (1997) and De Mantaras and Armengol (1998) presented a review of instance-based learning classifiers. Thus, in this study, apart from a brief description of the *nearest neighbour* algorithm, we will refer to some more recent works.

k-Nearest Neighbour (kNN) is based on the principle that the instances within a dataset will generally exist in close proximity to other instances that have similar properties (Cover and Hart, 1967). If the instances are tagged with a classification label, then the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbours. The kNN locates the *k* nearest instances to the query instance and determines its class by identifying the single most frequent class label. In Figure 8, a pseudo-code example for the instance base learning methods is illustrated.

```
procedure    InstanceBaseLearner(Testing
Instances)
    for each testing instance
    {
    find the k most nearest instances of
    the training set according to a
    distance metric
    Resulting Class= most frequent class
    label of the k nearest instances
    }
```

Figure 8. Pseudo-code for instance-based learners

In general, instances can be considered as points within an *n*-dimensional instance space where each of the *n*-dimensions corresponds to one of the *n*-features that are used to describe an instance. The absolute position of the instances within this space is not as significant as the relative distance between instances. This relative distance is determined by using a distance metric. Ideally, the distance metric must minimize the distance between two similarly classified instances, while maximizing the distance between instances of different classes. Many different metrics have been presented. The most significant ones are presented in Table 3.

Minkowsky: $D(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^r\right)^{1/r}$
Manhattan: $D(x,y)=\sum_{i=1}^{m}\lvert x_i-y_i\rvert$
Chebychev: $D(x,y)=\max_{i=1}^{m}\lvert x_i-y_i\rvert$
Euclidean: $D(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^2\right)^{1/2}$
Camberra: $D(x,y)=\sum_{i=1}^{m}\frac{\lvert x_i-y_i\rvert}{\lvert x_i+y_i\rvert}$
Kendall's Rank Correlation: $D(x,y)=1-\frac{2}{m(m-1)}\sum_{i=j}^{m}\sum_{j=1}^{i-1}sign(x_i-x_j)sign(y_i-y_j)$

Table 3. Approaches to define the distance between instances (x and y)

For more accurate results, several algorithms use weighting schemes that alter the distance measurements and voting influence of each instance. A survey of weighting schemes is given by (Wettschereck et al., 1997).

The power of kNN has been demonstrated in a number of real domains, but there are some reservations about the usefulness of kNN, such as: i) they have large

storage requirements, ii) they are sensitive to the choice of the similarity function that is used to compare instances, iii) they lack a principled way to choose *k*, except through cross-validation or similar, computationally-expensive technique (Guo et al. 2003).

The choice of *k* affects the performance of the kNN algorithm. Consider the following reasons why a k-nearest neighbour classifier might incorrectly classify a query instance:

- When noise is present in the locality of the query instance, the noisy instance(s) win the majority vote, resulting in the incorrect class being predicted. A larger *k* could solve this problem.
- When the region defining the class, or fragment of the class, is so small that instances belonging to the class that surrounds the fragment win the majority vote. A smaller *k* could solve this problem.

Wettschereck et al. (1997) investigated the behavior of the kNN in the presence of noisy instances. The experiments showed that the performance of kNN was not sensitive to the exact choice of *k* when *k* was large. They found that for small values of *k*, the kNN algorithm was more robust than the single nearest neighbour algorithm (1NN) for the majority of large datasets tested. However, the performance of the kNN was inferior to that achieved by the 1NN on small datasets (<100 instances).

Okamoto and Yugami (2003) represented the expected classification accuracy of k-NN as a function of domain characteristics including the number of training instances, the number of relevant and irrelevant attributes, the probability of each attribute, the noise rate for each type of noise, and k. They also explored the behavioral implications of the analyses by presenting the effects of domain characteristics on the expected accuracy of k-NN and on the optimal value of k for artificial domains.

The time to classify the query instance is closely related to the number of stored instances and the number of features that are used to describe each instance. Thus, in order to reduce the number of stored instances, instance-filtering algorithms have been proposed (Kubat and Cooperson, 2001). Brighton & Mellish (2002) found that their ICF algorithm and RT3 algorithm (Wilson & Martinez, 2000) achieved the highest degree of instance set reduction as well as the retention of classification accuracy: they are close to achieving unintrusive storage reduction. The degree to which these algorithms perform is quite impressive: an average of 80% of cases are removed and classification accuracy does not drop significantly. One other choice in designing a training set reduction algorithm is to modify the instances using a new representation such as prototypes (Sanchez et al., 2002).

Breiman (1996) reported that the stability of nearest neighbor classifiers distinguishes them from decision trees and some kinds of neural networks. A learning method is termed "unstable" if small changes in the training-test set split can result in large changes in the resulting classifier.

As we have already mentioned, the major disadvantage of instance-based classifiers is their large computational time for classification. A key issue in many applications is to determine which of the available input features should be used in modeling via feature selection (Yu & Liu, 2004), because it could improve the classification accuracy and scale down the required classification time. Furthermore, choosing a more suitable distance metric for the specific dataset can improve the accuracy of instance-based classifiers.

7 Support Vector Machines

Support Vector Machines (SVMs) are the newest supervised machine learning technique (Vapnik, 1995). An excellent survey of SVMs can be found in (Burges, 1998), and a more recent book is by (Cristianini & Shawe-Taylor, 2000). Thus, in this study apart from a brief description of SVMs we will refer to some more recent works and the landmark that were published before these works. SVMs revolve around the notion of a "margin"—either side of a hyperplane that separates two data classes. Maximizing the margin and thereby creating the largest possible distance between the separating hyperplane and the instances on either side of it has been proven to reduce an upper bound on the expected generalisation error.

If the training data is linearly separable, then a pair $(\mathbf{w}, b)$ exists such that

$$\mathbf{w}^T\mathbf{x}_i + b \geq 1, \text{ for all } \mathbf{x}_i \in P$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1, \text{ for all } \mathbf{x}_i \in N$$

with the decision rule given by $f_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b)$ where $\mathbf{w}$ is termed the weight vector and b the bias (or $-b$ is termed the threshold).

It is easy to show that, when it is possible to linearly separate two classes, an optimum separating hyperplane can be found by minimizing the squared norm of the separating hyperplane. The minimization can be set up as a convex quadratic programming (QP) problem:

$$\underset{\mathbf{w},b}{\text{Minimize}}\ \Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \qquad (1)$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1, \ldots, l.$$

In the case of linearly separable data, once the optimum separating hyperplane is found, data points that lie on its margin are known as support vector points and the solution is represented as a linear combination of only these points (see Figure 9). Other data points are ignored.

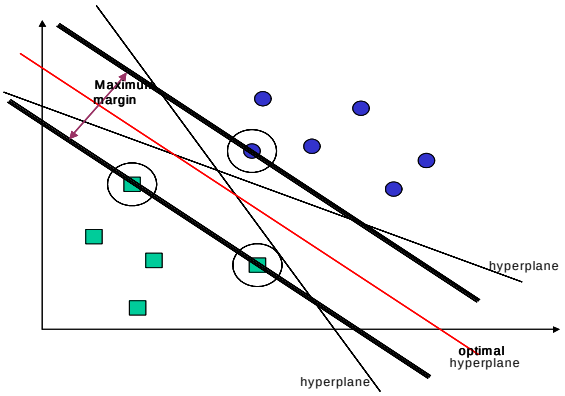

Figure 9. Maximum Margin

Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data (the number of support vectors selected by the SVM learning algorithm is usually small). For this reason, SVMs are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances.

A general pseudo-code for SVMs is illustrated in Figure 10.

```
1) Introduce positive Lagrange multipliers, one for each of the
inequality constraints (1). This gives Lagrangian:
   L_P ≡ (1/2)||w||^2 − Σ_{i=1}^{N} α_i y_i (x_i · w − b) + Σ_{i=1}^{N} α_i
   2) Minimize L_P with respect to w, b. This is a convex quadratic
programming problem.
   3) In the solution, those points for which α_i > 0 are called "support
vectors"
```

Figure 10. Pseudo-code for SVMs

Even though the maximum margin allows the SVM to select among multiple candidate hyperplanes, for many datasets, the SVM may not be able to find any separating hyperplane at all because the data contains misclassified instances. The problem can be addressed by using a *soft margin* that accepts some misclassifications of the training instances (Veropoulos et al. 1999). This can be done by introducing positive slack variables $\xi_i, \quad i = 1,...,N$ in the constraints, which then become:

$$w \cdot x_i - b \geq +1 - \xi \quad for \quad y_i = +1$$
$$w \cdot x_i - b \leq -1 + \xi \quad for \quad y_i = -1$$
$$\xi \geq 0,$$

Thus, for an error to occur the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors. In this case the Lagrangian is:

$$L_P \equiv \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i \{y_i (x_i \cdot w - b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i$$

where the μ_i are the Lagrange multipliers introduced to enforce positivity of the ξ_i.

Nevertheless, most real-world problems involve non-separable data for which no hyperplane exists that successfully separates the positive from negative instances in the training set. One solution to the inseparability problem is to map the data onto a higher-dimensional space and define a separating hyperplane there. This higher-dimensional space is called the *transformed feature space*, as opposed to the *input space* occupied by the training instances.

With an appropriately chosen transformed feature space of sufficient dimensionality, any consistent training set can be made separable. A linear separation in transformed feature space corresponds to a non-linear separation in the original input space. Mapping the data to some other (possibly infinite dimensional) Hilbert space H as $\Phi : R^d \rightarrow H$. Then the training algorithm would only depend on the data through dot products in H, i.e. on functions of the form $\Phi(x_i) \cdot \Phi(x_j)$. If there were a "kernel function" K such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, we would only need to use K in the training algorithm, and would never need to explicitly determine Φ. Thus, kernels are a special class of function that allow inner products to be calculated directly in feature space, without performing the mapping described above (Scholkopf et al. 1999). Once a hyperplane has been created, the kernel function is used to map new points into the feature space for classification.

The selection of an appropriate kernel function is important, since the kernel function defines the transformed feature space in which the training set instances will be classified. Genton (2001) described several classes of kernels, however, he did not address the question of which class is best suited to a given problem. It is common practice to estimate a range of potential settings and use cross-validation over the training set to find the best one. For this reason a limitation of SVMs is the low speed of the training. Selecting kernel settings can be regarded in a similar way to choosing the number of hidden nodes in a neural network. As long as the kernel function is legitimate, a SVM will operate correctly even if the designer does not know exactly what features of the training data are being used in the kernel-induced transformed feature space.

Some popular kernels are the following:

(1) $K(x, y) = (x \cdot y + 1)^P$,

(2) $K(x, y) = e^{-\|x-y\|^2 / 2\sigma^2}$,

(3) $K(x, y) = \tanh(\kappa x \cdot y - \delta)^P$

Training the SVM is done by solving N^{th} dimensional QP problem, where N is the number of samples in the training dataset. Solving this problem in

standard QP methods involves large matrix operations, as well as time-consuming numerical computations, and is mostly very slow and impractical for large problems. Sequential Minimal Optimization (SMO) is a simple algorithm that can, relatively quickly, solve the SVM QP problem without any extra matrix storage and without using numerical QP optimization steps at all (Platt, 1999). SMO decomposes the overall QP problem into QP sub-problems. Keerthi and Gilbert (2002) suggested two modified versions of SMO that are significantly faster than the original SMO in most situations.

Finally, the training optimization problem of the SVM necessarily reaches a global minimum, and avoids ending in a local minimum, which may happen in other search algorithms such as neural networks. However, the SVM methods are binary, thus in the case of multi-class problem one must reduce the problem to a set of multiple binary classification problems. Discrete data presents another problem, although with suitable rescaling good results can be obtained.

8 Discussion

Supervised machine learning techniques are applicable in numerous domains. A number of ML application oriented papers can be found in (Saitta and Neri, 1998) and (Witten and Frank, 2005). Below, we present our conclusions about the use of each technique. Discussions of all the pros and cons of each individual algorithms and empirical comparisons of various bias options are beyond the scope of this paper; as the choice of algorithm always depends on the task at hand. However, we hope that the following remarks can help practitioners not to select a wholly inappropriate algorithm for their problem.

Generally, SVMs and neural networks tend to perform much better when dealing with multi-dimensions and continuous features. On the other hand, logic-based systems tend to perform better when dealing with discrete/categorical features. For neural network models and SVMs, a large sample size is required in order to achieve its maximum prediction accuracy whereas NB may need a relatively small dataset.

SVMs are binary algorithm, thus we made use of error-correcting output coding (ECOC), or, in short, the output coding approach, to reduce a multi-class problem to a set of multiple binary classification problems (Crammer & Singer, 2002). Output coding for multi-class problems is composed of two stages. In the training stage, we construct multiple independent binary classifiers, each of which is based on a different partition of the set of the labels into two disjointed sets. In the second stage, the classification part, the predictions of the binary classifiers are combined to extend a prediction on the original label of a test instance.

There is general agreement that k-NN is very sensitive to irrelevant features: this characteristic can be explained by the way the algorithm works. Moreover, the presence of irrelevant features can make neural network training very inefficient, even impractical.

Bias measures the contribution to error of the central tendency of the classifier when trained on different data (Bauer & Kohavi, 1999). Variance is a measure of the contribution to error of deviations from the central tendency. Learning algorithms with a high-bias profile usually generate simple, highly constrained models which are quite insensitive to data fluctuations, so that variance is low. Naive Bayes is considered to have high bias, because it assumes that the dataset under consideration can be summarized by a single probability distribution and that this model is sufficient to discriminate between classes. On the contrary, algorithms with a high-variance profile can generate arbitrarily complex models which fit data variations more readily. Examples of high-variance algorithms are decision trees, neural networks and SVMs. The obvious pitfall of high-variance model classes is overfitting.

Most decision tree algorithms cannot perform well with problems that require diagonal partitioning. The division of the instance space is orthogonal to the axis of one variable and parallel to all other axes. Therefore, the resulting regions after partitioning are all hyperrectangles. The ANNs and the SVMs perform well when multicollinearity is present and a nonlinear relationship exists between the input and output features.

Lazy learning methods require zero training time because the training instance is simply stored. Naive Bayes methods also train very quickly since they require only a single pass on the data either to count frequencies (for discrete variables) or to compute the normal probability density function (for continuous variables under normality assumptions). Univariate decision trees are also reputed to be quite fast—at any rate, several orders of magnitude faster than neural networks and SVMs.

Naive Bayes requires little storage space during both the training and classification stages: the strict minimum is the memory needed to store the prior and conditional probabilities. The basic kNN algorithm uses a great deal of storage space for the training phase, and its execution space is at least as big as its training space. On the contrary, for all non-lazy learners, execution space is usually much smaller than training space, since the resulting classifier is usually a highly condensed summary of the data. Moreover, Naive Bayes and the kNN can be easily used as incremental learners whereas rule algorithms cannot. Naive Bayes is naturally robust to missing values since these are simply ignored in computing probabilities and hence have no impact on the final decision. On the contrary, kNN and neural networks require complete records to do their work.

Moreover, kNN is generally considered intolerant of noise; its similarity measures can be easily distorted by errors in attribute values, thus leading it to misclassify a new instance on the basis of the wrong nearest neighbors. Contrary to kNN, rule learners and most decision trees are considered resistant to noise because their pruning strategies avoid overfitting the data in general and noisy data in particular.

What is more, the number of model or runtime parameters to be tuned by the user is an indicator of an

algorithm's ease of use. As expected, neural networks and SVMs have more parameters than the remaining techniques. The basic kNN has usually only a single parameter (k) which is relatively easy to tune.

Logic-based algorithms are all considered very easy to interpret, whereas neural networks and SVMs have notoriously poor interpretability. k-NN is also considered to have very poor interpretability because an unstructured collection of training instances is far from readable, especially if there are many of them. While interpretability concerns the typical classifier generated by a learning algorithm, transparency refers to whether the principle of the method is easily understood. A particularly eloquent case is that of k-NN; while the resulting classifier is not quite interpretable, the method itself is quite transparent because it appeals to the intuition of human users, who spontaneously reason in a similar manner. Similarly, Naive Bayes' is very transparent, as it is easily grasped by users like physicians who find that probabilistic explanations replicate their way of diagnosing (Kononenko, 1993). Similarly, Naive Bayes' explanations in terms of the sum of information gains is very transparent, as it is easily grasped by users like physicians who find that explanations replicate their way of diagnosing (Kononenko, 1993).

Finally, decision trees and NB generally have different operational profiles, when one is very accurate the other is not and vice versa. On the contrary, decision trees and rule classifiers have a similar operational profile. SVM and ANN have also a similar operational profile. No single learning algorithm can uniformly outperform other algorithms over all datasets. Features of learning techniques are compared in Table 4 (from evidence of existing empirical and theoretical studies).

	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM	Rule-learners
Accuracy in general	**	***	*	**	****	**
Speed of learning with respect to number of attributes and the number of instances	***	*	****	****	*	**
Speed of classification	****	****	****	*	****	****
Tolerance to missing values	***	*	****	*	**	**
Tolerance to irrelevant attributes	***	*	**	**	****	**
Tolerance to redundant attributes	**	**	*	**	***	**
Tolerance to highly interdependent attributes (e.g. parity problems)	**	***	*	*	***	**
Dealing with discrete/binary/continuous attributes	****	***(not discrete)	***(not continuous)	***(not directly discrete)	**(not discrete)	***(not directly continuous)
Tolerance to noise	**	**	***	*	**	*
Dealing with danger of overfitting	**	*	***	***	**	**
Attempts for incremental learning	**	***	****	****	**	*
Explanation ability/transparency of knowledge/classifications	****	*	****	**	*	****
Model parameter handling	***	*	****	***	*	***

Table 4. Comparing learning algorithms (**** stars represent the best and * star the worst performance)

When faced with the decision "Which algorithm will be most accurate on our classification problem?", the simplest approach is to estimate the accuracy of the candidate algorithms on the problem and select the one that appears to be most accurate. The concept of combining classifiers is proposed as a new direction for the improvement of the performance of individual classifiers. The goal of classification result integration algorithms is to generate more certain, precise and accurate system results. Numerous methods have been suggested for the creation of ensemble of classifiers (Dietterich, 2000). Although or perhaps because many methods of ensemble creation have been proposed, there is as yet no clear picture of which method is best (Villada and Drissi, 2002). Thus, an active area of research in supervised learning is the study of methods for the construction of good ensembles of classifiers. Mechanisms that are used to build ensemble of classifiers include: i) using different subsets of training data with a single learning method, ii) using different training parameters with a single training method (e.g., using

different initial weights for each neural network in an ensemble) and iii) using different learning methods.

9 Conclusions

This paper describes the best-known supervised techniques in relative detail. We should remark that our list of references is not a comprehensive list of papers discussing supervised methods: our aim was to produce a critical review of the key ideas, rather than a simple list of all publications which had discussed or made use of those ideas. Despite this, we hope that the references cited cover the major theoretical issues, and provide access to the main branches of the literature dealing with such methods, guiding the researcher in interesting research directions.

The key question when dealing with ML classification is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem. Meta-learning is moving in this direction, trying to find functions that map datasets to algorithm performance (Kalousis and Gama, 2004). To this end, meta-learning uses a set of attributes, called meta-attributes, to represent the characteristics of learning tasks, and searches for the correlations between these attributes and the performance of learning algorithms. Some characteristics of learning tasks are: the number of instances, the proportion of categorical attributes, the proportion of missing values, the entropy of classes, etc. Brazdil et al. (2003) provided an extensive list of information and statistical measures for a dataset.

After a better understanding of the strengths and limitations of each method, the possibility of integrating two or more algorithms together to solve a problem should be investigated. The objective is to utilize the strengthes of one method to complement the weaknesses of another. If we are only interested in the best possible classification accuracy, it might be difficult or impossible to find a single classifier that performs as well as a good ensemble of classifiers. Despite the obvious advantages, ensemble methods have at least three weaknesses. The first weakness is increased storage as a direct consequence of the requirement that all component classifiers, instead of a single classifier, need to be stored after training. The total storage depends on the size of each component classifier itself and the size of the ensemble (number of classifiers in the ensemble). The second weakness is increased computation because in order to classify an input query, all component classifiers (instead of a single classifier) must be processed. The last weakness is decreased comprehensibility. With involvement of multiple classifiers in decision-making, it is more difficult for non-expert users to perceive the underlying reasoning process leading to a decision. A first attempt for extracting meaningful rules from ensembles was presented in (Wall et al, 2003).

For all these reasons, the application of ensemble methods is suggested only if we are only interested in the best possible classification accuracy. Another time-consuming attempt that tried to increase the classification accuracy without decreasing comprehensibility is the wrapper feature selection procedure (Guyon & Elissee, 2003). Theoretically, having more features should result in more discriminating power. However, practical experience with machine learning algorithms has shown that this is not always the case. Wrapper methods wrap the feature selection around the induction algorithm to be used, using cross-validation to predict the benefits of adding or removing a feature from the feature subset used.

Finally, many researchers in machine learning are accustomed to dealing with flat files and algorithms that run in minutes or seconds on a desktop platform. For these researchers, 100,000 instances with two dozen features is the beginning of the range of "very large" datasets. However, the database community deals with gigabyte databases. Of course, it is unlikely that all the data in a data warehouse would be mined simultaneously. Most of the current learning algorithms are computationally expensive and require all data to be resident in main memory, which is clearly untenable for many realistic problems and databases. An orthogonal approach is to partition the data, avoiding the need to run algorithms on very large datasets. Distributed machine learning involves breaking the dataset up into subsets, learning from these subsets concurrently and combining the results (Basak and Kothari, 2004). Distributed agent systems can be used for this parallel execution of machine learning processes (Klusch et al., 2003). Non-parallel machine learning algorithms can still be applied on local data (relative to the agent) because information about other data sources is not necessary for local operations. It is the responsibility of agents to integrate the information from numerous local sources in collaboration with other agents.

References

[1] Acid, S. and de Campos. L.M. (2003). Searching for Bayesian Network Structures in the Space of Restricted Acyclic Partially Directed Graphs. *Journal of Artificial Intelligence Research* 18: 445-490.

[2] Aha, D. (1997). *Lazy Learning*. Dordrecht: Kluwer Academic Publishers.

[3] An, A., Cercone, N. (1999), Discretization of continuous attributes for learning classification rules. Third Pacific-Asia Conference on Methodologies for Knowledge Discovery & Data Mining, 509-514.

[4] An, A., Cercone, N. (2000), Rule Quality Measures Improve the Accuracy of Rule Induction: An Experimental Approach, Lecture Notes in Computer Science, Volume 1932, Pages 119-129.

[5] Auer P. & Warmuth M. (1998). Tracking the Best Disjunction. *Machine Learning* 32: 127–150.

[6] Baik, S. Bala, J. (2004), A Decision Tree Algorithm for Distributed Data Mining: Towards Network Intrusion Detection, Lecture Notes in Computer Science, Volume 3046, Pages 206 – 212.

[7] Barto, A. G. & Sutton, R. (1997). *Introduction to Reinforcement Learning*. MIT Press.

[8] Batista, G., & Monard, M.C., (2003), An Analysis of Four Missing Data Treatment Methods for Supervised Learning, Applied Artificial Intelligence, vol. 17, pp.519-533.

[9] Basak., J., Kothari, R. (2004), A Classification Paradigm for Distributed Vertically Partitioned Data. Neural Computation, 16(7):1525-1544.

[10] Blum, A. (1997), Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain, Machine Learning, Volume 26, Issue 1, Pages 5-23.

[11] Bonarini, A. (2000), An Introduction to Learning Fuzzy Classifier Systems, Lecture Notes in Computer Science, Volume 1813, Pages 83-92.

[12] Bouckaert, R. (2003). Choosing between two learning algorithms based on calibrated tests. Proc 20th Int Conf on Machine Learning, pp. 51-58. Morgan Kaufmann.

[13] Bouckaert, R. (2004), Naive Bayes Classifiers That Perform Well with Continuous Variables, Lecture Notes in Computer Science, Volume 3339, Pages 1089 – 1094.

[14] Brazdil P., Soares C. and Da Costa J. (2003), Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results, Machine Learning, 50: 251-277.

[15] Breiman L., Friedman J.H., Olshen R.A., Stone C.J. (1984) Classification and Regression Trees, Wadsforth International Group.

[16] Breiman, L., Bagging Predictors. Machine Learning, 24 (1996) 123-140.

[17] Breslow, L. A. & Aha, D. W. (1997). Simplifying decision trees: A survey. *Knowledge Engineering Review 12:* 1–40.

[18] Brighton, H. & Mellish, C. (2002), Advances in Instance Selection for Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery* 6: 153–172.

[19] Bruha. I. (2000), From machine learning to knowledge discovery: Survey of preprocessing and postprocessing. , Intelligent Data Analysis, Vol. 4, pp. 363-374.

[20] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery.* 2(2):1-47.

[21] Camargo, L. S. & Yoneyama, T. (2001). Specification of Training Sets and the Number of Hidden Neurons for Multilayer Perceptrons. *Neural Computation* 13: 2673–2680.

[22] Castellano, G., Fanelli, A., & Pelillo, M. (1997). An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks 8:* 519–531.

[23] Cestnik, B., Kononenko, I., Bratko, I., (1987). Assistant 86: A knowledge elicitation tool for sophisticated users. In: Proceedings of the Second European Working Session on Learning. pp. 31-45.

[24] Cestnik, B. (1990), Estimating probabilities: A crucial task in machine learning. In Proceedings of the European Conference on Artificial Intelligence, pages 147-149.

[25] Cheng, J. & Greiner, R. (2001). Learning Bayesian Belief Network Classifiers: Algorithms and System, In Stroulia, E. & Matwin, S. (ed.), *AI 2001*, 141-151, LNAI 2056,

[26] Cheng, J., Greiner, R., Kelly, J., Bell, D., & Liu, W. (2002). Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence* 137: 43–90.

[27] Chickering, D.M. (2002). Optimal Structure Identification with Greedy Search. Journal of Machine Learning Research, Vol. 3, pp 507-554.

[28] Clark, P., Niblett, T. (1989), The CN2 Induction Algorithm. Machine Learning, 3(4):261-283.

[29] Cohen, W. (1995), Fast Effective Rule Induction. In *Proceedings of ICML-95,* 115-123.

[30] Cover, T., Hart, P. (1967), Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1): 21–7.

[31] Cowell, R.G. (2001). Conditions Under Which Conditional Independence and Scoring Methods Lead to Identical Selection of Bayesian Network Models. Proc. 17th International Conference on Uncertainty in Artificial Intelligence.

[32] Crammer, K. & Singer, Y. (2002). On the Learnability and Design of Output Codes for Multiclass Problems. *Machine Learning* 47: 201–233.

[33] Cristianini, N. & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, Cambridge.

[34] Csiszar, I. (1996), Maxent, mathematics, and information theory. In K. Hanson and R. Silver, editors, Maximum Entropy and Bayesian Methods. Kluwer Academic Publishers.

[35] De Mantaras & Armengol E. (1998). Machine learning from examples: Inductive and Lazy methods. *Data & Knowledge Engineering* 25: 99-123.

[36] Dietterich, T. G. (1998), Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10(7) 1895–1924.

[37] Dietterich, T. G. (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, *Machine Learning* 40: 139–157.

[38] Domingos, P. & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29: 103-130.

[39] Dutton, D. & Conroy, G. (1996), A review of machine learning, *Knowledge Engineering Review* 12: 341-367.

[40] Eklund, P., Hoang, A. (2002), A Performance Survey of Public Domain Machine Learning Algorithms Technical Report, School of Information Technology, Griffith University.

[41] Elomaa, T. & Rousu, J. (1999). General and Efficient Multisplitting of Numerical Attributes. *Machine Learning* 36, 201–244.

[42] Elomaa T. (1999). The biases of decision tree pruning strategies. Lecture Notes in Computer Science 1642. Springer, pp. 63-74.

[43] Flach, P.A. & Lavrac, N. (2000). The role of feature construction in inductive rule learning. De Raedt, L. & Kramer, S., (ed.), In *Proceedings of the ICML2000 workshop on Attribute-Value Learning and Relational Learning: Bridging the Gap*, Stanford University.

[44] Frank, E. & Witten, I. (1998). Generating Accurate Rule Sets Without Global Optimization. In Shavlik, J., (eds), *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA.

[45] Freund, Y. & Schapire, R. (1999), Large Margin Classification Using the Perceptron Algorithm, *Machine Learning* 37: 277–296.

[46] Friedman, J.H. (1989), Regularized Discriminant Analysis. Journal of the American Statistical Association.

[47] Friedman, N., Geiger, D. & Goldszmidt M. (1997). Bayesian network classifiers. *Machine Learning* 29: 131-163.

[48] Friedman, N. & Koller, D. (2003). Being Bayesian About Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning* 50(1): 95-125.

[49] Furnkranz, J. (1997). Pruning algorithms for rule learning. *Machine Learning* 27: 139-171.

[50] Furnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review* 13: 3-54.

[51] Furnkranz, J. (2001). Round Robin Rule Learning. *In Proceedings of the 18th International Conference on Machine Learning (ICML-01),* 146-153.

[52] Furnkranz, J., Flach, P. (2005), ROC 'n' Rule Learning—Towards a Better Understanding of Covering Algorithms, Machine Learning, Volume 58 (1), pp. 39 – 77.

[53] Gama, J. & Brazdil, P. (1999). Linear Tree. *Intelligent Data Analysis* 3: 1-22

[54] Gehrke, J., Ramakrishnan, R. & Ganti, V. (2000), RainForest—A Framework for Fast Decision Tree Construction of Large Datasets, Data Mining and Knowledge Discovery, Volume 4, Issue 2 - 3, Jul 2000, Pages 127 - 162

[55] Genton, M. (2001). Classes of Kernels for Machine Learning: A Statistics Perspective. *Journal of Machine Learning Research* 2: 299-312.

[56] Good I.J. (1950), Probability and the Weighing of Evidence, London, Charles Grin.

[57] Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. (2003), KNN Model-Based Approach in Classification, Lecture Notes in Computer Science, Volume 2888, Pages 986 – 996.

[58] Guyon, I, Elissee, A. (2003), An introduction to variable and feature selection. Journal of Machine Learning Research, 3:1157 1182.

[59] Hunt E., Martin J & Stone P. (1966), Experiments in Induction, New York, Academic Press.

[60] Heckerman, D., Meek, C. & Cooper, G. (1999). A Bayesian Approach to Causal Discovery. In Glymour, C. and G. Cooper, (ed.), *Computation, Causation, and Discovery*, 141-165. MIT Press.

[61] Hodge, V., Austin, J. (2004), A Survey of Outlier Detection Methodologies, Artificial Intelligence Review, Volume 22, Issue 2, pp. 85-126.

[62] Japkowicz N. and Stephen, S. (2002), The Class Imbalance Problem: A Systematic Study Intelligent Data Analysis, Volume 6, Number 5.

[63] Jain, A.K., Murty, M. N., and Flynn, P. (1999), Data clustering: A review, ACM Computing Surveys, 31(3): 264–323.

[64] Jensen, F. (1996). *An Introduction to Bayesian Networks*. Springer.

[65] Jordan, M.I. (1998), *Learning in Graphical Models*. MIT Press, Cambridge, MA.

[66] Kalousis A., Gama, G. (2004), On Data and Algorithms: Understanding Inductive Performance, *Machine Learning 54*: 275–312.

[67] Keerthi, S. & Gilbert, E. (2002). Convergence of a Generalized SMO Algorithm for SVM Classifier Design. *Machine Learning* 46: 351–360.

[68] Kivinen, J. (2002), Online Learning of Linear Classifiers, *Advanced Lectures on Machine Learning: Machine Learning Summer School 2002*, Australia, February 11-22, ISSN: 0302-9743, pp. 235 – 257.

[69] Klusch, M., Lodi, S., Moro, G. (2003), Agent-Based Distributed Data Mining: The KDEC Scheme. In Intelligent Information Agents: The AgentLink Perspective, LNAI 2586, pages 104-122. Springer.

[70] Kon, M. & Plaskota, L. (2000), Information complexity of neural networks, *Neural Networks* 13: 365–375.

[71] Kononenko, I. (1991), "Semi-Naive Bayesian Classifier", In Proceedings of the sixth European Working Session on Learning, 206-219.

[72] Kononenko, I. (1993), Inductive and Bayesian learning in medical diagnosis. Applied Artificial Intelligence 7(4): 317-337.

[73] Kononenko, I. (1994), 'Estimating attributes: analysis and extensions of Relief'. In: L. De Raedt and F. Bergadano (eds.): Machine Learning: ECML-94. pp. 171–182, Springer Verlag.

[74] Kubat, Miroslav Cooperson Martin (2001), A reduction technique for nearest-neighbor classification: Small groups of examples. Intell. Data Anal. 5(6): 463-476.

[75] Lindgren, T. (2004), Methods for Rule Conflict Resolution, Lecture Notes in Computer Science, Volume 3201, Pages 262 – 273.

[76] Littlestone, N. & Warmuth, M. (1994). The weighted majority algorithm. *Information and Computation 108*(2): 212–261.

[77] Liu, H. and H. Motoda (2001), Instance Selection and Constructive Data Mining, Kluwer, Boston.

[78] Madden, M. (2003), The Performance of Bayesian Network Classifiers Constructed using Different Techniques, Proceedings of European Conference on Machine Learning, Workshop on Probabilistic Graphical Models for Classification, pp. 59-70.

[79] Markovitch S. & Rosenstein D. (2002), Feature Generation Using General Construction Functions, *Machine Learning* 49: 59-98.

[80] McSherry, D. (1999). Strategic induction of decision trees. Knowledge-Based Systems, 12(5-6):269-275.

[81] Michalski, R. S., Chilausky, R. L. (1980), Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing and expert system for soybean disease diagnosis. Policy Analysis and Information Systems, 4(2)..

[82] Mika, S., Rätsch, G., Weston, J., Schölkopf, B. and Müller, K.-R. (1999), Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, Neural Networks for Signal Processing IX, pages 41-48. IEEE.

[83] Mitchell, T. (1997). *Machine Learning.* McGraw Hill.

[84] Murthy, (1998), Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, *Data Mining and Knowledge Discovery* 2: 345–389.

[85] Nadeau, C. and Bengio, Y. (2003), Inference for the generalization error. In Machine Learning 52:239–281.

[86] Neocleous, C. & Schizas, C., (2002), Artificial Neural Network Learning: A Comparative Review, LNAI 2308, pp. 300–313, Springer-Verlag Berlin Heidelberg.

[87] Nilsson, N.J. (1965). Learning machines. New York: McGraw-Hill.

[88] Olcay Taner Yıldız, Onur Dikmen (2007), Parallel univariate decision trees, Pattern Recognition Letters, Volume 28 , Issue 7 (May 2007), Pages: 825-832.

[89] Okamoto, S., Yugami, N. (2003), Effects of domain characteristics on instance-based learning algorithms. Theoretical Computer Science 298, 207-233.

[90] Parekh, R., and Yang, J., and Honavar, V. (2000), Constructive Neural Network Learning Algorithms for Pattern Classification. IEEE Transactions on Neural Networks. 11(2), pp. 436-451.

[91] Platt, J. (1999). Using sparseness and analytic QP to speed training of support vector machines. In Kearns, M., Solla, S. & Cohn, D. (ed.), *Advances in neural information processing systems*. MIT Press.

[92] Quinlan, J.R. (1979), "Discovering rules by induction from large collections of examples", D. Michie ed., Expert Systems in the Microelectronic age, pp. 168-201.

[93] Quinlan, J.R. (1993). C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco

[94] Rastogi, R. & Shim, K. (2000). PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. *Data Mining and Knowledge Discovery* 4: 315–344.

[95] Reinartz T. (2002), A Unifying View on Instance Selection, Data Mining and Knowledge Discovery, 6, 191–210, Kluwer Academic Publishers.

[96] Robert, J., Howlett L.C.J. (2001), Radial Basis Function Networks 2: New Advances in Design.

[97] Rosenblatt, F., (1962), Principles of Neurodynamics. Spartan, New York

[98] Roy, A. (2000), On connectionism, rule extraction, and brain-like learning. IEEE Transactions on Fuzzy Systems, 8(2): 222-227.

[99] Ruggieri, S. (2001). Efficient C4.5. *IEEE Transactions on Knowledge and Data Engineering* 14 (2): 438-444.

[100] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986), Learning internal representations by error propagation. In: Rumelhart D E, McClelland J L et al. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA, Vol. 1, pp. 318-362.

[101] Saad, D. (1998). *Online learning in neural networks.* London: Cambridge University Press.

[102] Sanchez, J., Barandela, R., Ferri, F. (2002), On Filtering the Training Prototypes in Nearest Neighbour Classification, Lecture Notes in Computer Science, Volume 2504, Pages 239 - 248

[103] Scholkopf, C., Burges, J. C. & Smola, A. J. (1999). *Advances in Kernel Methods*. MIT Press.

[104] Setiono R. and Loew, W. K. (2000), FERNN: An algorithm for fast extraction of rules from neural networks, Applied Intelligence 12, 15-25.

[105] Siddique, M. N. H. and Tokhi, M. O. (2001), Training Neural Networks: Backpropagation vs. Genetic Algorithms, IEEE International Joint Conference on Neural Networks, Vol. 4, pp. 2673–2678.

[106] Smyth, P, Goodman, R., M. (1990), Rule induction using information theory, In G. Piatetsky Shapiro and W. Frawley (eds), Knowledge Discovery in Databases, MIT Press.

[107] Tjen-Sien, L., Wei-Yin, L., Yu-Shan, S. (2000). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms. *Machine Learning* 40: 203–228.

[108] Utgoff, P., Berkman, N., Clouse, J. (1997), Decision Tree Induction Based on Efficient Tree Restructuring, Machine Learning, Volume 29, Issue 1, Pages: 5 – 44.

[109] Vapnik, V. (1995), The Nature of Statistical Learning Theory}. Springer Verlag.

[110] Veropoulos, K., Campbell, C. & Cristianini, N. (1999). Controlling the Sensitivity of Support Vector Machines. *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI99)*.

[111] Villada, R. & Drissi, Y. (2002). A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review* 18: 77–95.

[112] Vivarelli, F. & Williams, C. (2001). Comparing Bayesian neural network algorithms for classifying segmented outdoor images. *Neural Networks* 14: 427-437.

[113] Wall, R., Cunningham, P., Walsh, P., Byrne, S. (2003), Explaining the output of ensembles in medical decision support on a case by case basis, Artificial Intelligence in Medicine, Vol. 28(2) 191-206.

[114] Weigend, A. S., Rumelhart, D. E., & Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In: R. P. Lippmann, J. Moody, & D. S. Touretzky (eds.), Advances in Neural Information Processing Systems 3, San Mateo, CA: Morgan Kaufmann.

[115] Wettschereck, D., Aha, D. W. & Mohri, T. (1997). A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. *Artificial Intelligence Review* 10:1–37.

[116] Wilson, D. R. & Martinez, T. (2000). Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning* 38: 257–286.

[117] Witten, I. & Frank, E. (2005), "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[118] Yam, J. & Chow, W. (2001). Feedforward Networks Training Speed Enhancement by Optimal Initialization of the Synaptic Coefficients. *IEEE Transactions on Neural Networks* 12: 430-434.

[119] Yang, Y., Webb, G. (2003), On Why Discretization Works for Naive-Bayes Classifiers, Lecture Notes in Computer Science, Volume 2903, Pages 440 – 452.

[120] Yen, G. G. and Lu, H. (2000), Hierarchical genetic algorithm based neural network design, In: IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, pp. 168–175.

[121] Yu, L., Liu, H. (2004), Efficient Feature Selection via Analysis of Relevance and Redundancy, JMLR, 5(Oct):1205-1224.

[122] Zhang, G. (2000), Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C 30(4): 451-462.

[123] Zhang, S., Zhang, C., Yang, Q. (2002). Data Preparation for Data Mining. Applied Artificial Intelligence, Volume 17, pp. 375 - 381.

[124] Zheng, Z. (1998). Constructing conjunctions using systematic search on decision trees. *Knowledge Based Systems Journal 10:* 421–430.

[125] Zheng, Z. (2000). Constructing *X*-of-*N* Attributes for Decision Tree Learning. *Machine Learning* 40: 35–75.

[126] Zhou, Z. (2004), Rule Extraction: Using Neural Networks or For Neural Networks?, Journal of Computer Science and Technology, Volume 19, Issue 2, Pages: 249 – 253.

An Overview of Content-Based Spam Filtering Techniques

Ahmed Khorsi
Department of Computer Science,
Djillali Liabes University, Bel Abbes, 22000, Algeria
E-mail: ahmed-khorsi@univ-sba.dz

Overview paper

Keywords: antispam filters, text categorization, email classification

Received: May 26, 2007

So fast, so cheap, so efficient, Internet is nowadays incontestably communication mean of choice for personal, business and academic purposes. Unfortunately, Internet has not only this beautiful face. Malicious activities enjoy as well this so fast, cheap and efficient mean. The last decade, Internet worms took the lights. In the recent years, spams are invading one of the most used services of Internet: email. This paper summarizes most of techniques used to filter spams by analyzing the email content.

Povzetek: Članek pregledno opisuje metode za filtriranje elektronske pošte.

1 Introduction

With our agreement or without, Internet is increasingly becoming a favorite support of malicious activities. After worms which are always on the foreground of information technology problems, spams appeared and are really taking day after day more intensity.

Read an E-mail is nowadays a daily habit of many people. Indeed, emails are efficient, rapid and cheap mean of communication. This makes it favorite both in professional and personal correspondences. Additionally, Reading occasionally an E-mail from unknown source and content of which is not of the user interest is not really a misfortune. However, when more than 60% or even 90% of E-mails are of such kind, and often illicit; this is what one might call a nightmare. This kind of messages is said spams.

SpamCon Inc, estimated the cost induced by productivity and resources loss, filtering software, and support caused by only one unsolicited E-mail to from 1$ up to 2$[3]; multiplied by the number of spams sent and received everyday, the one dollar becomes then millions. International Data Corp. estimates the number of spams sent everyday through the net to 7.3 billions, where only AOL users recorded 5.5 millions by March, 5, 2003 [55]. These statistics were sufficient to persuade big users of the E-mail service to forecast a supplementary budget to fight spams. UUNet, one of the most important ISPs has a group of six persons with a budget of 1 million dollars, just to fight spams [56]. Netcom estimated that 10% of the end-user invoice is dedicated to filter spams [54]. A study of International Data Corporation (IDC) ranked spams in the second position of the ISPs' problems. One question arises then; *Why does someone enjoy sending so many E-mails, and how does he get so many addresses*? Although motivations are sometimes different, spams are generally of publicity-like contents. To broadcast a commercial through a TV channel costs hundreds times more than sending millions of spams. To get so many E-mail addresses is not at all difficult since many are available in the Internet itself. Some spammers, uses addresses found in newsgroups publicly accessible. Some others use webbots commonly called spambots, software which browses automatically the web seeking E-mail addresses. Generally, Spambots use keywords matching techniques to extract the email addresses. One evident way is to check for the character ''. Some others use software to generate random addresses then record all addresses from which they do not receive a reply of a delivery failure. More advanced techniques are summarized in [57].

Actually, fighting spams takes various forms. Juridical one came early in US by adopting an anti-spam law [30]. International cooperation is growing as well [47]. Simpler method might consist in some good practi-

	Computer viruses	**Spams**
1.	their authors are human programmer who tempt to workaround anti-viruses.	Senders are humans who tempt to avoid characteristics which denounce their spams.
2.	tempt to infect as many as possible systems.	spammer tempt to send their messages to as many as possible Internet users.
3.	may cause lot of damage in the infected system.	Consumes a large part of the Internet bandwidth, and causes a considerable productivity loss.
4.	it introduce itself discreetly in systems.	they are generally unsolicited

Table 1: Similarities between viruses and spasm.

ces. For instance, do not publish textual E-mail addresses, and use images instead, do not reply to suspicious E-mails. Some other techniques use features of headers such as E-mail origin [58][18][61]. These methods are widely supported by the most used Email servers without need of filtering software. However, all these methods are not the purpose of our discussion.

An antispam filter is similar to an anti-virus which scans files to check for virus signatures. Indeed, there are many similarities between computer viruses and spams. Table 1 enumerates some of them. In the following

Actually, many of the filtering techniques are based on text categorization methods [28] [8]. Thus filtering spams turns on a classification problem. Roughly, we can distinguish between two methods of machine classification. The first one is done on some rules defined manually. The typical example is the rule based expert systems. This kind of classification can be used when all classes are static, and their components are easily separated according to some features. The second one is done using machine learning techniques [23]. It is more convenient when the characteristics of discrimination are not well defined. These techniques attempt to generate on a set of samples, quasi or semi automatically a classifier with an acceptable error rate.

2.1 Generalities

Getting an E-mail *m*, we have to define a decision function *f* which assigns to *m* its class, *S* for Spams or *L* for legitimate. Let G_M be the set of messages. *f* is then :

$$f : G_M \rightarrow \{S,L\}$$

In such techniques, we first check some characteristics on which we may classify the message into the class *S* or *L*. We will refer to such characteristics by a vector *x*.

Let *P(x/c)* be the probability that the class *c* generates a message which its characteristic vector is *x*. If we suppose that a legitimate message never contains the text *t* ="*Buy now*" and that *x = (m = utv)* with *u*, *v* are two strings, the probability *P(x/L) = 0*. Then, the problem is to compute the probability that a message which has a characteristic vector *x* belongs to the class *c* say *P(c/x)*. We obtain then by observing the rule of Bayes:

$$P(c/x) = \frac{P(x/c)P(c)}{P(x)} = \frac{P(x/c)P(c)}{P(x/S)P(S) + P(x/L)P(L)}$$

Where *P(x)* indicates the a-priori probability of occurrence of a message which characteristic vector is *x*, and *P(c)* indicates the probability that a random Email belongs to the class *c*. Knowing the probability *P(c)* and the probability *P(x/c)* suffices to deduce *P(c/x)*. We have then the following rule of classification:

If *P(S/x) > P(L/x)* (the a-posteriori probability that the E-mail which has the characteristic vector *x* belongs to the class *S* is greater than that the same E-mail belongs to the class *L*) then classify *m* as being unsolicited.

sections we will briefly present some content-based filtering techniques. The main idea behind such techniques is to classify an email into unsolicited or legitimate by checking some features in its content. It is not our aim to make an extensive comparison of these different methods when many papers propose comparisons between two or more of these techniques[34][20][2][60]

2 Bayesian classifier

This rule is called the rule of the maximum aposteriori probability(MAP). It can be written as follow: If

$$\frac{P(x/S)}{P(x/L)} > \frac{P(L)}{P(S)}$$

classify the message as being unsolicited and legitimate otherwise.

We often note the resemblance fraction:

$$\Lambda(x) \quad \frac{P(x/S)}{P(x/L)}$$

The MAP rule is written then:

$$\Lambda(x) \underset{L}{\overset{S}{\gtrless}} \frac{P(L)}{P(S)}$$

Let us note $L(c_1, c_2)$ the function which determines the cost of a bad classification of an occurrence of the class c_1 as being of the class c_2. It is logic to say whereas $L(L,L) = L(S, S) = 0$. We can then define a function of risk:

$R(c/x) = L(S, c)P(S/x) + L(L, c)P(L/x)$

Obviousness would be to classify the message by minimizing the function of risk. From which the rule:

If R(S/x) < R(L/x) classify m unsolicited and legitimate otherwise.

This last rule is called bayesian rule of classification or Bayes classifier[44].

We write the classification rule in term of resemblance fraction as follow:

$$\Lambda(x) \underset{L}{\overset{S}{\gtrless}} \lambda\frac{P(L)}{P(S)}$$

Where

$$\lambda = \frac{\mathcal{L}(L,S)}{\mathcal{L}(S,L)}$$

Intuitively, this parameter indicates the risk taken when we classify a legitimate E-mail as being unsolicited. Clearly, more λ is great; more the false positive error is small.

2.2 Application

In this subsection, we highlight the practical application of the theoretical principle of the bayesian classifier.

As already mentioned, to be able to determine the classification parameter, one must determine the probabilities *P(x/c)* and *P(c)* for any message *m*. Obviously, that cannot be made in exact manner. However, we can approximate these probabilities on the basis of a training sample. For example, the probability *P(S)* would be roughly given by calculating the ratio of the spams on the number of all messages in the training sample.

For simplicity, we consider that the characteristic vector is a binary one, where the presence of a catchword w in the message *m* is represented by one 1. That is to say then:

$$P(x_w = 1/S) \approx \frac{\text{Number of spams in which} w \text{ is checked}}{\text{number of all spams}}$$

Algorithm 1 summarizes the training where Algorithm 2 the classification steps.

Algorithm 1 Training algorithm of Bayes's classifier

1: **for all** $c \in \{S, L\}$ **do**
2: Estimate $P(c)$
3: **for all** $x_w \in \{0, 1\}$ **do**
4: Estimate $P(x_w/c)$
5: **end for**
6: **end for**
7: **for all** $x_w \in \{0, 1\}$ **do**
8: Calculate $P(c/x_w)$ using Bayes's rule
9: **end for**
10: **for all** $x_w \in \{0, 1\}$ **do**
11: Estimate $\Lambda(x_w)$
12: **end for**
13: Calculate $\lambda \frac{P(L)}{P(S)}$

Algorithm 2 Classification based on Bayes's classifier

1: Calculate the vector x_w of the input message m
2: **if** $\Lambda(x_w) > \lambda \frac{P(L)}{P(S)}$ **then**
3: m is unsolicited
4: **else**
5: m is a legitimate email
6: **end if**

In general, we represent the presence of a word w_i in the message *m* by the value 1 in the characteristic vector $x = (x_1, x_2, \ldots, x_n)$. However, the algorithm 1 will have to compute 2^n values of x which is unpractical. To avoid this, the assumption is introduced that the presence of two words is independent one of the other, which allows us to write:

$$P(x/c) = \prod_{i=1}^{n} P(x_i/c) \qquad \Lambda(x) = \prod_{i=1}^{n} \Lambda_i(x_i)$$

In [32] T.A Meyer and B Whateley report that using foure corpora gathered from SpamBayes users and SpamAssassin public corpus they obtained the following results. The bayesian classifier constitutes one of the most used techniques in antispam filters such as ' *spamassassin*' [53] and SpamBayes [32] or [49]. Although the assumption of mutual independence between word's occurrences is false, the recorded results remain very good for the traditional text messages. It often serves as a baseline to compare performances of other methods [60].

