Submitted to the International Conference on Rewriting Techniques and Applications
http://rewriting.loria.fr/rta/

AUTOMATA FOR DATA WORDS AND DATA TREES

MIKOŁAJ BOJAŃCZYK

University of Warsaw
E-mail address: bojan@mimuw.edu.pl
URL: www.mimuw.edu.pl/∼bojan

Abstract. Data words and data trees appear in verification and XML processing. The term “data” means that positions of the word, or tree, are decorated with elements of an infinite set of data values, such as natural numbers or ASCII strings. This talk is a survey of the various automaton models that have been developed for data words and data trees.

A *data word* is a word where every position carries two pieces of information: a *label* from a finite alphabet, and a *data value* from an infinite set. A data tree is defined likewise.

As an example, suppose that the finite alphabet has two labels `request` and `grant`, and the data values are numbers (interpreted as process identifiers). A data word, such as the one below, can be seen as log of events that happened to the processes.

request	request	request	grant	request	grant	request	request	. . .
1	2	1	1	3	3	7	7	

The example with processes and logs can be used to illustrate how data words are used in verification. In one formulation, verification is a decision problem with two inputs: a correctness property, and a scheduling mechanism. The correctness property is some set K of “correct” logs. Often in verification, one talks about infinite words. For example, we might be interested in the following liveness property:

> “For every position with label `request`, there exists a later position with label `grant` and the same data value.”

The scheduling mechanism is represented as the set L of logs consistent with the mechanism. For example, a rather unwise scheduler would result in the following logs:

> “Every position with label `grant` carries the same data value as the most recent position with label `request`.”

The verification problem is the question: does the scheduling mechanism guarantee the correctness property? (In the example given here, the answer is no.) In terms of languages, this is the question if the difference $L - K$ is nonempty. To complete the description of the

1998 ACM Subject Classification: PREFERRED list of ACM classifications.

Key words and phrases: Automata, Infinite State Systems.

Work supported by the FET programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599.

problem, one should indicate some way of representing the languages L and K; a typical solution is to use some variant of logic or automata.

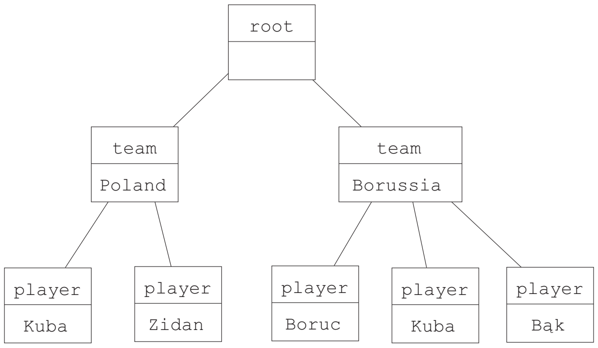
Another application of data is in XML. Here, (finite) data trees are the pertinent object. The data values are used to model the text content in an XML document, while the labels are used to model tag names, as illustrated in the following example. The finite alphabet describes three possible tag names: `root`, `team` and `player`. The data values are ASCII strings that describe team names and player names. The picture below depicts an XML document and its interpretation as a data tree.

```
<root>

<team> Borussia
   <player> Kuba </player>
  <player> Zidan </player>
</team>

<team> Poland
  <player> Boruc </player>
  <player> Kuba </player>
  <player> Bąk </player>
</team>

</root>
```



One might want to express correctness properties of an XML document, such as "no player is a member of two different teams". A typical algorithmic problem would concern the relation between two correctness properties, such as: does correctness property L imply correctness property K? This is a problem like the one in the verification example, although the logics used for describing XML documents usually have a different flavor than the logics for describing behavior of processes. Another algorithmic problem is to query documents, such as "find the nodes that describe a player who plays in two different teams". For querying, algorithms should be very fast, for instance linear in the document size. For verification, sometimes even decidability is hard to get.

The problems described above are well understood in the data-free setting, where positions carry only labels and not data values. Automata techniques have been highly successful in this area.

What about data? What is the right automaton model for data words and data trees? Recent years have seen a lot of work in this direction, with many incompatible definitions being proposed. Some of the approaches are listed in the references. Which one is the right one? What is a "regular language" in the presence of data? We do not know yet, and maybe we never will. It is difficult (indeed, impossible, under a certain formulation) to design an automaton model that is robust (the languages recognized have good closure properties, such as boolean operations and projections), expressive (captures some reasonable languages, such as "all positions with label a have the same data value") and decidable (e.g. has decidable emptiness). Undecidable problems, like the Post Embedding Problem, can be easily encoded in data words, in many different ways.

The talk surveys the varied landscape of automata models for data words and data trees. I will talk about the technical aspect of deciding emptiness, including the connection with vector addition systems (Petri Nets), as well as the connection with well-quasi-orders.

I will also talk about the technical aspect of efficient evaluation, including the connection with semigroup theory.

References

[1] Michael Benedikt, Wenfei Fan, and Floris Geerts. XPath satisfiability in the presence of DTDs. In *PODS*, pages 25–36, 2005.

[2] H. Björklund and T. Schwentick. On notions of regularity for data languages. In *FCT*, pages 88–99, 2007.

[3] Mikołaj Bojańczyk, Sławomir Lasota. An extension of data automata that captures XPath . To appear in *LICS*, 2010.

[4] Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3), 2009.

[5] Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *LICS*, pages 7–16, 2006.

[6] Mikołaj Bojańczyk, Paweł Parys. Efficient evaluation of nondeterministic automata using factorization forests. To appear in *ICALP*, 2010.

[7] Stéphane Demri and Ranko Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[8] Diego Figueira. Forward-XPath and extended register automata on data-trees. In *ICDT*, 2010. To appear.

[9] Marcin Jurdziński and Ranko Lazić. Alternation-free modal mu-calculus for data trees. In *LICS*, pages 131–140, 2007.

[10] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[11] Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, pages 41–57, 2006.