Reasoning about data in a simple process algebra

Robert Meolic, Tatjana Kapus, Zmago Brezočnik
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ul. 17, SI-2000 Maribor, Slovenia
{meolic,kapus,brezocnik}@uni-mb.si

Abstract

Various process algebras have been introduced for reasoning about concurrent systems. Some of them include explicit data-passing mechanisms, while others do not. This paper presents a non-trivial problem involving data, which can also be comprehensively solved with a simple process algebra without explicit data-passing. The presented problem is very suitable for comparing the strength of different process algebra software tools.

1 Introduction

Process algebras have become widely used formalisms in the verification of concurrent systems. Most likely, their success results from the algebraic nature of their system descriptions. A number of operators have been suggested for process algebras, among them many different kinds of process compositions and abstractions. Moreover, many preorders and equivalence relations have been proposed and model checking has been successfuly adapted to process algebras specifications [1, 8].

Most popular process algebras origin in Milner's CSS (Calculus of Communicating Systems) and Hoare's CSP (Communicating Sequential Processes). The behaviour of concurrent systems is described by a set of processes, where a process is an entity which performs actions and transits between its internal states. Processes can synchronise with each other by simultaneously performing synchronisation actions. A great weakness of such basic process algebras is that they cannot explicitly deal with data. This causes many expressivity problems if one tries to describe not just trivial systems. Therefore, it is useful to extend process algebras with the possibility of data-passing during process synchronisation [2, 3]. However, the explicit introduction of data complicates the algorithms for automatic verification considerably.

Further in this paper, we introduce a problem which we called the Euro 2000 problem. The problem involves data, but it can also be comprehensively solved with a simple process algebra without explicit data-passing. In Section 3 we present the process algebra and software package EST (Efficient Symbolic Tools) which we used for reasoning about the given problem. Section 4 gives our solution and the results we got with EST. In a short conclusion we talk about our future research directions.

2 The EURO 2000 problem

The European football championship EURO 2000 held in Belgium and Netherlands will serve us as an idea for a concurrent system. More precisely, only the matches played in group C will be observed. In each group, four teams played one match with each other. After each team had played all three matches, two teams with the most points continued to play in the quarterfinal of the championship, while the other two teams went home. If two or more teams had the same number of points, then other criteria were used to determine the final order of teams.

In group C, Slovenia, Yugoslavia, Spain, and Norway competed. There was a very interesting situation in this group and nobody really dared to forecast the two teams which would be the lucky ones. Slovenia and Norway are small countries in the football sense and they participated in the European football championship for the first time. Yugoslavia had a lot of good football players, but the quality of the team on the whole was unknown. The team from Spain was favourite, but football is a very unpredictable game.

Informally, the problem proposed as a benchmark study can be given as follows. The concurrent system consists of four football teams. In the first part of system execution, teams play a match with each other. Each match can end with a winner or in a tie. The winner of a match gets 3 points. If there is no winner, each team gets 1 point. In the second part of system execution, the points of all four teams are compared. Each team gets a label which reflects its place in the table. We take that if two teams have the same number of points, any of them can be the better one. Properties of the described system that are interesting for us are:

- possible quadruples of points in the final table,
- the minimal number of points that can be enough to place in the quarterfinal,
- the minimal number of points and special situations which guarantee placing in the quarterfinal etc.