3 *k* nearest neighbors

The principle of this technique is rather simple. Let us suppose that similarities among messages are measurable using a measure of distance among the characteristic vectors. To decide whether a message is legitimate or not, we look at the class of the messages that are closest to it. Generally, this technique does not use a separate phase of training and the comparison between the vectors is a real time process. This has a time complexity of *O(nl)* where n is the size of the characteristic vector and l the sample size. This can be circumvented by using a traditional indexing methods [13][19][35]. To adjust the risk of false classification, t/k rule is introduced. What can be read:

> If at least *t* messages in *k* neighbors of the message *m* are unsolicited, then *m* is unsolicited email, otherwise, it is legitimate.

We should note that the use of an indexing method in order to reduce the time of comparisons induces an update of the sample with a complexity *O(m)*, where *m* is the sample size. An alternative of this technique is known as memorybased approach [2][46].

TiMBL [11] is a software package developed by ILK Research Group that implements a collection of machine learning algorithms. Results of the implementation of this technique in spam filtering reported in [2] seems to be comparable to those of bayesian classifiers.

4 Technique of Support Vector Machine (SVM)

Support vector machine [9][10][7] is one of the most recent techniques used in text classification. In machine learning the training sample is a set of vectors of n attributes. We can then assume that we are in a hyper-space of *n* dimensions, and that the training sample is a set of points in the hyper-space. Let us consider the simple case of just two classes (as it is the case of spam problem). The classification using Support vector machine look for the hyper plane able to separate the points of the first class from those of the second one such that the distance between the hyper plane and points of each class is maximum see Figure 1.

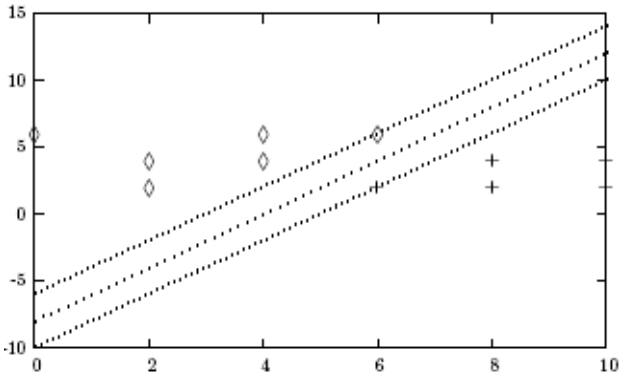

Figure 1: hyper-plane that separate two classes.

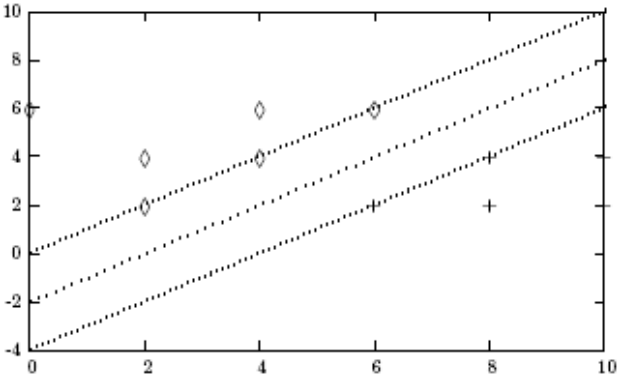

Figure 2: hyper-plane that separate two classes and is far from each class.

One question may be how we can find the hyper plane when the classes are not linearly separable (eg. XOR function). In this case the hyper space is extended to more dimensions. This insures the existence of hyper plane that separates the two classes. One interesting feature is that to find the appropriate plane, SVM method explore just the nearest points. One of the most efficient SVM algorithms was proposed in [39]. An implementation of the SVMmethod in spam filtering is proposed in [12] where Dricker et all also provide also a comparison with other methods.

5 Technique of maximum entropy

Maximum entropy is a classical model often used in natural language processing [41]. The principle is to find the appropriate probability distribution *p(a, b)* that maximizes the entropy:

$$H(p) = -\sum_{x \in \mathcal{A} \times \mathcal{B}} p(x) \log p(x)$$

where $\mathcal{A}$ denotes the set of possible classes, and $\mathcal{B}$ the set of possible values of vectors of features. This maximization should keep *p* consistent with evidence (i.e., should meet all known values in the training set). *p* becomes is then:

$$p(a,b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}$$

where *k* is the size of the vector of features and

$$Z(b) = \sum_{a} \prod_{j=1}^{k} \alpha_j^{f_j(a;b)}$$

is a normalization factor that ensures

$$\sum_a p(a,b) = 1.$$

α_j can be computed using the Generalized Iterative Scaling [15]. *f* is defined as follows:

$$f_{cp,a'}(a,b) = \begin{cases} 1 & \text{if } a = a' \text{ and } cp(b) = true \\ 0 & \text{otherwise} \end{cases}$$

where cp maps a pair *(a, b)* to *{true, false}* Results reported in [59] show an error rate better than that of bayesian classifier when the training sample grows.

6 Technique of neural networks

Neural network is a well known model [51][31][42][14][17][17][50] which has been designed by McCuloch on the basis of work carried out on the human neurons. The neural networks are quite famous to be well adapted for problems of classification. Without being spread out over the model, we will retain in what follows the characteristics which contribute to the design of an antispam filter.

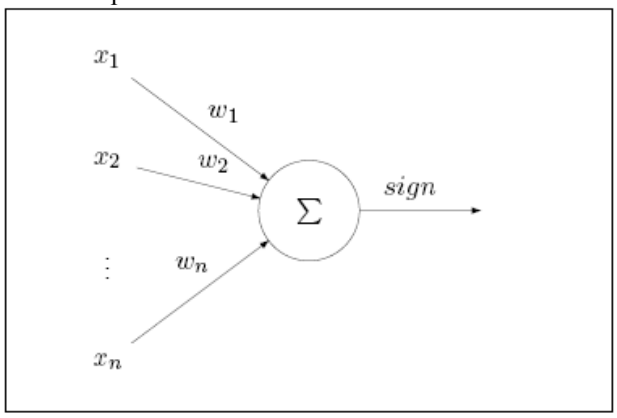

Figure 3: The perceptron.

6.1 Perceptron

The idea is to define a linear function *f(x) = wx + b* where:

$$f(x) \begin{cases} > 0 & \text{If } x \text{ is of the first class} \\ < 0 & \text{else} \end{cases}$$

where *w* is a vector of weights and *b* a bias vector. We can simplify the function to obtain a decision function from it *d(x) = sign(wx + b)*[43]. Figure 3 shows a graphical representation of the perceptron. The training of the perceptron is performed using an iterative method, where the weight and bias vectors are initialized then adjusted each iteration in such manner to ensure the classification of a new occurrence of the training sample. For instance let *x* be a vector that the perceptron fails to classify, and w_i, b_i the vector of weight and bias which corresponds to the i^{th} iteration. We have $sign(w_i x+bn) \neq c$

where c is the sign corresponding to the real class of the message that has the characteristic vector x. The new vectors w_i+1 and b_i+1 are calculated as follow:

$$w_{i+1} = w_i + cx \qquad b_{i+1} = b_i + c$$

The training continues until the perceptron manages to classify correctly all the messages of the training sample. In this case, we say that the perceptron converges. It is well-known that the perceptron does not converge in the case of non-linear classification problem [21][16].

In the case of spams filtering and if one makes a point of applying the technique of the perceptron, it is enough to choose a characteristic vector larger than that of the training sample to ensure the convergence. However such practice will heavily weigh down the computation.

The algorithm 3 summarizes the training of the perceptron.

Algorithm 3 Training algorithm of the perceptron

1: Initialize w and b.
2: **while** $\exists x \in$ training sample such that $sign(wx + b) \neq c$ **do**
3: $w \leftarrow w + cx$ and $b \leftarrow b + c$
4: **end while**

6.2 The multi-layer networks

As its name indicates, the multi-layer neural net is a network of connected perceptrons which form a network with successive layers. The outputs of each perceptron are inputs of perceptrons of the following layer. The inputs of the neurons of the first layer are the components of the characteristic vector, while the outputs of the last layer are the results of the classification. The layers between the first and the last are called *hidden layers*. The function of each neuron is somewhat different from the simple perceptron, although the training is also made in an iterative way as the simple perceptron. The output function is:

$$o = \phi(\sum_{i=1}^{k} w_i x_i + b)$$

where ϕ is a nonlinear function such as

$$\frac{1}{1+e^{ax}}$$

Or tanh(x). Figure 4 shows a graphical representation of a multi-layer neural network. The training of the neural network means the readjustment of the weights and bias in such manner to minimize the sum of the errors of the output, that is to say:

$$E(f) = \sum_{i=1}^{n} |f(x_i) - c_i|^2$$

The tuning of these parameters is described in details in [21][16]. In [38] Levent ¨Osg¨ur and all reported a 90% accuracy in a filter based on coupling neural network technique and bayesian classifier. [33] is a Semantec white paper on how and why neural network should be implemented in an antispam system.

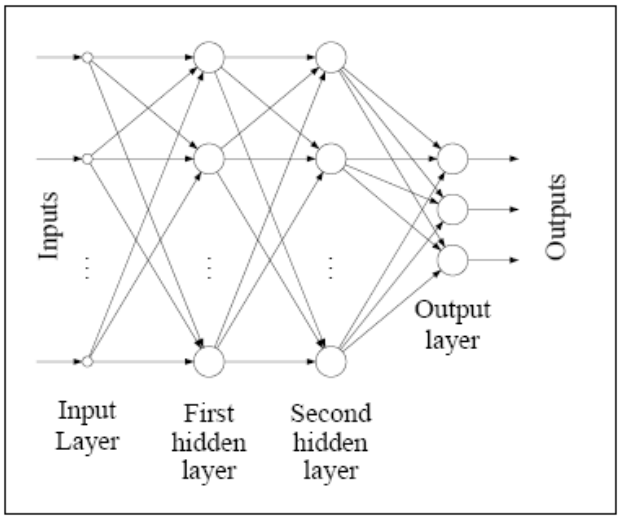

Figure 4: Multi-layer neural network.

7 Technique of search engines

When it acts on text e-mails, classification techniques of text seem to be efficient. However, spammers do not cease to invent tricks to circumvent filters. One of these tricks is to include in the body of the message only the hyperlink to a Web page which contains the advertising text. The problem become then a web content classification. A proposed technique to overcome this kind of spams is to use the public search engines which offer a mean to classify the websites [22]. The principle of this technique is to analyze automatically the contents of the pages referred by the links sent in the messages likely to be spams. The analysis starts by using the public search engines such as Yahoo and Google. A comparison then with the user interest can judge the convenience of the message with the requests of the user. If the search engines do not label the referred pages, a later step consists to analyze contents of the pages by traditional Bayes's classifier. Initial classification by the search engines is also used to enrich the sample of the bayesian classifier. This makes the model more dynamic. The main drawback of this approach is that to judge whether a mail is legitimate or not its important use of the band-width.

In [22], the author reports a false positive rate of 0.0032% on a sample of 1191 legitimate emails and 493 spams.

Type	Operation	Description
Arithmetic	+,-,/,*,$\sqrt{}$	
Relational	=,$\neq$,$\geqslant$,¡	
Boolean	AND,OR,NOT	
Non-linear	max,min,ABS	
Of words	Freq(x)	return the frenquency of the word$x$ in the message body
	Exists(x)	Return 1 if the word x occurs in the message and 0 otherwise

Table 2: Operation represented in the tree.

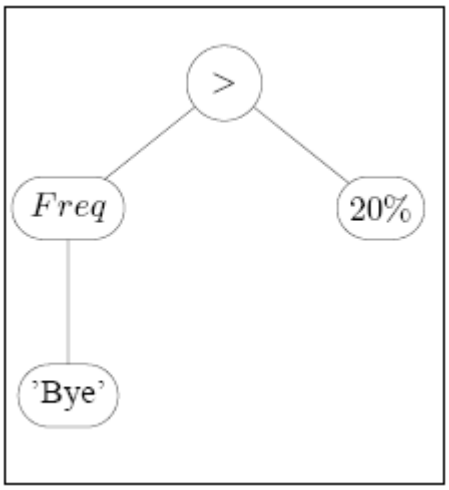

Figure 5: An example of a tree of a bayesian classifier.

8 Technique of genetic programming

In the design of a bayesian filter, the characteristic vector may include the frequencies of some words [45] generally selected by human experts. In fact, this construction is sometimes decisive in the performances of the filter. In [20], Hooman proposes a method to build automatically the bayesian filter. This method is based on the genetic programming. Thus, the frequency of a word ' *buy* ' for example ' *60*' % in an E-mail can argument the classification of the message as unsolicited. As genetic programming suggested by Koza [25][26][24][27][1], the filter is represented by a syntactic tree where nodes are :

- numbers that represent the frequencies
- operations on numbers
- words
- operations on words

see Table 2.

A syntactic tree of a filter should be built according to a precise syntax. Syntactic rules then can be used to check the correctness of the tree by checking whether we are able to reduce the tree to some number. Figure 5 gives an example of a tree of a filter. Artificial genetic approaches use functions that evaluate the fitness of a population to some criteria. This is also the case of the approach being explained.

The fitness function is then defined as follows:

$$Fitness = \frac{1}{|m_S|}\sum_{i=1}^{|m|} S(m_i)(v_i - a_i)^2 + \frac{1}{|m_L|}\sum_{i=1}^{|m|} L(m_i)(v_i - a_i)^2 \quad (1)$$

Where

a_i □ *{0, 100}* Is the correct classification of the message m_i

$0 \leq vi \leq 100$ the classification of mi returned by the filter.

m_S The number of spams.

m_L The number of legitimate E-mails.

$S(m_i)$ returns 1 if m_i is a unsolicited and 0 otherwise $L(m_i)$ returns 1 if mi is legitimate and 0 otherwise

Figure 6 gives an example of cross-over. According to the author of this approach, the experimental results reported in [20] shows an effectiveness close to those of a bayesian classifier manually built

9 Technique of Artificial Immune System

The almost obvious similarities between spams and computer viruses let think that the traditional and new techniques of anti-viruses can be applied to fight spams. One of these techniques is computer immune systems [36][52][48]. In [37] Terri Oda and Tony White suggest to design an anti-spams filter based on the generation of artificial lymphocytes using gene database. Genes are regular expressions which represent mini-languages likely to contain keywords that are usually checked in spams. The use of the regular expressions aims according to the author at increasing the accuracy as well as the general information hold in the detecting lymphocytes. The generation of lymphocytes is based on a training sample. The lifespan of these lymphocytes can be tuned

in order to ensure the system dynamicity. This technique represents only an attempt to assist the classical techniques already proved. Membranar proteins of biological cells allow a deterministic way to check whether a cell is self or not. It remains too difficult to find efficient discrimination between viruses and legitimate objects of computer systems as well as between spams and legitimate Emails

10 Conclusion

It is now well known that no technique can be claimed alone to be the ideal solution with 0% false positive and 0% false negative. Currently used antispam systems couples several machine learning techniques for content classification. Spamassassin uses the genetic programming to generate its bayesian classifier for each release. Text classification techniques, such as bayesian classifiers and neural networks offer a good theoretical and practical background to fight the problem of spams. However, two disadvantages are opposed to such relatively simple approaches. First, the definition of unsolicited E-mails varies from one to another. A generalized classification can penalize some users interested by some products advertised electronically. The second disadvantage is that a mail can be other thing than simple text. Take the example of the multimedia messages (images, voice-email, and movies). If methods of text classification are allowed to wander the text of each message, wandering tens of thousands of images or movies to classify them is surely not a practical solution. A solution of the first problem is to base classification on user profiles rather than impose characteristic vectors issued from perceptions other than those of the users. More general solution would be an hierarchical filter. In each node of the hierarchical tree the filter should block all E-mails which seem to be unsolicited from the users of all its sub-trees. Regarding the second disadvantage, the methods of classifications of multimedia documents exist [40][6][29] but their time and space complexities remain far from the requirements of a real time computation. Recently, new approches which count links of spam have been also investigating [4][5] Fighting spams is series of chess parts between industrial

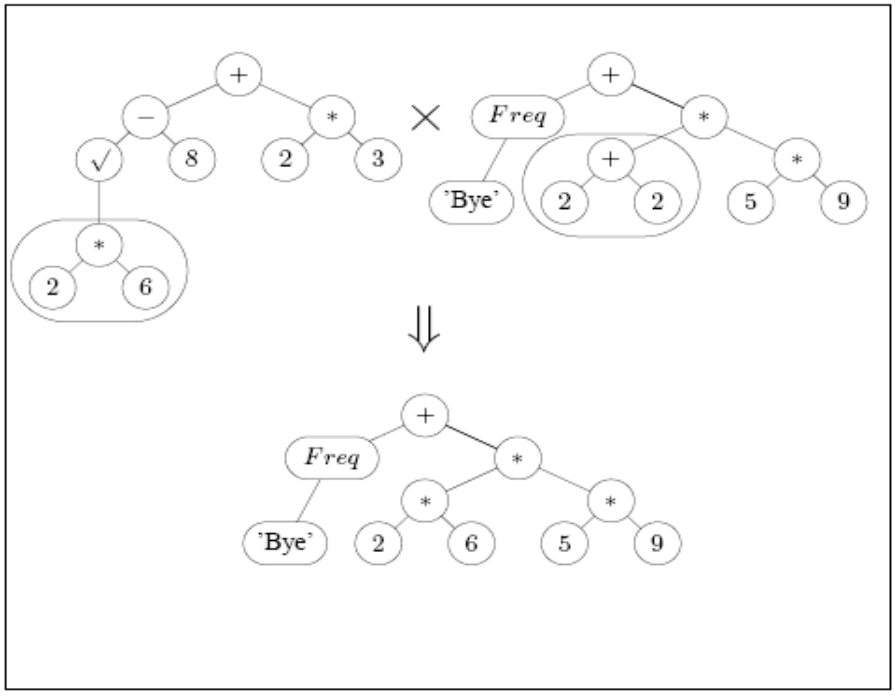

Figure 6: Example of a cross between two trees

and researchers in one side and spammers in the other side. Until the day the latter will decide to do not play any more, researchers will abstain from shouting victory, just like industrials and researchers of the anti-viruses. Spams may be a misfortune for simple users, but it seems to be a new big market for information technology industrials.

Acknowledgement

My thanks go to Dr. Bendahmane, Dr. Djelloul Ziadi, and all persons who contributed to this work.

References

[1] M. Ahluwalia and L. Bull. A genetic programming-based classifier system. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 11–18, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[2] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C.D. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memorybased approach. In H. Zaragoza, P. Gallinari, , and M. Rajman, editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, pages 1–13, 2000.

[3] S. Atskins. Size and cost of the problem. In *Priceedings of the Fifty-sixth Internet Engineering Task Force(IETF) Meeting*, (San Francisco, CA), March 16-21 2003. SpamCon Foundation.

[4] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. Technical report, DELIS – Dynamically Evolving, Large-Scale Information Systems, 2006.

[5] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *Workshop: The Future of Web Search*, Barcelona, May 2006.Universitat Pompeu Fabra.

[6] J. S. De Bonet. Novel statistical multiresolution techniques for image synthesis, discrimination, and recognition. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1997.

[7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[8] W. W. Cohen. Learning to classify English text with ILP methods. In Luc De Raedt, editor, *Advances in inductive logic programming*, pages 124–143. IOS Press, Amsterdam, NL, 1995.

[9] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[10] N. Cristiatnini and J. Shawe-Taylor. *An introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2003. `http://www.support-vector.net`.

[11] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. Timbl: Tilburg memory based learner, version 1.0, reference guide. ILK Technical Report 98-03, Tilburg, 1998.

[12] H. Drucker, V. Vapnik, and D. Wu. Support vector machines for spam categorization. I*EEE Transactions on Neural Networks*, 10(5):1048–1054, 1999. Available: `http://www.monmouth.edu/ druker/SVM spam article compete.PDF`.

[13] P. Ferragina and R. Grossi. Improved dynamic text indexing. *J. Algorithms*, 31(2):291–319, 1999.

[14] M. Glick and D. Rumelhart. *Neuroscience and Connectionist Theory*. The Developpement in Connictionist Theory. Erlbaum Assiciates, Hilldsdale,

[15] NJ, 1989.J. Goodman. Sequential conditional generalized iterative scaling. In *ACL '02: Proceedings of the40th Annual Meeting on Association for Computational Linguistics*, pages 9–16, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[16] S. Haykin. *Neural Networks: A Fomprehensive Foundation*. Printice Hall, 1998.

[17] J. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

[18] G. Hulten, J. Goodman, and R. Rounthwaite. Filtering spam e-mail on a global scale. Technical report, Microsoft Corp, 2004.

[19] J. K¨arkk¨ainen and E. Ukkonen. Lempel-ziv parsing and sublinear-size index structures for string matching. In *Proc WSP'96*, pages 141– 155. Carleton University Press, 1996.

[20] H. Katirai. Filtering junk e-mail: A performance comparison between genetic programming and naive bayes.

[21] V. Kecman. *Learning and Soft Computing*. The MIT Press, 2001.

[22] O. Kolesnikov, W. Lee, and R. Lipton. Filtering spam using search engines. Technical Report GIT-CC-04-15, Georgia Tech, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, 2004-2005.

[23] A. Konar. *Artificial Intelligence and Soft Computing : Behavioral and Congnitive Modeling of the Human Brain*, chapter 8. CRC Press, washington, 2000.

[24] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, MA, 1992.

[25] J. R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. In N. S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, volume 1, pages 768–774. Morgan Kaufmann, 20-25 aug 1989.

[26] J. R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University, June 1990.

[27] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.

[28] D. Lewis. (spam vs.) forty years of machine learning for text classification. In *Proceedings of the Spam Conference*, 2003.

[29] L. Lu, H. Jiang, and H. Zhang. A robust audio classification and segmentation method. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 203–211, New York, NY, USA, 2001. ACM Press.

[30] D. Platt Majoras, O. Swindle, Commissioner, T. B. Leary, P. O. Harbour, Commissioner, and J. Leibowitz. A can-spam informant reward system : A report to congress. Technical report, Federal Trade Commission, US, 2004.

[31] W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in nervous activity.

Bulletin of Mathematical Biophysics, 5:115– 133, 1943.

[32] T. A. Meyer and B. Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004. Available: http://www.ceas.cc/papers-2004/136.pdf.

[33] C. Miller. Neural network-based anti-spam heuristics, white paper.

[34] K. Mock. An experimental framework for email categorization and management. In *24th Annual ACM International Conference on Research and Development in Information Retrieval*, New Orleans, LA, September 2001.

[35] G. Navarro and R. Baeza-Yates. A practical q-gram index for text retrieval allowing errors. *CLEI Electonic Journal*, 1(2), 1998.

[36] G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, editors. *Artificial Immune Systems, Third International Conference, ICARIS 2004, Catania, Sicily, Italy, September 13-16, 2004*, volume 3239 of *Lecture Notes in Computer Science*. Springer, 2004.

[37] T. Oda and T. White. Developing an immunity to spam. In *GECCO*, pages 231–242, 2003.

[38] L. ¨Osg¨ur, T. G¨ung¨or, and F. G¨urgen. Adaptive anti-spam filtering for agglutinative languages: a special case for turkish. *Pattern Recogn. Lett.*, 25(16):1819–1831, 2004.

[39] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[40] *29th Applied Image Pattern Recognition Workshop (AIPR 2000),16-18 October 2000, Washington, DC, USA, Proceedings*. IEEE Computer Society, 2000.

[41] A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical report, University of Pennsylvania, 1997.

[42] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, Washington, 1958.

[43] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*, pages 138–149. Kaufmann, SanMateo, CA, 1990.

[44] M. Sahami. Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery in Databases*, 1996.

[45] M. Sahami. A bayesian approach to filtering junk e-mail. In *Proceedings of AAAI-98 workshop on Learning for Text Categorization*, Madison, Wisconsin, USA, 1998.

[46] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. A memory-based approach to antispam filtering for mailing lists. *Information Retrieval*, 6:49–73, 2003.

[47] C. Sarrocco. Spam in the information society: Building frameworks for international cooperation. Technical report, OECD Task force on spam, 2004.

[48] A. Secker, A Freitas, and J. Timmis. Aisec: An artificial immune system for e-mail classification. In R. Sarker, R. Reynolds, H. Abbass, T. Kay-Chen, R. McKay, D Essam, and T. Gedeon, editors, *Proceedings of the Congress on Evolutionary Computation*, pages 131–139, Canberra. Australia, December 2003. IEEE.

[49] S. Shakeri and P. Rosso. The bsp spam filter. In *Proc. Confer. Information Technologies Int. Symposium*, Tutan, Morocco, June 2005. IEEE.

[50] J. Shavlik, R. Mooney, and G. Towell. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143, 1991.

[51] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995.

[52] A. Somayaji, S. Hofmeyr, and S. Forrest. Principles of a computer immune system. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, pages 75–82, New York, NY, USA, 1997. ACM Press.

[53] Spamassassin. Spamassassin website. http://spamassassin.org.

[54] New York Times, March 19, 1998.

[55] J.Weaver. Aol escalates spam warfare, March 5, 2003.

[56] Internet Week, May 4, 1998.

[57] Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.

[58] H. K. Yoke. Curbing spam via technical measures: an overview. Technical report, ITUWSIS Thematic Meeting on Countering Spam, 2004.

[59] L. Zhang and T. Yao. Filtering junk mail with a maximum entropy model. In *Proceeding of 20th International Conference on Computer Processing of Oriental Languages (ICCPOL03)*, pages 446–453, 2003.

[60] L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACMTransactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.

[61] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proc. of ACM/IFIP/USENIX Intl. Middleware Conf*, 2003.

Comparative Study and Techno-Economic Analysis of Broadband Backbone Upgrading: a Case Study

Borka Jerman-Blažič
Laboratory for Open systems and Networks, Jožef Stefan Institute, Slovenia
E-mail: borka@e5.ijs.si
http://www.e5.ijs.si/staff

Keywords: broadband communications, techno-economic study, backbone network upgrading

Received: October 2, 2006

In this paper we compare and evaluate suitable methods for delivering broadband services at the metro and backbone level. In particular, the paper presents a techno-economic analysis of two key broadband technologies for metropolitan and backbone networks: The studies are based on a modelling methodology for network value analysis that involves CAPEX and OPEX calculations, while the overall technology deployment financial assessment is based on techno-economic evaluation measures such as net present value (NPV) and internal rate of return (IRR). The studied case is the Telekom Slovenia network which is the incumbent operator in Slovenia. The study is based on the operator data and plans. It deals with the upgrading the current network backbone and takes in account future development and trends of broadband services in the country.

Povzetek: Narejena je primerjalna študija in tehnično-ekonomska analiza primera nadgradnje širokopasovnega hrbteničnega omrežja.

1 Introduction

Definitions of broadband have continued to evolve and are changing with time and place. Initially a simple notion was anything perceptibly better than a basic ISDN line. This implies a rate around or exceeding 144 kbps, although customers did accept less if this was the best available to them. A common current understanding is "a service that is always on, and can scale up to at least 2 Mbps". Other definitions do not specify transmission capacity because of the continued evolution of bandwidth. A 2004 Commission Communication (COM(2004) 369: "Connecting Europe at High Speed: National Broadband Strategies") referred to "a wide range of technologies that have been developed to support the delivery of innovative interactive services, equipped with an always-on functionality, providing broad bandwidth capacity that evolves over time and allowing the simultaneous use of both voice and data services". One of the basic components of the broadband development is the convergence of the backbone, metro and the access networks. The convergence is expected to contribute to the integration of all electronic services. The convergence is based on digitalisation which is a major parameter of this process; however it is still only one amongst several parameters that influence the convergence at the infrastructure level. It is also important to emphasise that the success or failure of convergence is not directly connected to the capability of one infrastructure to integrate all services. None of the infrastructures available can integrate all the services in their current state. While integration of the back-bone parts of the networks have higher possibility to evolve due to common ownership and technology convergence, the integration of the last mile which cover the access network has been shown to be dependent on many different parameters. Common trends appear in the development of the access technologies like decentralised architectures and common optical transport, new physical and MAC layers, common IP architectures (including QoS, charging, device provisioning and security) as well as in the approaches taken in the fixed and mobile networks. The progressive introduction of the new IP protocol, IPv6 in the access, then in the backbone is also developed. "Backbone and Metro" networks have similar trends with different constraints and roadmaps; the clear recent tendency is an evolution towards lower CAPEX & OPEX architectures with "multiple services centric" compared to the old "voice centric services". A clear trend is a progressive evolution of SDH technology which remains clearly dominant in that field. Trends present at the scene are dynamic networking, more efficient robust modulation and transport format, reduction of protocol stacks, multiprotocol support with GMPLS, and the progressive apparition of Ethernet in the Metro. In this paper we compare and evaluate suitable methods for delivering broadband services at the metro and backbone level. In particular, the paper presents a techno-economic analysis of two key broadband technologies for metropolitan or backbone networks: Our studies are based on a modelling methodology for network value analysis that involves CAPEX and OPEX calculations, while the overall technology

deployment financial assessment is based on techno-economic evaluation measures such as net present value (NPV) and internal rate of return (IRR). The studied case is the Telekom Slovenia network which is the incumbent operator in Slovenia. The study address the upgrading of the current Telekom Slovenia network backbone and takes in account future development of broadband services in the country.

2 Network and technology development consideration

"All-Optical" is the vision for future wired networks, where fibre is used for all wires in the WAN, MAN and Access. Nowadays, optical fibres are ubiquitous in the backbone network, and an extension to access networks is the logical next step. An optical fibre based access network offers of all available technologies by far the highest speed and can support an unlimited set of services. The backbone, national or wide area network may extend over distances of thousands of kilometres and provides an interconnection fabric for regional and metropolitan networks. In recent years considerable capacity has been installed in this network layer, so major investment is not expected in the near future. Current investment is focused on developing revenue streams or reducing operational expenditure. There is a trend towards reducing the number of major network nodes and building a very high capacity backbone, essentially a fabric of very high capacity pipes, with much of the processing and routing devolved to the regional and metro layers. The deployment of Wavelength Division Multiplexing (WDM) techniques and equipment in the field has provided backbone networks with high capacity and long reach capabilities. In this part of the network, in order to maximise the use of the available fibre bandwidth, the trend has been to develop systems with more WDM channels together with higher bit rates. Currently deployed systems could transmit 160 channels each at 10Gbit/s. In reality, as yet, few links use more than a handful (~20) of these channels however. Although practical 40Gbit/s systems have been developed, the economic downturn in telecoms has delayed their take up. The optical links are point to point and are terminated in electronic SDH/SONET switches. The SDH/SONET layer provides network management and switching of the links. The functions it provides include:

- Connection set-up
- Connection and link performance monitoring
- Management data communications
- Protection and restoration

The network providers have a large investment in SDH/SONET equipment. The downturn in the telecommunications sector from 2000 onwards has restricted investment and limited expectations to more realistic horizons than those professed in the late 1990s. The implication of this is that near term investments are likely to be based mainly on SDH/SONET technology variants. Another consequence is that any new technology deployed in an existing network will necessarily have to work alongside SDH/SONET. As voice and data networks converge there is a force for upgrading SDH/SONET. Next Generation SDH/SONET is being employed to provide an evolutionary upgrade to the legacy infrastructure by introducing new SDH switches. The Slovenian national operator Slovenia Telecom is facing the same dilemma regarding the upgrading the current national backbone network. In the sections that follow the selection of the appropriate technology is presented based on the information and data provided by the operator and by assessment of the future demands in the country. The technology is evaluated in view of the current state of the network and the future demands of the Slovenian customers. The study starts by identifying the network loops and the required upgrade to cover the next 5 years. The technical assessment is followed by economical evaluation of the OPEX and CAPEX index. Net Present Value and Internal Rate of Return are used as well as indicators that have contributed to the final selection of the technical variants.

3 The case study network and the future prospects

In Slovenia, households are using one of the following types of broadband access:

- ADSL that is being followed now by ADSL +2, offered by Telecom Slovenia
- Coaxial networks with cable modems, offered mainly by the cable operators and
- HFC Network (Hybrid Fibre Coax) which is combination of optical and coaxial networks, also offered by the cable operators.

In the business sector, SME's are mostly using ADSL and cable modems, while middle and large enterprises are using leased lines with ATM switches or Frame Relay (block mediation). Some of the business customers are still using narrowband dial-up modems due to the limitation of their business capacity. There are also residential users that are using wireless access and business customers using Ethernet access over optical lines. In order to optimize the resources the incumbent operator owning the largest part of access networks is using the DSLAM (Digital Subscriber Line Multiplexer) equipment located in urban areas where the concentration of users is highest. The estimated population coverage in towns/cities with this technology by the end of 2005 was around 95%. The growth of BB access is a result of the higher usage of different e-services such as TV, e-banking, e-learning, movie and music download and real time gaming. In order to provide an extension and upgrading of the current Telecom network the following parameters must be considered:

- DSLAMs are connected to edge routers that are or will be positioned on urban locations within cities with

larger population. In case of customers that are located on distance smaller than 50 km, the DSLAMs equipment is directly connected to the customer equipment using fibber. In that context, we can allocate the geographical points where the upgrading of the connectivity to the backbone network will take place These location are the following:
Koper, Postojna, Nova Gorica, Idrija, Kranj, Bled, Novo Mesto, Krško, Celje, Velenje, Ravne, Maribor, Ptuj and Murska Sobota,
•Traffic evaluation anticipates connection of DSLAM units with GbE interfaces, so the edge routers will have nxGbE or 10GbE interfaces, which requires in the transmission systems of the backbone network channels with 10G/bit bandwidth.
•The transmission systems will be built in ring topology.

In addition to that the Telecom network will still offer connectivity with use of Dial-up and Ethernet. Dial-up access (analogue and ISDN) is carried out over the existing SDH network, and no essential growth of the existing traffic is foreseen. Regarding the use of Ethernet connectivity over optical connections, Telekom Slovenia is expecting similarly to the other operators growth of the demand. Ethernet connectivity over optical fibre is offered on the network side, using a specific switch or router. It is also anticipated that there will be an increased demand for the provision of virtual private networks (VPNs), which may have an impact on the decision to enlarge the network topology and consequently on additional network-access locations around the country. Telekom Slovenia's backbone network is also used to run the operator service and for the leased-line market. This leased-line market is well developed and leased lines are used by many enterprises, such as banks, mobile operators, (e.g., Western Wireless International, Simobil/Vodafone, and Mobitel), dedicated networks under state ownership and management (e.g., ARNES, the academic network) and HKOM, the network services provided to the Slovenian government and administration. Leased lines are also used by new entrants to the telecommunications market, e.g., internet service providers like T2, Voljatel and Medinet. These users are mostly using 2M and 34 M PDH or STM-1 interfaces, providing 155Mbit/s bandwidth capacity.

The network topology is presented on Fig.1. The optical part of the network consists of single-mode optical fibres. They are spread over the whole country, providing a transmission capacity of more than 10 GB/s. The redundancy of the lines is due to operational requirements, as Telekom Slovenie provides basic telephone, data transmission and other (TDM) services with a QoS (quality of service) provision.

Figure 1: Telecom Slovenia backbone network.

Increase of data communication traffic is expected in the international connections of the network, in the mobile network with the UMTS technology implemented by the operator daughter company – Mobitel. Other demands are expected regarding bandwidth for the DRC (Disaster Recovery Centre), which are becoming important customers. Inquiries for such centres come from Slovenian State administration, various Ministries and banks. These users in majority of cases need SAN (Storage Area Network) interfaces. Last larger group of customers that influence the demand for bandwidth are systems or enterprises which are asking for VPNs and own private Ethernet networks.

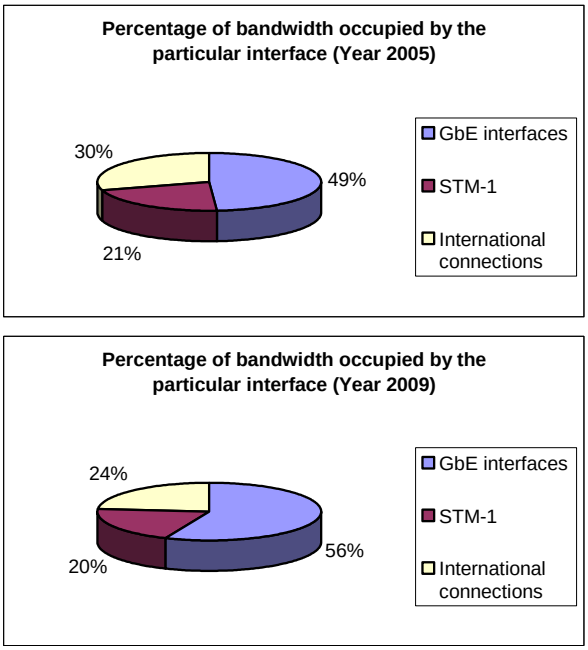

Figure 2: Comparison of bandwidth occupied by particular interfaces in year 2005 and 2009.

They are interested in FE (Fast Ethernet) or GbE bandwidth on the backbone level. The expected changes in interface occupancy regarding bandwidth in 5 years time span is presented on Figure 2.

The fast growing needs for additional bandwidth, required by the new services, are important, additional challenge is the merge of different kind of traffic on one backbone platform. The decision for a transmission technology is simple as optical cable with single-mode

fibres is the only acceptable solution in economical and physical aspects. The media for upgrading and extension of the backbone network is known, the question that needs an answer is the appropriate technology. We have identified two technologies to be available for the backbone upgrading and extension of the Telecom network, wavelength multiplexing (WDM) and SDH systems of transmission. The selection depends on different factors and criteria. The next section provides more information about the approach and the upgrading proposal.

4 Technical and cost evaluation of the upgrading

Telekom network customers are familiar with the SDH interfaces as the current network is built up with SDH technology. They are cheaper than WDM (for the present needs); however they still have weakness in comparison to WDM. With enlargement of the network and the upgrading of the bandwidth capacities parallel systems will be required as the current can not answer to the increased demand. This can become expensive and time wasting. Technical solution for upgrading the existing network with WDM technology requires 16 channel DWDM transmission systems. The number of channels is specified according to the number of network elements in the loops and the estimated traffic. In case of SDH based equipment an SDH STM-64 transmission system is required capable to provide capacity of 10Gbit/s. In both cases the bandwidth per channel is same, and as a consequence the locations of the network elements are same. The upgrading with SDH technology requires for the Telecom network upgrading the following SDH equipment: network elements on all specified relations shown at the network topology (Fig.1), SDH STM-64 modules and compensation fibres. The WDM technology upgrading variant requires network elements on all locations with appropriate number of channels (DWDM systems are 16 channelled), multiplexing equipment (for locations, with highest traffic), compensating fibres, since in the 10Gbit/s transmission, the dispersion needs to be compensated.

In order to evaluate the optimal solution for the network upgrading in financial aspect CAPEX (capital expenditure) index and OPEX (operational expenditure) index were calculated for period of 5 year time span (later, new technologies are expected). In the CAPEX index price of equipment, software, installation, takeover test, setting and wiring are considered. (Caballero, 2005, page 190). In addition to that the operational cost that was also calculated includes depreciation of the equipment, maintenance of the connections, monitoring the functioning of the system and corrections, creation of new connections, system support: - maintenance contract and costs of used transmission media. The accounting regulations of Telekom Slovenie, allow appreciation of 14, 3% for the equipment and 5% for the media per year.

5 Findings and results

The calculation provided numerical values for both indexes: CAPEX for SDH and for DWDM. They are compared and presented on Figure 3.

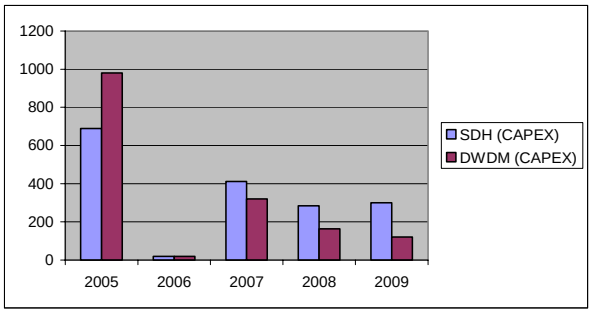

Figure 3: Comparison of CAPEX values in millions of euros for SDH and DWDM.

The data on Figure 3 shows clearly that the investments cost in fixed assets for the DWDM variant is significantly higher in the first year of investment compared to the cost for SDH equipment. In the second year both indexes come close together. In the last three years of the 5 year time span the DWDM cost of investment is much lower compared to the cost required for the SDH. The higher cost in the first year for DWDM is due the purchase of expensive amplifiers and compensating fibres, later only line cards are necessary to be purchased. In the case of SDH variant each expansion of the line capacity, requires a purchase of additional amplifiers, compensation fibres and expensive line cards. In long term the investment in SDH demands more spending and this is especially obvious if the considered time period is longer e.g. 6 or 7 years. Feeding interfaces are same in both variants since the same access traffic is expected in both cases.

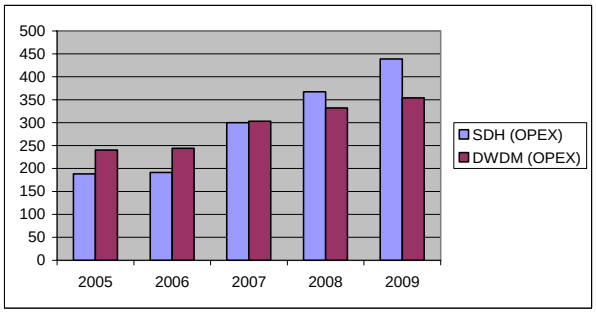

Figure 4: Comparison of OPEX value in millions of EUR between SDH and DWDM solutions.

The graph in Figure 4 shows the differences between the two OPEX indexes and they are results of two factors. The first is due to the large investments required in the DWDM case variant in the first year. In line with this the depreciation cost and maintenance contract are also somehow higher than for the SDH variant. Cost of personnel for maintenance of the equipment or systems is the same for both variants, so this does not contribute to the difference between both OPEX indexes. The increase of a bandwidth requires

upgrading of the line capacity. Sufficient line capacity in DWDM variant system is provided in the first year and the system continuously works over two fibres, upgrading, or additional fibres are not required. In contrast to that in the case of SDH equipment usage each increase of line speed demands additional two fibres. This is why the media item appears as a constant in the DWDM variant, while in the SDH variant this item is much higher. In case of unexpected additional needs for bandwidth, this variant becomes much more disadvantageous. This is the second factor that influence on the difference of the two OPEX indexes. The next Figure 5 presents the comparison of both variants with aggregation of both indexes. The lines represent cumulative values of the aggregated indexes.

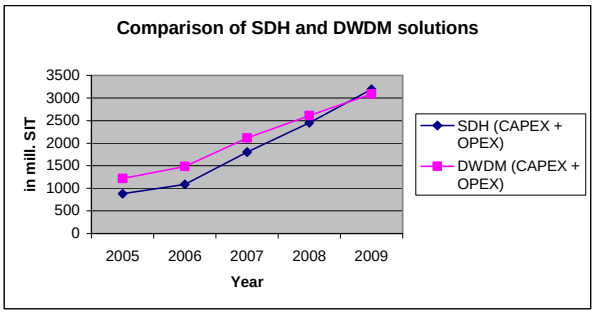

Figure 5: Comparison of SDH and DWDM variants with aggregated indexes (CAPEX + OPEX).

The findings from the two variant comparison of particular index as presented on Table 1 and Table 2 are reflected as well in the graphs presented on Figure 5. The smaller cost of investment in SDH equipment is followed with the higher costs of maintenance for upgrading, in absolute value comes close to the cost of the DWDM variant. The line for the SDH variants in the second year is rising with bigger slope then the line for the DWDM variant. In case of prolonged time period e.g. for two more years (all together 7 years were considered according to the accounting regulation of Telekom Slovenia) the lifecycle of such equipment, and the slope of the line will become even bigger and the indexes will show bigger difference in favour of the DWDM variant.

By taking into account all previously stated findings it can be concluded that for the upgrading of the Telekom Slovenia network much more attractive variant is the selection of DWDM technology. However, in this type of analysis the time component, Network Present Value and Internal Rate of Return are necessary to be considered as well. In this type of calculation the network present value that is taken in account contributes to positive and negative cash flows. Positive cash flows are the inflows from bandwidth lease, negative cash flows are the costs associated with the investment (equipment, reserve units and services), salaries, cost of the maintenance contract. In addition to that as negative cash flow are considered also the use of fibres. NPV (Net Present Value) is calculated according to the dynamic investment evaluation method that considers the life cycle of the investment (I), the positive cash flow in particular year, the interest rate and the initial investment.

$$NPV = \left(\sum_{t=1}^{n} \frac{D_t}{(1+r)^t} \right) - \left(\sum_{t=0}^{n} \frac{I_t}{(1+r)^t} \right)$$

Here r is the interest rate and I am the investment and D is the positive cash flow in particular year. The calculation method clearly shows the dependence of the NPV from the value of the capital used for the investment in course of the time (1 to t) and the interest rate associated with it. The other parameter calculated is the Internal Rate of Return (IRR). In our case the Modified IRR was used as this parameter enables re-investing of the cash flow with lower interest rate. The formula as suggested by Brigham (Brigham, 1977, p.411) was applied:

$$I_0 = \frac{\sum_{t=1}^{n} D_t (1+r)^{n-1}}{(1+MISD)^n}$$, where MISD is the modified IRR.

Both variants were compared also with the value of the Internal Rate of Return (IRR), which is calculated three times with different value for the interest rate. In the expression for NPV, r is the interest rate, I am the investment and D is the positive cash flow in a particular year. The other measure calculated is IRR. In our case, the modified IRR (MIRR, [14]) was used, as this parameter allows for the re-investment of the cash flow with a lower interest rate. In Fig 6 the NPV profiles for both variants (SDH and DWDM) are shown.

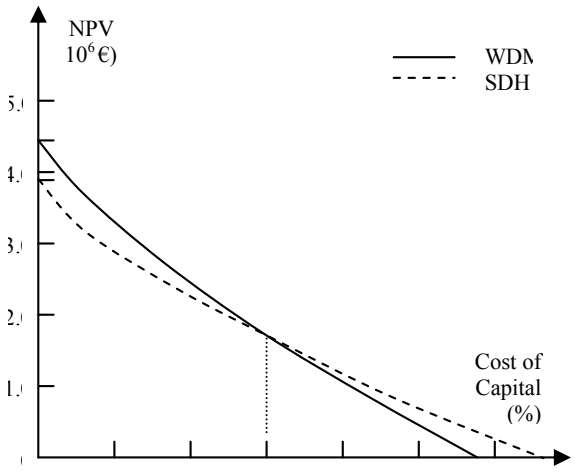

Figure 6: Net present value profiles for SDH and DWDM solutions.

We can see that the IRR method favours the SDH technology (IRR = 33%) over DWDM (IRR = 28%), but the NPV is the same for both variants at 15% ("the crossover rate"), and higher for DWDM below that

point. Since both options are mutually exclusive, we can say that if the capital costs are lower than 15% (as they are in our case) the DWDM option is preferable, since we can expect a greater NPV from it.

From the technological point of view the upgrade of Telekom Slovenia backbone can be carried out with either the SDH or DWDM technology. However, a comparison of the costs of the concerned technologies shows that the DWDM variant is a better as a long-term choice, especially with current costs of capital below 15%, which enables greater NPV value. The DWDM variant is better in longer time period e.g. 5-7 years and could be said that is optimal. This solution offers as well better price performance ratio. The same applies in the case of capital investment and return.

6 Conclusion

Over the years there has been an evolution in the backbone networks from analogue to digital transmission, from Plesiochronous Digital Hierarchy (PDH) to Synchronous Digital Hierarchy (SDH) and recently from SDH to SDH upon Wavelength Division Multiplexing (WDM). Initially digital transmission was introduced with a capacity of 2 Mbit/s (the primary multiplexers) and a granularity of 64 kbit. A next step was to improve the transmission efficiency by allowing higher bitrates and introducing cross-connection, so that currently, with SDH, the granularity is 155 Mbit/s and a line capacity of 10 Gbit/s is possible. Advances in electronic processing could not follow the traffic growth so the next step is obviously the introduction of WDM. Here currently deployed systems are capable of capacity of 1600 Gbit/s (160 wavelengths) or even more, with a granularity of 10 Gbit/s. While the capability to use 160 wavelengths allows growth capacity, few systems currently deploy the full capability. Recent technology evolution gives the possibility of the WDM transport layer migrating from simple transmission links into an elaborate network providing switching, with higher manageability, lower complexity and cost. In such network scenarios, optical routes form connections between discrete point network locations through optical add/drop and cross connect nodes and provide traffic allocation, routing and management of the optical bandwidth. They also facilitate network expansion, traffic growth, churn and network resilience. Optical cross-connects are located at nodes cross-connecting a number of fibre pairs and also support add and drop of local traffic providing the interface with the service layer. Our study results show also that WDM technology in the case of the Telecom Slovenia network appears to present a better economic solution (based on the NPV and IRR indicators) compared to a SDH network upgrading. Thus, an investment in WDM presents improved prospects for answering to the increased demand of bandwidth and services. The results were based on exactly the same market demand and revenue assumptions for both technologies considered in the study, a particular concrete market with differentiation only in technology chains and related cost assumptions.

