A framework for XML Data Streams History Checking and Monitoring

Alessandro Campi Paola Spoletini
Politecnico di Milano, Dipartimento di Elettronica e Informazione
P.zza Leonardo da Vinci 32, Milano, Italy
campi|spoleti@elet.polimi.it

ABSTRACT

The need of formal verification is a problem that involves all the fields in which sensible data are managed. In this context the verification of data streams became a fundamental task. The purpose of this paper is to present a framework, based on the model checker SPIN, for the verification of data streams.

The proposed method uses a linear temporal logic, called TRIO, to describe data constraints and properties. Constraints are automatically translated into Promela, the input language of the model checker SPIN in order to verify them.

Categories and Subject Descriptors

D.2.2 [**Design Tools and Techniques**]: User interfaces; D.2.4 [**Software/Program Verification**]: Model checking

General Terms

Verification

Keywords

XML, semi-structured data, verification

1. INTRODUCTION

In many applications, data may take the form of data streams. Several aspects of data management need to be reconsidered in the presence of data streams, offering new research directions. In this paper we focus primarily on the problem of defining a framework that implements methods to verify properties on data streams.

Typically data streams are XML data flowing throughout the net (streams of stock quotes, of medical data,...). In particular XML dialects for financial data spread out in the last years: FinXML [2], ebXML [1], and FpML [3] are some examples of these standards. Especially in this context we need to specify very complex business rules regarding XML streams. Such rules predicate on historical trends of chosen values. At the state of the art those rules are checked only by ad-hoc applications, which correctness and reliability are related only to the test cases run. The need to cover the gap between the state of art and the importance of having techniques to guarantee the correctness of streams has caused an increasing interest on formal methods.

However the application of verification techniques in such a context it is not an easy task. The temporal description need not only to be able to describe the ordering among events, but also to describe specific and detailed temporal constraints among data. For these reasons we consider as specification language a first order linear time temporal logic called TRIO [5].

```
<stock date="2005_08_24">
  <share name="Google" price="12.678"
         minPrice="12.478" maxPrice="12.900"
         quantity="4000"/>
  <share name="IBM" price="12.279"
         minPrice="12.125" maxPrice="12.698"
         quantity="8000"/>
</stock>
<exchanges>
<exchange seller="E4566" buyer="E5667" name="HP"
          timestamp="2005_08_26_10_01_34"
          price="25900" quantity="499"/>
<exchange seller="E3333" buyer="E4535" name="IBM"
          timestamp="2005_08_26_10_03_34"
          price="12.675" quantity="199"/>
</exchanges>
```

Figure 1: Running example

Hence the basic idea is that the specification of the system in TRIO and generated the translation in Promela, the input language of SPIN, the framework states if the data already present satisfy the specification; if not the model checker shows where the specification is violated.

2. VERIFICATION WITH SPIN

Our approach to model history checking data, whose specifications is given in TRIO, is based on the translation of the TRIO formulae into Promela explained in details in [6, 4].

In figure 1 we show stock performance data. Information contained in the `stocks` element describe the performance data at the beginning of the day. Instead, data in the `exchanges` element represent the stream of stock sales during the day. Given this scenario, we want to monitor these business rules: (a) between the buying and the selling of stocks of the same society should be at least a fixed temporal distance, (b) during the day the gap between the minimum price and the maximum price cannot be greater than a fixed limit. The translation in TRIO of the properties is the following:

(a)$Alw(\forall p \forall a(buy(p,a) + \sum_{t \in Day} past(buy(p,a),t) < k))$

(b)$Alw(\forall t_1, t_2, a(|past(price(a),t_1) - past(price(a),t_2)| \leq k))$

The formulae are defined on the following domains: p is on the domain of persons, a takes value in the possible stocks and t_i on the timestamps.

Now if we consider the data shown in Figure 1 and we add two channels to connect them to the acceptor automaton, one for the stock information and one for the exchange in-

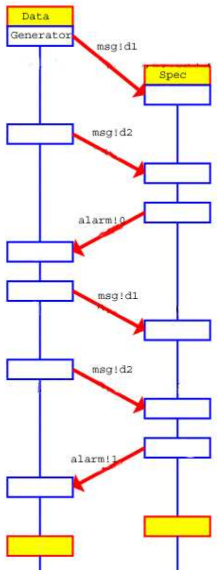

Figure 2: The output of the SPIN model checker

formation, we can verify if the data stream is conform to the specification. In this example we obtain the counterexample shown in a visual representation in Figure 2. The first process (the one on the left) represents the history manager and the second one (on the right) the specification. Each time instant the history manager sends to the specification data d_1 and d_2 through the channel message. After evaluating the data, the specification sends to the history manager the value of the alarm signaling anomalies with respect to the received data. In this example at the second time instant the sent data violates the specification.