Acknowledgment

Part of this work was carried out within European project BReATH from FP6, BroadBand for all Strategic Objective, 2004-2006.

References

[1] D. J. Bishop, C. R. Giles, and G. P. Austin 2002, "The Lucent lambdarouter: MEMS technology of the future here today," *IEEE Commun.Mag.,* pp. 75–79.

[2] P. B. Chu, S.-S. Lee, and S. Park 2002 "MEMS: The path to large opticalcross-connects," *IEEE Commun. Mag.,* pp. 80–87, *Photon. Technol. Lett., vol. 10,* pp. 896–898.

[3] P. De Dobbelaere, K. Falta, L. Fan, S. Gloeckner, and S. Patra, 2004 "Digital MEMS for optical switching," *IEEE Commun. Mag*., pp. 88.

[4] R. DeSalvo et al., 2002, "Advanced Components and Sub-Systems Solutions for 40 Gb/s Transmission", *J. Lightwave Technol*., vol. 20.

[5] E. Iannone, and R. Sabella, 1996, "Optical Path Technologies: A Comparison Among Different Cross-Connect Architectures", *J. Lightwave Technology* Vol. 14, pp. 2184 - 2196,.

[6] M. Jose Caballero 2005, "Migration to Next Generation SDH", Barcelona: *Printulibro, Intergroup, pp.*, 182-191.

[7] Costas Courcoubetis, Weber Richard, 2003, "Pricing Communication Networks, Economics, Technology and Modelling*",* West Sussex, John Willey and Sons, 347.

[8] Sato Ken-ichi 2004, "Key enabling Technologies for Future Networks", *Optics and Photonics News"*, p. 34, 2004.

[9] J. Lacey 2002, "Optical cross-connect and add/drop multiplexers: technologies and applications", (Tutorial), WT1, *83 IST OPTIMIST consortium: Technology Trend Documents, (www.ist-optimist.org) pp.83-86.*

[10] B. Jerman Blažič Borka 2000, "Notes from the subject Contemporary telecommunications techniques and services, Stack of TCP/IP protocols*"*, *Ljubljana: Faculty of Economics*, p. 149-152.

[11] A. Jelinek 2003, "SURPASS hiT 70, A New Generation of Networking", *ISN Carrier Networks"*, Siemens, p. 32-33.

[12] Ministry for information society 2004, "Strategy of development of broadband data networks in Republic of Slovenia"*; (URL: http://www2.gov.si/mid/mid/.nsf/)*

[13] R. Ramaswami and K. N. Sivarajan 1998, "Optical Networks, A Practical Perspective". *San Fransisco, CA: Morgan Kaufmann,* p.159-162.

[14] R. Ramaswam 2001, OFC"Using All-Optical Crossconnects in the Transport Network", (invited), *WZ1-I pp.8-10,*

[15] A. Tzanakaki, I. Wright and S. S. Sian SCI2002, "Wavelength Routed Networks: Benefits and Design Limitations*", SCI Proceedings, Orlando, Florida,* p.34-38.

[16] G. Wilfong, B. Mikkelsen, C. Doerr, and M. Zirngib 1999, "WDM Cross-Connect Architectures with Reduced Complexity", *J. Lightwave Technology* Vol. 17, pp. 1732 – 1741

Efficient Constraint Validation for Updated XML Databases

Béatrice Bouchou, Ahmed Cheriat and Mírian Halfeld Ferrari
Université François-Rabelais de Tours/Campus Blois - LI - France
E-mail: beatrice.bouchou@univ-tours.fr, ahmed.cheriat@univ-tours.fr, mirian.halfeld@univ-tours.fr

Dominique Laurent
Université de Cergy-Pontoise - ETIS - UMR CNRS 8051 - France
E-mail: dominique.laurent@dept-info.u-cergy.fr

Maria Adriana Lima
Pontifícia Universidade Católica de Minas Gerais - Poços de Caldas - Brazil
E-mail: adriana@pucpcaldas.br

Martin A. Musicante
Universidade Federal do Rio Grande do Norte - DIMAp - Natal - Brazil
E-mail: mam@dimap.ufrn.br

Keywords: XML, schema, constraints

Received: May 12, 2006

XML constraints are either schema constraints representing rules about document structure (e.g. a DTD, an XML Schema definition or a specification in Relax-NG), or integrity constraints, which are rules about the values contained in documents (e.g. primary keys, foreign keys, etc.).
We address the problem of incrementally verifying these constraints when documents are modified by updates. The structure of an XML document is a tree, whose nodes are element (or attribute) names and whose leaves are associated to values contained in the document. Considered updates are insertion, deletion or replacement of any subtree in the XML tree. Schema constraints are represented by tree automata and tree grammars. Key and foreign key constraints are represented by attribute grammars, adding semantic rules to schema grammars, to carry key and foreign key values (to verify their properties).
Our incremental validation tests both schema and integrity constraints while treating the sequence of updates, in only one pass over the document. Only nodes involved in updates trigger validation tests. An analysis of complexity shows that worst cases are determined by the shape of the XML tree being processed (asymptotic upper bounds are presented). Experimental results show that our algorithms behave efficiently in practice.

Povzetek: Opisana je inkrementalna verifikacija podatkovnih baz XML.

1 Introduction

XML is now a standard for exchanging data and, by extension, for representing information. For reliable exchange as well as for information system design, it is necessary to define rules that data must conform to.

XML documents can be represented as unranked trees: nodes are identified by a position and associated to a label, which is the name of an element or an attribute. Our tree representation of an XML document is exemplified in Fig.1. This figure shows the representation of part of an XML document, which will be used in some of our examples. Notice that the data values in the XML document appear as leaves of its tree representation.

XML documents can be constrained either by *structural rules* or *semantic rules*. Rules about the structure of documents are called a *schema*: there are several formalisms to express these constraints, *e.g.* DTD, XML Schema (XSD) [4] or Relax-NG [41]. It is a well known fact that a schema can be represented by a tree grammar. From this grammar, it is easy to derive a tree automaton [20, 43]: the validation of the document wrt the schema is the run of this tree automaton over the XML tree.

Integrity constraints impose restrictions to the values that may appear in XML documents. Integrity constraints (*e.g.*, primary and foreign keys) are devised to improve the semantic expressiveness of XML, in the same way as their counterpart in relational databases. In this paper we propose to represent integrity constraints by attribute grammars [8, 40], adding semantic rules to schema grammars, to carry key and foreign key values (to verify their properties).

We address the problem of *incremental* validation of *updates* performed on a valid XML document (*i.e.*, one that respects a given set of constraints), represented by the XML tree T. In our approach, update operations corre-

spond to the insertion, the deletion and the replacement of subtrees of T. They are results of a user request's preprocessing. A user can modify an XML document either by using a text editor such as *XmlSpy* [1] or *Stylus studio* [46], or by means of a program, written in an update language such as *XSLT* [26] or *UpdateX* [47], which is embedded in XQuery [16].

In the case of a text editor, different scenarios are possible: validity is tested while the user changes his document (this method is cumbersome and not realistic since it implies verification after each change) or validity tests are explicitly activated (by the user) after the performance of several changes. In the latter context, some updates can be "subsumed" by others (*e.g.*, the user can build or change a part of the document and finally delete it). Thus, before applying an *incremental* validation method one should define which updates are to be taken into account. To this end, one can either use pre-processing over the set of changes performed by the user or compute the difference between the last verified version and the current (modified) one. In both cases, the overhead caused by this pre-processing may be greater than incremental validation profit, this is why most of the existing XML editors apply a complete re-validation.

The use of a language to specify updates open different possibilities of work. Considering the integration of such an approach to a text editor framework, we may imagine that an update language editor would be capable of specifying update positions and then our algorithms would directly apply. More generally, an update language allows to specify updates such as, for example: *increment by 10 percents all accounts of a given consumer*. Thus, it allows to manage XML documents from a database point of view. To this end, update languages integrated in XQuery are the most promising solutions. The W3C has edited a first public working draft on *XQuery Update Facility* [22]: it recommends that the evaluation of any expression produces either a new XML document or a pending update list, which is a set of update primitives. An update primitive has a target node and consists in insertions, deletions or replacements to be operated at this node (or just before, or just after or just under this node, in case of insertions). The update list can be held in wait until an operation (upd:applyUpdate) is performed. An operation of validation (upd:revalidate) must exist.

Update operations considered in this paper can be seen as updates in such an update pending list.

Only updates that do not violate constraints are accepted. Thus, before applying updates on a valid tree, we need to test whether these updates do not violate the validity of the tree. The goal of an incremental validation method is to perform these tests without testing the entire tree, but just the part of it which is concerned by the updates. In accordance with the *snapshot semantics* [14, 47], document validity is assured just after considering the *whole* sequence of updates. The snapshot semantics consists in delaying update application to the end of query evaluation. In this way, all updates always refer to the original document[1].

In the following, we roughly explain how the incremental validation tests are performed. Let *UpdateTable* be the sequence of updates to be performed on an XML tree T. To visit T we proceed in a depth-first visit of the XML document, triggering tests and actions according to the required updates (in *UpdateTable*). To simplify our explanation, schema and integrity constraints are treated as separated sub-routines. A system that makes these routines work together is a simple generalization of the one presented in this paper. The sub-routines can be summarized as follows.

Schema constraint routine:

• When an update node is reached, the required update is taken into account (considered as done). Nodes which are descendant of update nodes are skipped *i.e.*, they are not treated.

• For each node p which is an ascendant of an update position a *validation step* is activated when reaching the close tag corresponding to p. A validation step at position p verifies whether p's children (in the updated tree version) respect schema constraints. For instance, if a sequence requires updates on positions $0.1.2$ and 0.3 of Fig. 1 then a validation step is activated on nodes 0.1, 0 and ϵ.

• All other nodes (those that are not on the path between the root and an update position) are skipped, *i.e.*, no action is triggered on them.

Integrity constraint routine:

• Similarly to the schema constraint routine, when an update node is reached, the required update is taken into account (considered as done). However, contrary to the schema constraint routine, in the integrity constraint routine the removal of a subtree (rooted at an update position p) triggers the subtree traversal for searching key and foreign key values involved in the update. For instance, consider a key constraint on the document of Fig. 1 establishing that in a *collection*, a *recipe* is uniquely identified by its *name* and *author*. We assume that a deletion is required at position 0.1. The deletion "treatment" is activated (in order to find key values) when the open tag `<recipe>` is reached. After the traversal of the subtree rooted at position 0.1, *i.e.*, when reaching the close tag `</recipe>` a verification test is performed. This test consists in checking whether the key value $\langle Shrimp\ Soup, J.Fox\rangle$ (involved in the deletion) is referenced by a non deleted foreign key. To perform this test the routine uses an auxiliary structure (called *keyTree*) created during the first key validation (from scratch). During the tree traversal necessary to analyze an update sequence, some marks can be inserted into the *keyTree* to indicate that the document is temporarily invalid. A subsequent update in the same transaction can reestablish validity and remove the mark.

[1]The language XQuery! (*"XQueryBang"*) [33] extends XQuery 1.0 with compositional updates, offering user-level control over update application. The user controls update semantics via an operator called *"snap"* (for snapshot). However, it can also implements snapshot semantics.

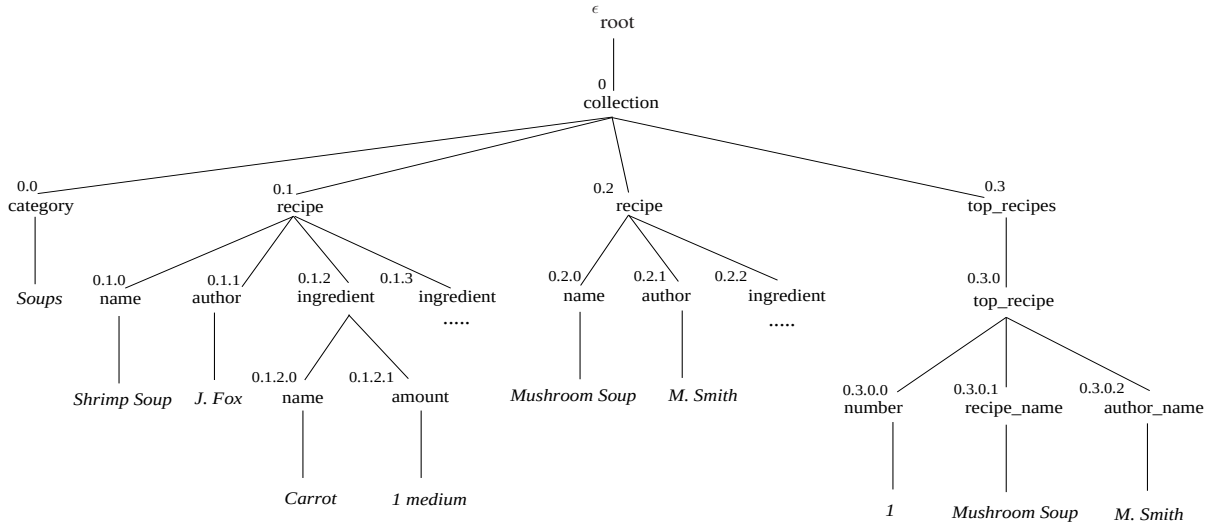

Figure 1: Tree representation of an XML document.

• When reaching the root, a final test verifies if any violation mark exists in *keyTree* (*i.e.*, a constraint violation not corrected until the end of the traversal).

• All other nodes are skipped. □

XML trees are treated in the style of SAX [3], in order to allow the treatment of much bigger XML documents [42][2]. This choice implies a complete XML tree traversal but it avoids the use of an auxiliary structure for incrementally verifying some kinds of schema constraints. It is worth noting that, although our algorithm visits all nodes in the XML tree, only some nodes trigger validation actions while others are just "skipped" (*i.e.*, no action is activated when reaching them). The cost of skipping nodes is not relevant when compared to the cost of validation actions. Thus, in Sections 3.2 and 4.3 we consider the complexity of our method only wrt the total number of validation actions that should be performed.

The algorithms presented in this paper have been implemented in Java, and experimental results demonstrate advantages of the incremental schema verification over the verification from scratch, for multiple updates over large XML documents. Experimental results obtained with our key and foreign key validation routines are also good, despite their theoretical complexity: curves grow almost linearly with the size of the processed document. Tests for the incremental key verification programs give even better results than those for the verification from scratch.

The main contribution of this paper is the integration of incremental schema, key and foreign key validation, while dealing with multiple updates. It extends our previous work [7, 17, 18] not only in this aspect but also in the use of attribute grammars to deal with integrity constraint verification.

[2]It is also possible to consider our routines in a DOM context. In this case space requirements are bigger, but time complexity can be smaller.

By dealing with unranked trees, usually much shorter than the binary ones, the complexity of our incremental schema validation method happens to be similar to the one proposed in [11, 45]. However, contrary to [11, 45], our update operations can be applied at any node of the XML tree and auxiliary structures are not necessary when using DTD and XSD. In terms of key validation, our proposition is close to [23], but contrary to them, we treat foreign keys and multiple updates. Moreover, our routines for schema validation and integrity constraint verification can work simultaneously.

This paper is organized as follows. Section 2 introduces some necessary concepts. In Section 3 we consider the validation wrt schema constraints while in Section 4 we present our validation method wrt key and foreign key constraints. In section 5 we conclude and discuss some perspectives.

2 Background

An XML document is an unranked labeled tree where: (i) the XML outermost element is the tree root and (ii) every XML element has its sub-elements and attributes as children. Elements and attributes associated with arbitrary text have a *data* child. Fig. 1 shows an XML tree.

To define our trees formally, let U be the set of all finite strings of positive integers (which usually we separate by dots) with the empty string ϵ as the identity. The *prefix relation* in U, denoted by $\preceq$ is defined by: $u \preceq v$ iff $u.w = v$ for some $w \in U$.

Now, if Σ is an alphabet then a Σ-valued tree T (or just a tree) is a mapping $T : dom(T) \rightarrow \Sigma$, where $dom(T)$ is a tree domain. A finite subset $dom(T) \subseteq U$ is a (finite) *tree domain* if: (1) $u \preceq v$, $v \in dom(T)$ implies $u \in dom(T)$ and (2) $j \geq 0, u.j \in dom(T), 0 \leq i \leq j \Rightarrow u.i \in dom(T)$.

Each tree domain may be regarded as an unlabeled tree, *i.e.*, a set of tree positions. We write $T(p) = a$ for $p \in$ *dom*(T) to indicate that the symbol a is the label in Σ associated with the node at position p. For XML trees, Σ is composed by element and attribute labels together with the label *data*. We consider the existence of a function *value* that returns the value associated with a given *data* node.

Let T be an XML tree, valid wrt some integrity and schema constraints, and consider updates on it. We assume three update operations (*insert*, *delete* and *replace*) whose goal is to perform changes on XML trees. Fig. 2 illustrates the individual effect of each update operation over T.

In this paper we are interested in multiple updates, *i.e.*, we suppose an input file containing a sequence of update operations that we want to apply over an XML tree The effective application of this sequence of updates depends on its capability of preserving document validity. In other words, a valid XML tree is updated (according to a given update sequence) only if its updated version remains valid. The acceptance of updates relies on *incremental validation*, *i.e.*, only the validity of the part of the original document directly affected by the updates is checked.

A sequence of updates is treated as one unique transaction, *i.e.*, we assure validity just after considering the whole sequence of updates - and not after each update of the sequence, independently. In other words, as a valid document is transformed by using a sequence of primitive operations, the document can be temporarily invalid but in the end validity is restored. This extends our previous work in [7, 17, 18] and follows the ideas in [14, 47].

Let $UpdateTable$ be the relation that contains updates to be performed on an XML tree. Each tuple in $UpdateTable$ contains the information concerning the update position p and the update operation op. In this paper we assume that $UpdateTable$ is the result of a pre-processing over a set of updates required by a user. In the resulting $UpdateTable$ the following properties hold:

P1 - An update position in an $UpdateTable$ always refers to the original tree. Consider for instance the tree of Fig. 1. In an $UpdateTable$ an insertion operation refers to position 0.3 even if a deletion at position 0.1 precedes it.

P2 -An update on a position p excludes updates on descendants of p. In other words, there are not in $UpdateTable$ two update positions p and p' such that $p \preceq p'$.

P3 - $UpdateTable$ then one of the operations involving p can be *replace* or *delete*, but all others are *insert*.

P4 - Updates in an $UpdateTable$ are ordered by position, according to the document order.

3 Schema verification

A tree automaton can be built from a schema specified using schema languages such as DTD, XSD or RELAX NG. In our approach, we use a bottom-up unranked tree automaton capable of dealing with both (unordered) attributes and (ordered) elements in XML trees [17].

Definition 1 - Non-deterministic bottom-up finite tree automaton: A tree automaton over Σ is a tuple $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ where Q is a set of states, $Q_f \subseteq Q$ is a set of final states and Δ is a set of transition rules of the form $a, S, E \rightarrow q$ where *(i)* $a \in \Sigma$; *(ii)* S is a pair of disjoint sets of states, *i.e.*, $S = (S_{compulsory}, S_{optional})$ (with $S_{compulsory} \subseteq Q$ and $S_{optional} \subseteq Q$); *(iii)* E is a regular expression over Q and *(iv)* $q \in Q$. □

The tree automaton $\mathcal{A}$ obtained from a schema specification D may have different characteristics according to the schema language used for D. These characteristics, discussed below, match the taxonomy of regular tree grammars introduced in [42].

Let $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ be a tree automaton. Two different states q_1 and q_2 in Q are *competing* if Δ contains different transition rules ($a, S_1, E_1 \rightarrow q_1$ and $a, S_2, E_2 \rightarrow q_2$) which share the same label a. Notice that we assume that no two transition rules have the same state in the right-hand side and the same label in the left-hand side, since two rules of this kind can be written as a single one. A regular expression E in a transition rule *restrains competition* of two competing states q_1 and q_2 if for any sequence of states α_U, α_V, and α_W, either $\alpha_U q_1 \alpha_V$ or $\alpha_U q_2 \alpha_W$ fails to match E.

Based on the concepts of competing states and competition-restrictive regular expressions, regular tree languages are classified as follows:

$C1-$ *Regular tree languages* (RTL): A regular tree language is a language accepted by any tree automaton specified by Definition 1. The automaton obtained from a specialized DTD recognizes languages in this class.

$C2-$ *Local tree languages* (LTL): A local tree language is a regular tree language accepted by a tree automaton that does not have competing states. This means that, in this case, each label is associated to only one transition rule. The automaton obtained from a DTD recognizes languages in this class.

$C3-$ *Single-type tree languages* (STTL): A single-type tree language is a regular tree language accepted by a tree automaton having the following characteristics: (i) For each transition rule, the states in its regular expression do not compete with each other and (ii) the set Q_f is a singleton. The combination of these two characteristics implies that although it is possible to have competing states, the result of a successful[3] execution of such an automaton can consider just a single type (state) for each node of the tree. The automaton obtained from a schema written in XSD recognizes languages in this class.

$C4-$ *Restrained-competition tree languages* (RCTL): A restrained-competition tree language is a regular tree language accepted by a tree automaton having the following characteristics: (i) For each transition rule, its regular expression restraints competition of states and (ii) the set Q_f is a singleton. No schema language proposed for XML is classified as a RCTL.

[3] See below the definition of the *run* of a tree automaton over a tree.

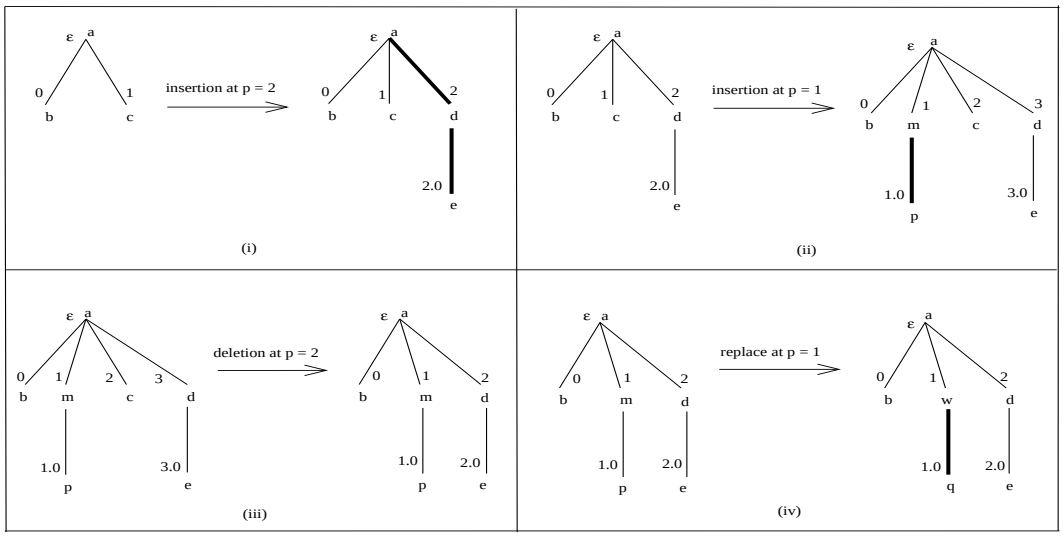

Figure 2: Update operations over a tree T. (i) Insertion at a frontier position. (ii) Insertion at a position of $dom(T)$. Right siblings are shifted right. (iii) Deletion. Right siblings are shifted left. (iv) Replace.

Notice that, as shown in [42], the expressiveness of the above classes of languages can be expressed by the hierarchy LTL $\subset$ STTL $\subset$ RCTL $\subset$ RTL (where $L_1 \subset L_2$ means that L_2 is strictly more expressive than L_1).

Example 1 Let $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ be a tree automaton where $Q_f = \{q_C\}$. Consider the following transition rules with q_{A1} and q_{A2} as competing states. Assume that states q_1, q_2 and q_3 are defined by simple transition rules that do not introduce competition and do not have regular expressions involving q_{A1} and q_{A2}.

$$
\begin{array}{lll}
(1) & a, (\emptyset, \emptyset), q_1 q_2? & \rightarrow q_{A1} \\
(2) & a, (\emptyset, \emptyset), q_1 q_3? & \rightarrow q_{A2} \\
(3) & b, (\emptyset, \emptyset), (q_{A1} \mid q_{A2})^* & \rightarrow q_B \\
(3') & b, (\emptyset, \emptyset), (q_{A1})^* & \rightarrow q_B \\
(4) & c, (\emptyset, \emptyset), (q_B)^* & \rightarrow q_C
\end{array}
$$

In this context, consider different sets Δ containing subsets of the above rules: Firstly, consider a set Δ with rules (1), (2), (3) and (4). The language recognized by $\mathcal{A}$ is a RTL which is not a STTL. Secondly, assume that Δ contains rules (1), (2), $(3')$ and (4). Then the language recognized by $\mathcal{A}$ is a STTL which is not a LTL. Finally, assume Δ has no competing states (*e.g.*, from the above rules, only rules (1), $(3')$ and (4) are in Δ). In this case, the language recognized by $\mathcal{A}$ is a LTL. □

The execution of a tree automaton $\mathcal{A}$ over an XML tree corresponds to the validation of the XML document wrt to the schema constraints represented by $\mathcal{A}$. In its general form, a *run* r of $\mathcal{A}$ over an XML tree T is a tree such that: (i) $dom(r) = dom(T)$ and (ii) each position p is assigned a set of states $\mathcal{Q}_p$. The assignment $r(p) = \mathcal{Q}_p$ is done by verifying whether the attribute and element constraints imposed to p's children are respected. The set $\mathcal{Q}_p$ is composed by all the states q such that:

1. There exists a transition rule $a, (S_{compulsory}, S_{optional}), E \rightarrow q$ in $\mathcal{A}$.
2. $T(p) = a$.
3. $S_{compulsory} \subseteq \mathcal{Q}_{att}$ and $\mathcal{Q}_{att} \backslash S_{compulsory} \subseteq S_{optional}$ where $\mathcal{Q}_{att}$ is the set containing the states associated to each attribute child of p.
4. There is a word $w = q_1, \ldots, q_n$ in $L(E)$ such that $q_1 \in \mathcal{Q}_1, \ldots, q_n \in \mathcal{Q}_n$, where $\mathcal{Q}_1 \ldots \mathcal{Q}_n$ are the set of states associated to each element child of p.

A run r is *successful* if $r(\epsilon)$ is a set containing at least one final state. A tree T is *valid* if a successful run exists on it. A tree is *locally valid* if the set $r(\epsilon)$ contains only states that belong to $\mathcal{A}$ but that are not final states. This notion is very useful in an update context [17].

Restricted forms of schema languages permit simplified versions of *run*. For instance, in a run of a tree automaton for (simple) DTD, the sets $\mathcal{Q}_p$ are always singleton. This situation considerably simplifies the implementation of item (4) above [17]. Moreover, implementations can be enhanced by considering the fact that XML documents should be read sequentially to be validated. While reading an XML document, we can store information useful to avoid the possible ambiguity of state assignment, expressed by the transition rules. For instance, in the implementation of the run of a tree automaton for XSD, each tree node can always be associated to one single state. This state is obtained by intersecting a set of "expected" states (computed during the sequential reading of the document so far) and the set of states obtained by the bottom-up application of the rules of the automaton (Proposition 1).

3.1 Incremental schema verification

Given a tree T and a sequence of updates over T, the incremental validation problem consists in checking whether the updated tree complies with the schema, by validating only the part of the tree involved by the updates. We propose a method to perform the incremental validation of an XML

tree T by triggering a local validation method only on positions that are a prefix of an update position p (including both the root position ϵ and p itself).

Remark: When considering the most general classes of schema languages, during an incremental validation, we need to know which states were assigned by a previous validation to each node of the tree being updated. A data structure containing the result of the run over the original document should be kept. However, schema verification for languages in STTL (*i.e.*, XSD) and LTL (*i.e.*, DTD), does not impose the need for auxiliary, permanent data structures [42]. □

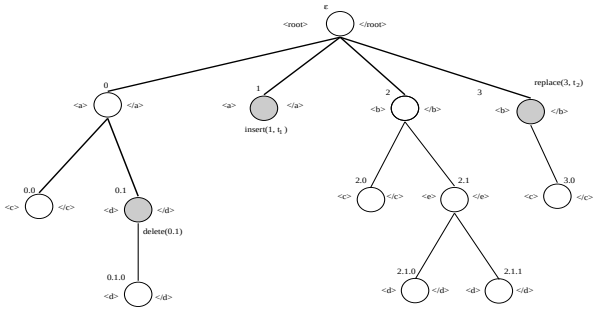

Figure 3: XML tree and update operations.

The following example illustrates our method in an intuitive way.

Example 2 Consider the XML tree of Fig. 3, where update positions are marked. Bold arcs represent the necessary tree transversal for our method, to check the validity of the updates we want to perform.

Let us suppose that there is a tree automaton $\mathcal{A}$, representing the schema to be verified. We also suppose that the original tree (Fig. 3) is valid w.r.t. $\mathcal{A}$, and that the subtrees being inserted are *locally valid* w.r.t. $\mathcal{A}$. It is interesting to remark that:

• When the open tag <d> (at position 0.1) is reached, the deletion operation is taken into account and the subtree rooted at this position is skipped. To verify whether this deletion can be accepted, we should consider the transition rules in $\mathcal{A}$ associated to the parent of the update position. This test is performed when the close tag (position 0) is found. Notice that to perform this test we need to know the state assigned to position 0.0, but we do not need to go below this position (those nodes, when they exist, are skipped).

• When the second open tag <a> (position 1) is reached, the insertion operation is taken into account and the new, locally valid subtree, t_1 (rooted at this position) is inserted. This implies that if all the updates are accepted, right-hand side siblings of position 1 are shifted to the right. We proceed by reading nodes at (original) positions 1 and 2. Notice that we can skip all nodes below position 2, since there is no update position below this point.

• The replace operation at position 3 combines the effects of a deletion and an insertion.

• The close tag </root> activates a validity test that takes into account the root's children. This test follows the definition of the run of the tree automaton. □

The implementation of this approach is done by Algorithm 1. This algorithm takes a tree automaton representing the schema, an XML document and a sequence of updates to be performed on the document. The algorithm checks whether the updates should be accepted or not. It proceeds by treating the XML tree in document order. During the execution, the path from the root to the *current* position p defines a borderline between nodes already treated and those not already considered.

Our algorithm keeps two structures in order to perform the validation. The first one stores the states *allowed* at the current position by the tree automaton. The second one contains, for each position p' on the borderline path, the states *really assigned* to the left-hand side children of p'. In the following, we formally define these structures.

Definition 2 ***- Permissible states for children of a position*** *p: Let PSC(p) be inductively defined on positions p as follows:*

$$\begin{aligned} PSC(\epsilon) = \{q \mid \exists\, a, S, E \rightarrow q_a \in \Delta \\ \textit{such that } t(\epsilon) = a,\ q \textit{ is a state} \\ \textit{appearing in } E \textit{ and } q_a \in \mathcal{Q}_f\} \\ PSC(pi) = \{q \mid \exists\, a, S, E \rightarrow q_a \in \Delta \\ \textit{such that } t(pi) = a,\ q \textit{ is a state} \\ \textit{appearing in } E \textit{ and } q_a \in PSC(p)\} \end{aligned}$$ □

Roughly speaking, for each position pi (labeled a, child of node p), the set $PSC(pi)$ contains the states that *can be* associated to pi's children. To this end, we find those states appearing in the regular expression E, for each rule associated to the label a. Notice however that we consider only transition rules that can be applied at the current node, according to the label of pi's father (*i.e.*, only those rules having a head that belongs to the set of states $PSC(p)$). For instance, suppose two rules $a, S_1, E_1 \rightarrow q_{a1}$ and $a, S_2, E_2 \rightarrow q_{a2}$. Consider now that an element $struct_1$ has a child that should conform to rule $a, S_1, E_1 \rightarrow q_{a1}$ while an element $struct_2$ has a child that should conform to rule $a, S_2, E_2 \rightarrow q_{a2}$. When computing the state associated to an element labeled a that is a child of $struct_1$, state q_{a2} is not considered. This is possible because $PSC(p)$ for the element being treated contains just q_{a1}.

The following definition shows how each position p is associated to a list composed by the set of states assigned by the tree automaton to p's children. This list is built taking into account the updates to be performed on the XML document.

Definition 3 ***- State attribution for the children of a position*** *p: Given a position p, the list SAC(p) is composed by the sets of states associated (by the schema verification process) to the children of position p, i.e., $SAC(p) = [\mathcal{Q}_{att}, \mathcal{Q}_1, \ldots, \mathcal{Q}_n]$, where each set $\mathcal{Q}_i$, for $1 \leq i \leq n$ is calculated as described in Fig. 4. Moreover, the set $\mathcal{Q}_{att}$ contains the states associated to p's children that are attributes. The set Φ contains the states associated to the root of the subtree being inserted at position p, i.e., the result of a successful local validation.* □

$$\mathcal{Q}_i = \begin{cases} \{\ \} & \textbf{If } pi \textbf{ is a delete position} \\ \Phi \cap PSC(p) & \textbf{If } pi \textbf{ is an insert or a replace position} \\ \{q \mid q \in PSC(p), t(pi) = a, \text{there is a rule } a, S, E \rightarrow q\} & \\ & \textbf{If } pi \textbf{ has no descendant update positions} \\ \{q \mid \text{there is a rule } a, (S_{compulsory}, S_{optional}), E \rightarrow q, \text{ such that} & \\ \quad t(pi) = a, \text{ for } \mathcal{Q}_{att} \text{ of } SAC(pi) \text{ we have } S_{compulsory} \subseteq \mathcal{Q}_{att} & \\ \quad \text{and } \mathcal{Q}_{att} \setminus S_{compulsory} \subseteq S_{optional} \text{ and } L(E) \cap L(SAC(pi)) \neq \emptyset\ \} & \\ & \textbf{If } pi \textbf{ is an ascendant of an update position} \end{cases}$$

Figure 4: Calculation of the sets $\mathcal{Q}_i$.

The construction of each set $\mathcal{Q}_i$ in the list $SAC(p)$ depends on the situation of p wrt the update positions. When pi is an update position the set $\mathcal{Q}_i$ takes into account the type of the update. If pi is an ancestor of an update position then it represents a position where a validity test may be necessary. In this case, $\mathcal{Q}_i$ is the set of states associated to pi when the validation at this position succeeds. If there is no update over a descendant of pi, then $\mathcal{Q}_i$ contains all possible states for pi. Now we present Algorithm 1 that is responsible for the construction of both $PSC(p)$ and $SAC(p)$, for each position p.

Algorithm 1 - Incremental Validation of Multiple Updates
Input:
(i) *doc*: An XML document
(ii) $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$: A tree automaton
(iii) $UpdateTable$: A relation that contains updates to be performed on *doc*.
Each tuple in $UpdateTable$ has the form $\langle pos, op, T_{pos}, \Phi\rangle$ where pos is an update position (considering the tree representation of *doc*), op is an update operation, T_{pos} is the subtree to be inserted at pos (when *op* is an insertion or a replace operation) and Φ is the set of states associated to the root of T_{pos} by the execution of $\mathcal{A}$ over T_{pos} (*i.e.*, the result of the local validation). All inserted subtrees are considered to be locally valid.
Output: If the XML document remains valid after all operations in $UpdateTable$ the algorithm returns the Boolean value $true$, otherwise $false$.

(1) **for** *each event v in the document*
(2) *skip*:= **false**;
(3) **switch** v **do**
(4) **case** *start of element a at position p*:
(5) **if** $a \neq$"<root>"{
(6) **if** $\exists\ u = (p, \mathsf{delete}, T_p, \Phi) \in UpdateTable$ **then** *skip*:= **true**;
(7) **if** $\exists\ u = (p, \mathsf{replace}, T_p, \Phi) \in UpdateTable$ **then** {
(8) Compute $\mathcal{Q}_p$ (Definition 3);
(9) **if** $(\mathcal{Q}_p = \emptyset)$ **then** report "invalid" and halt;
(10) $SAC(father(p)) = SAC(father(p))@\mathcal{Q}_p$;
//Append $\mathcal{Q}_p$ to $SAC(father(p))$
(11) *skip*:= **true**;
(12) }
(13) **for each** $u = (p, \mathsf{insert}, T_p, \Phi) \in UpdateTable$ **do** {
(14) Compute $\mathcal{Q}_p$ (Definition 3);
(15) **if** $(\mathcal{Q}_p = \emptyset)$ **then** report "invalid" and halt;
(16) $SAC(father(p)) = SAC(father(p))@\mathcal{Q}_p$;
(17) }
(18) **if** $\nexists u' = (p', op', T', \Phi') \in UpdateTable$ *such that* $p \prec p'$ {
//If there is no update over a descendant of p
(19) Compute $\mathcal{Q}_p$ (Definition 3);
(20) $SAC(father(p)) = SAC(father(p))@\mathcal{Q}_p$;
(21) *skip*:= **true**;
(22) }
(23) }
(24) **if** $a =$"<root>" **or** $\neg skip$ **then** {
//If p is an ascendant of an update position
(25) Compute $PSC(p)$ (Definition 2);
(26) $SAC(p) = SAC(p)@\mathcal{Q}_{att}$;
//Starting the construction of the list $SAC(p)$
(27) }
(28) **if** $skip$ **then** $skipSubTree(doc, a, p)$;
(29) **case** *end of element a at position p*:
(30) **foreach** $u = (p.i, \mathsf{insrt}, T_{p.i}, \Phi) \in UpdateTable$ **where** $p.i$ is a frontier position **do** {
(31) Compute $\mathcal{Q}_{p.i}$ (Definition 3);
(32) **if** $(\mathcal{Q}_{p.i} = \emptyset)$ **then** report "invalid" and halt;
(33) $SAC(p) = SAC(p)@\mathcal{Q}_{p.i}$;
(34) }
(35) Compute $\mathcal{Q}_p$ (Definition 3);
(36) **if** $(\mathcal{Q}_p = \emptyset)$ **then** report "invalid" and halt;
(37) **if** $a \neq$"</root>" **then** $SAC(father(p)) = SAC(father(p))@\mathcal{Q}_p$;
(38) report "valid" □

Algorithm 1 processes the XML document as it is done by SAX [3]. While reading the XML document, the algorithm uses the information in *UpdateTable* to decide which nodes should be treated. When arriving to an open tag representing a position p concerned by an update, different actions are performed according to the update operation:

delete: The subtree rooted at p is skipped. This subtree will not appear in the result and thus should not be considered in the validation process (line 6).

replace: The subtree rooted at p is changed to a new one (indicated by T_p in the *UpdateTable*). The set of states $\mathcal{Q}_p$ indicates whether the locally valid subtree T_p is allowed at this position. The set $\mathcal{Q}_p$ is appended to the list $SAC(father(p))$ to form the list that should contain the states associated to each sibling of p. The (original) subtree rooted at p is skipped (lines 7-12).

insert: The validation process is similar to the previous case for each insertion at p (lines 13-17), but the (original) subtree rooted at p is *not* skipped since it will appear in the updated document on the right of the inserted subtrees.

When we are in a position p (labeled a) where there is no update over a descendant (lines 18-22) we can skip the subtree rooted at p. The list $SAC(father(p))$ is appended with the set $\{q \mid$ there is a rule $a, S, E \rightarrow q$ in Δ such that $q \in PSC(father(p))$ and $T(p) = a\}$. In other words, in $SAC(father(p))$, the set $\mathcal{Q}_p$ contains the permissible states for the child at position p. We use *skipSubTree* (line 28) to "skip nodes" until reaching a position important for the incremental validation process. Notice that when such a position is reached, *skipSubTree* changes the value of the variable *skip* accordingly.

When reaching an open tag representing a position p that is an ascendant of an update position, the structures $PSC(p)$ and $SAC(p)$ should be initialized (lines 24-27) . The set $PSC(p)$ contains the states that can be associated to the children of the element at position p (labeled a). To this end, we find the states appearing in the regular expression E of rules associated to label a. Only rules that can apply to the current element are considered, *i.e.*, only those having a head that belongs to $PSC(father(p))$ (see Definition 2).

When reaching a close tag representing a position p we verify firstly if there is an insert operation on the frontier position (*i.e.*, on a position $pi \notin dom(T)$ such that $p \in dom(T)$). In this case, the insertion is performed (lines 30-34) .

Next (lines 35-37), we should test whether the p's children respect the schema. In fact, reaching a (not skipped) close tag (representing position p), means that updates were performed over p's descendants.

Schema constraints for the current node p (labeled a) are verified by taking into account the list $SAC(p)$ (*i.e.*, $[\mathcal{Q}_{att}, \mathcal{Q}_1, \ldots, \mathcal{Q}_n]$) which, at this point, is completely built. Recall that the set $\mathcal{Q}_{att}$ contains the states associated to the attributes of p while the sets $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ contain the states associated to each element child of p (in the document order). In fact, at this point of the algorithm, our goal is to find the set $\mathcal{Q}_p$ to be appended to $SAC(father(p))$. This computation corresponds to the last case of Definition 3.

More precisely, we consider the language $L(SAC(p))$ which is defined by the regular expression $(q_1^0 \mid q_1^1 \mid \ldots \mid q_1^{k_1}) \ldots (q_n^0 \mid q_n^1 \mid \ldots \mid q_n^{k_m})$ where each $k_i = |\mathcal{Q}_i|$ and each $q_i^j \in \mathcal{Q}_i$ (with $1 \leq i \leq n$). The resulting set of states $\mathcal{Q}_p$ is composed by all states q for which we can find transition rules in Δ of the form $a, (S_{compulsory}, S_{optional}), E \rightarrow q$, that respect all the following properties: (1) q is a state in $PSC(father(p))$; (2) $S_{compulsory} \subseteq \mathcal{Q}_{att}$; (3) $\mathcal{Q}_{att} \setminus S_{compulsory} \subseteq S_{optional}$ and (4) $L(E) \cap L(SAC(p)) \neq \emptyset$.

Notice that Algorithm 1 considers all the updates over the children of a node p before performing the validity test on p.

Example 3 Let $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ be a tree automaton where $Q_f = \{q_r\}$. The set Δ contains all the transition rules below together with rules $data, (\emptyset, \emptyset), \epsilon \rightarrow q_{data}$, and $\alpha, (\emptyset, \emptyset), q_{data} \rightarrow q_\alpha$ (for $\alpha \in \{c, d, m, n\}$).

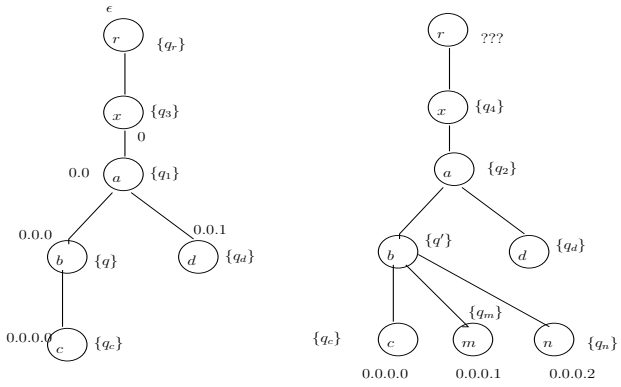

Figure 5: XML tree and its running tree: (a) before updates (b) after insertions.

$(1) r, (\emptyset, \emptyset), q_3 \mid q_4.q_c \rightarrow q_r$ $(2) x, (\emptyset, \emptyset), q_2 \rightarrow q_4$
$(3) x, (\emptyset, \emptyset), q_1 \rightarrow q_3$ $(4) a, (\emptyset, \emptyset), q.q_d^* \rightarrow q_1$
$(5) a, (\emptyset, \emptyset), q'.q_d^* \rightarrow q_2$ $(6) b, (\emptyset, \emptyset), q_c \rightarrow q$
$(7) b, (\emptyset, \emptyset), q_c.q_m.q_n \rightarrow q'$

Fig. 5(a) shows a valid XML tree wrt the schema constraints expressed by $\mathcal{A}$. Fig. 5(b) shows the tree we shall obtain after two insertions on the frontier position 0.0.0.1. In both cases, the result of the running is showed by the set of states associated to each position. Notice that the two insertions on 0.0.0.1 generate changes not only on the set of states associated to its father (0.0.0) but also on the sets of states associated its *ancestors*. Note that even the set of states associated to the root changes and the tree is not valid anymore! This example illustrates why, when dealing with non LTL , the validator tests the ancestors of the update position (and not just its parent). We propose an algorithm that performs tests until reaching the root, but optimizations are possible: one should perform tests until reaching a node whose associated set of states does not need to be changed.

To illustrate the execution of Algorithm 1, consider, in the table below, the situation of structures $PSC()$ and $SAC()$ at some specific instants, during the incremental validation process that take into account the updates illustrated by Fig. 5(b).

Position	Situation of $PSC()$ when we reach	Situation of $SAC()$ when we reach	Situation of $SAC()$ when we reach <d>
0.0.0	$\{q_c, q_m, q_n\}$	$[\{q_c\}, \{q_m\}, \{q_n\}]$	
0.0	$\{q, q', q_d\}$	[ ]	$[\{q'\}]$
0	$\{q_1, q_2\}$	[ ]	[ ]
ϵ	$\{q_3, q_4, q_c\}$	[ ]	[ ]

According to Definition 2, when is reached, states q_c, q_m, q_n are put in $PSC(0.0.0)$ since they appear in rules (6) and (7) (which have label b and whose right-hand side are in $PSC(0.0)$).

When we reach the list $SAC(0.0.0)$ is complete since it now contains the set of states assigned to all the children of position 0.0.0. Notice that $SAC(0.0)$ is an empty list since we still do not know the states that are effectively associated to its children. As is found, $SAC(0.0.0)$ is popped and the state associated to the leftmost child of position 0.0 can be computed.

Indeed, when we reach <d>, the list $SAC(0.0)$ is not empty anymore - it contains the set of the states we effectively associate to position 0.0.0 (chosen among those in $PSC(0.0)$ and taking into account the transition rules that apply). □

Our algorithm is general, however, as shown in the following proposition, for the case of single-type tree languages, $SAC(p)$ (for each position p) is a list of singleton sets. In this case, the operation $L(E) \cap L(SAC(p)) \neq \emptyset$ is reduced to the verification of a word to belong to a regular language.

Proposition 1 *Given a schema defining languages in STTL, for each position p in Algorithm 1, we have that the set of states assigned to element children that are not delete positions is always a singleton set, that is:*
If $SAC(p) = [\mathcal{Q}_{att}, \mathcal{Q}_1, \ldots, \mathcal{Q}_n]$ then $\mid \mathcal{Q}_i \mid = 1$, for all $1 \leq i \leq n$. □

PROOF: By cases, on the definition of each $\mathcal{Q}_i$:
If pi is an insert or a replace position. In this case, we have $\mathcal{Q}_i = \Phi \cap PSC(p)$. If we suppose that $\{q_1, q_2\} \subseteq \mathcal{Q}_i$, then (i) Both states q_1 and q_2 are in competition (since they belong to Φ) and (ii) they belong to the same regular expression in a transition rule (since they belong to $PSC(p)$). The conjunction of the two conditions above contradicts the definition of a single-typed language, to which the schema belongs.
If pi has no descendant update positions. In this case the set $\mathcal{Q}_i = \{q \mid q \in PSC(p), T(pi) = a$, there is a rule $a, S, E \rightarrow q\}$.
If we suppose that $\{q_1, q_2\} \subseteq \mathcal{Q}_i$, then, by the definition of $\mathcal{Q}_i$, the states q_1 and q_2 compete to each other (since there is only one label associated to the position pi of the tree), leading to a contradiction.
If pi is an ascendant of an update position. In this case the set $\mathcal{Q}_i$ is also defined as being composed by states which are in the right-hand side of a rule for a given label a. The existence of more than one state in this set contradicts the definition of a single-typed language, to which the schema belongs. □

Proposition 1 allows us to define a simplification on Algorithm 1, to use single states instead of sets of states, in such a way that, for these classes of tree languages, we can represent $SAC(p) = [\mathcal{Q}_{att}, \mathcal{Q}_1, \ldots, \mathcal{Q}_n]$ as a set of states $\mathcal{Q}_{att}$ and a word of states.

Notice that while performing validation tests, a new updated XML tree is being built (as a modified copy of the original one). If the incremental validation succeeds, a *commit* is performed and this updated version is established as our current version. Otherwise, the *commit* is not performed and the original XML document stays as our current version. The next section considers the implementation of our method, comparing it to a validation from scratch.

3.2 Complexity and experimental results

As said in Section 1, the complexity of our method is presented taking into account that the cost of "skipping" nodes is not relevant when compared to the cost of validation actions. In this context, notice that in Algorithm 1, validation steps are performed only for those nodes $p \in dom(T)$ which are ascendants of update positions.

Let E be the regular expression defining the structure of p's children. When a DTD or XSD schema is used, each validation step corresponds to checking whether a word w is in $L(E)$. The word w is the concatenation of the states associated to p's children. Thus, for DTD or XSD schema each validation step is $O(|w|)$.

When a specialized DTD schema is used, each validation step corresponds to checking if there is a word $w = q_i, \ldots, q_n$ in $L(E)$ such that $q_i \in \mathcal{Q}_i, \ldots, q_n \in \mathcal{Q}_n$ (see the definition of a run, in Section 3). In other words, we should test if $L(E_{aux}) \cap L(E) \neq \emptyset$, where E_{aux} is the regular expression representing all the words we can build from the concatenation of the states associated to p's children, *i.e.*, $E_{aux} = (q_1^0 \mid q_1^1 \mid \ldots \mid q_1^{k_1}) \ldots (q_n^0 \mid q_n^1 \mid \ldots \mid q_n^{k_n})$. This test is done by the intersection of the two automata $M_{E_{aux}}$ and M_E. The solution of this problem runs in time $O(n^2)$ where n is the size of the automata [35, 36].

Thus, if we assume that n is the maximum number of children of a node (fan out) in an XML tree T, then a *validation step* runs in time $O(n)$ (for DTD or XML schema) or in time $O(n^2)$ (for specialized DTD).

Let m be the number of updates to be performed on a tree t (of depth h). Given an update position p, in the worst-case, a validation step should be performed for each node on the path between p and the root. For a worst-case analysis, we can also consider that all m updates are performed on the leaves. We also suppose that all the paths from nodes p to the root are disjoint. In this case, the complexity of our algorithm can be stated as $O(m.n.h)$ (DTD or XML Schema) or $O(m.n^2.h)$ (when specialized DTD is considered).

Notice that, in a general XML setting, updates can happen at any level of the tree (so that the limit imposed by h is seldom reached). Moreover, when dealing with multiple updates, part of the paths between the update nodes and the root are common to two or more of these updates. In this case, only one validation action is performed for the shared nodes.

The worst-case of our algorithm is reached for a very unusual configuration of the tree being processed (the configuration maximizing the product $n.h$). In this configuration, one half of the nodes are leafs, children of the root, while the others form a list (pending from the root).

Other singular configurations are:

- A flat tree, where all the nodes (except the root) are leaves, and children of the root node. In this case, the depth of the tree is one and the complexity expressions are reduced to $O(m.n)$ (DTD or XML Schema) or $O(m.n^2)$ (specialized DTD).

- A list, where there is exactly one leaf node and the maximum fan-out of the tree nodes is one. In this case, both complexity expressions are reduced to $O(m.h)$ (DTD,

XML Schema or specialized DTD).

• If we consider an n-ranked, balanced tree t, its depth is given by $h = \log_n |t|$, where $|t|$ is the size of the tree. In this case, the complexity expressions are reduced to $O(m.n.\log_n |t|)$ (DTD or XML Schema) or $O(m.n^2.\log_n |t|)$ (specialized DTD).

Notice that when dealing with a DTD, we just need to verify whether the state associated to the father of an update position changes (see [17]). This is easily done by using transition rules. Thus, for multiple updates with DTD, we just need to perform verifications until reaching the father of the update position which is the nearest to the root. Example 3 illustrates that this optimization cannot be applied to non-LTL schemas.

Experimental results (Table 1) show that our incremental algorithm behaves very efficiently in practice. In order to compare these approaches we use ten XML documents of different sizes (from $3,000$ to $61,000,000$ nodes). These documents are in the STTL class of languages and describe different car suppliers. They are valid wrt an XSD whose principal schema constraints are expressed by the following transition rules:

$$\begin{array}{ll} supplier, (\emptyset, \emptyset), q^+_{shop} q^*_{garage} & \rightarrow q_{supplier} \\ shop, (\emptyset, \emptyset), q^*_{newVeh} & \rightarrow q_{shop} \\ garage, (\emptyset, \emptyset), q^+_{oldVeh} & \rightarrow q_{garage} \\ vehicle, (\{id\}, \{type\}), q_{name} q_{cv} q_{cat}? & \rightarrow q_{newVeh} \\ vehicle, (\{id\}, \emptyset), q_{name} q_{cv} q_{km}? & \rightarrow q_{oldVeh} \end{array}$$

Given an XML document, we consider a sequence of 50 updates over it. Our implementation is just a prototype in Java. Experiments were performed on a 1.5 GHz Pentium M system with 512MB of memory and a 40GB, 5400rpm hard drive.

Table 1 and Fig. 6 show the superiority of Xerces [2] when only validation from scratch is considered. This is a natural result when comparing a prototype with a commercial product. However, when we compare our incremental validation method (Algorithm 1) to a validation from scratch approach Table 1 and Fig. 7 show that our incremental validation method is very efficient for large documents[4]. Indeed, it takes almost a third of the time needed for Xerces to validate 50 updates on a document having $61,000,000$ nodes. Similarly, it takes about half of the time needed for Xerces to validate documents having $10,000,000$ nodes.

A sufficient condition for *commutative* update lists is given in [34]. In this paper, our *UpdateTable* respects this condition and thus, single updates can be performed in any order, without changing the result of the global update on the XML tree. However, as our validation process is done by using SAX, a left-right computation of updates is more efficient than one that considers tree nodes at random. If we do not care about loading all the XML file, a bottom up computation might be proposed, since all update positions activate validation verification on their ancestor nodes. Indeed, schema validation performance depends on the depth of update nodes as discussed before.

[4]For small documents, the number of updates represents changes over a high percentage of the document. In this case, incremental validation cost is close to our validation from scratch cost and thus it is worse than Xerces, a commercial product.

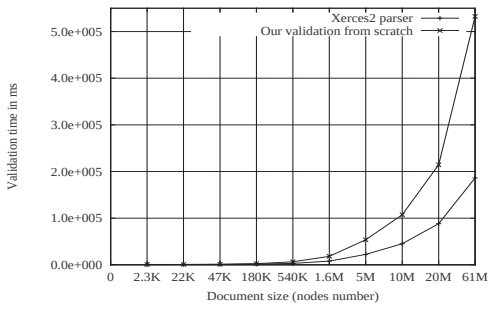

Figure 6: Comparing validation from scratch performed by Xerces and our prototype.

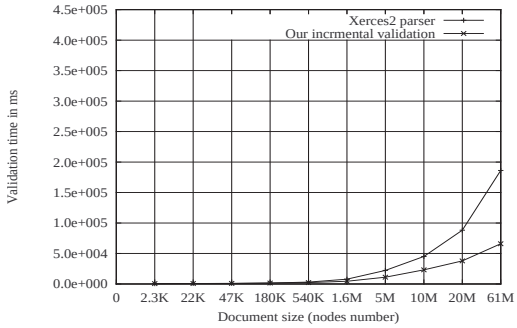

Figure 7: Xerces (validation from scratch) $\times$ Our incremental validation method.

3.3 Related work

The importance of algorithms for the efficient validation of XML documents grows together with the use of schema languages. Algorithms for the incremental validation are very useful when dealing with very large XML documents, since they can substantially reduce the amount of time of the verification process. Several XML document editors such as XML Mind [5] and XMLSpy [1] include features for the validation of documents (wrt one or more schema languages). However, the documentation of most of these tools include little or no information on their validation algorithms.

In [42] validation algorithms are presented but incremental validation is not considered. One of the most referenced work dealing with incremental validation of XML documents wrt schema constraint is [11, 45]. In those papers incremental validation methods wrt DTD, XSD and specialized DTDs are presented. Their approach is based on word automata, built from schema descriptions. The words considered are *lines* (*i.e.*, paths) computed from a *binary version* of the document tree. Not only the binary version of the document tree is stored as auxiliary data, but also

Document characteristics		**Existing product**	**Our method**	
Attributes	Attributes + Elements	Xerces (ms)	From scratch (ms)	Incremental (ms)
700	3K	451	683	658 (50 updates)
6.5K	25K	727	814	773 (50 updates)
13K	50K	929	1 156	991 (50 updates)
52K	200K	1 509	2 435	1 520 (50 updates)
170K	600K	3 095	6 357	2 258 (50 updates)
500K	1.7M	7 855	18 100	4 526 (50 updates)
1.5M	5.2M	22 208	53 625	11 160 (50 updates)
2.4M	10.5M	45 065	107 195	23 200 (50 updates)
6M	21M	88 270	214 280	37 883 (50 updates)
18M	61.5M	186 144	532 505	66 009 (50 updates)

Table 1: Experimental results.

trees of transition relations, one for each *line*, representing the potentially legal evolutions of the line. This is important to notice because this is the reason why the method can hardly generalize to updates on whole subtrees: such an update would require to compute again ALL auxiliary data, *i.e.*, (i) the binary tree, (ii) the set of *lines* and (iii) trees of transition relations. Indeed, the method is applied only for updates on nodes: insertion or deletion of *leaves*, and renaming of one node. In [11, 45], they propose two main incremental algorithms to validate a number m of updates on the leaves of a given tree T. The first algorithm, for DTD and XSD, has time complexity $O(m.log|T|)$, and uses an auxiliary structure of size $O(|T|)$. The second algorithm, for specialized DTDs, has time complexity $O(m.log^2|T|)$ and also uses an auxiliary structure of size $O(|T|)$.

Our approach deals with multiple updates and incremental validation over an unranked tree. As we have already said, our algorithm visits all the nodes of an XML tree but only some nodes trigger the validation actions that represent the relevant cost of the approach. Thus, we can say that our time complexity is similar to the one found in [11, 45]. For updates on leaves and for node renaming, their method may be more efficient than ours because the use of trees of transition relations is quite efficient for incremental word verification. However these updates are very special ones: for more general updates, as considered in our paper (involving whole subtrees), our method performs a minimum of tests without maintaining huge auxiliary structures.

Due to our use of unranked trees instead of the binary trees of [11, 45], each validation step on our method can be more expensive, but our XML tree is usually much shorter. Moreover, our work differs from that in [45] in four main aspects:

1. Our update operations can be applied at any node of the tree, and not just on the leaves. This feature permits us to change large areas of the XML tree with one single operation.

2. In [45], testing whether a word belongs to a language is done in an incremental way by storing an auxiliary structure. This optimization might be easily integrated in our algorithm for verifying child state word, but it seems to be interesting only in case of very large fan-out.

3. When dealing with DTD and XSD we do not use any auxiliary structures to store the result of a previous validation process.

4. Integrity constraint verification for keys and foreign keys is naturally integrated to our algorithm (Section 4).

4 Integrity constraint verification

In this section we present our constraint language, called CTK (for Context-Target-Key), and we show that constraints in CTK can be compiled to an attribute grammar. CTK is used to specify absolute and relative keys, together with foreign keys. As in [21], CTK uses path expressions built from a fragment of XPath. This XPath fragment includes (a) the empty path ϵ, (b) an element or attribute name, (c) a wildcard matching any single node name (_), (d) an arbitrary downward path ($//$) and (e) the concatenation of paths (P/Q where P and Q are paths, as defined by these rules). Indeed, this fragment is the same used by XML Schema to specify integrity constraints ([15]). Notice that a given path defines a language whose symbols are XML labels. Path expressions are, in fact, regular expressions over XML labels.

The following example illustrates how keys and foreign keys are specified. Next we define their syntax and semantics.

Example 4 The XML document represented by the tree T in Fig. 1 describes a collection of cooking recipes. Each collection maintains a categorized list of recipes, and a list of the top recipes. We want to validate this document wrt the following keys and foreign key, defined in CTK:

- K_1 : (/, (*./collection*, {*./category*}))
  It indicates that, for the whole document, every collection must be uniquely identified by its category.
- K_2: (*/collection*, (*.//recipe*, {*./name*, *./author*}))
  It means that, in the context of a collection, a recipe is uniquely identified by its name and author.
- K_3: (*//recipe*, (*./ingredient*, {*./name*}))
  Similarly to K_2, it indicates that, in the context of a recipe, each ingredient is uniquely identified by its name.

- FK_4: ($/$*collection*, ($.$/*top_recipes*/*top_recipe*, {$.$/*recipe_name*, $.$/*author_name*}) $\subseteq$ ($.$//*recipe*, {$.$/*name*, $.$/*author*})) where ($/$*collection*, ($.$//*recipe*, {$.$/*name*, $.$/*author*})) is the key K_2.
The foreign key constraint indicates that, in a collection, the name and the author of a top recipe must already exist as the name and the author of a recipe (in the same collection, but disregarding the order). □

Definition 4 - CTK key and foreign key syntax [21]: A key is represented by an expression $(P, (P', \{P^1, \ldots, P^m\}))$ in which the path P is called the *context path*; P' is the *target path* and $P^1, \ldots, P^m$ are the *key paths*.

A foreign key is represented by $(P, (P'_0, \{P^1_0, \ldots, P^m_0\}) \subseteq (P', \{P^1, \ldots, P^m\}))$ where $(P, (P', \{P^1, \ldots, P^m\}))$ is a key K and $P^1_0, \ldots, P^m_0$ are called *foreign key paths*. □

In this paper, key and foreign key specifications respect the following constraints: (a) Context and target paths should reach element nodes; (b) Key paths always exist and are unique and (c) Key (or foreign key) paths are required to end at a node associated to a value, *i.e.*, attribute nodes or elements having just one child of type *data*. Notice that our key specification corresponds to a special case of strong keys defined in [21] and thus it imposes the *uniqueness* of a key and *equality* of key values (*i.e.*, keys values cannot be null). This concept is similar to the concept of key in relational databases.

In the following definition we use the notion of tuple in a *named perspective* as described in [6]. Thus, tuples are functions that associate a non-null value to each component (name). The order of values appearing in the tuple is not important since each component value is associated to its name (in our case, the name of the tuple component is an XML label, *i.e.*, the name of the element or attribute whose value we want to consider). Thus, wrt the textual representation of an XML element, the definition below states that the order of elements (or attributes) is unimportant in defining equality.

Definition 5 - Semantics of CTK keys and foreign keys: An XML tree T satisfies a key $(P, (P', \{P^1, \ldots, P^m\}))$ if for each context position p reached by following path P from the root, the following two conditions hold:

(i) For each target position p' reachable from p via P' there exists a unique position p^h from p', for each $P^h (1 \leq h \leq m)$, and

(ii) For any target positions p' and p'', reachable from p via P', whenever $\tau' = \tau''$ (where tuples[5] τ' and τ'' are built following $P^1 \ldots P^m$ from p' and p'', respectively) then p' and p'' must be the same position.

Similarly, an XML tree T satisfies a foreign key $(P, (P'_0, \{P1_0, \ldots, P^m_0\}) \subseteq (P', \{P^1, \ldots, P^m\}))$ if:

(i) It satisfies its associated key $K = (P, (P', \{P1, \ldots, P^m\}))$, and

(ii) For each target position p'_0 reachable from the context position p via P' there exists a unique position p^h from p', for each $P^h (1 \leq h \leq m)$, and

(iii) Each tuple $\tau_0 = [\tau_0(P^1_0) : v_1, \ldots, \tau_0(P^m_0) : v_m]$, that was built following the paths $P/P'_0/P^1_0, \ldots, P/P'_0/P^m_0$ is equivalent in value to a tuple $\tau = [\tau(P^1) : v_1, \ldots, \tau(P^m) : v_m]$, built following paths $P/P'/P^1, \ldots, P/P'/P^m$. In other words, for each $1 \leq h \leq m$, the τ_0-component name P^h_0 corresponds to the τ-component name P^h and their values are equal. □

Example 5 In Example 4, if we assume an inversion of the two rightmost children of position 0.3.0 of Fig. 1 (*author_name* on position 0.3.0.1 and *recipe_name* on position 0.3.0.2) then we obtain tuples [*recipe_name*: *Mushroom Soup*, *author_name*: *M. Smith*] and [*name*: *Mushroom Soup*, *author*: *M. Smith*] which are equivalent in value. This is because foreign key specification relates *recipe_name* to *name* and *author_name* to *author*. Thus, according to Definition 5, the foreign key is satisfied. This remark is also valid when comparing key values.

□

In order to perform the validation of key constraints, we represent the paths in key definitions by finite state automata: for a context path P, we have the automaton $M = \langle \Theta, \Sigma, \delta, e, F \rangle$. This automaton will be referred to as the context automaton. It is defined to recognize the language generated by the path (regular expression) P.

Similar automata are defined for target, key and foreign key paths: For a target path P', its corresponding target automaton is defined as $M' = \langle \Theta', \Sigma, \delta', e', F' \rangle$; and for key or foreign key paths $P^1, \ldots, P^m$, their key or foreign key automata are $M'' = \langle \Theta'', \Sigma, \delta'', e'', F'' \rangle$.

We denote by $M.e$ the current state e of the finite state automaton M. $M.e$ is the *configuration* of the automaton, representing a snapshot of it during its run. We illustrate the above definitions with an example.

Example 6 Fig. 8 illustrates finite state automata that correspond to the (context, target and key) paths in K_1, K_2, K_3 and FK_4 of Example 4. Given the XML tree of Fig. 1, those finite state automata are used in the tree traversal to perform constraints validation. □

4.1 Key and foreign key validation: attribute grammar approach

We consider a context-free grammar G = (V_N, V_T, P, B) where V_N is the set of non-terminals, V_T is the set of terminals, P is the list of productions, and B is the start symbol. In order to add "extra" information to a non-terminal symbol we can attach a set of attributes to it. An attribute can represent anything: a string, a number, a type, a memory location or whatever [8]. An *attribute grammar* is G augmented by semantic rules, which are declarative specifications describing how the attached attributes are computed.

[5] A tuple τ has the general format $\tau = [\tau(P^1) : v_1, \ldots, \tau(P^m) : v_m]$. We use $\tau(P^h)$ to denote the name (label) of the component of τ corresponding to the node reached via P^h from a given target position.

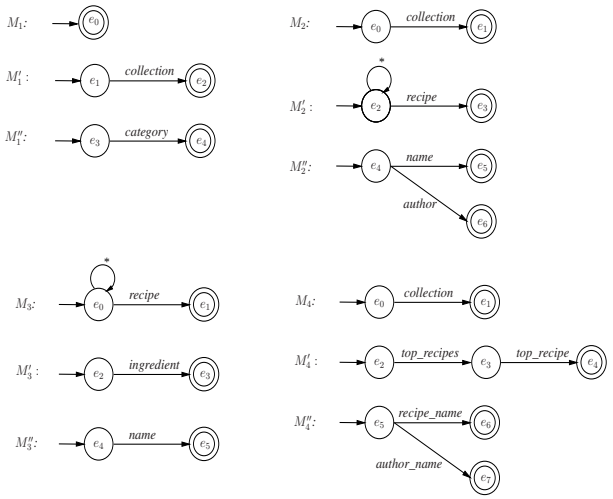

Figure 8: Context, target and key automata corresponding to keys K_1, K_2, K_3 and to the foreign key FK_4.

Production	Attributes

$R \rightarrow \alpha_1 \ldots \alpha_m$	$R.conf$:= { $M.q_0$ }
	/* *Inherited Attributes* */
	for each α_i $(1 \leq i \leq m)$ do
	$\alpha_i.conf$:= { $M.q'$ | δ_M $(q_0, \alpha_i) = q'$ }
	if $(q_0 \in F_M)$ then
	$\alpha_i.conf$:= $\alpha_i.conf \cup$
	{ $M'.q_1'$ | $\delta_{M'}(q_0', \alpha_i) = q_1'$ }
	end for
	/* *Synthesized Attributes* */
	if $(q_0 \in F_M)$ then
	$R.c^K$:=$\langle \forall w, z$: $w \neq z \Rightarrow \alpha_w.t \cap \alpha_z.t = \emptyset \rangle$
	if $(\exists\, FK \subseteq K) \wedge (q_0 \in F_{M^{FK}}) \wedge$
	$(R.t^{FK} \subseteq R.t^K)$ then
	$R.c^{FK} := \langle true \rangle$
	if $(q_0 \notin F_M)$ then
	$R.c := \langle \forall x_w : \alpha_w.c = \langle x_w \rangle \Rightarrow \bigwedge_{w=1}^{m} x_w \rangle$

Table 2: Attribute Grammar for Keys and Foreign Keys (Root block).

Thus, the value of an attribute at a parse tree node is defined by a semantic rule associated to the production used at that node [8].

We recall that a schema can be seen as an extended context free grammar G (regular tree grammar)[37]. Thus, in the context of integrity constraint verification one may augment G, whose production rules are those defining the schema, by semantics rules [44], using attributes which represent information about integrity constraints. In our case, as we assume that integrity constraint validation is independent from schema verification, we consider G as a simpler grammar just describing any XML tree. We write $A \rightarrow \alpha_1 \ldots \alpha_m$ to indicate that the semantic rule applies to the node labeled a and to its children (labeled α_i). The semantic rules provide a mechanism for annotating the nodes of a tree with attributes, which can work either bottom-up for *synthesized attributes* or top-down for *inherited attributes*.

Tables 2, 3 and 4 present the attribute grammar for keys and foreign keys. In these tables the definition of semantic rules for a key K (or a foreign key FK) is given according to the kind of production rules. Indeed, we classify our production rules into three kinds, according to the XML node over which they can be applied:
1: Rules applied at the root node (root block in Table 2).
2: Rules applied at the leaves (or data) nodes (data block in Table 4).
3: General rules, applied at all other nodes (general block in Table 3).

We use the top-down direction, *i.e.*, *inherited attributes*, in order to determine the role of each node wrt keys and foreign keys, defined according to language CTK. We define just one inherited attribute, called *conf*. For each node, *conf* is computed by executing the finite state automata that recognize the paths in the definition of K (or FK).

In this way, Table 2 defines, for each key or foreign key, how the attributes *conf* are computed for a rule in which the left-hand side is a symbol that represents the root node. To this end, we firstly assign to the root node the set of values $\{M.q_0\}$ where $M.q_0$ is the initial configuration of the context automaton M. Then, we compute the value of *conf* for each child of the root (by executing M), without forgetting to verify whether we need to change from the context to the target automaton.

In Table 3 the computation of *conf* is similar to the one performed for the root's children in Table 2. Notice that to compute the value of *conf* for a node α, we start by considering each configuration $\overline{M}.q$ in the set of values associated to the attribute *conf* of its parent A. We should also verify if it is necessary to change from one automaton to another.

Example 7 - Consider the XML document, key K_2 and foreign key FK_4 of Example 6 with their corresponding finite state automata (respectively M_2, M_2', M_2'' and M_4, M_4', M_4''). We have one inherited attribute *conf* for each key or foreign key, as illustrated in Fig. 9. The attributes $conf^{K_2}$ and $conf^{FK_4}$ are computed top-down by the execution of their finite state automata. For instance, at the root node, we assign to $conf^{K_2}$ and $conf^{FK_4}$ their corresponding initial configuration, respectively the sets $\{M_2.e_0\}$ and $\{M_4.e_0\}$. In order to compute $conf^{K_2}$ for node *collection* we execute a first transition in M_2 using the label *collection* as input. The result is the set $\{M_2.e_1\}$. Similarly, the computation of $conf^{FK_4}$ for node *collection* results in $\{M_4.e_1\}$. □

We use the bottom-up direction, *i.e.*, *synthesized attributes*, to carry the values that are part of a key or foreign key up to their context node. At this level, we verify if the integrity constraints are respected. For each key (or foreign key) definition K (according to language CTK), we use three attributes, called c, t and k, for (respectively) *context values*, *target values* and *key values*. At each node, these attributes receive values depending on the role of the node for K (*i.e.*, the value of attribute *conf*), and depending also on values of c, t and k from children nodes.

In this context, Table 4 shows that an attribute k obtains the data value of a leaf whose parent is a key node for some K. Then, Table 3 defines the values of the synthesized attributes concerning a node p (labeled A) in the following way:

Production	Attributes

$A \rightarrow \alpha_1 \ldots \alpha_m$	/* *Inherited Attributes* */
	for each α_i $(1 \leq i \leq m)$ do
	for each $\overline{M}.q \in A.conf$ do
	$\alpha_i.conf := \{\overline{M}.q' \mid \delta_{\overline{M}}(q, \alpha_i) = q'\}$
	if $(\overline{M} = M) \wedge (q \in F_M)$ then $\alpha_i.conf := \alpha_i.conf \cup \{M'.q'_1 \mid \delta_{M'}(q'_0, \alpha_i) = q'_1\}$
	if $(\overline{M} = M') \wedge (q \in F_{M'})$ then $\alpha_i.conf := \alpha_i.conf \cup \{M''.q''_1 \mid \delta_{M''}(q''_0, \alpha_i) = q''_1\}$
	end for
	end for
	/* *Synthesized Attributes* */
	for each configuration $\overline{M}.q$ in $A.conf$ do
	if $(\overline{M} = M'') \wedge (q \notin F_{M''})$ then $A.k := < \alpha_1.k \ \ldots \ \alpha_m.k >$
	if $(\overline{M} = M') \wedge (q \in F_{M'}) \wedge (|< \alpha_1.k \ \ldots \ \alpha_m.k >| = n)$ then $A.t := A.t \cup \{ < \alpha_1.k \ldots \alpha_m.k > \}$
	if $(\overline{M} = M') \wedge (q \notin F_{M'}) \wedge (\forall w, z : w \neq z \Rightarrow \alpha_w.t \cap \alpha_z.t = \emptyset)$ then $A.t := \bigcup_{w=1}^{m} \alpha_w.t$
	if $(\overline{M} = M) \wedge (q \in F_M)$ then $A.c^K := \langle \forall w, z : w \neq z \Rightarrow \alpha_w.t \cap \alpha_z.t = \emptyset \rangle$
	if $(\exists\, FK \subseteq K) \wedge (\overline{M} = M^{FK}) \wedge (q \in F_{M^{FK}}) \wedge (A.t^{FK} \subseteq A.t^K)$ then $A.c^{FK} := \langle true \rangle$
	if $(\overline{M} = M) \wedge (q \notin F_M)$ then $A.c := \langle \forall x_w : \alpha_w.c = \langle x_w \rangle \Rightarrow \bigwedge_{w=1}^{m} x_w \rangle$
	end for

Table 3: Attribute Grammar for Keys and Foreign Keys (General block).

Production	Attributes

$A \rightarrow data$	/* *Synthesized Attributes* */
	for each configuration $\overline{M}.q$ in $A.conf$ do
	if $(\overline{M} = M'') \wedge (q \in F_{M''})$
	then $A.k := < value(data) >$
	end for

Table 4: Attribute Grammar for Keys and Foreign Keys (Data block).

```
<!DOCTYPE keyTree[
<!ELEMENT keyTree (context*)>
<!ATTLIST keyTree nameKey CDATA #REQUIRED>
<!ELEMENT context (target+)>
<!ATTLIST context pos CDATA #REQUIRED>
<!ELEMENT target (key+)>
<!ATTLIST target pos CDATA #REQUIRED
                  refCount CDATA #REQUIRED>
<!ELEMENT key #PCDATA>]
```

Figure 11: DTD specifying the structure *keyTree*

1. If p is in a key path, then its attribute k is the tuple composed by the key values (those associated to the attribute k of each child of p).

2. If p is a target node, then its attribute t is a set containing the tuple composed by the key values carried up from p's children. This tuple is computed from the values of attributes k, as explained in item 1. Notice that the assignment of a value to the attribute t depends on the verification of the key size (*i.e.*, the size of the key tuple must respect the key definition).

3. If p is in a target path and if all the tuple values carried up by p's children are distinct, then the attribute t for node p is assigned with the union of the sets containing these tuples.

4. If p is a context node for a key and the tuple values carried up by p's children are distinct, then the attribute c (wrt the key K) is assigned with a tuple containing the value *true*. Otherwise the tuple contains the value *false*.

5. If p is a context node for a foreign key and the tuple values carried up by p's children are also key values, then the attribute c (wrt the foreign key FK) is assigned with a tuple containing the value *true*. Otherwise the tuple contains *false*.

6. If p is in a context path, then the attribute c is assigned with a tuple containing the conjunction value of c's values obtained from p's children.

Finally, Table 2 shows how to compute synthesized attributes for the root, distinguishing whether it is a context (for a key or a foreign key) or not.

From the above explanation, we see that the values of attributes c are computed from those of attributes t which are, in turn, built from the values of the attributes k.

Example 8 - As in Example 7 we show in Fig. 10 how the synthesized attributes are computed for K_2 and FK_4 for nodes in the key, target and context paths:

Key path: The data values for K_2, obtained from nodes $name$ and $author$, are collected in each k_2. For FK_4, $recipe\_name$ and $author\_name$ are the foreign key nodes and their data values are collected in attributes k_4.

Target path: For K_2, the key values are also concatenated into a tuple and kept in a singleton set when reaching target node $recipe$. At target node top_recipe, the attribute t_4 receives a singleton set containing the tuple obtained by the concatenation of the key values for FK_4. Node $top\_recipes$ is in the target path and its attribute t_4 groups all the target tuples coming from target nodes.

Context path: At the context node $collection$, as all the tuples collected in the various t_2 are all distinct, then the attribute c^{K_2} is set to $true$. Still at node $collection$, the set of tuples coming from t_4 is compared to those coming from the various t_2. As all the tuples in t_4 are contained in the set formed by the various t_2, then the attribute c^{FK_4} is set to $true$. It means that for the concerned $collection$ node, the foreign key FK_4 is valid.

At the root node, the attributes c_2 and c_4 are set to $true$, indicating that the document respects K_2 and FK_4. In order to show neatly the attribute values that are synthesized, those attributes (c, t or k) for K_2 or FK_4 that are not concerned in the position are hidden. □

At the same time that we compute the synthesized attributes for a key, we build its corresponding *keyTree*. The

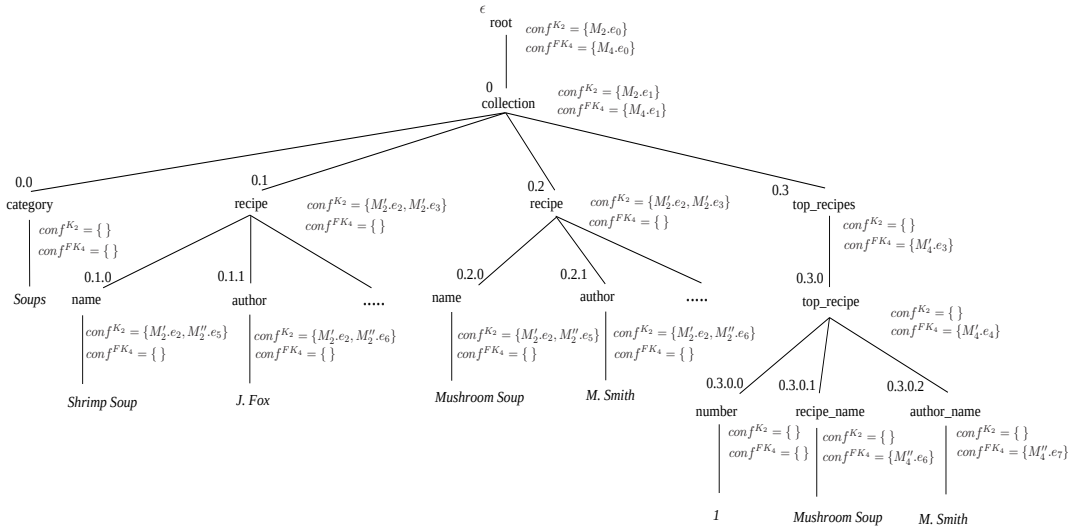

Figure 9: Inherited Attribute *conf* for K_2 and FK_4.

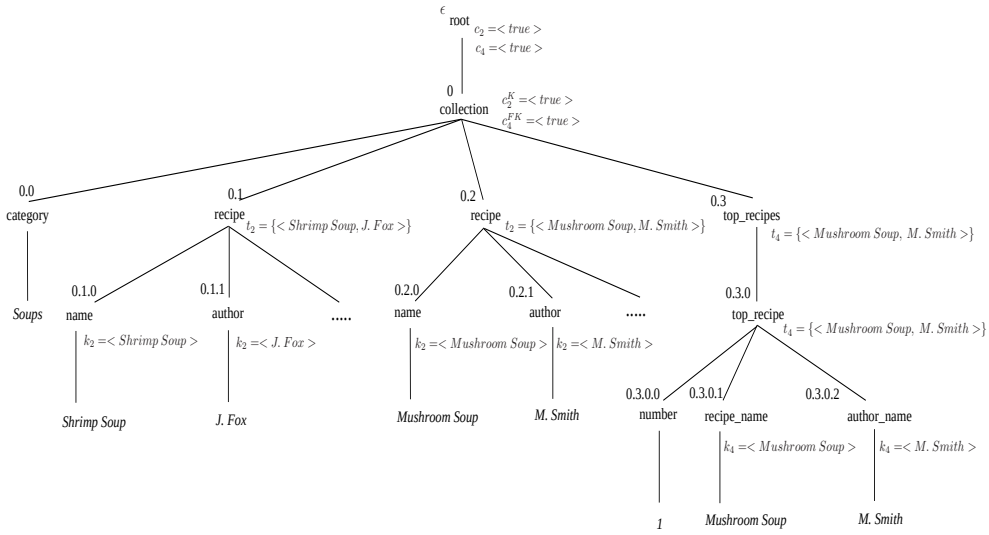

Figure 10: Synthesized Attributes for K_2 and FK_4.

*keyTree*s are structures storing the position of each context and target nodes together with the key values associated to each key node. Fig. 11 describes this structure index using the notation of DTDs. For each key constraint K that should be respected by the XML document, we keep its $keyTree_K$. Also, for each key, a reference counter *refCount* is used to store how many times the key is referenced by a foreign key. The *keyTree*s are kept to facilitate validation of keys and foreign keys in update operations, as the acceptance of an update operation wrt integrity constraints relies on information about key values. Fig. 12 shows $keyTree_{K_2}$ that stores the key values and information for K_2.

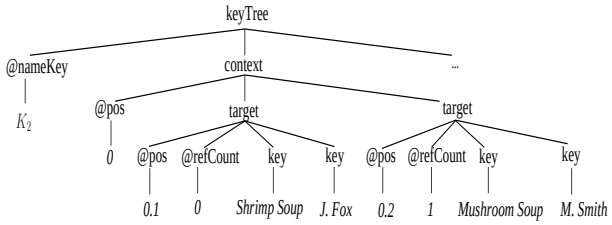

Figure 12: $KeyTree_{K_2}$ built over the XML document of Fig. 1.

4.2 Incremental integrity constraint verification

We consider a collection of keys K_j $(1 \leq j \leq m)$ and foreign keys FK_j $((m+1) \leq j \leq n)$ constraints that should be respected by a subtree T' being inserted, replaced or

deleted. The execution of our key and foreign key constraints validator over a tree T' gives a tuple $\langle\langle l_1, \ldots, l_n\rangle$, $\langle$*keyTree*$_{K_1}[T', \epsilon]$,..., *keyTree*$_{K_m}[T', \epsilon]\rangle\rangle$ where:

- $\langle l_1, \ldots, l_n\rangle$ is a n-tuple of tuples $\langle c, t, k\rangle$. Each tuple $\langle c, t, k\rangle$ represents the synthesized attributes computed for one key (or foreign key) at the root position ϵ of T'.
- $\langle$*keyTree*$_{K_1}[T', \epsilon], \ldots,$ *keyTree*$_{K_m}[T', \epsilon]\rangle$ is a m-tuple containing one *keyTree* for each key.

Notice that the n-tuple that represents the synthesized attributes has two distinct parts: tuples $l_1, \ldots, l_m$ represent keys and tuples $l_{m+1}, \ldots, l_n$ represent foreign keys.

We introduce now the notion of local validity for keys and foreign keys. When performing an insertion we want to ensure that the new subtree T' has no internal validity problems (as, for instance, duplicated values for a key K).

Definition 6 - Local Validity: Let T' be an XML tree. Let K_j $(1 \leq j \leq m)$ be a collection of keys and FK_j $((m+1) \leq j \leq n)$ be a collection of foreign keys, both defined according to language CTK. The tree T' is locally valid if the result of the validation gives a tuple $\langle\langle l_1, \ldots, l_n\rangle, \langle$*keyTree*$_{K_1}[T', \epsilon], \ldots,$ *keyTree*$_{K_m}[T', \epsilon]\rangle\rangle$ respecting the conditions below.

For each tuple l_j $(1 \leq j \leq n)$ we have:

(i) If the root of T' is a target position for K_j (or FK_j) or a position in the target path then all tuples in the set specified by the attribute t_j, which is in tuple l_j, has length m_j (*i.e.*, its length equals the number of elements composing a key (or foreign key) tuple).

(ii) If the root of T' is a context position for K_j (or FK_j), or a position in the context path, then the attribute c_j in l_j is a tuple containing the value *true*. □

Given a tree T and a sequence of updates over T, the incremental validation problem wrt integrity constraints consists in checking whether the updated tree does not violate any constraints, by visiting only the part of T involved by the updates. We propose an algorithm, similar to Algorithm 1, to perform the incremental validation of an XML tree T.

Example 9 Consider the XML tree of Fig. 3 where update positions are marked. As in Section 3, updates are treated in the document order. Now, we remark that, in the incremental validation wrt integrity constraints the following steps are performed:

1. When the open tag <d> (position 0.1) is reached, the deletion operation is taken into account and a search of key and foreign key values is triggered in order to verify whether there are key or foreign key values in the subtree rooted at the deletion position. If such kind of values are found, we use the corresponding *keyTree* to test whether the deletion does not imply any violation of integrity constraints. If a violation is detected we mark position 0.1 (in *keyTree*) and the document is considered to be temporarily invalid. Only after analyzing late updates we can decide whether this deletion operation can be performed. A late update can indeed restore document validity (*e.g.*, by removing foreign key references from a key involved in our delete operation). In this case, our deletion can be performed.

2. When the open tag <a> at position 1 is reached, we use the corresponding *keyTree* to test whether the insertion does not imply any violation of integrity constraints. We consider the insertion as performed (similarly to Section 3). If the insertion represents a violation of an integrity constraint, then there is a subtree where key values are duplicated wrt those being inserted. We mark this subtree in *keyTree* and proceed as in the above item. Notice that, similarly to Section 3, we can skip nodes below position 2.

3. The replace operation at position 3 combines the effects of a deletion and an insertion.

4. At the end of the tree traversal a final validity test is executed. It consists on verifying whether the resulting *keyTree* does not contain any violation label (meaning that a postponed violation correction was not performed).

The implementation of this approach is done by the following algorithm. Notice that it uses the *UpdateTable* to obtain the update positions. When an update position is reached, different tests are performed, according to the update operation.

Algorithm 2 - Incremental Validation of Keys in Multiple Updates

Input:

(i) *doc*: An XML document.

(ii) $UpdateTable$: A relation that contains updates to be performed on *doc*. Each tuple has the general form $\langle pos, op, T_{pos}, \Phi\rangle$, similar to Algorithm 1.

(iii) $KeysFSA$: Set of finite state automata describing the paths that appear in the keys and foreign keys.

(iv) $KeyTree$: Structure that contains the key values resulting from the last validation performed on *doc*.

Output:

If *doc* remains valid after all operations in $UpdateTable$ the algorithm returns the Boolean value $true$, otherwise $false$.

Local Variables:

(i) $CONF$: structure storing the inherited attributes.

(ii) $SYNT$: structure storing the synthesized attributes.

(iii) $keyTreeTmp$: copy of the initial $keyTree$.

(1) $keyTreeTmp$:= $KeyTree$
(2) $CONF_\epsilon$:= $InitializeInhAttributes(KeysFSA)$
(3) **foreach** *event* v *in doc* **do**
(4) **switch** (v) **do**
(5) **case** *start of element* a *at position* p
(6) Compute $CONF_p$ using $KeysFSA$
(7) **foreach** $u = (p, \mathsf{insert}, T_p, \Phi) \in UpdateTable$ **do**
(8) **if** $(\neg\mathsf{insert}(doc, p, T_p, keyTreeTmp))$
(9) **then** report "invalid" and halt
(10) **if** $\nexists u' = (p', op', T', \Phi') \in UpdateTable$ *such that* $p \prec p'$
(11) **then** $skipSubTree(doc, a, p)$;
(12) **case** *end of element* a *at position* p
(13) Compute $SYNT_p$ using $CONF_p$
(14) **if** $\exists\ u = (p, \mathsf{delete}, \Phi)\ \in\ UpdateTable$
(15) **then if** $(\neg\mathsf{delete}(doc, p, SYNT_p, keyTreeTmp))$
(16) **then** report "invalid" and halt
(17) **if** $\exists\ u = (p, \mathsf{replace}, T_p, \Phi)\ \in\ UpdateTable$
(18) **then if** $\neg\mathsf{replace}(doc, p, SYNT_p, T_p, keyTreeTmp)$
(19) **then** report "invalid" and halt
(20) **case** *value*
(21) str := $value(p)$
(22) Compute $SYNT_p$ using $CONF_p$ and str
(23) **if** $(\neg valid(keyTreeTmp))$
then report "invalid" and halt

(24) **return** $true$ □

Algorithm 2 describes the incremental validation of key and foreign key constraints while reading the XML tree in the document order. Firstly, we make a work version of the initial valid *keyTree*. This work version will be changed as we process a sequence of update operations. At the end of this sequence we check whether the obtained *keyTreeTmp* is valid and, only in this case, it will replace the original *keyTree*.

The algorithm uses two structures to store the attribute values, namely *CONF* and *SYNT*. At each position p, the structure $CONF_p$ keeps the roles of p wrt the keys and foreign keys being verified. Indeed, $CONF_p$ contains one inherited attribute *conf* (as defined in Tables 2 and 3) for each key and foreign key at position p. The structure $SYNT_p$ contains a tuple formed by the synthesized attributes k, t, and c (see Tables 2, 3 and 4) for each key and foreign key at position p.

When reaching an open tag at position p, the inherited values to be stored in $CONF_p$ are computed and all requested insertions at this position are performed according to our insertion method (Algorithm 3). On the other hand, when reaching a close tag in position p, we compute the synthesized values to be kept in $SYNT_p$ and we perform the requested deletions and replacements. Recall that the computation of synthesized values depends not only on $CONF_p$ but also on the result of each $SYNT_{p'}$ (where p' is a child of p). Notice that $SYNT_p$ is necessary in delete and replace operations, since for each key and foreign key, we need to remove key and foreign key values from the corresponding *keyTreeTmp* (see Algorithm 4).

Algorithm 3 defines the operation $\mathsf{insert}(T, p, T', keyTreeTmp)$ to include a new subtree T' in an XML tree T at position p. It first computes the tuple τ which is the result of the local validation for T'. Recall that τ is composed by tuples containing the synthesized attributes and by the (partial) *keyTree*s associated to T'.

An insert operation is accepted or rejected. If an insertion is accepted then the key or foreign key values found in T' are inserted in the corresponding *keyTreeTmp*. Notice that an insertion is temporarily accepted in two cases, namely:

• If the key tuple value corresponding to a foreign key instance being inserted does not exist. In this case, we add the new key tuple value to *keyTreeTmp* and we mark it with a null value at attribute pos.

• If the insertion of a key instance generates duplicated tuple values for a key. In this case, we mark the original tuple in *keyTreeTmp* as a duplicate value.

In both cases, we postpone the final decision of rejecting or accepting the insertion. We just mark the involved tuples in *keyTreeTmp*, keeping the document temporarily invalid. We should notice the difference between duplicating key values and key (or foreign key) components. Indeed, if the subtree to be inserted requests a duplication of key (or foreign key) component nodes, then it is rejected since we assume that nodes composing a key (or a foreign key) must be unique. For instance, if we consider the key K_1 of Example 6, then the insertion at position 0.0 of a new category under a collection (position 0) is rejected. As *category* is the key node for K_1, the subtree rooted at *collection* cannot have more than one *category* child.

Algorithm 3 - The insert operation: $\mathsf{insert}(T, p, T', keyTreeTmp)$

(1) **if** ((τ := LocalValidation(T, p, T')) = 'invalid') **then** return **false**
(2) **for** each tuple $l_i = \langle k_i, t_i, c_i\rangle$ $(1 \leq i \leq m)$ in τ, corresponding to K_i **do**
(3) **if** $(k_i \neq <>)$ **then** return **false**
(4) **if** $(t_i \neq \emptyset)$ **then**
Find the context position p' (above p) for K_i
(5) **for** each tuple $v \in t_i$ **do**
(6) **if** $\exists u = v$ in $keyTreeTmp_{K_i}[T, p']$ **then**
(7) **if** u is associated to an attribute $pos \neq null$
(8) **then** Mark u in $keyTreeTmp_{K_i}[T, p']$ with *dup* = *"yes"*
(9) Add $keyTree_{K_i}[T', p]$ to $keyTreeTmp_{K_i}[T, p']$
(10) **if** $(c_i = \langle true\rangle)$ **then**
(11) Add $keyTree_{K_i}[T', p]$ to $keyTreeTmp_{K_i}[T, \epsilon]$
(12) **for** each tuple $l_j = \langle k_j, t_j, c_j\rangle$ $(m+1 \leq j \leq n)$, corresponding to FK_j **do**
(13) **if** $(k_j \neq <>)$ **then** return **false**
(14) **if** $(t_j \neq \emptyset)$ **then**
(15) Find the context position p' (above p) for FK_j
(16) **for** each tuple $v \in t_j$ **do**
(17) **if** $\exists u = v$ in $keyTreeTmp_{K_i}[T, p']$ **then**
(18) increment *refCount* of tuple u by 1
(19) **else** build a subtree β which corresponds to tuple u as follows:
(20) ⋆ Attribute pos is null
(21) ⋆ Attribute $refCount$ is 1
(22) ⋆ Key values compose the tuple u
(23) add the subtree β in $keyTreeTmp_{K_i}[T, p']$
(24) as rightmost child of context node p'. □

The insertion of a new subtree rooted at position p is possible if the following tests succeed:

1. The subtree being inserted is locally valid.
2. For each key K_i $(1 \leq i \leq m)$:

(a) If p is a position in the key path (*i.e.*, a position below the target node) then the update operation implies the duplication of key nodes. In this case, the insertion is rejected.

(b) If p is a position in the target path, then we insert the new key values in $keyTreeTmp_{K_i}$ under the corresponding context. The key value may already exist in $keyTreeTmp_{K_i}$. It is necessary to test if it exists as a duplicate or as an "incomplete" insertion triggered by the previous insertion of a foreign key. The test consists in verifying whether attribute pos is null.

If there are duplicated key values wrt the one being inserted, then the duplication is annotated by adding *dup* = *"yes"* in $keyTreeTmp_{K_i}$. If the insertion corresponds to an incomplete previous insertion, it is completed by changing the value of attribute pos.

(c) If p is a position in the context path, then the insertion is accepted since tree T' is locally valid wrt key and foreign keys (item (1)) .

3. For each foreign key FK_j $(m+1 \leq j \leq n)$:

(a) If p is a position in the foreign key path, then we proceed as in item 2(a).

(b) If p is a position in the target path, then we are inserting new foreign keys wrt a key K_i. We increment the *refCount* (in $keyTreeTmp_{K_i}$) of each K_i tuple corresponding to a foreign key tuple being inserted. If there is not a corresponding tuple in $keyTreeTmp_{K_i}$ then a subtree containing the key values which correspond to the inserted foreign key is added to $keyTreeTmp_{K_i}$ as the rightmost child of the concerned context. For this subtree, attributes $pos = null$ and $refCount = 1$.

Algorithm 4 defines the update operation delete$(T, p, \sigma, keyTreeTmp)$ where p is the position to be removed from the XML tree T, and σ is the tuple resulting from the local validation of the subtree T' originally rooted at p.

If a deletion is accepted, then the key and foreign key values are removed from the corresponding *keyTreeTmp*. Notice that a deletion is temporarily accepted if it concerns a key tuple whose *refCount* is not 0. Indeed, in this case, the deletion is finally accepted only if all the foreign keys referencing this tuple are also removed from T. We postpone the final decision of accepting or rejecting the deletion to the end of the update sequence. We consider that key (or foreign key) nodes cannot be deleted.

Algorithm 4 - The delete operation: delete$(T, p, \sigma,$ ***keyTreeTmp***$)$

(1) **for** each tuple $l_i = \langle k_i, t_i, c_i \rangle$ $(1 \leq i \leq m) \in \sigma$, corresponding to K_i at position p **do**
(2) **if** $(k_i \neq <>)$ **then** return **false**
(3) **if** $(t_i \neq \emptyset)$ **then**
(4) Find the context position p' (above p) for K_i.
(5) **for** each tuple $v \in t_i$ **do**
(6) Find the tuple $u = v$ in $keyTreeTmp_{K_i}[T, p']$
(7) **if** *refCount* associated to tuple u is 0
(8) **then** remove u from $keyTreeTmp_{K_i}[T, p']$
(9) **else** mark tuple u in $keyTreeTmp_{K_i}[T, p']$ with *del* = *"yes"*
(10) **if** $(c_i = \langle true \rangle)$ **then**
(11) **for** each context pos p' under or equal p **do**
(12) remove $keyTreeTmp_{K_i}[T,p']$from $keyTreeTmp_{K_i}[T,p]$
(13) **for** each tuple $l_j = \langle k_j, t_j, c_j \rangle$ $(m+1 \leq j \leq n)$ in σ, corresponding to FK_j **do**
(14) **if** $(k_j \neq <>)$ **then** return **false**
(15) **if** $(t_j \neq \emptyset)$ **then**
(16) Find the context position p' (above p) for FK_j
(17) **for** each tuple $v \in t_j$ **do**
(18) Obtain the tuple u that is referenced by v in $keyTreeTmp_{K_i}[T, p']$;
(19) Decrement *refCount* associated to u by 1;
(20) **if** (*refCount* = 0) **and** (*del* = yes) in u
(21) **then** remove u from $keyTreeTmp_{K_i}[T, p']$ □

To remove a subtree rooted at position p we execute the following tests for keys and foreign keys:

1. For each key K_i $(1 \leq i \leq m)$:

(a) If p is a position in the key path, then the deletion is rejected.

(b) If p is a position in the target path and if each target to be deleted is not referenced by any foreign key, then the deletion is accepted. Otherwise, the corresponding target in $keyTreeTmp_{K_i}$ is labeled to be deleted subsequently.

(c) If p is a position in the context path, then the deletion is possible for all contexts, since we are removing key values and also the foreign key values that reference them.

2. For each foreign key FK_j $(1 \leq j \leq n)$:

(a) If p is a position in the foreign key path, then the deletion is rejected.

(b) If p is a position in a target path, then the deletion is accepted and we decrease the corresponding *refCount*s in $keyTreeTmp_{K_i}$. If *refCount* turns to 0 and the target tuple in $keyTreeTmp_{K_i}$ was earlier marked to be deleted, then this tuple is removed.