3. ARCHITECTURE

Our framework is fully implemented in a tool environment consisting of about 30 Java classes. Figure 3 shows the architecture of our framework.

The **XML data** block represents the data stream to be monitored. These data are processed by the **Data translator**, a component formed by an ad-hoc XML parser and a generator of Promela **finite history** (the generated code contains the channels simulating the data stream). The **TRIO formulae** represent the constraints to be checked by the framework. These formulae are the input of the **Promela translator** [4]. The tranlation generates the **Language recognizer**: a piece of Promela code devoted to verify properties. This piece of Promela is coupled with the Promela representation of data by the **Model generator** in order to obtain the **Finite operational model**. The analysis conducted by the **SPIN model checker** allows to identify the instant in which a violation is performed. The output of the SPIN model checker is parsed by a tool tailored to show the results in a way comprehensible for not skilled

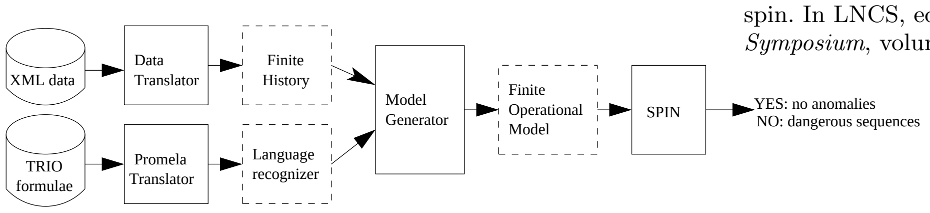

Figure 3: Architecture of the entire system

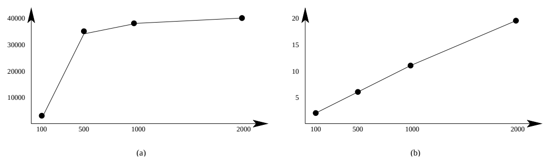

Figure 4: Depth reached (a) and memory usage (b)

users. The visualization of the result is strictly coupled by the semantics of the data stream. In order to facilitate the rapid creation of new interfaces, the visualization of the result is parametric and is described in a configuration file very easily to manage.

We now present some experiments conducted on a series of XML data streams (in the form presented in Figure 1), varying in size from 100 to 2000 data (generated re-mapping data from a real stock quotation Web site) in order to evaluate the performance of our approach. We refer to the constraints of example (a) previously presented. Figure 4a shows the time needed to verify a constraint after each arrival of a new message from the stream, Figure 4b shows the required memory. Indexes are indicated on y-axis (depth in milliseconds, memory in MB) and represent the average of the measured times of 100 attempts for each experiment. These results show that our approach is feasible also for a considerable size of data streams. We observe that, in our analysis, we do not have to take into account the time spent to produce the Promela version of the constraint, because it is generated only once at schema design time and thus do not interfere with run time performance. On the contrary the time needed to translate data from XML to Promela is considered.

4. CONCLUSIONS AND FUTURE WORKS

In this work, we proposed a framework for the history checking of the data based on TRIO, a linear temporal logic, that can be used to describe data constraints and properties. We showed how the constraints can be verified over data streams. The constraints are automatically translated in Promela, the input language of SPIN, to verify the properties.

5. REFERENCES

[1] ebXML. http://www.ebxml.org.

[2] Finxml home page. http://www.finxml.org.

[3] Fpml. http://xml.coverpages.org/fpml.html.

[4] Verification of temporal logic via SPIN automata, PhD thesis. www.elet.polimi.it/upload/spoleti/ PhDThesis_PaolaSpoletini.pdf.

[5] C. Ghezzi, D. Mandrioli, and A. Morzenti. Trio: A logic language for executable specifications of real-time systems. *The Journal of Systems and Software*, 12(2):107–123, May 1990.

[6] A. Morzenti, M. Pradella, P. San Pietro, and P. Spoletini. Model checking of trio specifications in spin. In LNCS, editor, *Proc. of 12th International FME Symposium*, volume 2805, Sep 2003.