The replace operation combines the deletion and the insertion but it is not equivalent to the update sequence (insert$(T, p, T', \tau,$ *keyTreeTmp*); delete$(T,\ p,\ \sigma,$ *keyTreeTmp*)) since it allows the replacement of key (or foreign key) nodes. For instance, the replace operation allows the substitution of a recipe author, even if *author* is part of key K_2 of Example 6 (recall that the delete operation does not allow this removal). If there is a replace operation to be performed at position 0.2.1 (to change the value *M. Smith* to *L. Greene*), it is accepted if key K_2 is still respected after the update.

Example 10 We suppose the XML tree of Fig. 1 and an update sequence composed by: (1) An insertion of a new *recipe* at position 0.1; (2) A deletion at position 0.2; (3) A deletion at position 0.3.0.

The initial XML tree is valid, and we keep a work version of its *keyTree* in the new structure *keyTreeTmp*, used here to verify the validity of the updates wrt key and foreign keys. In this example we consider keys K_2 and FK_4 defined in Example 6. The update sequence is examined while reading the XML tree, as shown in Algorithm 2. In this way, the tests are performed in the following order:

1. When we reach the open tag of element *recipe* at position 0.1, we find that there is an insertion to be performed. The new subtree to be inserted is a new recipe that contains instructions and ingredients for preparing a broccoli soup. To check if this insertion is valid, the insert operation in Algorithm 3 is performed. The new key values are inserted in $keyTreeTmp_{K_2}$ under the collection of soups (context at position 0, prefix of position 0.1), as shown in Fig. 13. Notice that attributes pos are not updated yet.

2. When the close tag of element *recipe* at position 0.2 is reached, there is a deletion to be performed (the removal of the mushroom soup recipe). To verify if this deletion is accepted, the delete operation in Algorithm 4 is performed. It also concerns key K_2, and the key values under position 0.2 must be removed from $keyTreeTmp_{K_2}$ under the collection of soups (context at position 0). Fig. 14 shows that this deletion is postponed, since the *refCount* associated to mushroom soup in $keyTreeTmp_{K_2}$ is 1.

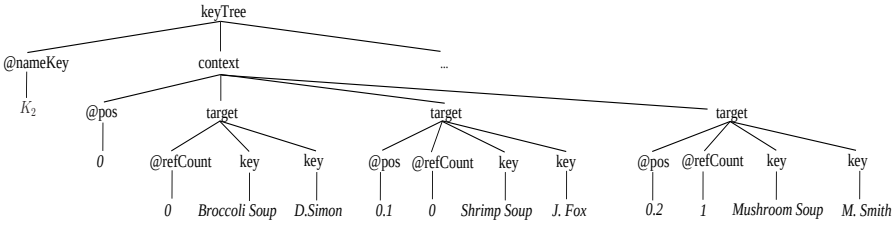

Figure 13: $keyTreeTmp_{K_2}$ after insertion at position 0.1.

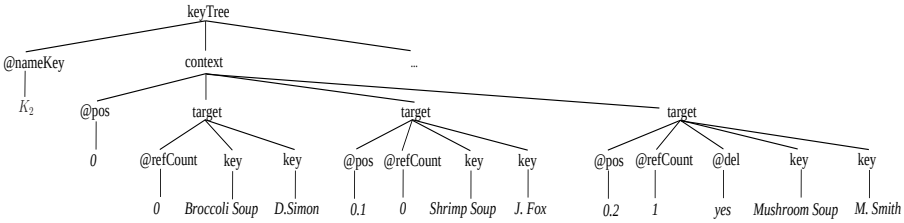

Figure 14: $keyTreeTmp_{K_2}$ after deletion at position 0.2.

3. When the close tag of element *top_recipe* at position $0.3.0$ is reached, the deletion of the top recipe with *number = 1* (mushroom soup) should be done. Since this deletion concerns FK_4 (that references K_2), the foreign key values under position $0.3.0$ are removed from *keyTreeTmp*$_{K_2}$ in the collection of soups (context at position 0). This is done by decreasing of 1 the corresponding *refCount* for mushroom soup. At this point, as *refCount* becomes 0 and the mushroom soup tuple was already marked to be deleted, then the deletion (earlier postponed) is accepted, as illustrated in Fig. 15. At the end of the update sequence, we traverse *keyTreeTmp*$_{K_2}$ in order to: (a) update attributes pos and (b) verify the existence of violation marks.

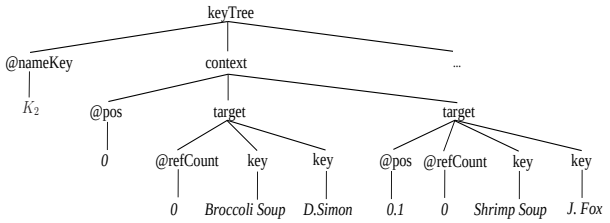

Figure 15: $keyTreeTmp_{K_2}$ after deletion at position $0.3.0$.

Notice that we consider that the user provides our algorithm with a *set* of updates. The processing of these updates is performed in the document order, *i.e.*, in the order in which the nodes of the XML tree are visited[6]. This ordering is natural to our algorithm and is the most efficient one, since we will perform only one pass over the tree.

The semantics of our method is independent of the order in which the update operations are processed. Our method has the following properties:

[6]This is why we present *UpdateTable* as a sequence built in a preprocessing phase from a given set of updates and respecting properties stated in Section 2.

• The operations are performed only whenever possible. In the presence of errors in the verification of constraints, our algorithm will not perform the updates. (this condition is similar to that for transactions, in the context of relational databases).

• The updates are performed if and only if there exists an ordering for them, whose final result preserves the validity of the document wrt the verified constraints.

Notice that the second condition above is verified since our method is as general as possible. It has the same effect as the ordering which enables us to accept most update sequences, that is: (i) remove foreign keys; (ii) remove primary keys; (iii) insert primary keys; (iv) insert foreign keys.

This condition means that our method is as general as possible, accepting the largest class of possible sets of updates.

4.3 Complexity and experimental results

The validation method proposed in Section 4.1 requires only one pass on the XML document. As in [24], its running time is linear in the size of the XML document. This complexity is not affected by the shape of the XML document, but it can be affected by the size of the *keyTree*. Indeed the time and space complexities of our method are based on the size of the *keyTree* which indicates the number of key instances existing in the document (*i.e.*, the number of target nodes). When the *keyTree* is big, the following steps are time consuming: the set inclusion test done (once) at the root level (Table 2) and list comparisons at the context level, which are performed once for each key. The space complexity for our method corresponds to the size of *keyTree*. Each *keyTree* is a subset of the document, containing only the necessary information to verify the validity wrt a given integrity constraint.

In Algorithm 2, each update position p activates a *validation step* (*i.e.*, Algorithm 3 or Algorithm 4) which depends on the kind of the required update. To find if key and foreign key values are involved in the update, algorithms consult the list τ containing the result of the local validation of the subtree T_p (being inserted of deleted). The list τ is built in time $O(|T_p|)$. We recall that insertions and deletions trigger a subtree traversal to find the key and foreign key values.

Algorithm 3 and 4 go through the resulting list τ in order to detect non empty lists (*i.e.*, those that correspond to the key (or foreign) values being inserted or deleted). Then the algorithms compare these values to those in the *keyTree*. To this end, Algorithm 3 and 4 visit key nodes in the *keyTree* corresponding to the context under which the insertion or deletion is required. Let n be the number of key and foreign key constraints and let n_{target} be the maximum number of target nodes (*i.e.*, key tuples). This operation runs in time $O(n.n_{target}.c_2)$ where c_2 is the maximum number of components of a composed key (*i.e.*, the number of key nodes under a target).

In the worst case, each insertion (validation of the subtree being inserted and Algorithm 3) and each deletion (Algorithm 4) runs in time $O(|T_p| + n.n_{target}.c_2)$. Let m be the number of updates. Since validation steps are needed only on update positions, if we disregard the value of c_2 then the cost of incremental validation under m updates is $O(m.(|T_p| + n.n_{target}))$. The last step of our incremental validation corresponds to visiting and updating *keyTree*s. To this end, we need to visit all the attribute nodes of each *keyTree* concerned by the updates. This routine runs in time $O(n.n_{target})$.

The implementation of our validation method was done in Java, and the Xerces SAX Parser was used for reading the XML documents and loading them into our data structures. Each key or foreign key is checked by running the finite state automata that corresponds to its path. We use two stack structures to store the inherited and synthesized attributes.

We now present the results of a preliminary experimental analysis of our validation method. All tests were executed on the same 1.7GHz Pentium 4 machine with 256MB memory, running Windows 2000. For these tests, we used 4 XML documents, varying in the number of nodes (XML elements and attributes) from $250,000$ to $1,000,000$.

To verify the validity of a set of keys and foreign keys over an XML document, we fixed the number of keys to 2 and foreign keys to 1, and we varied the size of the XML document to verify the method's performance. The CPU time required to check the validation from scratch of the XML document wrt the given keys and foreign key is shown in Fig. 16 (a).

To verify the checking time wrt the number of integrity constraints, we fixed the size of the XML document to $500,000$ and we varied the number of keys and foreign keys from 1 to 5. The results (for validation from scratch) are shown in Fig. 16 (b).

The implementation of our incremental validation method was also done in Java. The sequence of updates is treated as a unique transaction and the updates are tested in one traversal of the XML tree. In fact, the sequence of updates is treated before being applied, so that the update positions are sorted in the sequential order of the XML tree lecture.

Our method incorporates the operations *insert* and *delete* and tests for each operation, by using the $KeyTree$, if the XML tree still respects the predefined set of integrity constraints. The $KeyTree$ is stored in a hash table structure that associates each constraint (primary key) with values (the various contexts obtained for the constraint and their corresponding targets, attributes, key data values and foreign key references). The hash table structure organizes the constraints information reducing look up time and it also improves the verification performance (Fig. 17).

The local validation of each subtree to be inserted or deleted is triggered when the update position is reached. The result of this local validation is loaded into new data structures and a new $KeyTree$ is built for the subtree. We can note that if the update does not concern the set of keys and foreign keys, then the local validation is an empty structure, meaning that there are no verifications to be executed. On the other hand, updates concerning constraints that have composed key paths are the ones which demand more comparisons.

Fig. 18 summarizes the behavior of our algorithm for the incremental validation of keys and foreign keys. We ran two experiments, similarly to the ones for the validation from scratch, using a sequence of 50 updates. In the first one we varied the size of the XML document to test the updates wrt five fixed keys and foreign keys. The second experiment was done by fixing the XML document size ($500,000$ nodes) and varying the number of constraints.

Our last experiments aim at assessing the systems capabilities with respect to keyTree construction. As an example consider an absolute key, composed of three parts (an order is identified by its number, the product number and the supplier number). The validation of this key generates a keyTree containing all values found for this key if they are all different. Clearly, the size of the keyTree may vary drastically with respect to the number of key instances found in the document, and it is of particular interest for incremental validation. Fig. 19 shows the time response considering XML documents that contain from 10^2 to 10^5 key instances for validation from scratch. A sequence of 50 updates was considered for incremental validation.

4.4 Related work

In this paper, we present the validation of XML documents wrt schema and its verification wrt integrity constraints in an independent way. However, if we assume that the given integrity constraints are consistent with the given schema, then the integration of both validation routines is a straightforward generalization of our method.

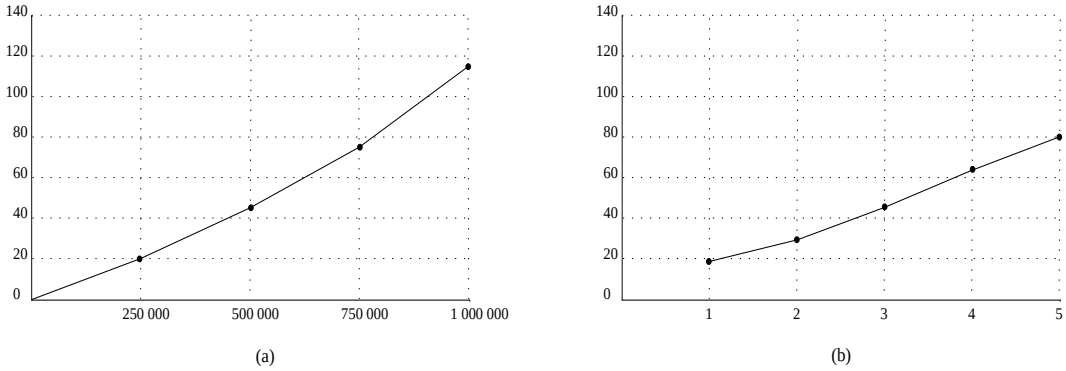

Figure 16: Validation from scratch: (a) Number of nodes $\times$ checking time (in seconds). (b) Number of integrity constraints $\times$ checking time (in seconds).

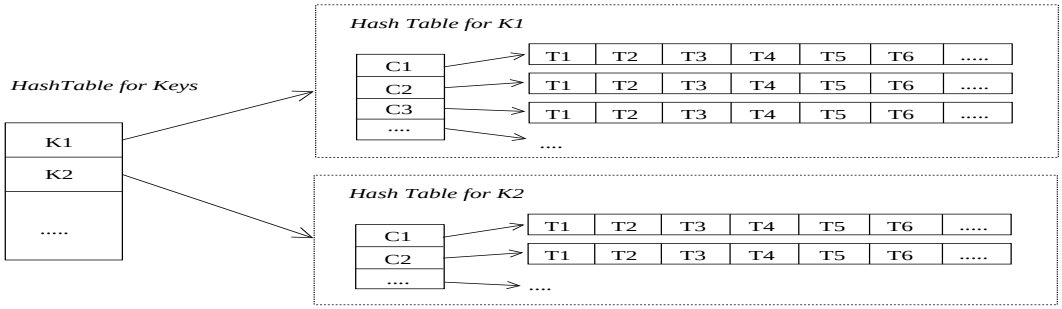

Figure 17: Two-level hash table structure to represent $KeyTrees$.

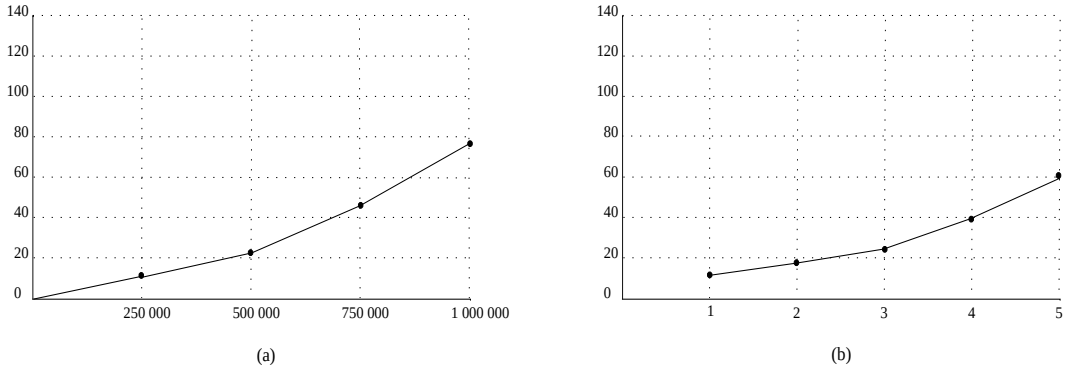

Figure 18: Incremental validation: (a) Number of nodes $\times$ checking time (in seconds). (b) Number of integrity constraints $\times$ checking time (in seconds).

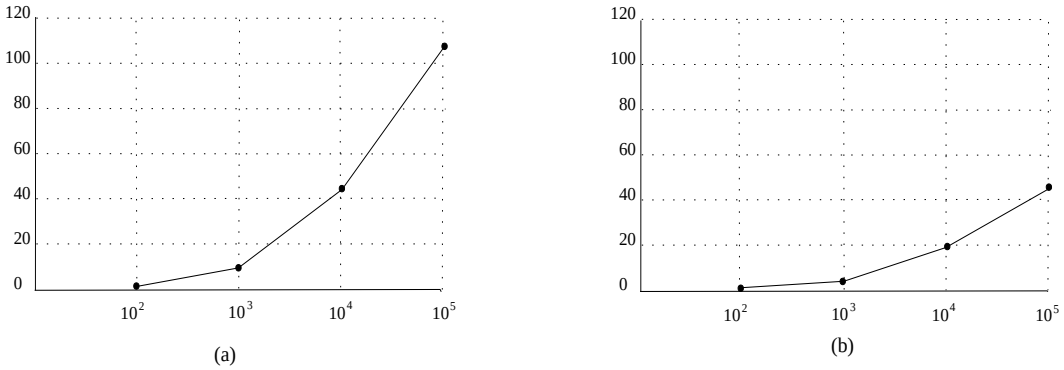

Figure 19: Number of key instances $\times$ checking time (in seconds): (a) Validation from scratch for an absolute key. (b) Incremental validation.

In [21], the independence of key syntax from schema constraints is presented as an advantage. One can wonder if this advantage is so clear since structural and integrity constraints are both parts of database design, implementation and optimization. However, as shown in [10], verifying consistency of structural and integrity constraints for XML documents is a difficult problem. In several cases, in order to reason about satisfiability and implication, it is better to consider only integrity constraints.

Proposals that deal with both structural and integrity constraints usually impose restrictions on schema definitions. In [30], for instance, the authors propose an Unified Constraint Model (UCM) which is tightly coupled with the schema model. The schema is defined using *the type system* of XQuery algebra [32] which captures the structural aspects of XSD. Validating a document wrt schema constraints corresponds to finding a *unique* type assignment to each position on an XML tree. In other words, schemas must only define LTL (Section 3). In UCM it is possible to define the "key of a type" by specifying the type name together with the key components (defined by path expressions). A drawback of [30] is the lack of relative and foreign key constraints.

In [31] key paths are composed of a (possibly empty) ranked sequence of up symbols (*i.e.*, Kleene closure cannot be used) followed by a nonempty simple (downward) path. For instance, consider a document with information about conference proceedings. Proceedings are identified by an absolute key $key_{Proc} = (/bib/proc/, \{confName, confYear\})$ containing the name of the conference and its year. As articles are identified by numbers inside a proceedings, we need a relative key to express this constraint. Thus, we write $key_{Article} = (/bib/proc/article/, \{numArticle, [up]/confName, [up]/confYear\})$ where up is a wildcard denoting a move up on the XML tree. Such a syntactic possibility can be translated into CTK. Let $K_1 = (/, (bib/proc/, \{confName,\ confYear\}))$ and $K_2 = (/bib/\ proc/, (article/, \{numArticle\}))$ be two CTK keys. Notice that $key_{Article}$ holds if K_1 and K_2 do. More generally, let k be the length of the longest upward wildcard sequence in a constraint C. It can be shown that there is a set $\{C_1, ..., C_k\}$ of CTK constraints such that if $\{C_1, ..., C_k\}$ is verified then C is verified too.

The goal of [12], where XPath and XSD are used to express constraints, is to perform incremental constraint checking. Their approach differs from ours since in [12] constraints are translated into logic formula and updates on documents are related to computation of incremental versions of the formula. They use schema information to optimize their computation.

In [24], a validator for XML constraints is presented for incrementally checking updates. Its performance is linear in the size of the part being updated, for each key being checked. Although this work is similar to ours, it is worth noting that in [24] foreign keys are not treated and the incremental validation is considered only for a single update. In our approach, as the validation of keys and foreign keys is done for multiple updates, the maintenance of the *keyTree* is performed progressively (while considering the update sequence). The *keyTree* can be temporarily invalid. In this way, our complexity is bound to the number and type of update operations, the number of constraints that are being verified and the number of tuple values corresponding to a constraint.

The current work extends and merges our previous proposals which have considered XML document validation only under single updates. In [17], we propose an incremental schema validation method, but only wrt DTD. In [7, 18] we use a tree transducer to address the problem of the key validation. In the current paper integrity constraints are specified by an attribute grammar which makes our validation strategy simpler. At this point, some similar aspects with [13] can be observed. In [13] XML documents are mapped to relational databases. Indeed, relational DBMS are tuned to efficiently validate integrity constraints. Nevertheless, when relying on a relational DBMS, the system must factorize XML documents before it can store them in its data structures. This entails format mappings and interchanges between XML's hierarchical structure and the DBMS structures, which raises non trivial, theoretical questions about the relational database design on

the one hand, and XML update (and XML query) translations on the other hand (as shown in, *e.g.*, [19, 48, 38]).

Proposals for mapping XML in RDBMS by considering XML integrity constraints exist [25, 28, 27, 39]. However, in [28, 27], the authors state that it is impossible to effectively propagate all forms of XML constraints supported by XML Schema, including keys and foreign keys, even when the transformations (from XML to relational) are trivial. In contrast [39] does not deal with such theoretical consideration, but the authors present an ad hoc translation of XML constraints in a relational framework. Experimental results reported in [39] compare this proposal to the one in [9], and conclude that the translation of keys and foreign keys leads to improve query execution time in the context of relational databases.

Authors in [25] notice that, in current propositions, the design of the relational database aimed to store XML data is tuned either for updates (enforcing the constraints efficiently), or for queries workload (to achieve better performance), however, to find a good compromise for both seems to be still an open problem. Indeed, as noticed in [38], one big, open challenge is to efficiently process queries to hierarchical XML data, in a database whose fundamental storage is table-based and whose fundamental query engine is tuple-oriented.

Update processing is even a bigger challenge in this context: for instance one can notice that since a single XML update may affect several tuples in the relational store, transactions must be carefully used to prevent anomalies. Moreover, the XML "view" of the relational database must be *updatable*, *i.e.* there must be a unique, *side effect free* translation from any update on this view to the underlying relations. In [19] the authors show that this is still an open problem for XML views that are not defined by general nested relational algebra or that can not be rewritten into a *nest-last* relational form.

To conclude, the use of relational databases for allowing data to be imported, accessed and exported in the XML format is still an important challenge, addressed by some of the so-called "native XML data stores". As argued in [48], a natural implementation of such systems keeps an XML logical format (even if the underlying storage is relational or object-oriented) in order to achieve scalability, data-access speed and reliability. We can place our work in this context.

5 Conclusions and perspectives

In this paper we present a method to incrementally verify multiple updates in the presence of schema and integrity constraints in XML documents. Update operations are the insertion, deletion and replacement of any subtree of the XML tree. The validity of the resulting XML document is determined only after having analyzed all update operations in a given set of updates. If the resulting document does not violate the imposed constraints, then updates are committed and the document is permanently changed.

Though there exist previous works on schema validation on the hand, and key verification on the other hand, our approach is new in the sense that it considers them both simultaneously, in one single pass over the XML document being processed. This implies to deal with interesting research topics, including incremental validation, unranked tree processing (not translated into ranked trees) and attribute grammars, as well as with non-elementary and non-single update operations on XML documents.

The incremental verification of schema constraints is performed by using a bottom-up tree automaton to revalidate just the parts of the XML document affected by the updates. Our algorithm is not restricted to schemas specified by DTDs or XML Schema, but it also works on schemas obtained from any regular tree grammar, even those which are not local or single typed tree grammars. Attribute grammars are used to formalize the process of integrity constraints verification.

The algorithms presented here have been implemented in Java, and experimental results show that the incremental schema verification has advantages over the verification from scratch, for multiple updates and for large XML documents. In large scale tests, our incremental schema validation algorithm (although implemented without optimization) uses almost one third of the time needed for the validation from scratch performed by a highly optimized commercial product.

The experimental results obtained with our key and foreign key validation routines are also encouraging. Despite their theoretical complexity, their behavior led to a graphic which grows almost linearly with the size of the processed document (Fig. 16). Results for the incremental key verification programs are similar, but more efficient, to those of the verification from scratch. Indeed, the incremental verification of keys is, in the limit, better than the verification from scratch (compare the results in Fig. 16 and 18).

Our asymptotic time complexity and experimental results show the efficiency of our incremental validation of updates: it is at least as efficient as those proposed by [23, 45] (even if it is hard to compare precisely). Moreover, it has some advantages over them, such as space requirements, multiple updates on any tree node and flexibility of integrating schema and key constraint validation.

Several directions have to be investigated for future work:

Integration of our approach into an existing update framework: We have designed algorithms to run in the "back office" of a framework enabling the performance of updates over XML documents. As discussed in Introduction, it can be either a text editor and/or an update language. Next step will be to integrate our routines in such a framework.

New update operations: Our set of update operations conforms to recent recommendations for updates in

XQuery ([22]), but it does not include insertion or deletion of a node in a path, say under position p, *i.e.*, an update operation allowing changes on the hierarchy by the addition or the removal of one level. Although these operations are not explicitly recommended by [22], they may be useful for some applications. Nevertheless, before implementation, their semantics must be carefully specified: this new kind of insertion consists in adding to the child axis a new node that becomes the father of one child of p. Moreover, it can also be the nesting of several children of p under one new level: in that case, the update must contain the specification of which children become children of the newly inserted node. The new kind of deletion might be the inverse operation.

New classes of constraints: Our integrity constraint validator can be adapted to verify different constraints such as XML functional (XFD) and inclusion dependencies. Such constraints can be expressed in a language very similar to CTK. The approach to implement them can be exactly the same as the one presented here for keys and foreign keys (using the attribute grammar): we just need to adapt the positions where tests must be performed and the kind of tests to be performed.

Use of a more expressive constraint language: One of the first possibilities to consider is to extend our key constraint language to include other XPath expressions, such as predicates or the use of other axes. XPath has been widely used in XML query languages and in XML specifications. In practice, many applications do not need the excessive power of the full XPath (which makes it rather expensive to process); they use only a fragment of XPath. Considering the sub-languages of XPath studied in [15], we notice that path expressions in CTK belongs to $\mathcal{X}_r$ the XPath sublanguage which allows navigation along the ancestor and descendant axes (while others permit only parent and child axes); but does not allow upward navigation or the use of qualifiers (predicates). Indeed, since $\mathcal{X}_r$ is the sublanguage used by XML Schema to specify integrity constraints ([15]) we have (until now) considered that our CTK was well adapted to most of the practical cases. Nevertheless, it would be interesting to extend our proposition to a quite general constraint language such as the one presented in [29].

References

[1] Altova XMLSpy. At http://www.altova.com.

[2] Apache XML editor. Available at http://www.apache.org/xerces-j/.

[3] Official website for SAX. Available at http://www.saxproject.org/.

[4] XML schema. Available at http://www.w3.org/XML/Schema.

[5] XMLmind. Available at http://www.xmlmind.com/xmleditor/.

[6] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Publishing Company, 1995.

[7] M. A. Abrão, B. Bouchou, M. Halfeld Ferrari, D. Laurent, and M. A. Musicante. Incremental constraint checking for XML documents. In *XSym*, number 3186 in LNCS, 2004.

[8] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley, 1988.

[9] Sihem Amer-Yahia, Fang Du, and Juliana Freire. A comprehensive solution to the xml-to-relational mapping problem. In Alberto H. F. Laender, Dongwon Lee, and Marc Ronthaler, editors, *WIDM*, pages 31–38. ACM, 2004.

[10] M. Arenas and L. Libkin. A normal form for XML documents. In *ACM Symposium on Principles of Database System*, 2002.

[11] Andrey Balmin, Yannis Papakonstantinou, and Victor Vianu. Incremental validation of XML documents. *ACM Trans. Database Syst.*, 29(4), 2004.

[12] M. Benedikt, G. Bruns, J. Gibson, R. Kuss, and A. Ng. Automated update management for XML integrity constraints. In *Program Language Technologies for XML (PLANX02)*, 2002.

[13] M. Benedikt, C-Y Chan, W. Fan, J. Freire, and R. Rastogi. Capturing both types and constraints in data integration. In ACM Press, editor, *SIGMOD, San Diego, CA*, 2003.

[14] Michael Benedikt, Angela Bonifati, Sergio Flesca, and Avinash Vyas. Adding updates to XQuery: Semantics, optimization, and static analysis. In *XIME-P*, 2005.

[15] Michael Benedikt, Wenfei Fan, and Gabriel M. Kuper. Structural properties of XPath fragments. In *ICDT*, 2003.

[16] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simï£¡on. XQuery 1.0, w3c candidate recommendation 8 june 2006. Available at http://www.w3.org/TR/xquery.

[17] B. Bouchou and M. Halfeld Ferrari Alves. Updates and incremental validation of XML documents. In Springer, editor, *The 9th International Workshop on Data Base Programming Languages (DBPL)*, number 2921 in LNCS, 2003.

[18] B. Bouchou, M. Halfeld Ferrari Alves, and M. A. Musicante. Tree automata to verify key constraints. In *Web and Databases (WebDB)*, San Diego, CA, USA, June 2003.

[19] V. Braganholo, S. Davidson, and C. A. Heuser. On the updatability of xml views over relational databases. In *Web and Databases (WebDB)*, San Diego, CA, USA, June 2003.

[20] A. Brüggeman-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over nonï£¡ranked alphabets. Technical Report HKUST TCSC 2001 05, Hong Kong Univ. of Science and Technology Computer Science Center (available at http://www.cs.ust.hk/tcsc/RR/2001ï£¡05.ps.gz), 2001.

[21] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. C. Tan. Keys for XML. In *WWW10, May 2-5*, 2001.

[22] D. Chamberlin, D. Florescu, and J. Robie. XQuery update facility, W3C Working Draft 11 July 2006. Available at http://www.w3.org/TR/2006/WD-xqupdate-20060711/.

[23] Y. Chen, S. Davidson, and Y. Zheng. Validating constraints in XML. Technical Report MS-CIS-02-03, Department of Computer and Information Science, University of Pennsylvania, 2002.

[24] Y. Chen, S. B. Davidson, and Y. Zheng. XKvalidator: a constraint validator for XML. In ACM Press, editor, *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 446–452, 2002.

[25] Yi Chen, Susan B. Davidson, and Yifeng Zheng. Constraints preserving schema mapping from xml to relations. In *WebDB*, pages 7–12, 2002.

[26] J. Clark. XSL transformations (XSLT 1.0) -w3c recommendation 16 november 1999. Available at http://www.w3.org/TR/xslt.

[27] Susan B. Davidson, Wenfei Fan, and Carmem S. Hara. Propagating xml constraints to relations. *J. Comput. Syst. Sci.*, 73(3):316–361, 2007.

[28] Susan B. Davidson, Wenfei Fan, Carmem S. Hara, and Jing Qin. Propagating xml constraints to relations. In Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman, editors, *ICDE*, pages 543–. IEEE Computer Society, 2003.

[29] A. Deutsch and V. Tannen. XML Queries and Constraints, Containement and Reformulation. *Theoretical Computer Science*, 336(1), 2005.

[30] W. Fan, G. M. Kuper, and J. Siméon. A unified constraint model for XML. In *WWW10, May 2-5*, 2001.

[31] W. Fan, P. Schwenzer, and K. Wu. Keys with upward wildcards for xml. In *DEXA '01: Proceedings of the 12th International Conference on Database and Expert Systems Applications*, pages 657–667. Springer-Verlag, 2001.

[32] Mary F. Fernández, Jérôme Siméon, and Philip Wadler. An algebra for xml query. In *FSTTCS*, 2000.

[33] G. Ghelli, C. Rï£¡, and J. Simï£¡on. XQuery! an XML query language with side-effects. In *DATA-X colocated with EDBT 2006*, 2006.

[34] Mírian Halfeld Ferrari Alves. *Aspects dynamiques de XML et spécification des interfaces des services web avec PEWS*. Habilitation à diriger des recherches, Université François Rabelais de Tours, 2007. In preparation.

[35] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley Publishing Company, second edition, 2001.

[36] George Karakostas, Richard J. Lipton, and Anastasios Viglas. On the complexity of intersecting finite state automata. In *IEEE Conference on Computational Complexity*, 2000.

[37] C. Koch and S. Scherzinger. Attribute grammars for scalable query processing on XML streams. In *Workshop on Data Base Programming Languages (DBPL)*, pages 233–256, 2003.

[38] Muralidhar Krishnaprasad, Zhen Hua Liu, Anand Manikutty, James W. Warner, Vikas Arora, and Susan Kotsovolos. Query rewrite for xml in oracle xml db. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 1122–1133. Morgan Kaufmann, 2004.

[39] Qiuju Lee, Stéphane Bressan, and J. Wenny Rahayu. Xshrex: Maintaining integrity constraints in the mapping of xml schema to relational. In *DEXA Workshops*, pages 492–496. IEEE Computer Society, 2006.

[40] M. Mernik and D. Parigot (Eds). Special issue on attribute grammars and their applications. In *Informatica, Vol 24 No 3, September*, 2000.

[41] M. Murata. Relax (REgular LAnguage description for XML). Available at http://www.xml.gr.jp/relax/, 2000.

[42] M. Murata, D. Lee, and M. Mani. Taxonomy of XML schema language using formal language theory. In *Extreme Markup Language, Montreal, Canada*, 2001.

[43] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of XML Schema Language using Formal Language Theory. In *ACM, Transactions on Internet Technology (TOIT)*, 2004.

[44] Frank Neven. Attribute grammars for unranked trees as a query language for structured documents. *J. Comput. Syst. Sci.*, 70(2):221–257, 2005.

[45] Y. Papakonstantinou and V. Vianu. Incremental validation of XML documents. In *Proceedings of the International Conference on Database Theory (ICDT)*, 2003.

[46] Stylus Studio. XMLStylus. Available at http://www.stylusstudio.com.

[47] G. M. Sur, J. Hammer, and J. Simï£¡on. An XQuery-based language for processing updates in XML. In *PLAN-X - Programming Language Technologies for XML A workshop colocated with POPL 2004*, 2004.

[48] Athena Vakali, Barbara Catania, and Anna Maddalena. Xml data stores: Emerging practices. *IEEE Internet Computing*, 9(2):62–69, 2005.

Web-based Decision Support System for the Public Sector Comprising Linguistic Variables

Jože Benčina
University of Ljubljana, Faculty of Administration
Gosarjeva 5, Ljubljana
E-mail: joze.bencina@fu.uni-lj.si

Keywords: decision support system, decision-making, municipality, fuzzy logic, linguistic variable

Received: September 7, 2006

Decision support systems are often demanding in terms of modelling and use, while purchasing software can also represent too large an investment for many organizations. The level of maturity of an organization influences the use (or non-use) of methods and tools for decision support, which is definitely lower in the public sector than in the commercial sector. The public sector uses considerable assets in its operation and investments, and therefore good decisions are of crucial importance for further development (at state, regional and local levels). In this article we present a decision support system which incorporates a process, approach and web-based software that is simple enough to use and be accepted in organizations and environments less inclined to use a systematic approach to decision making. The system is implemented as a web application, and the simplicity of the system is enhanced by the use of fuzzy logic. With this article we are opening discussion on the question of implementing management support systems in the public sector, where, with a suitable approach and the support of responsible persons, such solutions could play an important role. The results of the case studies on the use of the system in Slovenian municipalities indicate that the task will not be an easy one, because the opinions of the participants concerning a systematic approach to decision making are widely divergent.

Povzetek: V prispevku je predstavljen internetni sistem za podporo odločanju v javnem sektorju.

1 Introduction

Decision support systems are gaining recognition in the public sector, which seeks solutions to various problems in a number of diverse areas. Many solutions are closely tied to individual fields, such as medicine [1], ecology [2] and spatial planning [3]. Others, in a more general way, are directed towards support in strategic planning and solving problems in management [4], [5]. Lately, due to the redirection of politics away from ascertaining public opinion about the functioning of the public sector towards public engagement and cooperation in decision-making processes, the number of solutions in the area of e-democracy is increasing [6], [7], [8], [9]. Support systems and cooperation in decision making are, however, still used mainly in narrow professional circles and have not found their way to political decision makers or to the public [10]. The challenge of successful implementation of a decision support system in the public sector, with engagement over the whole spectrum of decision making, is still unmet.

An important negative effect is also the conviction that there are great differences in decision making in the private and public sectors. This conviction is perpetuated by stereotypes of decision-making processes in both sectors, as shown in Table 1 [11]. The authors of a comparison, Bots and Lootsma [11], argue that all the mentioned approaches, with regard to the areas of operation and specifics of the branch of activity, can occur in either the private or public sector. For this reason, the question of decision making cannot be clearly separated into public and private decision making, yet we must take into account that the public sector has numerous specific features. If we add to this the increasingly emphasized demand for the engagement and co-deciding of the civil society, it becomes clear that in the development of decision support systems for the public sector, as well as in cases of direct transfer of solutions for the private sector to the public sector, we must also take into consideration the specific needs and demands of the public sector. Certain of these demands can be addressed by adapting existing solutions, but there are also numerous issues which demand special treatment and the development of a specific solution adapted to the environment.

Private sector	Public sector
Decisions are made by a single agent (individual manager or management team) whose authority is defined by a hierarchical organization structure.	Decisions are not made but "happen" as a result of a complex interaction between administrators, trade unions, pressure groups, etc.
Decisions are dominated by a single interest, typically the competitive position of the company.	Decisions involve many and often divergent interests of a society, and aggregation into such

	notions as "general welfare" only masks the conflict
Decision alternatives are evaluated on the basis of a limited set of quantitative economic criteria such as market share, bottom line profit or shareholder value.	The set of evaluation criteria is large and has a wide variety of both quantitative and qualitative criteria, whose values are difficult to establish and/or aggregate.
Decisions typically have a planning horizon of months to at most several years (e.g. new products and markets).	Decisions have a planning horizon of several decades (e.g. decisions on infrastructure).

Table 1: Perceptions of decision-making processes

When we speak about decision support in the public sector today, it is best to observe the issue in its most general form. Representative democracy, as we know it today, has a range of shortcomings. For this reason and thanks to the development of information and telecommunication technology, the public sector and politicians are seeking possible solutions to enable an approach to participatory democracy of an Athenian type. It requires that citizens be involved in all phases of decision making. They need, therefore, to learn about the problem, its alternative solutions and their implications, and about their own and other participants' interests and constraints. Since these interests may produce conflicts, the citizens need to be able to identify these conflicts and resolve them. It is also necessary that they be able and willing to take responsibility for their decisions [8]. E-democracy is today one of the principal challenges in the development of e-government [10], [15], [16].

Any consideration of decision making in the public sector must take into account that events take place in a triangle – politics ←→ civil society ←→ administration, where the civil society should be understood in the broadest sense as non-political and non-administrative (Figure 1) [7].

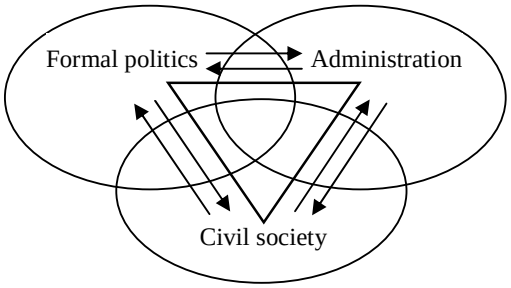

Figure 1: Basic spheres and relations in a democratic governmental system [7].

In the figure, arrows indicate influence and circles indicate domains of control. Intersections indicate "transaction zones" where control is negotiated by lobbyists and media, for example, on the left-hand side, intermediary service deliverers on the right-hand side and professional interaction in government boards and committees on the top [17].

Joint decision making by all three groups of participants is possible only if all of them are sufficiently acquainted with the subject of their decision making. Decision making should therefore be treated comprehensively as a process which implements all the phases necessary for high-quality decisions. The general process framework of decision making must take into consideration at least three phases (Figure 2) [18]:

- Formulating: Actors become aware of a decision problem at the "Doing" level, which represents the implementation phase, and initiate a decision process instance. Depending on their own background, experiences and agenda, as well as predefined goals and constraints, they formulate alternatives and criteria while seeking and filtering information about the problem. The produced alternatives and values are then passed to the appraising stage.
- Appraising: The role of actors at this stage is to assess the alternatives produced in the previous stage. Input that has been passed from the "Formulating" phase is evaluated and alternatives examined by the decision makers.
- Evaluating: The actors who are involved in this stage devise a framework for the evaluation of alternative interventions. Decision analysts or expert decision makers calculate the consequences of alternatives and choose a technique for the appraisal of alternative interventions.

Support for the decision-making process must be ensured with appropriate information and telecommunication technology and tools. The question of decision making in the public sector motivated our work and research with the aim of contributing to solving the issue and to adding a new solution to the range of current tools, specific to the environment in question. We are not alone in this, since the necessity of solving this type of problem is recognized throughout the world [11], [15], [16], [19], [20], [21].

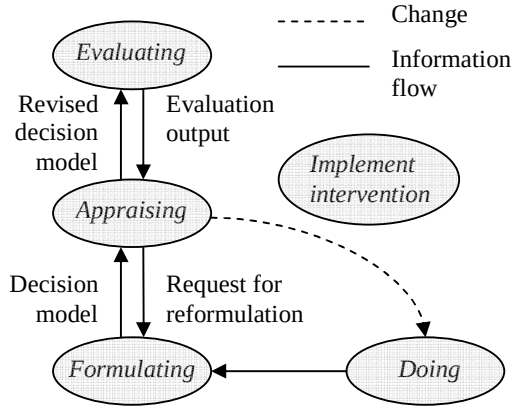

Figure 2: General decision process framework [18].

We have established improving the quality of decision making as our foremost goal. The concept of quality decision making is defined by efficiency, effectiveness, future-influencing capacity and legitimacy

[22], [11]. Efficiency is the ratio between invested effort and achieved results. In decision making, the internal efficiency that we would normally find in the production of products and services has little significance, since the number of decisions with respect to the time they take can only be an informative indicator, not to specify effectiveness. For this reason, effectiveness is linked to the decision-making results, that is, the effectiveness of reaching goals.

If we wish to achieve the long-term positive effects of the decision, we must set long-term goals, given that such goals are an important element of effectiveness. This especially holds true for the public sector. Legitimate decisions are those which the participants accept, and therefore the views of the participants must be incorporated in the goals, meaning that in assessing the quality of decisions in the public sector we are dealing with only two aspects: invested effort or the time consumed, and effectiveness, which is exemplified in achieving goals and legitimacy. We must chose suitable goals within a reasonable time and then select the optimum path towards reaching them.

The paths to high-quality decisions can be very diverse. In our case we must answer to various desires and the needs of a large number of participants, and therefore cooperation and reaching a consensus is undoubtedly the correct path. Experience indicates that effort invested in finding a consensus is rewarded with better, more innovative and efficient solutions, which are willingly accepted by the key participants [23]. Innes and Booher [24] specified three classes of the effects of the process of finding a consensus, which promise that efforts invested in consensus will be richly rewarded (Table 2):

First-order effects
➢ Social Capital: Trust, Relationships
➢ Intellectual Capital: Mutual Understanding, Shared Problem Frameworks, Agreed Upon Data
➢ Political Capital: Ability to Work Together for Agreed-Upon Ends
➢ High-Quality Agreements
➢ Innovative Strategies
Second-order effects
➢ New Partnerships
➢ Coordination and Joint Action
➢ Joint Learning Extends into the Community
➢ Implementation of Agreements
➢ Changes in Practices
➢ Changes in Perceptions
Third-order effects
➢ New Collaborations
➢ More Coevolution, Less Destructive Conflict
➢ Results on the Ground: Adaptation of Cities, Resources, Services
➢ New Institutions
➢ New Norms and Heuristics
➢ New Discourses

Table 2: Potential outcomes of consensus building [24]

2 DSS in public sector

Before we start with detailed aspects of the issue, it would be prudent to devote a few words to the definition of decision support systems [12].

Decision Support Systems (DSSs) are interactive computer-based systems intended to help decision makers utilize data and models to identify and solve problems and make decisions. The "system must aid a decision maker in solving unprogrammed, unstructured (or 'semistructured') problems...the system must possess an interactive query facility, with a query language that ...is ...easy to learn and use" [13]. DSSs help managers/decision makers use and manipulate data, apply checklists and heuristics, and build and use mathematical models. According to Turban [14], a DSS has four major characteristics: it incorporates both data and models; it is designed to assist managers in their decision processes in semistructured (or unstructured) tasks; it supports, rather than replaces, managerial judgment; and its objective is to improve the effectiveness of decisions, not the efficiency with which decisions are being made. The five types of Decision Support Systems are:

- Communications-Driven DSS – uses network and communications technologies to facilitate collaboration and communication;
- Data-Driven DSS – emphasizes access to and manipulation of a time-series of internal company data and sometimes external data;
- Document-Driven DSS – integrates a variety of storage and processing technologies to provide complete document retrieval and analysis;
- Knowledge-Driven - intended to suggest or recommend actions to managers. These DSSs are personal computer systems with specialized problem-solving expertise;
- Model-Driven DSS or Model-oriented DSS – emphasizes access to and manipulation of a model, e.g. statistical, financial, optimization and/or simulation. Simple statistical and analytical tools provide the most elementary level of functionality.

Most current advanced DSSs are combinations of all, or nearly all, five generic types. In the public sector, as a result of problem scope, social diversity and dynamics, the stakeholder network is generally more complex and less transparent, and its interests are more diverse. The variety of interests in particular seems to favor multi-criteria decision analysis (MCDA) approaches to decision support [11]. Thus at least two of the types of DSSs listed above (communications-driven and model-driven DSSs), especially for the public sector, should be able to handle a multi-criteria decision analysis approach.

Group Decision Support Systems (GDSSs) are interactive, computer-based systems that facilitate the solution of unstructured and semi-structured problems by a set of decision makers working together as a group. A GDSS aids groups in analyzing problem situations and in performing group decision-making tasks. Any of the five generic types of DSSs can be built as a GDSS.

Usually a DSS is tailored to either a specific application area (e.g. strategic planning, water management or policy making) or a particular decision-making phase (e.g. problem framing or decision-tree development), or both. According to Bots and Lootsma [11], they can be divided into three categories.

- Generic DSSs. These decision-support information technology (IT) applications are domain-independent (spreadsheets, generic DSSs for conceptual modeling, based on a particular problem solving method, and generic DSSs consisting of electronic meeting systems that support problem solving in a group).
- Domain-specific DSSs. The core of these systems is a model that computes the impact of measures on a given subsystem (economic, biological, or other) and presents the results in tabular or graphic form.
- Phase-specific DSSs. These applications aim to support one particular phase in the decision-making process (problem formulation phase, choice phase, negotiation support systems).

The aim of our research is the development of a model of a decision support system for the public sector and usable solutions for a chosen research environment. In this we have bound ourselves to the principle that the solution must be as general as possible. However, given the fact that for the development of e-democracy the local environment is the most suitable [16], we have decided to focus on local self-government and its key development problem: deciding on investment projects in local communities, with the aim of ensuring good decisions, which is related to the quality of selection of such projects. As the key point in ensuring the quality of decision making, we have focused on cooperation and reaching a consensus in determining that well prepared investment projects will be selected [24] and that they will be possible to realize within the set framework. It will in turn have a beneficial effect on efficient use of the local community's budget and ensure that the chosen investment projects will bring the participants long-term positive results.

A systematic approach to decision making is new to most Slovenian municipalities. For this reason, and because of time and financial limitations, we have limited the scope of the planned model and solutions in the sense of the typology of the projects [11] and a multi-phase approach [10], taking into consideration only two groups of participants. The subject of the research was the decision-making process concerning investment projects which are included in the Plan for Programme Development, as well as the annual budget in the local community. The aim of the research was to shape a decision-making model and the use of support system in a chosen environment within the following framework:

- Model-Driven Group Decision Support System based on Multi-Criteria Decision Analysis with suitable elements of Communications-Driven DSS:
- Domain-specific DSS, with the intention nearing a generic DSS, covering local government decision making on investment projects (plans for development programs and investments from the municipal budget);
- Phase-specific DSS with emphasis on the choice phase with the intention to cover some aspects of the problem-formulation phase and negotiation support systems;
- Considering two basic areas of democratic governmental systems (formal politics and administration) with the intention to make the participation of the civil society possible.

The research thesis refers to the feasibility of the model in given circumstances in a given environment and asserts: "The decision support system in a chosen environment and set framework enable simple expressions of appraisal and balanced participation in decision making for all participants and ensure a final solution which the decision makers and responsible persons consider to be suitable". We have checked this with case studies in three Slovenian municipalities.

3 Research design

We have addressed the issue by a review of the literature and by setting basic guidelines for a solution. We then studied the environment for dealing with the issue, i.e. municipalities. Within the aim of our research, that is, finding a solution to the issue, which encompassed a study of documentation and interviews with participants in the decision-making process, we sought answers to the questions of *how* decision making progresses and *why* undesired results occur.

We found answers to the following questions:

How…

- does decision making progress in including investment projects in the municipal budget?
- are the interests of various political options asserted?
- are various expert opinions and interests asserted in decision making?
- do expert opinions affect political decisions?
- does adjustment of opinions about individual projects and groups of projects that have been selected take place?

Why…

- do the selected projects often fail to meet expectations?
- can evidently less suitable projects dominate clearly more suitable ones?
- are attempts in adjusting opinions often unsuccessful?

On the basis of the case study we defined a solution model and developed web-based software for support of the model specified in the decision-making process. The research theses were analysed in case studies in three Slovenian municipalities within which we verified the suitability of the model, functioning of the software and the response of decision makers to this new, systematic approach to decision making.

We first presented the solution and its goals to the leaders in the municipality. We then analysed the situation in the area of investment projects and chose

projects (five to seven projects) about which decisions had been made. In two cases the responsible persons in the municipality invited experts and municipal counsellors, while in one case, due to local elections, we decided to first make an appraisal with the municipal expert services and postpone the counsellor appraisals to a time after the elections. The number of questionnaires handed in by representatives of expert services was six, nine and ten, while the municipal counsellors contributed two and four. The assessment was initially conceived as anonymous; however, in the first case it was decided to have personal signatures, and in the second case, a statement of affiliation to the expert services department. In the case of the counsellors the process was anonymous, while in the third case, appraisal by expert services was also fully anonymous.

We processed the results and presented them to the participants in the appraisals. During the presentation we initiated a discussion concerning the usefulness of the approach, and in two cases we made a survey by which we measured the opinions of the participants concerning the approach and the end results. On the basis of the results we have made an appraisal of the model and solutions, and developed guidelines for further steps.

4 Solution model design

4.1 Investigative case studies

The Local Government Act [25] and Municipal Statutes [26] regulate the functioning of municipalities. The statutes define the organisational structure of the municipality in detail (division into sub-units of local self-government – local communities, specification of the committees and boards of the municipal council and the organizational structure of the municipal government). In deciding on investment needs, the public participates through a council of presidents of the local communities, which is the mayor's counselling body, expert staff of the municipal government represented by heads of the department and the mayor's collegium, and municipal counsellors who work through committees and the municipal council. If we add to this the forms of direct public decision making (people's assembly, referendum and public initiative), we find that the organizational structure of decision making in the municipality is sufficiently comprehensive.

However, for efficient progress in decision making this is not nearly enough. We would need a firm framework for decision making which would define the procedures, roles, inter-relations and limitations of the decision-making process. Unfortunately, we have not found this in any of the processed cases.

Due to this, preparation of the projects and preliminary appraisal of the participants' response (civil society, politicians and municipal government) progresses in an unsystematic and non-transparent manner. Investment projects are prepared within narrow political or expert circles, and the number of people who are well acquainted with all the parameters is low. Cooperation and balance between the participants is lacking, and the opinions and arguments of those who think in a different way are often ignored. This causes frequent situations where poorly prepared projects without prior discussion within the civil society and/or interaction between expert staff of the municipal government and/or counsellors reach the phase of final decision making by a poorly informed municipal council.

This situation is an ideal environment for asserting informal or formal power over arguments and the needs and desires of different-thinking people. The municipal government is aware of this problem, but has neither the knowledge nor the motivation to alter the situation. Numerous urgent and "urgent" projects, the first based on the actual needs of the participants and the other supported by informal (political) power, give the government little hope that this situation can be changed.

On the basis of the collected answers in our research, we can confirm that the environment in question is relatively immature in the area of decision making and that municipalities are confronting numerous difficulties:

- unorganized progress of opinion adjustment and deciding on preparing investment projects and their inclusion in the development plan or municipal budget
- powerful influence of the distribution of informal and formal powers in the municipality on the selection of possible investment projects and inclusion of approved projects in the budget
- difficulties in balancing opinions between professional fields, between political options and between or with the civil society
- the mayor's great direct influence on shaping expert opinion and political decisions
- absence of a comprehensive overview of the development of the local community
- poorly informed decision makers and public concerning the plans, realization and effects of the projects
- lack of qualifications and lack of motivation of the municipal government to improve the current situation
- unawareness and lack of motivation of the politicians to cooperate in solving issues
- a low level of public involvement in the preparation of decisions concerning solutions

Despite the relatively poor state of affairs, we were encouraged by the fact that leading staff of the municipal government understand the problem and are willing to invest the time and effort to find a solution.

4.2 Solution framework

The basic goal of our work is to improve the quality of decision making with the aid of a tool for decision support. Apart from ensuring the best possible participation of well-informed participants in the decision-making process, consensus is the central point of quality and success in decision making [23]. Implementing the principles of new public management in local self-government requires greater differentiation and responsibility in decision making, for which reason

the motivation and qualification of professional decision makers and especially municipal councillors is of key importance for successful development of the local community [27].

This is why the model, and with it the solution, had to ensure a decision-making framework in accordance with the conclusions expressed thus far, with a definition of the procedure and roles in decision making through the following steps:

- specification of the selection of alternatives,
- specification of the participants in decision making,
- specification of criteria and limitations,
- appraisal and choice,
- iteration (in case the solution is not satisfactory),
- documentation and archiving,

and a simple approach to enable appraisal without special knowledge in the area of decision-making or intuitive response, and by this to attract decision makers to cooperate and to appreciate the results of this method of decision making. Here we must point out the importance of knowledge exchange and constant adjustment, and improvement of the solution with regard to the capacity and motivation of the chosen environment, since this is the only correct way to achieve better decisions [28].

4.3 Solution model

The solution consists of the framework for decision making and web-based software for decision support within the following scope:

- specification of of alternatives
- specification of the participants in the decision making
- specification of attributes, criteria and limitations
- designing questionnaires
- appraisal
- analysis of the results and level of consensus
- selection
- export of the results by various cross-sections of the given structure.

The definition of a multiple-attribute decision problem encompasses the following:

- a set of attributes (parameters, factors, viewpoints, views, ranges) $C=\{c_1,\ldots,c_n\}$;
- a set of alternatives (possibilities, projects, scenarios, actions, goals, purposes) $A=\{a_1,\ldots,a_m\}$;
- specific information in each pair $(a_i, c_j); i \in \{1,\cdots,n\}, j \in \{1,\cdots,m\}$, ascertaining the relative importance of each attribute c_j – weight w_j;
- suitability r_{ij}, which is the decision maker's appraisal of the alternative a_i with regard to the attribute c_j;
- the merging function U, by which the appraisals of criteria r_{ij} for individual alternatives are aggregated into joint alternative appraisals;
- in group decision making, the given alternatives are appraised by the set of individuals $D = \{d_1,\ldots,d_k\}$.

The core of the solution is a three-dimensional group multi-attribute decision space:

- the basic structures are decision trees for each individual alternative, where the appraisals of the attributes c_{ij} (leaves) join into the appraisal of the alternative a_i by the aggregating function U_a,
- the individual alternatives a_i join into subsets of the set A, $V_k \in P(A)$ and the common appraisal for the subset of alternatives is given by the aggregating function U_v,
- the appraisals of individual appraisers d_l are joined into group appraisals for all the nodes and leaves of the decision tree by all the alternatives and variants $U_{G_p}; G_p \in P(D); p = 1,\cdots,|P(D)|$.

For the sake of simplicity, we have upgraded the decision-making model with fuzzy logic methods and implemented the appraisal with linguistic variables [29].

One thinks in terms of descriptive categories for which reason the appraisal by descriptive values demands much less mental effort. An appraisal method that demands less mental effort will be more precise than a method that demands greater mental capacity [30]. We can therefore claim that a descriptive appraisal is more precise than a numeric one. Additionally, a definition of the appraisal by linguistic variables is easier for the appraiser [31], [32]. These are undoubtedly sufficiently substantial arguments to support our approach.

In fuzzy logic theory we can find suitable solutions for joining values, based on the mapping of linguistic values into fuzzy numbers and the use of aggregation operators for fuzzy numbers in making the calculation.

The starting point is Zadeh's definition of linguistic variables [33, 34]:

A linguistic variable is defined by a quintuple $(\mathcal{H}, T(\mathcal{H}), U, G, \tilde{M})$ in which $\mathcal{H}$ is the name of the variable; $T(\mathcal{H})$ (or simply$T$) is the term set of $\mathcal{H}$, that is, the set of names for linguistic values $\mathcal{H}$, with each value being a fuzzy variable denoted generically by X and ranging over a universe of discourse U which is associated with the base variable u; G is a syntactic rule (which usually has the form of grammar) for generating names X of values of $\mathcal{H}$; and M is a semantic rule for associating each X with its meaning $\tilde{M}(X)$, which is a fuzzy subset of U. A particular X, that is, a name generated by G is called a term. A term consisting of a word or words which function as a unit (i.e. always occur together) is called an atomic term. A concatenation of components of a composite term is a subterm. An example of the term set (abbreviated by T instead of $T(x)$) is:

$$T = \begin{Bmatrix} Reject, Lowest, Very\ Low, Low, Middle, High, \\ Very\ High, Highest, Must\ Be \end{Bmatrix}.$$

The basic variable u is the assessed probability or degree of support, and comprises the values from the unit interval $u \in [0,1]$ [35]. The general rule that assigns a fuzzy set to the term X can be written as: $\tilde{M}(X) = \{(u, \mu_X(u)); u \in [0,1]\}$, which is for the term high: $\tilde{M}(High) = \{(u, \mu_{High}(u)); u \in [0,1]\}$.

What we have just written is in fact the definition of a fuzzy set:

Given a universe of discourse U, the fuzzy set $\tilde{A}$ in U is given by its membership function $\mu_{\tilde{A}}(u): U \rightarrow [0,1]$, in which the function $\mu_{\tilde{A}}(u)$ is interpreted as the degree of membership of u in the fuzzy set $\tilde{A}$. Clearly, the fuzzy set $\tilde{A}$ is fully determined by the set of ordered pairs $\tilde{A} = \{(u, \mu_{\tilde{A}}(u)); u \in U\}$.

Operations with fuzzy sets within given universe of discourse are operations with membership functions, which allows us relatively easy calculations with fuzzy sets defined with sufficiently simple membership functions. Expectations connected to this are fulfilled by Bonissone and Decker with the uniform scale for mapping linguistic conditional terms to fuzzy intervals [35], which are fuzzy subsets in the set of real numbers.

In this manner we have merged the advantages of using linguistic variables with the simplicity of mathematical operations over numeric variables. Thus we have avoided problematic and complex definitions of aggregation functions and unsurpassable limitations in respect of the branching out and size of the decision-making tree in aggregating linguistic values based on logical rules (rule-based aggregation). We have preserved the flexibility of modelling logical rules with a system of weights, which determine at each node the contribution of the child to the appraisal of the parent.

4.4 Software tool characteristics

The software itself consists of four modules (Figure 3). The module for implementing groups of cases is intended for defining the basic parameters of processing. With its help we can define the basic structure of the system and choose the methods of aggregation in tree structures. The number of questions is the number of the leaves of the appraisal tree; the content of the questions and the appraisal scale are determined with regard to the needs of each individual group of cases.

We first determine the decision-makers and the alternatives about which the decisions will be made for each separate case in the module for data collection, and if necessary we merge them into groups of decision-makers and subsets of projects – alternatives. The module also includes a user interface for appraisal. The job of calibrating the mapping is intended for registering the posture and mood of each individual decision-maker concerning the mapping of linguistic values into fuzzy numbers. With the help of the calculation module from the acquired appraisals, we fill in every point (leaf, node or root) of the given structure with three values: linguistic, fuzzy and real (Figure 4).

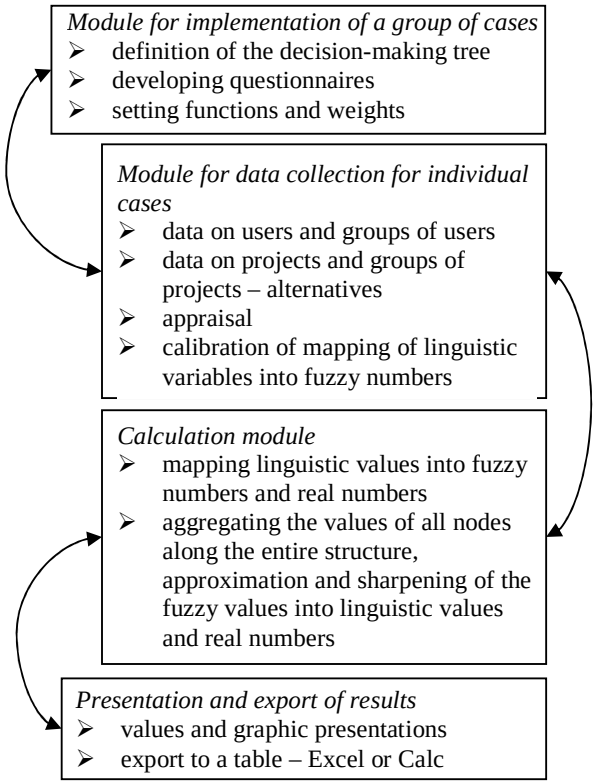

Figure 3: Software modules

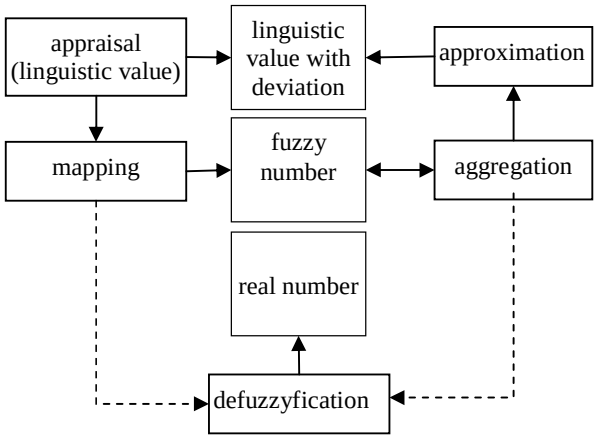

Figure 4: Functionality of the calculation module [36]

The module for presenting and exporting the results allows presentation and export along various cross-sections of the given structure.

The software tool is a web based application based on PHP 5 with database engine MySQL 5. It allows flexible settings of:

- structure of decision tree,
- appraisal scales (term sets) and mapping functions from linguistic values to fuzzy trapezoidal numbers,
- aggregation functions and distance measures.

With the intention of sharing ideas, knowledge and software itself after some tests the source code will be given in open source community.

The software tool is available at http://www.fu.uni-lj.si/bsc/. To obtain a user name and password, please contact the author of this article.

5 Case studies

5.1 Settings of the model

The research comprised three case studies in which we could, due to the similarity of the issues, use the same settings of the model, which include the decision-making tree, appraisal scale and functions of the mapping, aggregation, approximation and defuzzyfication.

Decision tree

Starting from the framework of deciding on capital investments in the public sector [36], legally prescribed definitions and the analysis of the method of decision making in local communities in Slovenia, we have determined the structure of the decision tree (Figure 5).

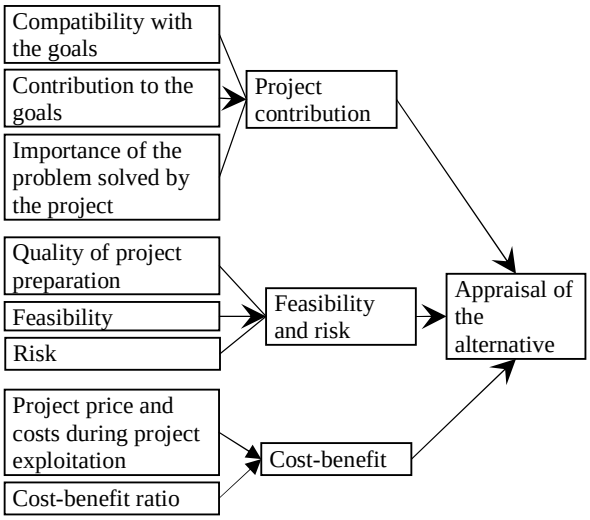

Figure 5: Decision tree of the module of multi-attribute appraisal of investment projects

Appraisal scale and mapping

The appraiser approves each attribute with a linguistic appraisal, which represents the degree of trust in the suitability of the project in terms of the given attribute (Table 3 and Figure 6).

Term	Fuzzy number	Label
Reject	0 0 0 0	L_1
Lowest	.01 .02 .01 .05	L_2
Very Low	.1 .18 .06 .05	L_3
Low	.22 .36 .05 .06	L_4
Medium	.41 .58 .09 .07	L_5
High	.63 .80 .05 .06	L_6
Very High	.78 .92 .06 .05	L_7
Highest	.98 .99 .05 .01	L_8
Must Be	1 1 0 0	L_9

Table 3: Linguistic values and equivalent fuzzy trapezoidal numbers

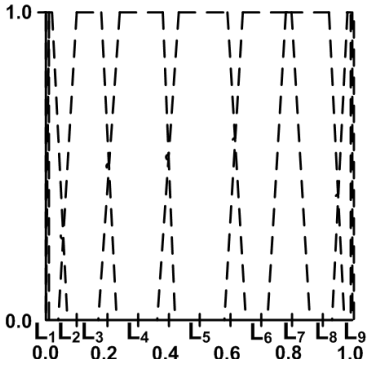

Figure 6: Graph of the mapping function

Arithmetic, aggregation function and distance measure

As a short break, have a look at a graph of a fuzzy number (more precisely, a fuzzy interval or trapezoidal fuzzy number, Figure 7):

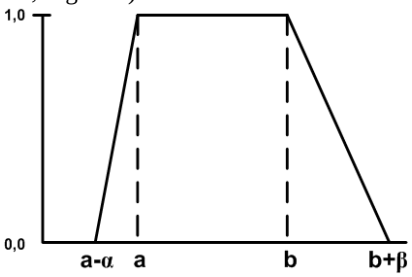

Figure 7: Graph of a fuzzy trapezoidal number

For fuzzy numbers, the computation necessary for algebraic operations are considerably simplified. The calculations within the decision-making framework are only done with positive fuzzy numbers $(\mu_{\tilde{A}}(x) = 0, \forall\, x < 0)$, and therefore only the arithmetic for a positive fuzzy number will be introduced (the definitions (Table 4) comprise the fuzzy numbers $\tilde{A} = (a, b, \alpha, \beta)\ and\ \tilde{B} = (c, d, \gamma, \delta)$):

Operation	Result
$\frac{1}{\tilde{A}}$	$\left(\frac{1}{b}, \frac{1}{a}, \frac{\beta}{b(b+\beta)}, \frac{\alpha}{a(a-\alpha)}\right)$
$\tilde{A} + \tilde{B}$	$(a + c, b + d, \alpha + \gamma, \beta + \delta)$
$\tilde{A} - \tilde{B}$	$(a - d, b - c, \alpha + \delta, \beta + \gamma)$
$\tilde{A} \cdot \tilde{B}$	$(ac, bd, a\gamma + c\alpha - \alpha\gamma, b\delta + d\beta - \beta\delta)$
$\frac{\tilde{A}}{\tilde{B}}$	$\left(\frac{a}{d}, \frac{b}{c}, \frac{a\delta + d\alpha}{d(d+\delta)}, \frac{b\gamma + c\beta}{c(c-\gamma)}\right)$

Table 4: Arithmetic operations for trapezoidal fuzzy numbers [35:230]

For the model to work, it will also need a aggregating operator. Following the simplicity principle, we have opted among the many operators for generalised operators of weighed mean expressed in the formula:

$$h_\alpha^w(a_1,\dots,a_n) = \left(\sum_{i=1}^n w_i a_i^\alpha\right)^{\frac{1}{\alpha}}, a_i \in [0,1], i \in \mathbb{N}_n, \alpha \in \mathbb{R}(\alpha \neq 0)$$

where for the components of the vector $\tilde{w} = (w_1,\dots,w_n)$ holds $\sum_{i=1}^n w_i = 1, w_i \geq 0 \forall i \in \mathbb{N}_n$. The vector $\vec{w}$ is termed the weighed vector, and its components w_i the weights. In the simplest version (equal weights $w_i = \frac{1}{n}$ and $\alpha = 1$) it is simply the arithmetic mean.

The final results of these calculations, trapezoidal fuzzy numbers, are not suitable for the presentation of results to appraisers. We must therefore map them back to linguistic values. We must find the linguistic value of which the fuzzy equivalent is the closest to the given trapezoidal fuzzy number.

For this purpose we need a metric of the fuzzy sets. The Tran-Duckstein distance takes into account the fuzziness of the fuzzy numbers and is confirmed in practice in an environmental-vulnerability assessment. We have, therefore, decided to choose it for our framework. For trapezoidal fuzzy numbers the general definition is simplified as $(f(\alpha) = \alpha, \tilde{A} = (a,b,\alpha,\beta), \tilde{B} = (c,d,\gamma,\delta))$:

$$\begin{aligned} D_T^2(\tilde{A},\tilde{B},\alpha) = {} & \left(\frac{a+b}{2} - \frac{c+d}{2}\right)^2 \\ & + \frac{1}{3}\left(\frac{a+b}{2} - \frac{c+d}{2}\right)[\beta - \alpha - \delta + \gamma] \\ & + \frac{2}{3}\left(\frac{b-a}{2}\right)^2 + \frac{1}{9}\left(\frac{b-a}{2}\right)[\beta + \alpha] \\ & + \frac{2}{3}\left(\frac{d-c}{2}\right)^2 + \frac{1}{9}\left(\frac{d-c}{2}\right)[\delta + \gamma] \\ & + \frac{1}{18}[\beta^2 + \alpha^2 + \delta^2 + \gamma^2] \\ & - \frac{1}{18}[\alpha\beta + \gamma\delta] \\ & + \frac{1}{12}[\beta\gamma + \alpha\delta + \beta\delta + \alpha\gamma] \end{aligned}$$

Figure 8: Tran-Duckstein distance for generalized left-right fuzzy numbers (GLRFN) [37:340].

Calculations

The linguistic values of the leaves are the direct result of the appraisal process, and the equivalent fuzzy numbers are the images of a simple mapping between them (Figure 6). The values of the parent nodes are calculated from the leaves towards the root of the tree as fuzzy arithmetic mean of fuzzy values of the children

$$\tilde{A}_{i,j} = \frac{1}{K_{i,j}}\sum_k \tilde{A}_{i+1,j,k}; i = I-1,\cdots,1; j = 1,\cdots,J_i; k = 1,\cdots,K_{i,j};$$

where I is the number of levels of the tree, i is the current level of the tree, J_i is the branching of the tree, j is the position of the node at the i-th level, $K_{i,j}$ is the number of children of the parent in question at the level i+1, and k is the position of the child of the parent in question.

The calculated trapezoidal fuzzy numbers $\tilde{A}$ are approximated back to linguistic values L so that the closest representative $\tilde{L}$ of the linguistic values L_i is found:

$$L_{app} = L: D_T(\tilde{A},\tilde{L},\alpha) = \min_i D_T(\tilde{A},\tilde{L}_i,\alpha); i = 1,\dots,n.$$

For higher granularity of the end results we introduced the approximation deviation. This is defined as the relative number of the difference in distance of the approximated fuzzy number and the fuzzy number image of the linguistic approximation and the difference between two adjacent linguistic values:

$$Dev = \begin{cases} -\dfrac{D_T(\tilde{A},\tilde{L}_{app},\alpha)}{D_T(\tilde{L}_{app-1},\tilde{L}_{app},\alpha)}, if\ D_T(\tilde{A},\tilde{0},\alpha) - D_T(\tilde{L}_{app},\tilde{0},\alpha) < 0 \\ \dfrac{D_T(\tilde{A},\tilde{L}_{app},\alpha)}{D_T(\tilde{L}_{app},\tilde{L}_{app+1},\alpha)}, if\ D_T(\tilde{A},\tilde{0},\alpha) - D_T(\tilde{L}_{app},\tilde{0},\alpha) \geq 0 \end{cases}$$

The approximation with the deviation is then labelled as:

$$\begin{aligned} & \leftarrow L_{app}, && if\ Dev < -0{,}25 \\ & L_{app}, && if -0{,}25 \leq Dev \leq 0{,}25 \\ & L_{app} \rightarrow, && if\ Dev < 0{,}25. \end{aligned}$$

This is our original solution, distinguished by its clarity, which allows precise and efficient presentation of the results to the decision makers [38].

The values of all the variables in the tree are merged into an appraisal of the variants V of the plan for the development programmes and into joint group appraisals of several appraisers G. This aggregation is also done with the calculation of the fuzzy arithmetic mean of the fuzzy numbers:

$$\tilde{A}_{i,j} = \frac{1}{|G|\cdot|V|}\sum_G\sum_V \tilde{A}_{i,j}; G \in P(D); V \in P(A); i = 1,\cdots I; j = 1,\cdots J_i;$$

for all subsets of the set of alternatives A and the set of appraisers D, for which it is reasonable in the given case.

5.2 Results

The case studies derive from the execution of the model in the chosen municipalities with the following steps:

- presenting solutions to the leaders (management director, heads of departments)
- preparation, adjustment and certification of the appraisal plan
- specification of the decision makers and groups
- specification of alternatives – projects
- presentation of the model and procedure to the appraisers
- executing the appraisal
- presentation of the results and discussions
- appraisal of the solution model

A detailed report about the progress and results of the case studies would unfortunately exceed the scope of this article, and we have therefore focused on the results, connected to the thesis of feasibility and usefulness of the model presented in the introduction. We verified the thesis by two methods:

- leading a discussion with the appraisers after presenting the results of the appraisal
- with a questionnaire about the progress and usefulness of the solution

The discussion revealed that the municipal government was well aware of the question of decision making. Municipal counsellors were less forthcoming.

They recognised the question of decision making, but they did not accept it as theirs.

The appraisers filled in the questionnaire in two cases, but only representatives of the municipal government responded. We have presented the appraisals in Appendix 1 at the end of this article.

The questionnaire referred to the following elements of the appraisal and attitude of the participants in the appraisal of the presented solutions:

1. Dissatisfaction with existing decision-making methods.
2. Willingness to cooperate in implementing new methods and approaches.
3. The method allows for easy expression of opinions about the projects and efficient cooperation in decision making.
4. The results will contribute to faster and better choices of projects.
5. The questionnaire is easily understood and allows for good expression of opinions.
6. I am content with the proposed decisions.
7. All the chosen projects are acceptable to me.
8. I wish to use the method in the future.

The results showed two very different facets of the municipalities. In the first municipality the collective satisfaction rating was High, an estimation given by all but one appraiser, who responded negatively (rejected) to the proposed choice of projects (Statement 7). Among other appraisers one could feel that all the proposed projects were not fully acceptable to them. Nevertheless, lower ratings than High were generally rare, so that the collective ratings, except two which refer to the results of the ratings themselves (Statements 6 and 7), were High. The highest estimated statement about willingness to cooperate in introducing new methods was Very High.

The results in the second municipality showed that the appraisers had different views of the presented method. The collective rating Medium was the result of three High and Medium ratings each and four ratings of Low. The collective rating Medium was given to almost all the individual statements, with the exception of willingness to cooperate in implementing new methods. Estimation of the method (Statements 3, 4 and 5) was in the opinion of most appraisers in accordance with their willingness to use the method in the future. Only one appraiser deviated from this pattern, who estimated willingness for future use of the presented method as High, but gave much lower ratings for the method.

The results of the case studies more or less confirm the thesis that the presented solution enables simple expression of the estimations in the municipalities, as well as balanced participation in decision making and ensures a final result which the decision makers and responsible persons accept as suitable. Of course, we cannot ignore the facts that necessitate an appropriate level of caution in confirming the thesis, since we have done the survey in only two municipalities. The appraisers in the second municipality were quite critical of the method, and we have yet to discover the reasons for such differences among appraisers in the same municipality. In any case we can claim that the method is suitable and enables, with correct preparation, swift execution of the appraisal procedure without major difficulties. Most of the participants in the three cases studies responded well to the method, and we thus can expect a positive response in the future.

We have attached the questionnaire and the appraisal results in both cases to this article.

6 Discussion

On the basis of the literature and studies of the situation and circumstances in the chosen environments we formulated certain principles and developed a decision-making model for investment projects in the public sector. We implemented a general model, which we have concluded must be simple for use in collecting data as well as in presentation of results, in a web-based software solution and tested it in three case studies. The study showed that we have fulfilled the requirement for simplicity and that the appraisers recognized the results as legitimate. We have confirmed the research statement within the given framework and have thus confirmed the approach and solution as a suitable tool for decision support in the public sector.

The formulated principles for the general decision-making model proved to be suitable and their realisation led to the successful execution of the appraisals. Our expectations concerning the approach to appraising and the quality of the results were met [30]. The chosen limitations described at the end of the first chapter prevented us from declaring the model a comprehensive solution to the question of decision making in the public sector. It will still have to be extended and generalized, which means that we must surpass the limitations and expand the model from the domain of local government to other domains, cover all the phases of decision making starting with recognition and definition of the problems and engage all participants involved in one way or another in the decision-making process.

The implemented solution showed that our approach is suitable and that we can ensure good-quality decision making. The approach with linguistic variables simplified the system, which is especially important from the point of view of presenting the results. A two-stage presentation of the results (with deviations and only with basic values of linguistic variables) allows overview on two levels of resolution (23 values and 7 values). Due to the limitations of the software, we could only test one setting of the model. This limitation will be rectified in the future version of the solution.

We have confirmed the research thesis in a given context, which is only the first step to comprehensive confirmation of the model as a good solution for the problem in question. Our expectations that the solution would make the decision process easier have been met. Unfortunately we were unable to attract a larger number of municipal counsellors. We therefore foresee additional presentations to the municipal council for the next appraisal, by which we will ensure the suitable participation of municipal counsellors. In addition, the expansion of the model to all phases of decision-making

and to other domains awaits us in the future. We will devote special attention to the question of reaching a consensus and the quality of decision-making, and develop a method for assessment of the quality of decision-making in the public sector, which will enable us to prepare a comprehensive estimation of the circumstances of decision-making in given environments and the usefulness and quality of the presented decision support model.

7 Conclusion

Using fuzzy logic, we extended the selection of cases for the use of these methods. By mapping linguistic variables into fuzzy numbers we avoided the limitations connected to indirect aggregation of linguistic values [39], [40] and to the breadth of the tree structure stemming from the approaches based on defining the values of the parents with logical expressions for all possible values of the children [41]. In this manner we managed aggregation without limiting the number of leaves, levels, alternatives and appraisers. We also enabled the formation of subsets of alternatives – variants and subsets of appraiser groups. The calculations are simple enough so that the applied system is not demanding in terms of computer capacity. Of course new dangers and limitations accompany new approaches. Since there are no directly comparable systems we will need more time and effort to confirm the results. This is similar to mapping the values of linguistic variables into fuzzy numbers. There are only a few cases in use, and it will require additional time and effort for further development in this area.

Implementation of the model in an environment of local self-government is a contribution to the development of decision support in a local environment and thus a contribution to the development of e-democracy in the matter of public co-deciding. The possibility of formation of variants (subsets of alternatives) is especially important here, as well as the groups (subsets of appraisers) that can profitably be used in seeking a consensus. Activation of the public in recognising the problems of decision-making and engaging them in seeking a consensus can considerably contribute to better understanding between all the participants in the local community [24].

The proposed solution is suitable for any system in which we wish to aggregate and compare values of various types of variables in organizations and systems with a hierarchical structure of goals and indicators, since the new version will accept all three types of entry data (numeric, linguistic and fuzzy numbers) and convert them into the other two forms according to rules prescribed for implementation of the solution. This will enable not only comparison of various types of variables, but also a flexible adaptation of the conversions, which will additionally enable comparability between the same types of variables from various definition areas.

Our article is a small piece in a mosaic of activity and research in the area of e-democracy [15]. It upgrades electronic election systems in which the voters choose between confirming or rejecting an individual alternative and enables estimation of the level of agreement among the participants. Based on fuzzy logic, which facilitates the comparability of various indicators, the solution also becomes a tool for monitoring success and outcomes of the functioning of the public sector.

8 References

[1] Plaper, G., Piletič, M., Bohanec, M., Rajkovič, V., Urh, B. (1999): "Hierarchical multi-attribute models for decision support in the management of diabetic foot syndrome". In: Kokol, P., Zupan, B., Stare, J., Premik, M., Engelbrecht, R.: *Medical Informatics Europe '99, Studies in Health Technology and Informatics*, Vol. 68. Amsterdam [etc.]: IOS Press; Tokyo: Ohmsha, pp. 970–972.

[2] Hämäläinen, R., Kettunen, E., Marttunen, M., Ehtamo, H. (2001): "Evaluating a Framework for Multi-Stakeholder Decision Support in Water Resources Management", *Group Decision and Negotiation*, Vol. 10, No. 4, pp. 331-353.

[3] Oñate, E., Piazzese, J. (2003) *Decision support system for risk assessment and management of floods*. International Center for Numerical Methods in Engineering (CIMNE) Gran Capitán, s/n, Campus Norte UPC, 08034 Barcelona, Spain. <http://www.eu-lat.org/eenviron/Onate.pdf> (accessed July, 2006).

[4] Benkovič, J., Bohanec, B., Rajkovič, V., Vrtačnik, M. (1998): "Knowledge-based evaluation of higher education institutions", *Preprints of the 6th IFAC-Symposium "Automated systems based on human skill"*, Kranjska Gora, Slovenia, Pergamon Press, pp. 157–160.

[5] Di Mauro, C., Nordvik, J.P., Lucia, A.C. (2002): *Multi-criteria decision support system and Data Warehouse for designing and monitoring sustainable industrial strategies – an Italian case study*. European Commission – Directorate General JRC – Joint Research Centre, Ispra Institute for the Protection and Security of the Citizen (IPSC) Technological and Economic Risk Management Unit I-21020 Ispra (VA), Italy <http://www.iemss.org/iemss2002/proceedings/pdf/volume%20uno/434_dimauro.pdf> (accessed July, 2006).

[6] Bouras, C., Katris, N., Triantafillou, V. (2003): "An electronic voting service to support decision-making in local government", *Telematics and Informatics*, Vol. 20, Issue 3, pp. 255–274.

[7] Grönlund, Å. (2003): "e-democracy: in search of tools and methods for effective participation", *Journal of Multi-Criteria Decision Analysis*, Vol. 12, No. 2-3, pp. 93-100.

[8] Kersten, G.E. (2003): "e-democracy and participatory decision processes: lessons from e-negotiation experiments", *Journal of Multi-Criteria Decision Analysis*, Vol. 12, No. 2-3, pp. 127-143.

[9] Walker, W.E. (2000): "Policy analysis: a systematic approach to supporting policymaking in the public

sector", *Journal of Multi-Criteria Decision Analysis*, Vol. 9, No. 1-3, pp. 11-27.

[10] Gammack, J., Barker, M. (2003): *E-democracy and public participation: a global overview of policy and activity*. School of Management, Griffith University Queensland.

[11] Bots, P.W.G., Lootsma, F.A. (2000): "Decision support in the public sector", *Journal of Multi-Criteria Decision Analysis*, Vol. 9, No. 1-3, pp. 1-6.

[12] Power, D.J. (1999): "Decision Support Systems Glossary", *DSS Resources*, World Wide Web, <http://DSSResources.COM/glossary/> (accessed July, 2006)).

[13] Bonczek, R.H., Holsapple, C.W., Whinston, A.B. (1981): *Foundations of Decision Support Systems*. Academic Press, New York.

[14] Turban, E. (1995): *Decision Support and Expert Systems: Management Support Systems*. 4th Edition, Macmillan Publishing Company, New York.

[15] Coleman, S., Norris, D.F.: (2004); "A New Agenda for e-Democracy", *Forum Discussion Paper No. 4*, Oxford Internet Institute. www.oii.ox.ac.uk/resources/publications/FD4.pdf> (accessed July, 2006).

[16] Rosen, T. (2001): *E-democracy in practice, Swedish experiences of a new political tool*. <http://www.sociedadinformacion.unam.mx/repositorio/documentos/E-democracySwedish.pdf> (accessed July, 2006).

[17] Molin, B., Mansson, L., Stromberg, L. (1975): *Offentlig forvaltning*, (Public Administration). Bonniers.

[18] Papamichail, K.N., Robertson, I. (2003): "Supporting societal decision-making: a process perspective", *Journal of Multi-Criteria Decision Analysis*, Vol. 12, No. 2-3, pp. 203-212.

[19] Perri, 6 (2004): *Re-Wiring Decision-making for Local Government, Not Local Administration*. <http://www.egovmonitor.com/features/perri6.html > (accessed July, 2006).

[20] Mittuniversitetet (2006): *Decision analytic support tools in e-government*. <http://www.miun.se/mhtemplates/MHPage____23601.aspx> (accessed July, 2006).

[21] Podger, A. (2002): *Improving Government decision-making.* <http://www.apsc.gov.au/media/podger310502.htm > (accessed July, 2006).

[22] Dror, Y. (1997): "Strengthening government capacity for policy development", *International Journal of Technical Cooperation*, Vol. 3, No. 1, pp. 1–15.

[23] Harter, P.J. (1997): "Fear of commitment: an affliction of adolescents", *Duke Law Journal*, No. 46, pp. 1389–1428.

[24] Innes, J.E., Booher, D.E., (1999): "Consensus Building and Complex Adaptive Systems – A Framework for Evaluating Collaborative Planning". *Journal of the American Planning Association*, Vol. 65, No. 4, pp. 412–423.

[25] Local Government Act (officially approved text) (2005) /ZLS-UPB1/ (Ur.l. RS, No. 100/2005).

[26] Municipality of Koper (2006): *Statute of the Municipality of Koper* (unofficially approved text). <http://www.koper.si/dokument.aspx?id=11415> (accessed July, 2006).

[27] Hansen, K. (2001): "Local Councillors: Between Local Government and Local Governance", *Public Administration*, Vol. 79, No.1, 2001 (105-123).

[28] Roberto, M.A. (2005): "Why making decisions the right way is more important than making the right decisions", *Ivey Business Journal Online*, Ivey Management Services, Richard Ivey School of Business. <http://www.iveybusinessjournal.com/view_article.asp?intArticle_ID=578> (accessed July, 2006).

[29] Zimmerman, H.J.: *Fuzzy Set Theory and Its Applications*, 4th ed. Kluver Academic Publishers, Boston/Dordrecht/London, 2001, p. 514.

[30] Zimmer, A.C. (1986): "What Uncertainty Judgement Can Tell About the Underlying Subjective Probabilities". In: *Uncertainty in Artificial Intelligence. Machine Intelligence and Pattern Recognition*, Kanal, L.N., Lemmer, J.F., Amsterdam, Elsevier Science Publisher B.V., Vol. 4, pp. 249–258.

[31] Chin, W.C., Ramachandran, V. (2000): "Fuzzy Linguistic Decision Analysis for Web Server System Future Planning", *International Journal of the Computer, the Internet and Management*, Vol. 8, No. 3.

[32] Herrera, F., Herrera-Viedma, E., Lopez, E. (1996): "On the Linguistic Approach in Multi-Person Decision-Making", *International Conference on Intelligent Technologies in Human-Related Sciences, ITHURS'96*. Leon (Spain), pp. 205–213.

[33] Zadeh, L. A. (1975): "The concept of a linguistic variable and its application to approximate reasoning – I", *Information Sciences*, No. 8, pp.199–249.

[34] Zadeh, L. A. (1973): *The concept of lingustic variable and its application to approximate reasoning*. Memorandum ERL-M 411, Berkeley, October 1973.

[35] Bonissone, P.P., Decker, K.S. (1986): "Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity". In: *Uncertainty in Artificial Intelligence. Machine Intelligence and Pattern Recognition*, Kanal, L.N. and Lemmer, J.F., Amsterdam, Elsevier Science Publisher B.V., Vol. 4, pp. 217–247.

[36] GAO/AIMD-98-110 (1998): *Leading Practices in Capital Decision-Making*, U.S. General Accounting Office, Washington, http://www.gao.gov/special.pubs/ai99032.pdf (accessed, June 2006).

[37] Tran, L., Duckstein, L. (2002): "Comparison of fuzzy numbers using a fuzzy distance measure", *Fuzzy Sets and Systems*, No. 130, pp. 331–341.

[38] Benčina, J. (2004): *Optimisation of the selection of investment projects in the public sector by methods of mathematical programming and fuzzy logic*. Doctorial dissertation. Ljubljana: Faculty of Economics.

[39] Herrera, F., Herrera-Viedma, E. (1997): "Aggregation operators for linguistic weighted information", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 27, pp. 646–656.

[40] Herrera, F., Martinez, L. (1999): *A 2-tuple Fuzzy Linguistic Representation Model for Computing with Words*. Dept. Computer Sciences and A.I., Granada University, Technical Report#DECSAI-990102. <http://citeseer.nj.nec.com/herrera99tuple.html> (accessed May, 2006).

[41] Von Altrock, C. (1997): *Fuzzy Logic and NeuroFuzzy Applications in Business and Finance*. Prentice Hall, Upper Saddle River, New Jersey.

9 Appendix

The data captured in two case studies and the appraisal results are presented in two following tables. The first information in appendix is the legend of tables which contains the labels of input data (variables - questions, Table 5) and the labels of linguistic values (results, answers, Table 6).

c_1	Dissatisfaction with existing decision-making methods.
c_2	Willingness to cooperate in implementing new methods and approaches.
c_3	The method allows for easy expression of opinions about the projects and efficient - cooperation in decision-making.
c_4	The results will contribute to faster and better choices of projects.
c_5	The questionnaire is easily understood and allows for good expression of opinions.
c_6	I am content with the proposed decisions.
c_7	All the chosen projects are acceptable to me.
c_8	I wish to use the method in the future.

Table 5: Labels of input variables

Label	Term
L_1	Reject
L_2	Lowest
L_3	Very low
L_4	Low
L_5	Medium
L_6	High
L_7	Very high
L_8	Highest
L_9	Must be

Table 6: Labels of linguistic values

The data of the case studies are given in the rows of the tables, while the results calculated are presented in the last columns (overall appraisal of the model given by a particular appraiser) and in the last rows (collective appraisal of an attribute), thus the collective overall appraisal occupies the outmost right cells in the last rows (Table 7 and Table 8).

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	**Σ**
1	L8	L5	L6	L6	L7	L6	L1	L6	L6 ←
2	L8	L8	L7	L6	L6	L6	L5	L6	→ L6
3	L5	L8	L6	L6	L6	L5	L5	L7	L6
4	L7	L7	L6	L6	L5	L5	L5	L7	L6
5	L3	L6	L5	L5	L7	L7	L5	L7	→ L5
6	L7	L8	L6	L6	L7	L6	L6	L7	L7 ←
Σ	L6	L7	L6	L6	L6	L6 ←	L5 ←	L7 ←	L6

Table 7: Data and results of the first case study

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	Σ
1	L5	L5	L6	L5	L5	L5	L5	L5	L5
2	L1	L7	L3	L7	L5	L5	L5	L5	L4 ←
3	L5	L8	L6	L5	L5	L5	L5	L7	→ L5
4	L8	L5	L6	L6	L6	L5	L5	L6	L6 ←
5	L5	L5	L4	L4	L5	L5	L5	L4	→ L4
6	L5	L7	L5	L5	L6	L5	L5	L5	L5
7	L2	L6	L4	L2	L5	L5	L5	L6	→ L4
8	L7	L7	L1	L1	L1	L5	L5	L2	L4
9	L7	L7	L8	L7	L7	L5	L5	L7	L6
10	L7	L6	L7	L7	L7	L5	L5	L7	L6
Σ	L5	L6	→ L5	L5	L5	L5	L5	L5	L5

Table 8: Data and results of the second case study

Dynamic Distribution of Java Applications

Gita Alagbhand and David Gnabasik
University of Colorado at Denver and Health Sciences Center
Department of Computer Science and Engineering
Campus Box 109, P.O. Box 173364, Denver CO 80217-3364, USA
E-mail: Gita.Alaghband@cudenver.edu P:303-556-2940 F:303-556-8369
E-mail: DavidGnabasik@comcast.net P:303-994-2740 F:303-617-7877

Keywords: Java, component streams, class loader, mobile devices

Received: April 10, 2007

This paper describes a streaming mechanism that distributes Java class bytecode streams to a client from a database server. The class server uses a 1st-order Markov probability model to effectively predict the client's next class request. Experimental results demonstrate that class prediction can deliver a class cache hit ratio of up to 54% using a modest cache size of 64kb on the client, whereas a 16kb cache delivers a hit ratio of 37%. The model is designed to mitigate the distribution and deployment problems of monolithic application software and is useful for applications running on resource-constrained, mobile computing devices.

Povzetek: Članek opisuje postopek dinamičnega porazdeljevanja aplikacij v Javi.

1 Introduction and problem description

Dynamic component streams can address several client software distribution and deployment issues, including the automated update of applications from a centralized software repository, as well as the delivery of application streams to resource-constrained mobile devices. We submit that the process of application deployment can take advantage of the dynamic linking and class loading [3] mechanisms in Java compilers to support a distributed component model that does not require the streaming of large monolithic applications to these devices. Java class hierarchies are stored as Java bytecode streams into database rows by a class author. A streaming mechanism then transmits a virtual application to a client's process space from a class server. Our class server uses a 1st-order Markov transition probability model to effectively predict the client's next class request based upon the statistical analysis of historical application runs. This permits for the effective overlapping of client-server communication with client processing. The Java linking model and class file format are suited for managing this problem since the class file is analyzed and stored as the unit of compilation [2]. We manage and optimize system performance by:

- decomposing and storing class hierarchies into a database;
- defining an effective asynchronous streaming component model;
- retrieving the most probable class subgraphs to be activated next;
- effectively managing limited available client memory.

This paper is organized as follows. In section 2, we describe each architectural component of the system and the underlying theory. Section 3 outlines its design and implementation. Section 4 describes our experimental measurements. Section 5 discusses related work, and section 6 presents our conclusions and summarizes the contributions of this paper.

2 System architecture

2.1 Class authoring and client access roles

The role of the class author is to correctly compile and store Java components or entire applications as binary streams of Java bytecode into a database. By doing so, the class authoring interface encapsulates much of the complexity of class relations and their distribution. The primary function of the interface is to scan each class for referenced classes and to store this list into the database as well. The authoring interface also manages the following information:

- appropriate class / application authorization and security measures;
- any class digital signatures or certificates;
- what foundation set of Java classes are required locally on the client;
- the set of client deployment preferences that indicate how and when the classes in a particular application trace should be updated.

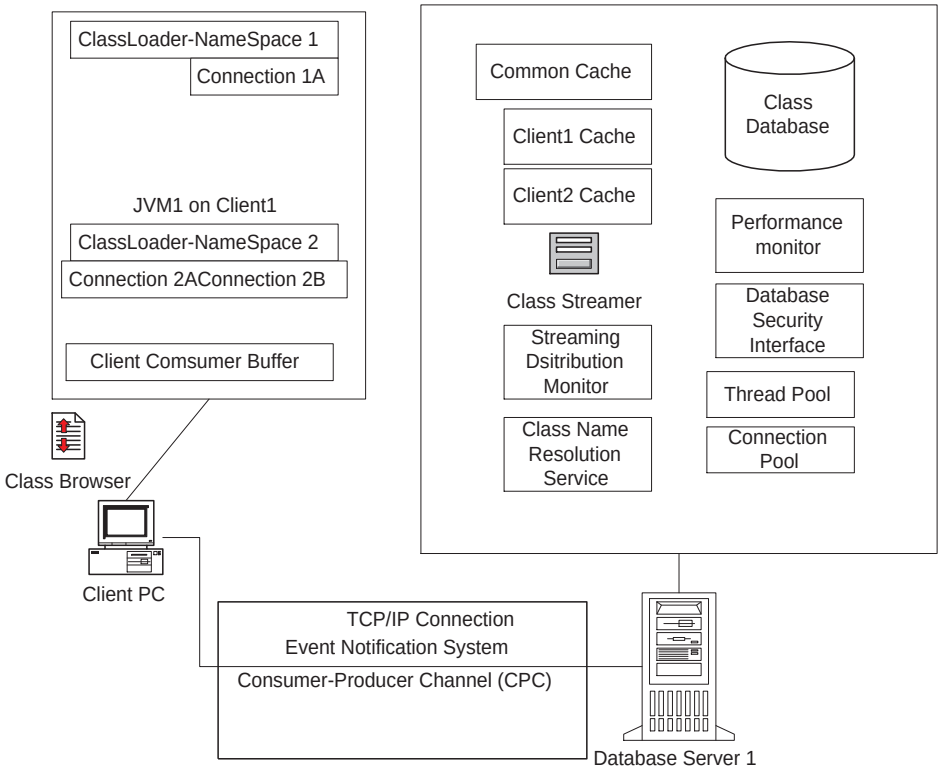

Figure 1: Overall architectural schema.

Clients use a class browser or the Java Naming Directory Interface (JNDI) to request a particular application through a well-known URL address. The client request establishes a consumer-producer channel with the class server. Once the client is authorized, the server retrieves and transmits a custom class loader to the client who loads it into his Java virtual machine (JVM). It is this custom class loader which initially loads an application's main class and which makes any further class requests on behalf of the application. No other changes to existing Java classes are necessary. Figure 1 diagrams our system architecture.

2.2 The class loading and linking process

Class loaders are responsible for importing binary data that define a running program's classes and interfaces. The JVM's flexible class loader architecture provides for dynamically extended applications. Linking a class stream into a JVM's run time state is divided into three steps: verification, preparation, and resolution [4]. Verification ensures that the stream is properly formed, preparation allocates memory needed by the stream, and resolution transforms symbolic class references into direct machine references for the sake of run time speed and efficiency. During the class activation process, the JVM must give the impression that it loads classes as late as possible, a process called *lazy activation* [3]. This on-demand activation process permits the transmission of individual Java classes to a client by a class server.

2.3 Class transmission mechanism

The server retrieves a requested set of classes, or class subgraphs, according to a class transition probability model. This model, which forms the basis for prediction, is described in section 1.4. Based on this model, the class server attempts to prefetch and transmit the next set of expected classes while the client is busy executing. Even though some classes are prefetched and delivered that may not be loaded by the client into his JVM, this network traffic occurs while the client is busy processing the current class, in effect, overlapping operations. Any successful access to a prefetched class is a measurable performance gain. Overall system performance is maintained by an efficient class prefetching algorithm, client and server caching, a simple database schema, and judicious threading.

When the client attempts to load another class, the custom class loader searches for classes in the following order: the client JVM, the client's class cache (CCC), the client's standard Java class libraries, and the class server. If the class is found in the client's cache, it is decompressed by the JCL or built-in classes of the JVM, loaded into the running JVM, and removed from the cache. If the requested class is not in the CCC, the JCL requests the class from the class server. The class is retrieved from the database along with its previously parsed list of symbolically referenced classes, which was generated when the class was inserted into the database by the class author. The list of classes is compressed and streamed to the CCC in the order in which a JVM internally resolves all of its referenced

classes. The amount of data transmitted is limited by the size of the client's cache, which is managed by the client's JCL.

Classes from the Java libraries are loaded by the native, primordial class loader. Classes in the class cache are loaded by the custom object class loader. The different class loaders are related by a policy called *delegation*. For each class it loads, the JVM keeps track of which class loader, whether primordial or object, loaded the class [4]. Our model uses a separately threaded class loader derived from the SecureClassLoader Java class to load all subsequent classes. SecureClassLoader extends ClassLoader with additional support for defining classes with an associated code source and permissions. The protection domain in the model is the class server database itself, which has permissions assigned to it as a code source. It also uses the *parent-delegation model* introduced in Java 1.2 to determine which class loader actually loads a given class. The rule is that the JVM uses the same class loader that loaded the referencing or calling class to load the referenced or called class. The custom class loader is the first class the client receives in an application stream.

A Java class is initialized in a certain order. In stage I, all the class's superclasses are recursively initialized, then the class itself, and then its inner classes, followed by the class's static variables. Once the class is initialized for active use, then the class's private variables are initialized followed by all classes referenced in any constructors. These references are necessary to initialize a class; therefore their invocation probability is 1. Class initialization is described in detail in the next section. Stage I classes are always transmitted to the client in their own package.

Stage II classes include any method argument classes, method return types, and all classes referenced in any method. These referenced classes are conditionally invoked by an application. Stage II classes are retrieved from the database, ordered by invocation probability, and transmitted to the client in a separate package. This strategy allows the class server to act as a predictive look-ahead class feeder for the client. Figure 2 diagrams these separate stages.

2.4 Transition probability model and class invocation

The delivery effectiveness ratio H is defined as the number of requests satisfied by a particular cache divided by the total number of requests during an entire application trace. Given that C_C is the number of class requests satisfied by the client cache, S_C is the number of class requests that had to be satisfied by the server, and C_R is the total number of class requests made by the client's class loader, such that $C_C + S_C = C_R$, then $H_C = C_C/C_R$. The effectiveness of prediction is then the ratio of class requests satisfied by the CCC to requests satisfied by the class server. The modeling question becomes: How can the server more effectively predict which classes to stream to the client?

The proposed model generates a set of invocation probabilities, one for each context $class_j$ that invokes a particular $class_i$. Given a class, the model enumerates all the invocation probabilities of the classes that it symbolically references. Since the probability of transmitting a specific class depends upon the class that calls it, the model establishes a conditional probability vector of $P(class_i|class_j) = P(class_i \cap class_j)/P(class_j)$ probability values, where $class_j$ is the context class for $class_i$. Each $class_i$ instance may be invoked by different classes; hence each class has multiple $class_j$ context classes. Each class also maintains a total count of class invocations per context class. These class values are accumulated during the execution of an application. Class invocation prediction implies distributing both the globally most frequently accessed classes as well as the class most likely to be invoked next at any point in the program. Good prediction means transmitting those classes that are most likely to be consumed by the client. Given a particular context class, stage I classes are always transmitted, but stage II classes are transmitted according to their invocation probabilities.

To illustrate, Table 1 lists the set of classes that are invoked by the class TelnetDriver over 20 program runs of a Telnet application. Stage I classes are DialerAccess, Plugin, Common and ReturnFocusRequest because TelnetDriver always invokes them. The invocation probabilities for the other classes are lower because they were conditionally invoked depending upon the flow of execution. Clearly, TelnetDriver is a stage I class because it was also invoked the same number of times the application was run. Each invocation probability is calculated as the number of invocations divided by the number of program runs (e.g., 20). For example, P(invoking Class OnlineStatusListener within Telnet) $= 2/20 = 0.1$. The 103 total classes and interfaces in this Telnet application range in size from 124 to 26158 bytes, with an average size of 2083 bytes for all classes.

2.5 Markov chain modeling

A 1st-order, finite state, probabilistic Markov model is proposed because a Markov model is suitable to the local dependencies embedded in Java class invocation structure, and the finite-state machine model accurately reflects the necessary and unique set of state transitions that occurs in an application trace. Since classes are conditionally invoked in an application trace, the model is able to characterize their invocations by probability values. The Markov model has an additional advantage in that it reveals an application's locality of reference, since "the performance of demand-driven caching depends on the locality of reference exhibited by the stream of requests made to the cache" [13](abstract). Vanichpun et al [13] further claim that "the two main contributors to locality of reference are temporal correlations in the streams of requests and the popularity distribution of requested objects", which are both accounted for by the model. The "temporal correlations" cor-

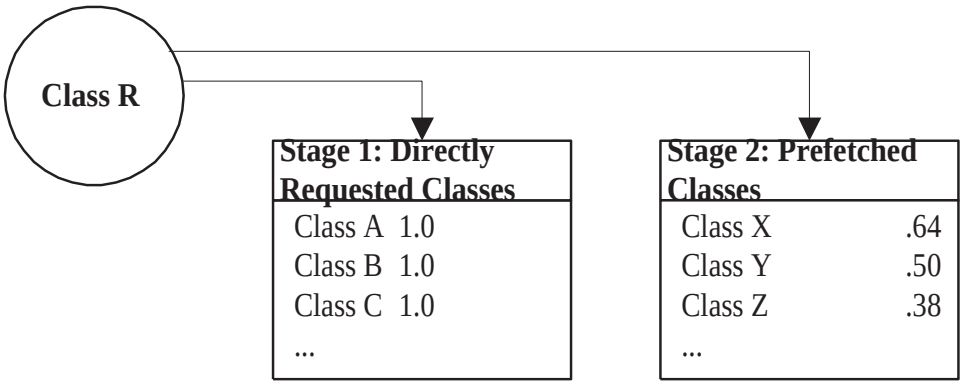

Figure 2: Invocation probability stages.

Table 1: Conditional probability vector (CPV) for class TelnetDriver.

Invoked Class Name	Invocation Probability	Number of Invocations
DialerAccess	1.0	20
Plugin	1.0	20
Common	1.0	20
OnlineStatusListener	.10	2
FocusStatusListener	.10	2
SocketRequest	.80	16
VisualTransferPlugin	.66	13
ReturnFocusRequest	1.0	20

respond to the sequential flow of class invocations and the "popularity distribution" corresponds directly to the probability of invocation for a particular class.

Markov chains can dynamically model these class invocation patterns found in applications. A discrete Markov chain model is defined as the tuple $< S, A, \lambda >$ where S corresponds to the state space, A_{ij} is a matrix representing the invocation probabilities from one state to another, and λ is the initial probability distribution of the states in $S(0)$. In our model, S represents every possible application trace, A_{ij} represents the invocation probabilities for each class, and λ is the initial class invocation distribution retrieved from the database. If the vector s_c denotes the probability vector for all the subsequent possible class invocations during execution of a specific class c, where $c \in S$, then the overall set of expected transition state values for class c is $\hat{s}_c(j) = s_c(i)A_{ij}$. The A matrix is recalculated during each application trace and stored in the database. The class request mechanism simply selects the largest probability values from the vector $\hat{s}_c$ either until a threshold probability value is reached or their cumulative class sizes are greater than the client's class cache size. The conditional probability vector for a class directly supports the calculation of $\hat{s}_c$.

For example, table 2 calculates $\hat{s}_c(j$ =TelnetDriver$\geq$ .75) for $s_c(i$ =TelnetDriver= .50). It is these combined stage I and II classes that are actually delivered to the client because the invocation probability for $\hat{s}_c(j)$ is $\geq .75$. $\hat{s}_c(i = .50)$ is the probability that the application would invoke TelnetDriver in the first place.

Given the semi-hierarchical structure of nearly all applications, this Markov chain model is not *irreducible*, that it is possible to get to any state from any other state. Some states would be *transient*, such that given a starting state there is a non-zero probability that the application would never return to that particular state. Most states would instead be *recurrent* or *persistent*, that at some point in time the application would return to that state. Most application states also avoid the Markov property of *absorbing*, where it is impossible to leave a particular state. Since our Markov chain model is not irreducible, there is no guarantee that the model provides a steady-state or equilibrium distribution. In practice, however, the probability transition matrix quickly approached a set of relatively stable values.

2.6 Threaded queue representation

Although we do not present a detailed queuing model in this paper, it is important to note that our system can also be represented by *queuing circuits* because of Java's stringent class loading requirements and the way the system is architected around several queue components. Relevant queuing centers are the *client request queue* (CRQ) and the more complicated *class server queue* (CSQ), each of which are distinguished by their own average service times. See sections 2.4 and 2.5 for complete descriptions of the queuing mechanism. The CSQ is accessed by two threads per client: one thread for handling class requests to the server and the

Table 2: Expected transition state values s_c for class TelnetDriver.

Class Transition	Transition Prob.
DialerAccess	$1.0 \times .50 = 0.50$
Plugin	$1.0 \times .50 = 0.50$
Common	$1.0 \times .50 = 0.50$
ReturnFocusRequest	$1.0 \times .50 = 0.50$
SocketRequest	$.80 \times .50 = 0.40$

second thread for receiving predicted, prefetched classes. The third client thread accesses locally referenced classes. The same queuing components are managed by each thread including a common cache, a client delivery cache, and a database fetch component. Figure 3 diagrams the fundamental queuing centers of the system. Operating system and database-specific queues are not included for the sake of simplicity.

Following Gunther [9], our system is characterized by a first-in, first-out (FIFO) service policy which assumes an exponential service time distribution for both queues, the custom and native class loaders. Under FIFO, the service time can only depend upon the queue length at each queuing center. The system is considered open because the server queuing center can access a possibly infinite number of elements or classes, even though only a limited and indeterminate number of classes are actually invoked in an application trace. Highleyman [11] argues that an open queuing center model is a reasonably accurate approximation if N is at least 10 times larger than the average queue length, which is indeed the case. The two queuing streams are also *separable* and *mergeable*. The streams are separable because it is possible to evaluate the performance measurements of the complete set of queuing centers as though each of the centers were evaluated separately in isolation. The streams are mergeable because the performance of the entire system is then built by combining the separate solutions. These queuing characteristics establish two different servers in an open, overlapping configuration with an infinite population, which we describe as a **M/M/2/FIFO** delay queue model.

3 Design and implementation

3.1 Optimizing overall system performance

Our system architecture attempts to minimize overall network traffic by delivering only the immediately needed classes of an application trace, since the model claims that overall delivery throughput is increased by incorporating and limiting the Markov probability model to the 1st-order when predicting the client's next class request. Network delivery time is reduced by writing the class subgraph into a single *Java ARchive* (JAR) or ZIP package for transmission.

The major problems to resolve in order to maintain adequate system performance are (along with

– decomposing and storing class hierarchies into a database where each Java class file is stored as precompiled Java bytecode in a database row;

– defining an effective asynchronous streaming component model that overlaps execution with network communication as implemented in the CSQ and CRQ queueing centers;

– retrieving the most probable class subgraphs to be activated next from the class server which uses a 1st-order Markov probability model to effectively predict the client's next class request to reduce invocation latency;

– caching the most frequently invoked bytecodes at the client while prefetching and caching class bytescodes at the server to reduce database access delay;

– managing class elements as discrete database rows, which allows for database access optimization;

– avoiding slow file system accesses and disk paging by storing the class bytecodes in database rows;

– effectively managing limited available client memory by allocating a minimum client cache;

– the use of judicious I/O threading.

Other design goals include:

– not modifying the client's virtual machine executable;

– not allowing the class server to manage client state beyond minimal client authentication, authorization and initialization, which reduces the overall complexity of the application;

– expending time and effort at the class-producing or -authoring stage instead of the class-consuming stage.

When non-duplicated classes are inserted into the CCC, they are associated with the class that invoked them. If a class is extracted from the CCC for activation, the discard policy marks the list of explicitly associated classes for discard, too. These classes are either sequentially pushed out of the CCC to make room for new classes or they are extracted for activation. Note that any inherited or parent classes or interfaces have themselves already been extracted from the CCC since a class's parents must be completely activated before the class itself. Since only the

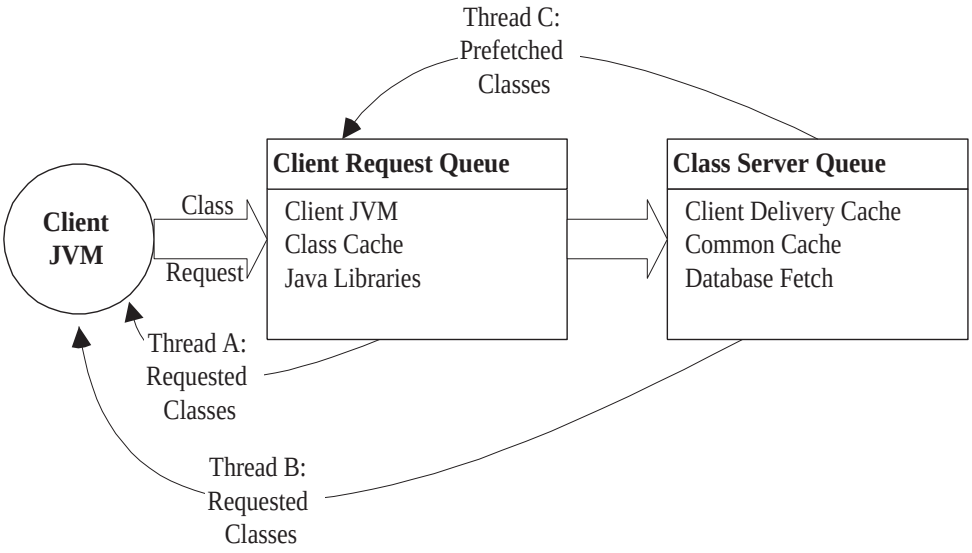

Figure 3: Queuing request centers.

client removes classes from the CCC, there is no need for the client to inform the server that the client has unloaded a class. The client JCL is responsible for cache overflows.

Storing Java bytecodes instead of source text offers several advantages: the bytecode does not have to be recompiled for each client; unlike machine code, the Java bytecode supports a heterogeneous computing environment. In addition, the source analysis and compilation is performed by several powerful class authors instead of by the clients themselves.

3.2 Server caching strategy

The server implements the following caching strategy. A common cache is maintained by the server which contains classes or resources that have recently been used by two or more active clients, as shown in Figure 1. A smaller, client-specific delivery cache is also established on the server for each client which receives the anticipated set of classes referenced in the class subgraph. These caches are informed and populated by the application's class invocation probabilities.

Using these previously computed probabilities for the requested class, the database keys of the requested class and its subgraph are fetched from the class server database as shown in Figure 3. Then the class fetching algorithm works as follows:

```
while requested classes not in client-specific
buffer do
  THREAD C:
    receive prefetched classes and buffer them
    locally
  while requested classes not in common buffer
    concurrently do
     THREAD A:
       fetch, buffer, transmit requested class
       stream
     THREAD B:
       fetch, order, buffer, transmit subgraph
       stream
  endwhile
endwhile
```

The Java environment also supports the effective use of multiple threads for asynchronous events. As shown above, each connected client allocates at least three threads: two to process class requests and one to receive class data. The server allocates and manages a separate thread from a thread pool per connected client. This coarse-grained parallelism, coupled with the use of dedicated socket ports, permits for effective overlapping of client-server communication with client processing. Note that a client does not establish a direct connection to the server database.

Determining the proper size of the client cache is of critical importance. This size must balance the critical constraints of limited client memory with the fact that class prediction may be incorrect, which therefore may transmit unneeded classes to the client. The viability of class prediction is demonstrated only if these constraints are effectively reconciled; i.e., if the delivery effectiveness ratio H can be increased.

4 Performance measurements

4.1 Experimental setup

In the experiments, a 600MHz computer simulates a resource-constrained client with a specified cache size. It is connected to a 1.7GHz server computer over a 10Mb TCP/IP network through a router. The server pro-

gram executes within a single Java JVM instance, version 1.4.1.02.b06. The representative Telnet client application is designed to exercise the two different stages of classes to be delivered. The application has a reasonably rich class hierarchy including superclasses, inner classes, static and private variables, class variables and constructors.

The goal of this paper is to present the feasibility and effectiveness of the proposed methods. Experiments with more applications and benchmarks running on more suitable hardware, say one of the newer mobile handsets, will be the subject of future publications.

4.2 Verifying the effectiveness of a common cache

A performance-experiment was conducted in order to test the effectiveness of a common cache on the server (as defined in the Server caching strategy section) under the assumption of multiple client JVMs. A common class package was transmitted to a set of 4, 16 and 32 clients all running under their own JVM instance on the same client computer. Figure 4 shows the effect of averaged package instantiation time with and without the common 256k server cache operating. The performance effect of using the 256k common cache is not significant until 32 clients are being serviced simultaneously, at which point the average package instantiation time is cut in half from 15 seconds to 7 seconds. We will use a common cache of 256K for the next set of experiments.

4.3 Experimental results

After exercising the client source application under multiple and various traces using different class method calls, the delivery effectiveness ratio H is calculated as a function of client cache size. Table 3 presents the class transfer data gathered for a complete application run using a 32 kb cache. In the table, the effectiveness ratio is calculated as $H_C = C_C/C_R$.

The server records the stream size it delivers to the client. Previous experiments [7] had demonstrated that activating the class prediction mechanism delivered twice as many bytes to the client as opposed to simply delivering every class stream on-demand. The separately threaded client class loader, threads B and C, records the total stream transfer time it took to receive requested data as the amount of time the class loader blocks on the server, not including actual class instantiation. However, the two-stage packaging mechanism that transmits the two separate class package streams does not double the amount of transfer time because the predicted classes are transmitted asynchronously to the client. The additional network traffic occurs while the client is busy processing the current class, in effect, overlapping operations. By convention, the custom class loader itself and the application's main class are counted in S_C. We do not count those classes that are already accessible within the client's JVM.

Recall that multiple stage I classes are usually required to initialize any particular class for active use, which simply activates the class in the client JVM. Depending upon the application's structural class hierarchy, these implicitly loaded classes and interfaces may be satisfied by either the CCC or the server. In either case, $C_R = 52$ reflects the actual number of classes the application trace instantiates, both implicit and explicitly named. Without prediction, the server has to deliver 27 out of 52 classes (52%) to the application stream while the client blocks. Note that the custom class loader makes a fewer number of explicit class requests than what is actually delivered to the CCC, and so does not produce a reliable number for comparison.

With prediction, the number of classes S_C satisfied by the server, where the client is forced to request, block and wait for a class stream, is 4 less because the server has anticipated their use, then prefetched and transmitted them to the CCC. Now the server delivers only 23 out of 82 classes (28%) to the application stream responding to a blocked request. The total client class cache count, $C_R = 82$, is larger because the server has transmitted additional, predicted stage II classes. However, the prediction process increases C_C, the number of classes fetched from the local client cache. We claim that any successful local access by the client to a prefetched class is a significant performance gain. The end benefit is that the client effectiveness ratio H_C, has increased from 0.48 to 0.72, an improvement of 50%.

Figure 5 illustrates the averaged instantiation times and H_C ratios, prediction over no prediction, under four different cache sizes: 8k, 16k, 32k, and 64k. A client cache size of 16k accommodates most class requests relatively efficiently. A client cache size of 32k nearly approximates immediate class activation, as measured by averaged time of class activation. The averaged class cache hit ratios approached 54% with a cache size of 64k, revealing the effective limits of the prediction mechanism as well as demonstrating the effective parallelization of class delivery with program execution. For a modest increase in client memory, say 32k, a large application can be effectively delivered to an otherwise resource-constrained client.

To show the effect of communication overlap of the required additional data transfer to the client's overall execution time, we compare estimates of the total application trace execution time t_e with and without prediction. Because we are interested in client I/O-bound applications, it is fair to factor out the common duration of in-memory class execution time and to concentrate on the I/O parameter of total blocking time t_b, which includes total class transmission time. We ask at what client class cache size, if any, does the ratio of total blocking time over trace execution time ever become less with prediction than without prediction? Tables 4 and 5 present the following averaged results for 1 client at various client class cache sizes. Again, the total size of the stream transmitted was 84746 compressed bytes without prediction and 169104 bytes with prediction.

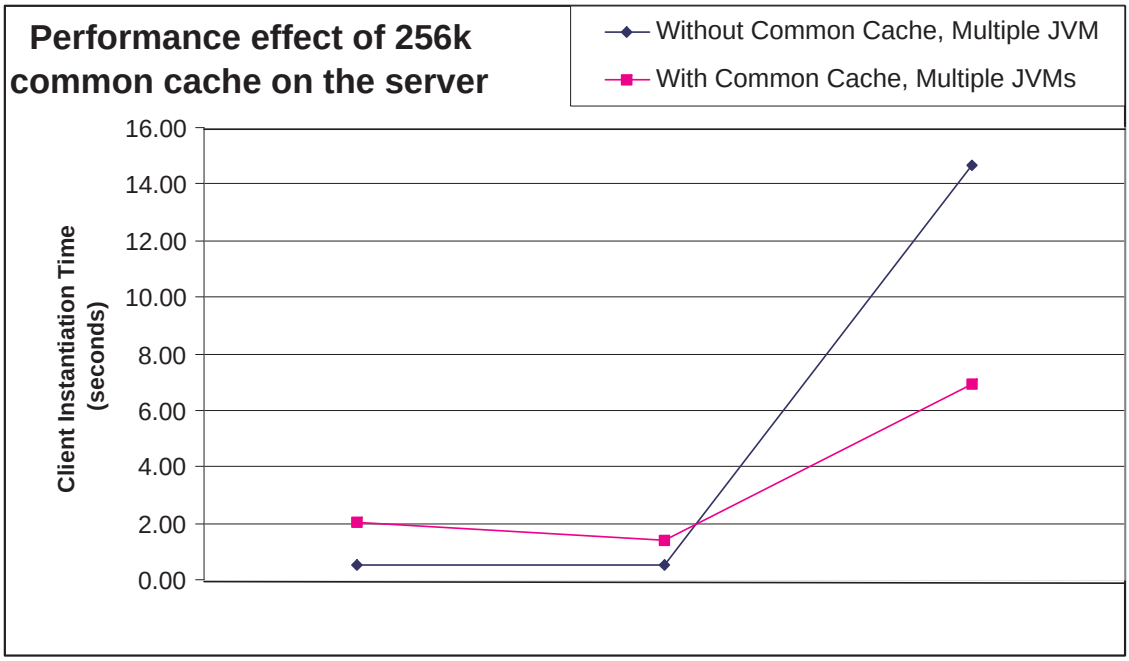

Figure 4: Common class package instantiation for 4, 16 and 32 clients.

Table 3: Effectiveness of class prediction for 1 client.

32kb Client Cache: 1 Active Client	**No Prediction**	**No Prediction**	**P/NP**
.	Stage I Only	Plus stage II	.
Total stream size (compressed bytes)	84746	169104	1.995
Total stream transfer time (secs) = t_b	14.2	19.3	1.36
Classes satisfied by server = S_C	27	23	0.85
Classes satisfied by client cache = C_C	25	59	2.36
Total client class cache count = C_R	52	82	1.58
Effectiveness ratio $H_C = C_C/C_R$	0.48	0.72	1.50

Table 4: Total execution t_e and blocking times t_b (sec) without prediction.

CCC Size	**8k**	**16k**	**32k**	**64k**
t_b	23.7	17.5	15.2	15.0
t_e	38.1	34.1	31.8	31.2
t_b/t_e	62%	51%	48%	48%

Table 5: Total execution t_e and blocking times t_b (sec) with prediction.

CCC Size	**8k**	**16k**	**32k**	**64k**
t_b	29.5	18.9	12.1	12.0
t_e	39.9	31.5	28.1	28.6
t_b/t_e	74%	60%	43%	42%

As shown in Figure 6, we conclude that prediction becomes more effective than not in terms of reducing total client execution time at a client cache size of 32k, even though twice as many class bytes are delivered to the client.

5 Related work

Arnold's recent survey [14] of adaptive optimization in virtual machines presents the three major developments of adaptive optimization technology in virtual machines over the last thirty years: 1) selective optimization, 2) feedback-directed code generation, and 3) other feedback-directed optimizations in order to improve VM performance. The survey discusses the benefits and drawbacks of just-in-time compilers, synchronized thread management, dynamic class loading, native code caches, class splitting and dynamic class caching. Most of these techniques exploit some form of temporal locality to be effective.

Krintz's work [5] proposes Java class file splitting and prefetching optimizations as an effective latency-hiding optimization for mobile programs which also does not require

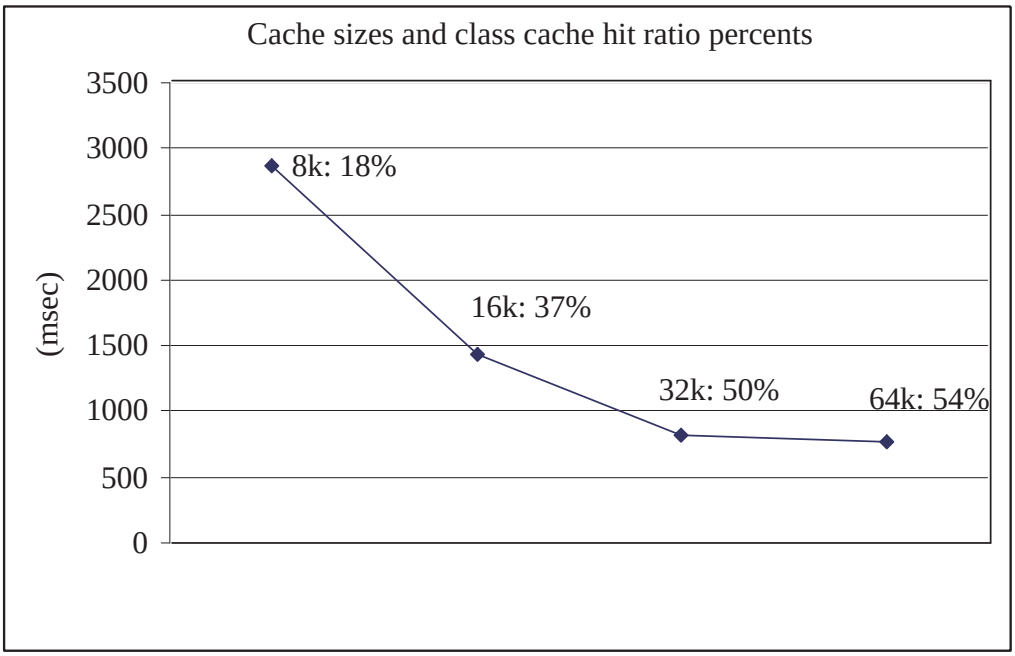

Figure 5: Client cache size vs. average time of class instantiation.

redefining the Java Virtual Machine specification. The combination of techniques reduces the overall transfer delay encountered during a mobile program's execution by 25% to 30% on average. However, Krintz's method requires inserting prefetch statements into the Java source code as well as compilation using a binary modification tool.

Bartels et al [1] describe an adaptive fault-based prefetching scheme based on a one-level Markov net that achieved high prediction accuracy for some classes of scientific applications.

Thiebaut's work [10] with synthetic traces demonstrated the structural importance of the locality of reference in real programs and its impact on hit and miss ratios.

Chilimbi [12] proposes a data reference framework and an exploitable locality abstraction called *hot data streams* as a quantitative basis for understanding and optimizing data reference locality.

Our general approach is indebted to Patterson's seminal paper [6] on informed aggressive disk prefetching and caching (*TIP*), where it is shown that prefetching can not only mask latency with asynchrony, by overlapping I/O with computation, but also expose parallelism for the sake of greater throughput.

6 Conclusions

Our paper describes a streaming mechanism that distributes Java class bytecode streams to a client from a class server using a 1st-order Markov probability model to predict the client's next class request. The experimental results demonstrate that a simple class prediction mechanism significantly reduces client blocking using a dedicated client cache size of 32k. At the cost of the client receiving twice as many bytes over the network and a modestly-larger cache, the client is able to execute rich and complex applications not otherwise possible. We acknowledge that this extra network processing is clearly a concern for power-sensitive devices.

Using the Java architecture requires writing and delivering a custom class loader for mobile devices. Sun's Java 2 Micro Edition (J2ME) CLDC, targeted to cell phones, requires 128K to 512K total memory available with less than 256K ROM/Flash and less than 256K RAM (JSR-000030). However, it currently does not support user-defined class loaders or native method access. The mobile device manufacturers would themselves have to embed modified class loaders into these devices in order to handle the proposed streaming mechanism.

To summarize, this paper makes the following contributions:

- The Java linking process permits dynamic class loading that delivers application streams to clients.
- A 1st-order Markov class invocation probability model effectively predicts the client's next class request in order to reduce invocation latency and transfer delay by overlapping program execution with network communication. Prediction is worth the effort.
- The current implementation does not require the modification of the Java Virtual Machine definition. However, due to the lack of a dynamic class loading mechanism, the current version of J2ME/CLDC would require significant modification.

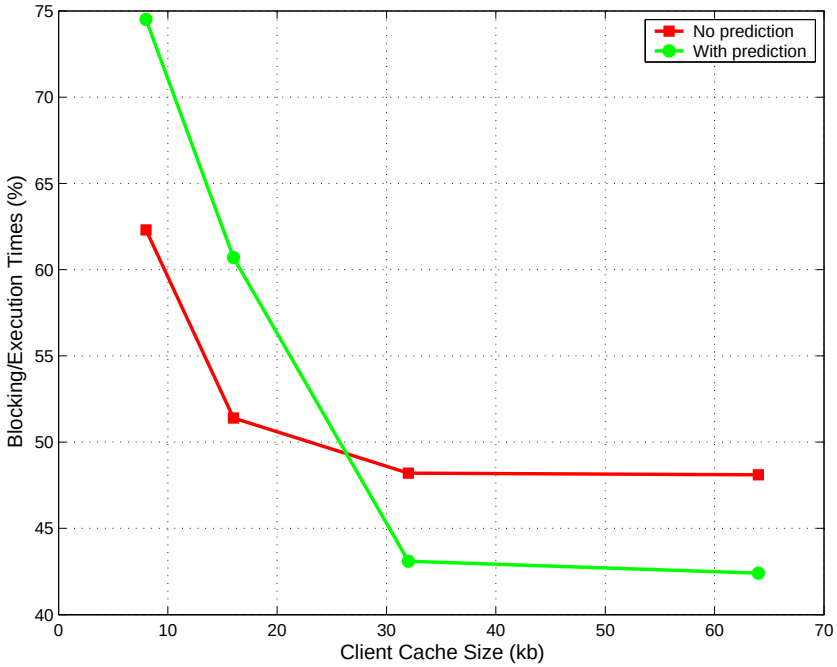

Figure 6: Execution and blocking time percent vs client cache size.

– We also plan to carry out the methods developed in this work with various benchmarks on appropriate hardware, such as mobile handsets.

References

[1] Bartels, G., Karlin, A., Levy, H., Voelker, G.; Anderson, D., Chase, J. (1999). *Potentials and Limitations of Fault-Based Markov Prefetching for Virtual Memory Pages*, ACM SIGMETRICS Performance Evaluation Review, Volume 27 , Issue 1, June 1999.

[2] Gosling, J., B. Joy, G. Steele, and G. Brach. (2000). *Java Language Specification, 2nd Ed*. Boston: Addison Wesley.

[3] Liang, Sheng and Bracha, Gilad. (1998). *Dynamic Class Loading in the Java Virtual Machine*, in Proceedings of OOPSLA '98, published as ACM SIGPLAN Notices, Volume 33, Number 10, October 1998, pages 36-44.

[4] Venners, Bill. (1999). *Inside the Java 2 Virtual Machine, 2nd Ed.*, New York: McGraw-Hill Companies.

[5] Krintz, C., Calder, B., and Holzle, U. (1999). *Reducing Transfer Delay Using Java Class File Splitting and Prefetching*, UCSD Technical Report, CS99-615, March 1999.

[6] Patterson, R.H. Gibson, G.A., Ginting, E., Stodolsky, D. and Zelenka, R. (1995). *Informed Prefetching and Caching*, J. Proc. of the 15th Symposium of Operating Systems Principles, Copper Mountain Resort, CO, December 3-6, 1995, pp. 79-95.

[7] Alagbhand, G., Gnabasik, D. (2004). *Streaming Java Applications to Mobile Computing Devices*, Proceeding of the 2004 International Conference on Wireless Networks, Monte Carlo Resort, Las Vegas, Nevada, June 21-24, 2004, pp. 637-643.

[8] Sun Microsystems. (1998). *Java Object Serialization Specification*. Available online: http://java.sun.com/products/jdk/1.2/docs/guide/serialization/spec/serialTOC.doc.html

[9] Gunther, N., (1998). *The Practical Performance Analyst: Performance-By-Design Techniques for Distributed Systems*, New York: McGraw-Hill Companies.

[10] Thiebaut, D., Wolf, J.L., and Stone, H.S. (1992). *Synthetic Traces for Trace-Driven Simulation of Cache Memories*. IEEE Trans. Computers. 41(4):388-410.

[11] Highleyman, W.H., (1989). *Performance Analysis of Transaction Systems*. Englewood Cliffs, N.J.: Prentice-Hall

[12] Chilimbi, Trishul. (2001). *Efficient Representations and Abstractions for Quantifying and Exploiting Data Reference Locality*. Microsoft Research, One Microsoft Way, Redmond, WA

[13] Vanichpun, S., Makowski, A.M., (2004). *Comparing strength of locality of reference - Popularity, ma-*

jorization, and some folk theorems. To appear in Performance Evaluation and Planning Methods for the Next Generation Internet, A. Girard, B. Sanso and F.J. Vazquez-Abad, Editors, Kluwer Academic Press.

[14] Arnold, M., Fink, S.J., Grove, D., Hind, M., Sweeney, P.F. (2004).*A Survey of Adaptive Optimization in Virtual Machines*. IBM Research Report, RC23143 (W0312-097).

A Formal Framework Supporting the Specification of the Interactions between Agents

Farid Mokhati
Département d'Informatique
Université d'Oum El Bouaghi - Algérie
E-mail: Mokhati@yahoo.fr

Mourad Badri and Linda Badri
Département de Mathématiques et d'Informatique
Université du Québec à Trois-Rivières - Canada
E-mail: Mourad.Badri@uqtr.ca, Linda.Badri@uqtr.ca

Keywords: multi-agent systems, RCA, Maude, translation, behavior, interactions, formal specification, verification and validation.

Received: May 24, 2005

In this paper we present a formal framework supporting the translation of interactions between agents (the interactions are described with the help of the RCA formalism) in a Maude specification. Based on rewriting logic, the formal and object-oriented language Maude supports formal specification and programming for a wide range of applications. The main motivations of our work are essentially: (1) to formally specify the behavior of multi-agent systems and (2) to provide a solid basis for their verification and validation. The translation process is illustrated by means of a real case study.

Povzetek:Opisan je formalni okvir za prevajanje interakcij med agenti.

1 Introduction

In Multi-Agent Systems (MAS), agents interact in order to exchange information, to cooperate and to coordinate their tasks [24]. The usual approach to the description of interactions between agents consists in using protocols [8, 26]. Several agents' interaction protocols (AIP) have been proposed in the literature [7]. They constitute an important part of MAS's infrastructures. However, most of the protocols published in the literature are semi-formal, vague or contain errors as mentioned in [23]. Knowing that AIP play a crucial role in MAS development [30], their formal specification as well as their verification constitute essential tasks [11]. In the field of agents' behavior specification, three major approaches emerge in the literature: state-charts based approaches [27, 22], Petri Nets based approaches [5, 1], and approaches representing an adaptation of object-oriented specification methods [19, 20].

Among the agents' interaction protocols proposed in the literature, we can mention the RCA formalism (Représentation des Comportements d'Agents) [27], which is based on strongly typed states-transitions graphs. The RCA formalism allows describing agents' behaviors graphically. This formalism has been used in the design of several Cooperative Information Systems (CIS) based on informational agents. We can mention, for example, the NetMan project based on the coordination of several agents [4], a project related to the reactive reorganization of production shops and treating the cooperation between agents having to solve a problem in a distributed and cooperative way [28], as well as a project on the hydraulic management of the Camargue ecosystem and based on a negotiation process between agents (Project SIMFONHYC) [18].

One of the strong points of the RCA formalism [18, 28] resides in the modular design of agents' behaviors. Indeed, the use of composite action states makes it possible the overlapping of behavioral plans and therefore a description by successive refinements of agents' behavior. This characteristic comes directly from the notion of composite state of RCA graphs. Nevertheless, some critiques on RCA graphs can be formulated, notably on their formalization and on the sequential aspect of the execution cycle of behavioral plans [28]. Furthermore, this formalism allows the visualization of the synchronization points between dual protocols thanks to the complementarity between communication states and external transition. It is then easy to recognize the coordination points between dual protocols [28]. However, RCA graphs as well as the existing formalisms in the literature describing agents' interaction protocols are not endowed again with a formal semantics [28]. They only offer a semi-formal specification [23] of interactions between agents. These weaknesses can generate several problems in MAS development and verification.

Using formal notations for the description of MAS' behavior offers several advantages. It essentially allows producing rigorous and precise descriptions supporting efficiently their verification and validation process. The Maude language, based on the rewriting logic, seems to

us to be an interesting candidate. It offers, through its rich notation, an interesting way for concurrent systems formal specification and programming. Furthermore, it also supports the description of multi-agent interactions [21, 16]. In this paper, we present a formal framework supporting the translation of multi-agent interactions, specified using the RCA formalism, in a Maude specification. The main motivations of our approach are essentially: (1) to specify formally the behavior of multi-agent systems, in particular, the interactions between agents, and (2) to provide a solid basis for their verification and validation process. The Maude specifications, generated in the context of the developed framework, have been validated using the platform supporting the Maude language. The remainder of the paper is organized as follows: Section 2 gives a brief survey on the main related works. We present summarily the RCA formalism in section 3. In section 4, we give the basic concepts related to the rewriting logic as well as the Maude language. Section 5 presents the translation process. The proposed approach is illustrated using a concrete case study in section 6. Finally, section 7 gives some conclusions and future work directions.

2 Related Work

We present briefly in this section three formalisms (AUML, CATN and RCA) supporting the description of agents' interaction protocols. AUML [19, 9] is an extension of the UML language allowing describing interactions between agents. To represent multi-agent interaction protocols, AUML adopts in fact an approach in three layers. It uses, in the first level, packages and templates to represent the protocol in whole. Sequence diagrams, collaboration diagrams, activity diagrams, and states-transitions diagrams are used to represent interactions between agents. Activity diagrams and states-transitions diagrams are also used to capture agents' internal behavior (for more details see [19]). However, AUML only offers a semi-formal specification of the interactions between agents.

The CATN formalism (Coupled Augmented Transition NetWork) [10] is a states-transitions machine, to which a particular goal (or significance) is associated. A CATN can be decomposed in sub-CATNs. Each of these components is a CATN, having its own goal. The components of a CATN are joined together by ad-hoc transitions named "interactions transitions". Among these, we distinguish the non-terminal interactions transitions of those that are terminal. These last correspond to language acts (between agents) or to private actions of agents. This recursive aspect of the CATN allows a top-down design approach, from the most abstract behavior of a group of agents until their most concrete actions (individual terminal actions and communications through the interactions transitions). Each agent can execute in a concurrent way several CATNs depending on the tasks that it has to achieve [10, 25].

The RCA formalism [27, 28], supporting the description of role protocols, is used to describe agents' behavior. It is based on states-transitions diagrams introducing seven types of states and two types of transitions. The seven states are: the initial state, the final state, the elementary action state, the composite action state, the communication state and the waiting states (limited and unlimited). The two types of transitions are the internal transition and the external transition. Using this formalism, it is easy to recognize the coordination points between dual protocols. The RCA formalism is not limited to the description of the exchanges of messages between agents (as the case in the other formalisms). It also allows clarifying the actions that they undertake. In addition, the RCA graphs describe the working of the agents and help thus the design of their interactions. The links that exist between the macro level (i.e. the system's behavior) and the micro level (i.e. the agent's behavior) may be considered in an integrated way [28, 29].

These different approaches certainly offer some elements of answer to some problems related MAS development. However, they only allow a partial formalization of MAS. Furthermore, some authors [6, 5] opposed to the use of formalisms based on state-transition graphs two major arguments: 1) the impossibility to be able to verify the consistency of the protocols thus specified; and 2) the absence of taking into account the concurrent aspects of protocols [28]. In spite of the advantages that it offers relatively to the other formalisms, the RCA formalism only offers a graphic semi-formal description [18]. Furthermore, it is not endowed again with a formal semantics. These weaknesses combined to the complexity of MAS can generate several problems in their development and verification processes. The use of an appropriate formal notation for the description of MAS' behavior offers several advantages. It essentially allows the production of rigorous and precise descriptions supporting efficiently their verification and validation process. Our approach is similar, in terms of objectives, to the previously quoted approaches. It consists, essentially, to support the important stage of the specification of agents' behaviors. However, we preferred to adopt a more formal approach in the specification of agents' behaviors in terms of interactions allowing, among others, to support the verification of consistency (internal and global) in the behavior. Our approach allows translating the interaction protocols described using the RCA formalism in the Maude language. The Maude system consists in a high-level language of programming, specification and modeling based on rewriting logic [2, 15, 21]. It is also endowed with a high performance interpreter. It allows describing concurrent systems and supports the formal specification of distributed systems [14, 29, 12].

3 RCA Formalism

RCA (Représentation des Comportements d'Agents) [27, 28] is a formalism allowing describing an agent's behavior graphically. It is based on a strongly typed graph: seven types of states and two types of transitions (figure 1). The seven states are the initial state (to show

the beginning of the graph), the final state (to mark the end of the graph), the elementary action state (that corresponds to the agent's simple action), the composite action state (it is in fact about the call to another protocol), the communication state (sending of message), and the limited and unlimited waiting states (waiting of treatments done by other agents). The two types of transitions are the internal transition (it corresponds to the end of an activity and provokes the passage to another state) and the external transition (it is in fact a reception of a message that provokes, like an internal transition, the change of the agent's activity). An external transition is triggered by a communication state at another agent.

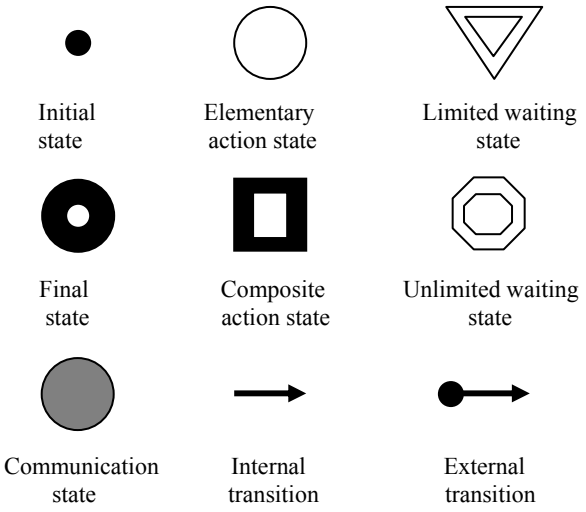

Figure 1 : Convention of representation of the RCA formalism.

The number of internal and external transitions depends on the type of the starting state and its transitions. It can be either null, limited or unlimited (figure 2).

Type of transition's departure state	Authorized internal transitions number	Authorized external transitions number
	[Min..Max]	[Min..Max]
Initial state	[0 .. 1]	[0 .. 1]
Elementary action state	[1 .. ∞]	[0 .. 0]
Composite action state	[1 .. ∞]	[0 .. 0]
Communication state	[1 .. 2]	[0 .. 0]
Limited waiting state	[1 .. 1]	[1 .. ∞]
Unlimited waiting state	[0 .. 0]	[1 .. ∞]
Final state	[0 .. 0]	[0 .. 0]

Figure 2 : Authorized transitions number according to the starting state.

Each states graph starts with a unique initial state and finishes by a unique final state. The internal events are the consequence of the agent's actions represented by action states (elementary or composite). They trigger the internal transitions. The external events result from communication activities of the agents, i.e. a reception of message constitutes an external event and provokes the crossing of an external transition. Of this fact, the type of allowed transition at a precise place of the graph depends exclusively of the origin state type of this transition:

- Initial state : only one transition (internal or external) may quit this state.
- Action state (simple or composite) : the internal transitions are in any number not null after action states.
- Communication state : one or two internal transitions may quit the communication state.
- Limited waiting state : the waiting may stop after the reception of a message (external transition), or if no message has been received beyond the waiting delay (internal transition). Furthermore, only one internal transition may quit a limited waiting.
- Unlimited waiting state : this waiting type remains while that it doesn't occur an external event (reception of message). It is therefore about a blocking state.

4 Rewriting Logic and Maude Language

4.1 Rewriting Logic

The rewriting logic, having a sound and complete semantics, was introduced by Meseguer [14]. It allows describing concurrent systems. This logic unifies all the formal models that express concurrence [13, 15]. In rewriting logic, the logic formulas are called rewriting rules. They have the following form: R:[t] → [t'] if C. Rule R indicates that term t becomes (is transformed into) t' if a certain condition C if verified. Term t represents a partial state of a global state S of the described system. The modification of the global state S of the system to another state S' is realized by the parallel rewriting of one or more terms that express the partial states. The distributed state of a concurrent system is represented as a term whose sub-terms represent the different components of the concurrent state. The concurrent state's structure can have a variety of equivalent representations because it satisfies certain structural laws (equivalence class).

```
1. sort Configuration .
2. sort Object .
3. sort Msg .
4. subsort Object < Configuration .
5. subsort Msg < Configuration .
6. op null : -> Configuration .
7. op_ _ : Configuration Configuration ->
          Configuration [assoc comm id : null] .
```

Figure 3 : Example of a portion of the Maude program.

For example, in an object-oriented system the concurrent state that is usually called configuration has the structure of a multi-set of objects and messages. Therefore, we can have configurations constructed by a binary operator applied to binary sets as illustrated in figure 3.

The portion of program illustrated in figure 3 gives a definition of three types: *Configuration*, *Object* and *Msg*. In lines 4 and 5, *Object* and *Msg* are sub-types of *Configuration*. Objects and messages are in fact multi-set configuration singletons. More complex configurations are generated from the application of the union on these multi-set singletons (objects and messages). Where there is neither floating messages nor live objects, we have in this case an empty configuration (line 6). The construction of a new configuration in terms of other configurations is done with line 7's operation. We can note that this operation has no name and that the two sub lines indicate the positions of two parameters of configuration type. This operation, which is the multi-set union, satisfies the structural laws of association and of commutation. It also possesses a neutral element null. For example, if we have a message *M1* that represents a configuration, and an object <*O : C|atts* > (please note that *O* is an object's identifier, *C* its class and *atts* the list of its attributes) that represents in itself another configuration, then we can construct another configuration in terms of those two configurations: *M1 < O : C | atts >*. This one is equivalent to the configuration *< O : C | atts > M1* because the __ operation is commutative.

4.2 Maude

Maude is a specification and programming language based on the rewriting logic [14, 3]. Three types of modules are defined in Maude. Functional modules allow defining data types and their functions through equations theory. Figure 4.a represents the functional module *Nat* specifying natural numbers. Such a module is imported in the module *FACT* (figure 4.b) to calculate the factorial of natural numbers. System modules define the dynamic behavior of a system. This type of modules extends functional modules by introducing rewriting rules. A maximal degree of concurrence is offered by this type of module. Finally, there are the object-oriented modules that can be reduced to system modules. In relation to system modules, object-oriented modules offer a more appropriate syntax to describe the basic entities of the object paradigm as, among others: objects, messages and configuration. Only one rewriting rule allows expressing the consumption of certain floating messages, the sending of new messages, the destruction of objects, the creation of new objects, state change of certain objects, etc.

Figure 5.a illustrates the use of a system module *BANK-ACCOUNT* to define an object counts banking *A* and the two operations capable to affect its content *credit* and *debit* while executing the rewriting rules defined in this module. Figure 5.b represents the same *BANK-ACCOUNT* module with a more appropriate object-oriented syntax.

```
fmod NAT is
sorts Zero NzNat Nat .
subsort Zero NzNat < Nat .
***constructors
op 0 : -> Zero .
op s_ : Nat -> NzNat .
….
endfm
```

(a)

```
fmod FACT is
Including NAT .
op _! : Nat -> NzNat .

var N : Nat .
eq 0 ! = 1 .
eq (s N) ! = (s N) * N !.
endfm
```

(b)

Figure 4 : Functional Modules *Nat* and *FACT*.

We note, that after the execution of the unconditional rule [credit], the message *credit(A, M)* is consumed and the content of the account is increased. In the same way, the execution of the conditional rule [debit] requires that the condition (N>=M) be verified. The execution of such rule generates the consumption of the message *debit(A,M)* and the reduction of the content of the account.

```
mod BANK-ACCOUNT is
protecting INT .
 including CONFIGURATION .
op Account : -> Cid.
op bal :_ : Int -> Attribute .
ops credit debit :    Oid Nat -> Msg .
var A  : Oid . vars M N : Int .

rl [credit] :    < A : Account | bal : N >   credit(A, M)
          =>   < A : Account | bal : N + M  > .

crl [debit] :   < A : Account | bal : N >   debit(A, M)
          =>  < A : Account | bal : N - M  >
              If N >= M .
endm
```

(a)

```
(omod BANK-ACCOUNT is
protecting MACHINE-INT .
class Account | bal : MachineInt .
msgs credit debit :   Oid MachineInt -> Msg .
var A : Oid .
vars M N : MachineInt .

rl [credit] :  < A : Account | bal : N >  credit(A, M)
          => < A : Account | bal : (N + M)  > .

crl [debit] :   < A : Account | bal : N >   debit(A, M)
          =>  < A : Account | bal : (N – M)  >
              If N >= M .
endom)
```

(b)

Figure 5 : The same *BANK-ACCOUNT* module in system module and O.O module forms.

5 Translating RCA Descriptions in Maude

We developed a formal framework allowing the formal specification of role protocols described using RCA formalism. The framework is composed, as illustrated by figure 6, of several modules: an object-oriented module (*ROLE-PROTOCOLE*) and several functional modules (the remainder of modules).

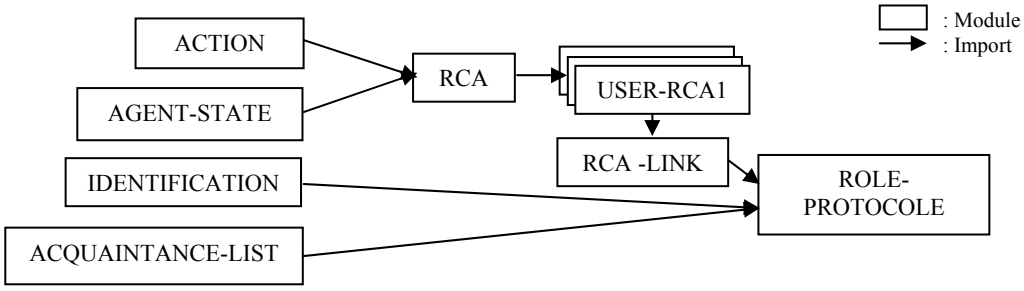

Figure 6 : *RCA-Maude* frameworks' architecture.

```
(fmod AGENT-STATE is
sorts AgentState KindAgentState NameAgentState .                          ***[1]

ops initial final communication elementary composite
                    limitedWaiting UnlimitedWaiting : -> KindAgentState .  ***[2]

op AgentState : NameAgentState KindAgentState -> AgentState .             ***[3]
op IsInitial : AgentState -> Bool .                                       ***[4]
op IsFinal : AgentState -> Bool .                                         ***[5]
op IsOfCommunication : AgentState -> Bool .                               ***[6]
op IsElementary : AgentState -> Bool .                                    ***[7]
op IsComposite : AgentState -> Bool .                                     ***[8]
op IslimitedWaiting : AgentState -> Bool .                                ***[9]
op IsUnlimitedWaiting : AgentState -> Bool .                              ***[10]

var k : KindAgentState . var ns : NameAgentState .

eq  IsInitial(AgentState(ns, k)) = if  k == initial then true                    ***[11]
            else false fi .
eq  IsFinal(AgentState(ns, k)) = if  k == final then true                        ***[12]
            else false fi .
eq  IsOfCommunication(AgentState(ns, k)) = if  k == communication then true      * **[13]
            else false fi .
eq  IsElementary(AgentState(ns, k)) = if  k == elementary then true              ***[14]
            else false fi .
eq  IsComposite(AgentState(ns, k)) = if  k == composite then true                ***[15]
            else false fi .
eq  IslimitedWaiting(AgentState(ns, k)) = if  k == limitedWaiting then true      ***[16]
            else false fi .
eq  IsUnlimitedWaiting(AgentState(ns, k)) = if  k == UnlimitedWaiting then true  ***[17]
            else false fi .
endfm)
```

Figure 7 : The functional module *AGENT-STATE*.

The functional module *AGENT-STATE* (figure 7) contains the different necessary type declarations for the definition of a state (line [1]) and, on the other hand, the definition of operations used for the construction and the manipulation of a state (lines [2, 3, 4, 5, 6, 7, 8, 9, 10]), as well as equations implementing these operations (lines [11, 12, 13, 14, 15, 16, 17]).

In the *ACTION* module (figure 8), in addition to the type Action, we define the two functions *IsSendingToOnlyOne* and *IsSendingToAll.* The first

function determines if an action is destined to only one agent's acquaintance, on the other hand the second function indicates if it is necessary to send a message to all agent's acquaintances. To describe the identification mechanism of agents, we define the functional module *IDENTIFICATION* (figure 9). Furthermore, an agent must be endowed with a list of its acquaintances allowing it to exchange messages with the other agents. We define for it the functional module *ACQUAINTANCE-LIST* to manage the lists of the agents' acquaintances . Due to imitation of space and a considerable size of this last module, we don't present it in this paper.

```
(fmod ACTION is
protecting BOOL .
sort Action .
op IsSendingToAll : Action -> Bool .
op IsSendingToOnlyOne : Action -> Bool .
endfm)
```

Figure 8 : The functional module *ACTION*.

```
(fmod IDENTIFICATION is
 sort AgentIdentifier .
 subsort AgentIdentifier < Oid .
 endfm)
```

Figure 9 : The functional module *IDENTIFICATION*.

To define an RCA diagram, we propose the *RCA* module (figure 10). This module reuses the *AGENT-STATE* and *ACTION* modules. It includes the definition of two operations: *TargetState* that determines the target state according to a state source and an action, and the *FeedBack* operation used in the case where the treatment accomplished by the agent takes place while toppling between two states during a limited length. To each event coming from a state source, such a function determines the appropriate action that should be executed from the target state as a feedback.

```
(fmod RCA is
protecting ACTION .
protecting AGENT-STATE .
op TargetState : AgentState Action -> AgentState .
op FeedBack : Action -> Action .
endfm)
```

Figure 10 : The functional module *RCA*.

For the construction of an RCA diagram for an application, we propose to extend the *RCA* module in another *USER-RCA* module (figure 11). In this module, the user must: mention all states constituting the RCA diagram, define all possible actions, attach the actions in the states using the *TargetState* function, determine the actions constituting feedbacks using the *Feedback* function, and finally specify for every communication action whether it is sent to all (using the *IsSendingActionToAll* function) or to only one (using *IsSendingActionToOnlyOne*). An *USER-RCA* module (figure 11) is associated with every category of agents (playing the same role).

```
(fmod USER-RCA is
extending RCA .

***User part***
endfm)
```

Figure 11 : The functional Module *USER-RCA*.

To respect the interaction protocol used between agents, we propose to realize a sort of link between the RCA diagrams of the different agents. Basing on the synchronization points, main characteristic of this formalism, such a link consists in guaranteeing that at the moment of the reception of a message, an agent can't consume such a message except if it is in the corresponding state of the state of the sender agent. An agent that is in a communication state generates an external event that causes an external transition at the agent receiver. To receive such an event, this last must be in a waiting state (limited or unlimited). Indeed, the sending actions accomplished by a sender agent represent events for receiver agent. Thus, there is a correspondence between the sending actions of the sender and the events received by the receiver. For it, the user must develop the *RCA-LINK* module (figure 12) that contains the correspondence on the one hand, between the different states of agents and, on the other hand, between the events generated by the sender and the events received by the receiver.

```
(fmod RCA-LINK is
protecting USER-RCA .
…
op CorrespondingState : AgentState -> AgentState .
op CorrespondingAction : Action -> Action .

***User part***************
…
endfm)
```

Figure 12 : The functional module *RCA-LINK*.

The object-oriented module *ROLE-PROTOCOL* (figure 13) represents the main module. It imports the *RCA-LINK*, *IDENTIFICATION,* and *ACQUAINTANCE-LIST* modules. For the formal description of agents, we propose the class *Agent* (line 2).

The definition of this class has as attributes *PlayRole*, *State,* and *AcqList,* to contain in this order, the agent's actual role, the current state of the agent, and the list of its acquaintances. In addition to different types of states defined in figure 7, we define in this module (figure 13)

the type *EventType* (line 1) relative to the two types of events used in this formalism (Internal and External). The appearance of an event is expressed by message *Event* (line 3) having as parameters an agent, a role, the type of the event, the agent's state, and an action.

In the RCA formalism, an agent changes state while doing either an internal transition or an external one. Figure 13 illustrates the necessary rewriting rules we developed modeling the possible cases of transitions (internal and external), while respecting the constraints of this formalism described by the table given in figure 2.

```
(omod ROLE-PROTOCOLE is
protecting RCA-LINK .
protecting IDENTIFICATION .
protecting ACQUAINTANCE-LIST .
sorts Agent Role EventType .

ops Internal External : -> EventType .                                    ***[1]
class  Agent | PlayRole : Role, State : AgentState, AcqList : acquaintanceList  .    ***[2]
Msg Event : Oid Role EventType AgentState Action -> Msg .                 ***[3]

*******************************************************************************
vars A A1 : Oid . var S : AgentState . vars R R1 : Role .
var Act : Action . var ACL : acquaintanceList .

*****************************Possible cases of internal transition**************************
*****************************************First case************************************
crl[InternalTransitionCase1] :                                            ***[4]
        Event(A, R, Internal, S, Act)
        < A : Agent | PlayRole : R, State : S, AcqList : ACL >
       =>
        < A : Agent | PlayRole : R, State : TargetState(S, Act), AcqList : ACL >
        if (IsInitial(S) or IsElementary(S) or IsComposite(S) or IslimitedWaiting(S)) .

*****************************************Second case***********************************
crl[InternalTransitionCase2] :                                            ***[5]
       Event(A, R, Internal, S, Act)
       < A : Agent | PlayRole : R, State : S, AcqList : ACL >
      =>
       < A : Agent | PlayRole : R, State : TargetState(S, Act), AcqList : TailA(ACL) >
       Event(HeadA(ACL), R1, External, CorrespondingState(S), CorrespondingAction (Act))
       if IsOfCommunication(S) and IsSendingToOnlyOne(Act) .

*****************************************Third case************************************
crl[InternalTransitionCase3] :                                            ***[6]
        Event(A, R, Internal, S, Act)
        < A : Agent | PlayRole : R, State : S, AcqList : ACL >
      =>
        < A : Agent | PlayRole : R, State : S, AcqList : TailA(ACL) >
     Event(A, R, Internal, S, Act)
       Event(HeadA(ACL), R1, External, CorrespondingState(S), CorrespondingAction(Act))
       if IsOfCommunication(S) and IsSendingToAll(Act) and ACL =/= EmptyacquaintanceList .

********************Possible case of External transition*********************************
crl[ExternalTransition] :                                                 ***[7]
        Event(A, R, External, S, Act)
        < A : Agent | PlayRole : Initiator, State : S, AcqList : ACL >
       =>
        < A : Agent | PlayRole : Initiator, State : TargetState(S, Act), AcqList : ACL >
        if  IsInitial(S) or IslimitedWaiting(S)  or  IsUnlimitedWaiting(S) .

*******************************************************************************************
…
endom)
```

Figure 13 : The object-oriented module *ROLE-PROTOCOLE*.

An agent doesn't do an internal transition except if it is in one of the following states: initial, elementary, composite, limited waiting or communication (see figure 2). In the first four states, an internal transition is described by the rewriting rule (line 4) of figure 13. Such a rule expresses that at the moment of the appearance of an internal event, the agent consumes the message and changes its state using the *TargetState* function defined in the *RCA* module (figure 10). We treated separately the case of a communication state, knowing that from this state the agent generates an external event (sending of message) allowing its acquaintances that are in waiting to change their states. A message can be sent by an agent to only one agent belonging to its acquaintance list or to all its acquaintances.

The first case is described by the rule of the line 5. Such a rule expresses, on the one hand, the consumption of an internal event, on the other hand, the generation of an external event sent to only one agent (here we adopt the strategy choosing the agent that is at the head of the acquaintances list using the *HeadA* function), if the agent sender is in a communication state. The second case is described by the rule of the line 6. Such a rule presents the sending of a message by the agent *A* to all its acquaintances. It presents a conditional loop. Indeed, it allows browsing the acquaintance list (*ACL*) of the agent, while using the two operations *HeadA* (determines the head of the list) and *TailA* (determines the rest of the list). Such a loop stops when the list is browsed completely. An agent doesn't do an external transition except if it is in a waiting state (limited either unlimited) or sometimes in its initial state (see figure 2). This is expressed by the rewriting rule of the line 7. When it occurs an external event to an agent, this last changes its state while doing an external transition, but the agent must be in an initial or waiting state (limited either unlimited).

6 Case Study : Auction Application

This section illustrates the application of our approach on a concrete example. It is about a simple example of an auction.

We have two kinds of agents: *Auctioneer* and *Bidder*. Each auction involves one Auctioneer and several Bidders.

The Auctioneer has a catalog of products. Before beginning the auction, the Auctioneer sends the catalog to all participants. Then, it begins the auction for all products. The products are proposed sequentially to the participants. Figures 14.a and 14.b describe the representation of the Auctioneer and Bidder roles respectively using the RCA formalism.

6.1 Application of the Translation Process

The formal description of the behaviors of the agents whose roles are described using the RCA formalism implies all defined modules previously with the definition of the *USER-RCA* and *RCA-LINK* modules. Figures 15 and 16 illustrate the defined modules corresponding to the Auctioneer and Bidder roles respectively. The correspondence between these roles is presented in figure 17. Indeed, the two modules *USER-RCA1* (figure 15) and *USER-RCA2* (figure 16) describe the Auctioneer and Bidder roles respectively in the same way. We limit ourselves to detail the *USER-RCA1* module only.

In figure 15, we define the different states of the Auctioneer agent (lines 1 and 2). For example, the state *AgentState(CommitmentDecision, communication)* means that the state named *CommitmentDecision* is a communication state (see figure 14.a). The actions given in figure 14.a are described by line 3. To determine the target state (line 4) according to a source state and a given action, we used the operation *TargetState* defined in figure 10. If the Auctionner agent is in its *CommitmentDecision* state, and the action to execute is *AcceptProposalSent*, the target state of this transition must be the final state *EndI*. To select the conditional rule to execute when the agent is in a communication state (see figure 13, lines 5 and 6), it is necessary to know the type of the action. For example, the line 5 of figure 15 indicates that the *CFP-Sent* action must be sent by the Auctioneer to all Bidders.

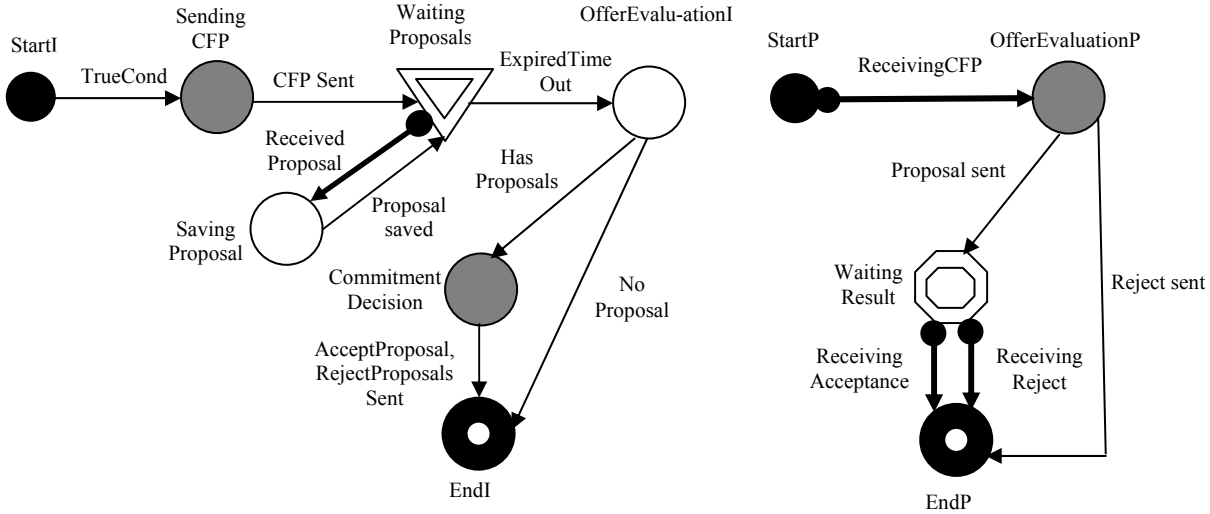

Figure 14 : Representation of the roles, *Auctioneer* and *Bidder* using RCA formalism.

```
fmod USER-RCA 1 is
extending RCA .

****************States of an Auctioneer************************************************
ops StartI SendingCFP WaitingProposals OfferEvaluationI SavingProposal
                                        CommitmentDecision EndI : ->  NameAgentState .      ***[1]

ops AgentState(StartI, initial) AgentState(SendingCFP, communication)
    AgentState(WaitingProposals, limitedWaiting) AgentState(OfferEvaluationI, elementary)
    AgentState(SavingProposal, elementary) AgentState(CommitmentDecision, communication)
                                                AgentState(EndI, final) : -> AgentState .      ***[2]

***************Actions to accomplish by an Auctioneer**********************************
ops TrueCondition CFP-Sent ExpiredTimeOut NoProposal HasProposal  ReceivedProposal
                          ProposalSaved AcceptProposalSent RejectProposalSent : -> Action .      ***[3]

***************Determination of the target state according to a state source and an action *********
eq TargetState(AgentState(StartI, initial), TrueCondition) = AgentState(SendingCFP, communication) .
…
eq TargetState(AgentState(CommitmentDecision, communication), AcceptProposalSent) =
                                                                AgentState(EndI, final) .      ***[4]
eq TargetState(AgentState(CommitmentDecision, communication), RejectProposalSent) =
                                                                AgentState(EndI, final) .

************* Determination of the type of an action ***************************************
eq IsSendingToAll(CFP-Sent) = true .                                                          ***[5]
eq IsSendingToOnlyOne(AcceptProposalSent) = true .

endfm
```

Figure 15 : The module *USE-RCA1* corresponding to the Auctioneer agent.

```
fmod USER-RCA2 is
extending RCA .

****************States of a Bidder*****************************************************
ops StartP OfferEvaluationP WaitingResult EndP : -> NameAgentState .

ops AgentState(StartP, initial) AgentState(OfferEvaluationP, communication)
          AgentState(WaitingResult, UnlimitedWaiting) AgentState(EndP, final) : -> AgentState .

***************Action to accomplish by a Bidder**************************************
ops ReceivingCFP ProposalSent RejectSent ReceivingAcceptance  ReceivingReject : -> Action .

****************Determination of the target state according to a state source and an action *****
eq TargetState(AgentState(StartP, initial), ReceivingCFP) = AgentState(OfferEvaluationP, communication) .
…
eq TargetState(AgentState(WaitingResult, UnlimitedWaiting), ReceivingAcceptance ) = AgentState(EndP, final) .
eq TargetState(AgentState(WaitingResult, UnlimitedWaiting), ReceivingReject ) = AgentState(EndP, final) .

*************** Determination of the type of an action**********************************
eq IsSendingToOnlyOne(ProposalSent) = true .
eq IsSendingToOnlyOne(RejectSent) = true .

endfm
```

Figure 16 : The module *USER-RCA2* corresponding to the Bidder agent.

The *RCA-LINK* module of figure 17, presents a correspondence on the one hand, between the different states of the agents Auctioneer and Bidder and, on the other hand, between the events they exchange. For example, if the Auctioneer agent is in its communication state *SendCFP*, the Bidder must be in its initial state *StartP* (line 1). In the same way, if the Bidder is in its communication state *OfferEvaluationP* (line 3), the Auctioneer must wait its decision. Indeed, an external event for an agent receiver corresponds to a message sent by a sender agent. For example, when the Auctioneer throws a call-for-proposal (*CFP-Sent*), the Bidder agent receives the call-for-proposal event (*ReceivingCFP*). This is expressed by the rule of the line 2. Also, when the Bidder accepts to propose, it sends its proposition (*ProposalSent*), and of the other side, the Auctionner receives its proposition (*ReceivedProposal*) (line 4).

```
fmod RCA-LINK is
protecting RCA1 .
protecting RCA2 .
sort EventType .

ops Internal External : -> EventType .
op CorrespondingState : AgentState -> AgentState .
op CorrespondingAction : Action -> Action .

**********************************Auctioneer Part**********************************

eq CorrespondingState(AgentState(SendingCFP, communication)) = AgentState(StartP, initial) .          ***[1]
eq CorrespondingState(AgentState(CommitmentDecision, communication)) =
                                              AgentState(WaitingResult, UnlimitedWaiting) .
…
eq CorrespondingAction(CFP-Sent) =  ReceivingCFP  .                                                  ***[2]
eq CorrespondingAction(AcceptProposalSent) = ReceivingAcceptance .

**********************************Bidder Part**************************************

eq CorrespondingState(AgentState(OfferEvaluationP, communication)) =
                                                    AgentState(WaitingProposals, limitedWaiting) .   ***[3]
…
eq CorrespondingAction(ProposalSent) = ReceivedProposal .                                            ***[4]

endfm
```

Figure 17 : The module *RCA-LINK*.

6.2 Validation of the Generated Description

The rewriting logic offers a great flexibility in terms of simulation of a specification, in particular, concerning the choice of the initial configuration. This choice plays a primordial role in the validation of the description of a system. Using all the system's description, we can validate a part of the system without involving the rest. For a validation of the AIP given by figure 14, we consider two essential cases: the case where there are Bidders that accept to propose and others do not, and the case where all Bidders refuse to propose. For the first case, we propose the following initial configuration :

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(StartI, initial), AcqList : ("Bidder1" :
                                                                        ("Bidder2" : "Bidder3")) >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
Event("Auctioneer", Initiator, Internal, AgentState(StartI, initial), TrueCondition)
Event("Auctioneer", Initiator, Internal, AgentState(SendingCFP, communication), CFP-Sent)
Event("Bidder1", Participant, Internal, AgentState(OfferEvaluationP, communication), ProposalSent)
Event("Bidder2", Participant, Internal, AgentState(OfferEvaluationP, communication), ProposalSent)
Event("Bidder3", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)  .
```

Figure 18 : Initial configuration.

We define an initial configuration including an agent initiator " Auctionneer ", and three agents participants ("Bidder1", "Bidder2", "Bidder 3"). In the beginning, every agent is in its initial state. From its *OfferEvaluationP* state a Bidder agent can send a proposition as it can refuse to propose. In the configuration of figure 18, Bidder1 and Bidder2 send their propositions whereas Bidder3 refuses to propose while sending a reject. The unlimited rewriting (without indicating the number of the rewriting steps) of this configuration gives the result illustrated by figure 19.

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(EndI, final), AcqList :
                                                        ("Bidder1" :  ("Bidder2" :  "Bidder3") >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
```

Figure 19: Auctioneer and Bidders in their final states.

After it sends a call for proposal to all Bidders, the agent Auctioneer begins to receive the proposal from Bidders agents. Once the considered deadline is expired (internal event) the initiator throws its evaluation process while choosing the most appropriate proposition (here we adopt the strategy based on the first proposing).

So, the Auctioneer sends to the chosen Bidder (here "Bidder1") an acceptance, and to the other (here "Biddert2") a reject. Bidder3 is not concerned because it refused to propose and therefore passes to its final state (see figure 14.b). For the second case, we propose the initial configuration of the following figure:

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(StartI, initial), AcqList :
                                                        ("Bidder1" : ("Bidder2" : “Bidder3”)) >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(StartP, initial), AcqList : "Auctioneer" >
 Event("Auctioneer", Initiator, Internal, AgentState(StartI, initial), TrueCondition)
 Event("Auctioneer", Initiator, Internal, AgentState(SendingCFP, communication), CFP-Sent)
 Event("Bidder1", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)
 Event("Bidder2", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)
 Event("Bidder3", Participant, Internal, AgentState(OfferEvaluationP, communication), RejectSent)  .
```

Figure 20 : Initial configuration.

The configuration of figure 20 looks like the one of figure 18 except that the Bidders refuse to propose. The unlimited rewriting (without indicating the number of the rewriting steps) of this configuration gives the result illustrated by figure 21.

```
< "Auctioneer" : Agent | PlayRole : Initiator, State : AgentState(EndI, final), AcqList :
                                                        ("Bidder1" : ("Bidder2" :  "Bidder3") >
< "Bidder1" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder2" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
< "Bidder3" : Agent | PlayRole : Participant, State : AgentState(EndP, final), AcqList : "Auctioneer" >
```

Figure 21: Auctioneer and Bidders in their final states.

Every participant who refuses to propose passes to the *EndP* state (see figure 14.b). In the same way, the initiator waits for the expiration of the deadline and as it doesn't receive any proposition during this interval of time, it passes on its turn in the *EndP* state (see figure 14.a). Indeed, the configuration of figure 21 seems to be the same that the one of figure 19. It is due to the fact that in the RCA formalism an agent can have only one final state. However, such configurations are different (for example, the *EndP* state of agent Bidder1 in figure 19 is a success state, but in figure 21 such a state presents a failure).

6.3 Implementation

Figure 22 illustrates a part of the code we developed. It visualizes the rewriting rule that describes the reception of an external event by the agent *A1* who plays the *Participant* role and exists in the state *S*. This rule also expresses the transition from the state *S* of the agent *A1* to another target state determined by the function

TargetState(S, Act). The triggering of such a transition only takes place if the agent *A1* is in one of waiting (limited or unlimited) or initial states. This is expressed in this conditional rule by the boolean functions *IsUnlimitedWaiting(S)*, *IslimitedWaiting(S)* and *IsInitial(S)* respectively.

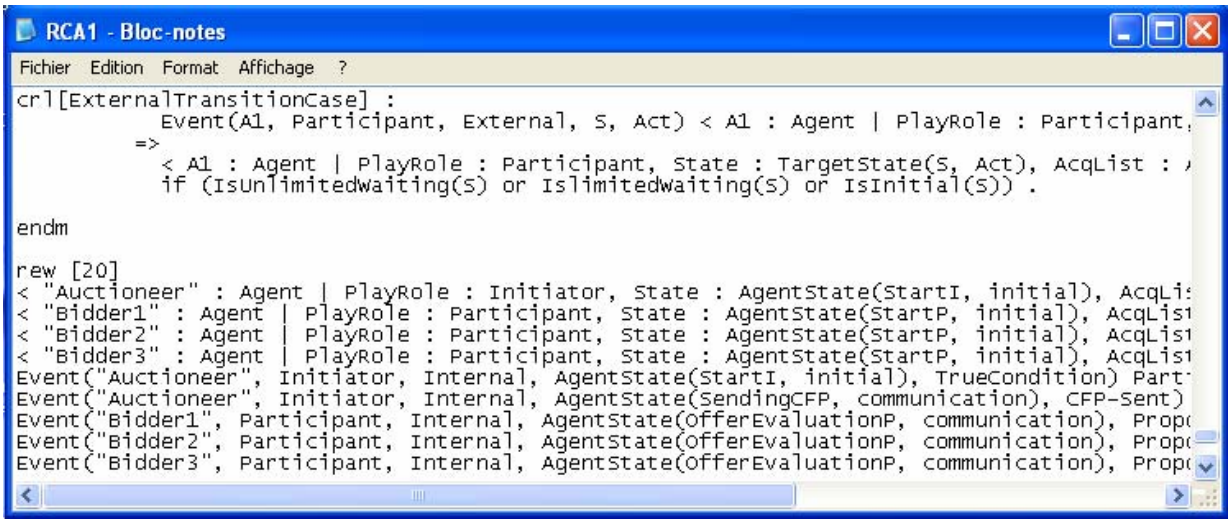

Figure 22 : Part of the developed code.

Furthermore, figure 22 shows the limited rewriting (after 20 rewriting steps) of an initial configuration. In this configuration, we have the agent " Auctioneer " playing the *Initiator* role, and the three agents " Bidder1 ", " Bidder2 " and " Bidder3 " each playing the *Participant* role. All agents are in the departure in their initial states (*StartI* for agent Auctioneer and *StartP* for the Bidders). We suppose, in this initial configuration, that after the sending of the call for proposal by the Auctionner to all Bidders, these last send propositions in the case where they are in state of evaluation of proposal *OfferEvaluationP*. This state is a communication state (see figure 14).

The result of rewriting of such an initial configuration is illustrated by figure 23. The Auctioneer throws its decision process, and all Bidders wait for an answer from it. The agent Auctioneer is in its elementary state *OfferEvaluationI* and all Bidders are in their unlimited waiting states *WaitingResult*.

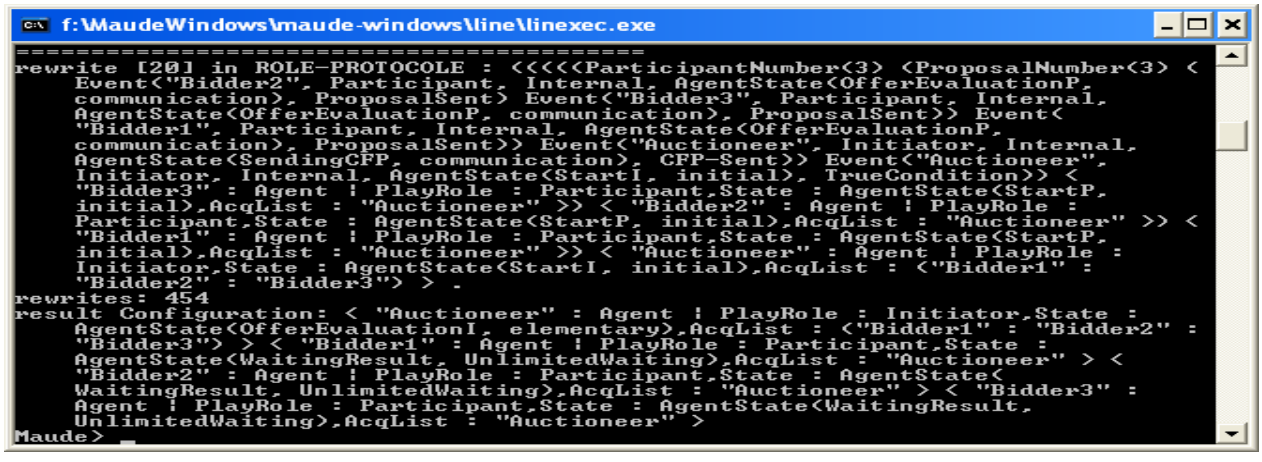

Figure 23 : Result of limited rewriting (after 20 steps) of the initial configuration.

7 Conclusions and Future Work

The RCA formalism allows specifying the roles protocols and is used to describe agents' behavior. Compared to others formalisms, RCA allows recognizing the synchronization points between dual protocols. As for the other existing formalisms, RCA is not endowed yet with a formal semantics [28]. Furthermore, it only allows a partial formalization of MAS [17, 22].

In this article, we proposed a formal framework supporting the translation of interactions between agents, specified using the RCA formalism, in a Maude specification. The translation process is based on the RCA graphs. All the concepts used by the RCA

formalism are supported by Maude. Based on rewriting logic, the formal and object-oriented language Maude supports formal specification and programming for a wide range of applications. The result of the translation procures a formal description of the interactions between agents preserving the consistency in their behavior. It offers a solid basis for their verification and validation process. The generated Maude specifications are flexible and remain open to extension.

Maude is supported by a tool. This allowed us, as a first experiment, in addition to the modeling, to perform a validation (based on a simulation) of our approach. Furthermore, we work on the extension of our approach in order to integrate the possibilities offered by the Maude language (model-checker) to verify some properties of the interactions between agents described using RCA graphs and translated in Maude.

References

[1] Bakam I., Kordon F., Le Page C., Bousquet F. « Formalization of a Spatialized Multiagent Model Using Coloured Petri Nets for the Study of a Hunting Management System ». First International Workshop, FAABS 2000, Greenbelt, MD, USA, April 2000. FAABS 2000.

[2] Bruni R., and Meseguer J., « Generalized rewrite theories ». In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP 2003), volume 2719 of Lecture Notes in Computer Science, pages 252-266. Springer, 2003.

[3] Clavel M., and al. "Maude : Specification and Programming in Rewriting Logic". Internal report, SRI International, 1999.

[4] Cloutier L. «Une approche multi-agents par conventions et contrats pour la coordination de l'entreprise manufacturière réseau », Université de Droit d'Economie et des Sciences d'Aix-Marseille III, DIAM-IUSPIM, Marseille, 1999.

[5] Cost R., and al. «Modeling Agent Conversations with colored Petri Nets», dans Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents'99, Seattle, Washington, mai 1999.

[6] El Fallah-Seghrouchni A., et Mazouzi H., «Une démarche méthodologique pour l'ingénierie des protocoles d'interaction», in. Actes Ingénierie des systèmes multi-agents, JFIADSMA'99, 8-10 novembre 1999, Saint-Gilles, Ile de la Réunion.

[7] Guessoum Z. «Modèles et Architectures d'Agents et de Systèmes Multi-Agents Adaptatifs ». Dossier d'habilitation à diriger des recherches de l'Université Pierre et Marie Curie. Décembre 2003.

[8] Huget M.P., «Model Checking Agent UML Protocol Diagrams ». Technical report ULCS-02-012 from the department of computer science, University of Liverpool. Version 2002/04/16.

[9] Huget M.P., and Odell J. «Representing Agent Interaction Protocols with Agent UML » AAMAS'04, July 19-23, 2004, New York, New York, USA.

[10] Lemaître C., Prat, X., Magnin, L. et Dury A. «Description, programmation et validation d'interactions par Coupled Augmented Transition Network(CATNs) ». In Actes des Secondes Journées Francophones sur les Modèles Formels d'Interactions (MFI'03). Lille, France, 20-23 mai 2003.

[11] Mazouzi H., El fallah Seghrouchni A.,, and Haddad S., «Open protocol design for complex interaction in multi-agent systems ». In Proceedings of the first international joint conference on Autonomous agent and multi-agent systems, pages 517-526. ACM Press, 2002.

[12] McCombs T., «Maude 2.0 Primer, Version 1.0». Internal report, SRI International, 2003.

[13] Meseguer J., «Rewriting as a unified model of concurrency» In Proceedings of the Concur'90 Conference, Amsterdam, Pg 384-400, Springer LNCS Vol. 458, 1990.

[14] Meseguer J., «Logical Theory of Concurrent Objects and its Realization in the Maude Language» In G. Agha, P. Wegner, and A. Yonezawa, Editors, Research Directions in Object-Based Concurrency. MIT Press, 1992.

[15] Meseguer J., «Rewriting Logic and Maude : a Wide-Spectrum Semantic Framework for Object-Based Distributed Systems» In S. Smith and C. L. Talcott, editors, Formal Methods for Open Object-Based Distributed Systems, FMOODS2000, 2000.

[16] Mokhati F., Boudiaf N., Badri L., & Badri M., «Generating Maude Specification from AUML Diagrams: Toward A Systematic Approach». In Proc of CSITeA-04 conference. Cairo, Egypt. December 27-29, 2004.

[17] Mokhati F., Boudiaf N., Badri M., & Badri L., «DIMA-Maude: Toward a Formal Framework for Specifying and Validating DIMA Agents». In Proc of the MOCA'04 conference. Arrhus, Denmark, October 11-13, 2004. pp. 169-187.

[18] Nathalie F., «Modélisation et simulation multi-agents d'écosystèmes antropisés : une application à la gestion hydraulique en grande Camargue», Université de Droit d'Economie et des Sciences d'Aix-Marseille III, IUSPIM-DIAM, Marseille, 2001.

[19] Odell J., Parunak H.V.D., Bauer B., «Representing agent Interaction protocol In UML» conférence AAAI Agents 2000, Barcelone, 3-7 juin 2000.

[20] Odell J., Parunak H. V. D., Bauer B., «Representing agent Interaction protocol In UML», Agent Oriented Software Enginering, Paolo Ciancarini and Michael Wooldridge (eds.), Springer-Verlag, Berlin, 2001, pp. 121-140.

[21] Olveczky P.C., «Modeling and Analyzing Protocols in Maude» 8th Brazilian Symposium on Programming Languages (SBLP'04). May 26-28, 2004.

[22] Paurobally S, Cunningham J., «Achieving Common Interaction Protocols in Open Agent

Environments», 2nd international workshop on Challenges in Open Agent Environments, AAMAS 2003, Melbourne, Australia 14-18th July 2003.

[23] Paurobally S, Cunningham J, and Jennings N R., «Developing Agent Interaction Protocols Using Graphical and Logical Methodologies» in Proc. AAMAS03 PROMAS Workshop on Programming Multi-Agent Systems , 2003.

[24] [24]Paurobally S., Cunningham J., and Jennings, N. R., «Verifying the contract net protocol: a case study in interaction protocol and agent communication semantics». In Proceedings of 2nd International Workshop on Logic and Communication in Multi-Agent Systems, Nancy, France 2004, pp. 98-117.

[25] Pham V. T., Laurent M., Houari S., «Adaptation dynamique des systèmes multi-agents basée sur le concept de méta-CATN». In Actes de la Deuxième Conférence Internationale Associant Chercheurs Vietnamiens et Francophones en Informatique, Hanoï Vietnam, 2-5 Février 2004.

[26] Toivonen S. and al. «Using Interaction Protocols in Distributed Construction Processes». In Seruca, I., Filipe, J., Hammoudi, S., and Cordeiro, J. (Eds.): Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'04), Porto, Portugal, April 2004, pp. 344—349

[27] Tranvouez E., Espinasse B., «Protocoles de coopération pour le réordonnancement d'atelier». In Actes des journées francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA'99) à Saint-Gilles, île de la Réunion, novembre 1999, Gleizes J.-P., Marcenac P., Ed. Hermès, 1999.

[28] Tranvouez E., «IAD et ordonnancement : une approche coopérative du réordonnancement par systèmes mulit-agents». Thèse de doctorat. Université de Droit, d'Economie et des Sciences d'Aix-Marseille III. 2001

[29] Wooldridge M., and al., «A Methodology for Agent-Oriented Analysis and Design». Proc. 3rd Int. Conf. On Autonomous Agents (Agents99), Seatle, WA, 1999.

[30] Wooldridge M., and al., «The gaia methodology for agent-oriented analysis and design». Autonomous Agent and Multi-agent Systems, 3(3):285-312, 2000.

Multi-resolution Parameterization for Texture Classification and Its Use in the Scintigraphic Image Analysis

Luka Šajn
Faculty of computer and information science, Tržaška 25, SI-1000 Ljubljana
Slovenia
E-mail: luka.sajn@fri.uni-lj.si
http://www.fri.uni-lj.si/file/75244/phd-sajn.pdf

Thesis Summary

Keywords: texture analysis, association rules, texture classification, multi-resolution texture parameterization, medical image analysis, scintigraphy analysis, whole-body bone scintigraphy segmentation

Received: July 17, 2007

Article presents the abstract of the dissertation [4] which studies multi-resolution approach for texture parameterization.

Povzetek: Članek predstavlja povzetek disertacije [4], ki preučuje uporabo večresolucijskega pristopa pri parametrizaciji tekstur.

1 Introduction

This dissertation [4, 2] addresses multi-resolution texture parameterization and proposes an original algorithm ARes for finding more informative resolutions in the sense of classification accuracy. ARes is designed to be used in combination with the existing parameterization algorithm ArTex [3]. The results obtained using the ArTex parameterization algorithm in combination with ARes are compared with standard parameterization methods such as Gabor filters, Haar and Laws wavelets and Image Processor. Primary application areas include whole-body bone scintigraphy for pathology detection and heart scintigraphy for diagnosing ishaemic heart disease.

2 Thesis overview

The idea on multi-resolution approach is based on the algorithm SIFT [1]. SIFT is a computer vision algorithm for extracting distinctive features from images, to be used in algorithms for tasks like matching different views of an object or scene and object recognition. The major step in the computation of the image features is scale-space extrema detection, which can be directly applied to the search of resolutions at which a geometric parameterization algorithm captures most textural information inside a certain region. The observed pixel neighborhood size in case of geometric algorithms is limited due to the time and computational complexity. To extract most rules inside a certain region the resolutions at which the most extremes take place should be used. This enables the parameterization algorithm to describe local characteristics which can be covered with the predefined region size.

Our study explores the multi-resolution texture parameterization approach based on the image content with regard to the parameterization quality, especially in case of the ArTex algorithm. ARes finds resolutions at which most local intensity peaks appear. It searches the scale space with a certain step calculated from the attributes of the used parameterization algorithm. ARes is, to our knowledge, the first algorithm to detect resolutions depending on the properties of the learning set for improving the classification quality. The tested parameterization algorithms (geometric algorithms, signal processing methods and statistical methods) using multi-resolution approach have demonstrated significant improvements in results over one scale parameterization. This supports the hypothesis that the resolution selection is important for texture parameterization.

2.1 Applications

For the multi-resolution parameterization applicative domain two medical cases have been used, sequential diagnostics of coronary artery disease (CAD) and diagnostics of whole-body bone scintigraphy.

The whole-body scintigraphy segmentation process is presented which uses reference points detected with local cumulative uptake extremes. Some standard image processing algorithms were tailored and used in combination to achieve the best reference point detection accuracy on scintigraphic images. In order to work satisfactorily, the presence of artifacts, pathologies and poor resolution of scintigraphic images, compared to radiography, requires algorithms to use as much background knowledge on anatomy and spatial relations of bones as possible (see example in fig. 2.1). This combination gives good results

and we expect that further studies on automatic scintigram diagnostics using reference points for image segmentation will improve the accuracy and reliability of results regarding previous approaches.

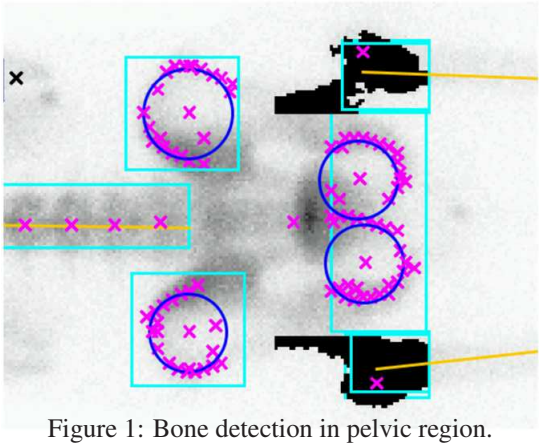

Figure 1: Bone detection in pelvic region.

In the case of coronary artery disease we have shown that multi-resolution ArTex parameterization using machine learning techniques can be successfully used as an intelligent tool for image evaluation, as well as as a part of the sequential diagnostic process. Automatic image parameterization and machine learning methods can help physicians to evaluate medical images and thus improve their combined performance (in terms of accuracy, sensitivity and specificity).

3 Results and conclusion

The developed algorithm ARes in combination with the ArTex algorithm achieves statistically significant improvements over single resolution and also over equidistant resolutions. ARes in many cases also improves the performance of other parameterization algorithms in comparison to single resolution approach, whereas compared to the equidistant resolution approach it usually shows no significant improvement. We have confirmed that the use of the equidistant resolution space when parameterizing textures significantly outperforms the use of the exponential resolution space, which is used by majority of authors.

The presented computer-aided system for bone scintigraphy is a step towards automating the routine medical procedures. This approach can be used as an additional tool for radiologists as it can point out some unregistered pathologies or even give some new insight on the patient condition. The reference point detection is evaluated on a clinical data-set and two methods for bone segmentation using the proposed reference points are presented.

The most significant contribution of our CAD study is the improvement of the predictive power of the sequential diagnostic process. Almost 10% improvement of positive and negative diagnosis of patients who would not need to be examined with costly additional tests, represents a significant contribution in quality and potential rationalization of the existing CAD diagnostic procedures.

References

[1] D. Lowe (2003) Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, pp. 91–110.

[2] L. Šajn, M. Kukar, I. Kononenko, M. Milčinski (2005) Computerized segmentation of whole- body bone scintigrams and its use in automated diagnostics, *Computer Methods and Programs in Biomedicine*, 80(1):47–55.

[3] M. Bevk (2005) *Feature extraction from textures with association rules*, PhD Thesis, University of Ljubljana, Faculty of Computer and Information Science.

[4] L. Šajn (2007) *Multi-resolution parameterization for texture classification and its use in the scintigraphic image analysis*, PhD Thesis, University of Ljubljana, Faculty of Computer and Information Science.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 800 staff, has 600 researchers, about 250 of whom are postgraduates, nearly 400 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit an email with the manuscript to one of the editors from the Editorial Board or to the Managing Editor. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than ten years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://www.informatica.si/

Referees:

Witold Abramowicz, David Abramson, Adel Adi, Kenneth Aizawa, Suad Alagić, Mohamad Alam, Dia Ali, Alan Aliu, Richard Amoroso, John Anderson, Hans-Jurgen Appelrath, Iván Araujo, Vladimir Bajič, Michel Barbeau, Grzegorz Bartoszewicz, Catriel Beeri, Daniel Beech, Fevzi Belli, Simon Beloglavec, Sondes Bennasri, Francesco Bergadano, Istvan Berkeley, Azer Bestavros, Andraž Bežek, Balaji Bharadwaj, Ralph Bisland, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Pavel Brazdil, Bostjan Brumen, Jerzy Brzezinski, Marian Bubak, Davide Bugali, Troy Bull, Sabin Corneliu Buraga, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Rajkumar Bvyya, Giacomo Cabri, Netiva Caftori, Particia Carando, Robert Cattral, Jason Ceddia, Ryszard Choras, Wojciech Cellary, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, Mel Ó Cinnéide, David Cliff, Maria Cobb, Jean-Pierre Corriveau, Travis Craig, Noel Craske, Matthew Crocker, Tadeusz Czachorski, Milan Češka, Honghua Dai, Bart de Decker, Deborah Dent, Andrej Dobnikar, Sait Dogru, Peter Dolog, Georg Dorfner, Ludoslaw Drelichowski, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Marjan Družovec, Jozo Dujmović, Pavol Ďuriš, Amnon Eden, Johann Eder, Hesham El-Rewini, Darrell Ferguson, Warren Fergusson, David Flater, Pierre Flener, Wojciech Fliegner, Vladimir A. Fomichov, Terrence Forgarty, Hans Fraaije, Stan Franklin, Violetta Galant, Hugo de Garis, Eugeniusz Gatnar, Grant Gayed, James Geller, Michael Georgiopolus, Michael Gertz, Jan Goliński, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Jozsef Gyorkos, Marten Haglind, Abdelwahab Hamou-Lhadj, Inman Harvey, Jaak Henno, Marjan Hericko, Henry Hexmoor, Elke Hochmueller, Jack Hodges, John-Paul Hosom, Doug Howe, Rod Howell, Tomáš Hruška, Don Huch, Simone Fischer-Huebner, Zbigniew Huzar, Alexey Ippa, Hannu Jaakkola, Sushil Jajodia, Ryszard Jakubowski, Piotr Jedrzejowicz, A. Milton Jenkins, Eric Johnson, Polina Jordanova, Djani Juričič, Marko Juvancic, Sabhash Kak, Li-Shan Kang, Ivan Kapustøk, Orlando Karam, Roland Kaschek, Jacek Kierzenka, Jan Kniat, Stavros Kokkotos, Fabio Kon, Kevin Korb, Gilad Koren, Andrej Krajnc, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Sofiane Labidi, Les Labuschagne, Ivan Lah, Phil Laplante, Bud Lawson, Herbert Leitold, Ulrike Leopold-Wildburger, Timothy C. Lethbridge, Joseph Y-T. Leung, Barry Levine, Xuefeng Li, Alexander Linkevich, Raymond Lister, Doug Locke, Peter Lockeman, Vincenzo Loia, Matija Lokar, Jason Lowder, Kim Teng Lua, Ann Macintosh, Bernardo Magnini, Andrzej Małachowski, Peter Marcer, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Daniel Memmi, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Armin R. Mikler, Gautam Mitra, Roland Mittermeir, Madhav Moganti, Reinhard Moller, Tadeusz Morzy, Daniel Mossé, John Mueller, Jari Multisilta, Hari Narayanan, Jerzy Nawrocki, Rance Necaise, Elzbieta Niedzielska, Marian Niedq'zwiedziński, Jaroslav Nieplocha, Oscar Nierstrasz, Roumen Nikolov, Mark Nissen, Jerzy Nogieć, Stefano Nolfi, Franc Novak, Antoni Nowakowski, Adam Nowicki, Tadeusz Nowicki, Daniel Olejar, Hubert Österle, Wojciech Olejniczak, Jerzy Olszewski, Cherry Owen, Mieczyslaw Owoc, Tadeusz Pankowski, Jens Penberg, William C. Perkins, Warren Persons, Mitja Peruš, Fred Petry, Stephen Pike, Niki Pissinou, Aleksander Pivk, Ullin Place, Peter Planinšec, Gabika Polčicová, Gustav Pomberger, James Pomykalski, Tomas E. Potok, Dimithu Prasanna, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Vojislav D. Radonjic, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Kai Rannenberg, Wolf Rauch, Peter Rechenberg, Felix Redmill, James Edward Ries, David Robertson, Marko Robnik, Colette Rolland, Wilhelm Rossak, Ingrid Russel, A.S.M. Sajeev, Kimmo Salmenjoki, Pierangela Samarati, Bo Sanden, P. G. Sarang, Vivek Sarin, Iztok Savnik, Ichiro Satoh, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Dennis Sewer, Zhongzhi Shi, Mária Smolárová, Carine Souveyet, William Spears, Hartmut Stadtler, Stanislaw Stanek, Olivero Stock, Janusz Stokłosa, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Leon Strous, Ron Sun, Tomasz Szmuc, Zdzislaw Szyjewski, Jure Šilc, Metod Škarja, Jiří Šlechta, Chew Lim Tan, Zahir Tari, Jurij Tasič, Gheorge Tecuci, Piotr Teczynski, Stephanie Teufel, Ken Tindell, A Min Tjoa, Drago Torkar, Vladimir Tosic, Wieslaw Traczyk, Denis Trček, Roman Trobec, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Tadeusz Usowicz, Romana Vajde Horvat, Elisabeth Valentine, Kanonkluk Vanapipat, Alexander P. Vazhenin, Jan Verschuren, Zygmunt Vetulani, Olivier de Vel, Didier Vojtisek, Valentino Vranić, Jozef Vyskoc, Eugene Wallingford, Matthew Warren, John Weckert, Michael Weiss, Tatjana Welzer, Lee White, Gerhard Widmer, Stefan Wrobel, Stanislaw Wrycza, Tatyana Yakhno, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Ales Zivkovic, Zonling Zhou, Robert Zorc, Anton P. Železnikar

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2007 (Volume 31) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printing: Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 31 Number 3 October 2007 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **31** (2007) Number 3, pp. 249–353