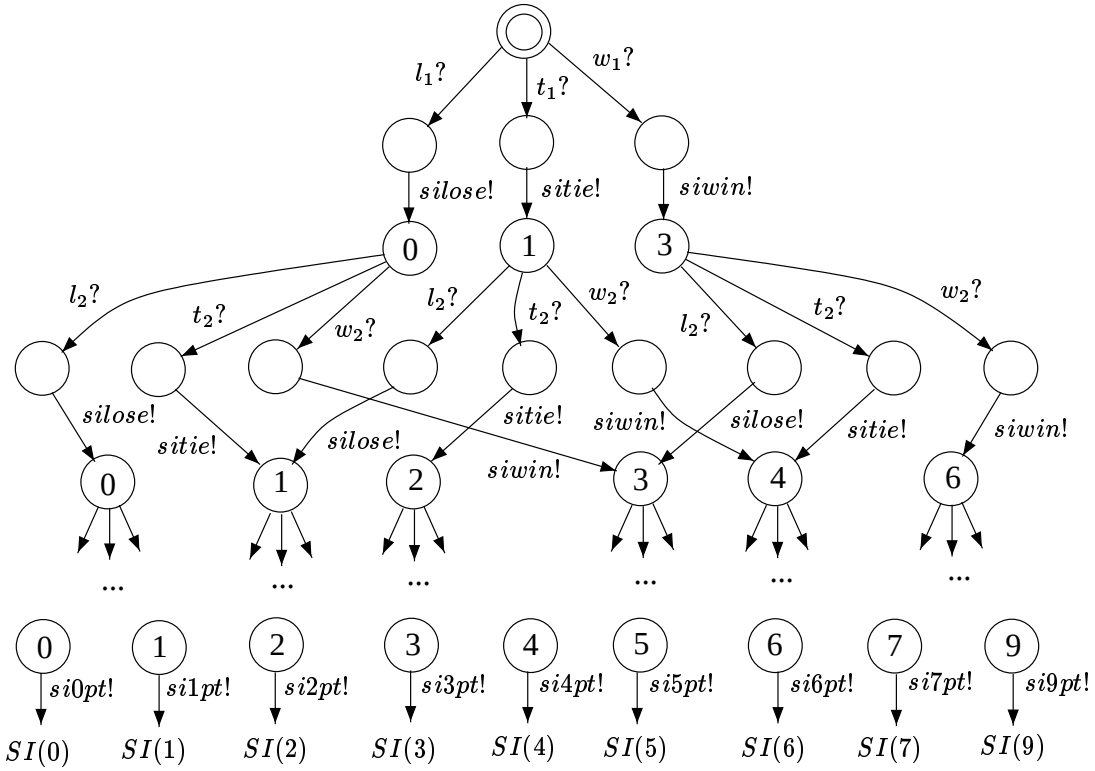

Figure 1: The structure of process representing the Slovenia team

3 Efficient Symbolic Tools package

Efficient Symbolic Tools (EST) is a software tool for formal verification of concurrent systems [4, 5, 7]. It is a small and efficient package with a simple user interface and readable source code. It runs on many different computers with different operating systems, including Linux and Windows 95/98/NT. EST uses symbolic methods to represent and manipulate processes. States and transitions are encoded by Boolean functions rather that being explicitly enumerated. Further, Boolean functions are represented with binary decision diagrams (BDDs).

The system being verified with EST is represented by a set of communicating processes. A formalism used for describing processes is derived from CCS and CSP. Processes are executed asynchronously. Each process transits between states and performs input and output actions. An action is an input action if its name terminates by '?', and it is an output action if its name terminates by '!'. Two actions whose names differ only in the terminating symbol, e.g. $a?$ and $a!$, are called complementary actions. Two processes synchronise with each other by simultaneously performing complementary actions. Processes may include nondeterminism and the special action τ, which is used to model internal transitions.

The major facilities of EST are parallel composition of processes, checking of observational equivalences, checking of testing equivalence, and ACTL model checking. Among them the latter one will be especially interesting for us because model checking allows reasoning about specific system properties. Action computation tree logic (ACTL) is a propositional branching-time temporal logic [6, 8, 9]. It is similar to the popular CTL and it has all the nice characteristics of it, including the feasibility of efficient symbolic model checking. The syntax of ACTL formulas consists of constants *true* and *false*, *action variables*, standard Boolean operators **OR**, **AND**, and **NOT**, *path quantifiers* **E** ("there exists a path") and **A** ("for all paths"), and *temporal operators* **U** ("until"), **U** or **UU** ("unless"), **X** ("for the next transition"), **F** ("for some transition in the future"), and **G** ("for all transitions in the future"). There are also two common abbreviations: $< \alpha > \varphi$ ("there exists a transition with action α leading to a state where ACTL formula φ is valid") and $[\alpha]\varphi$ ("all transitions with action α lead to a state where ACTL formula φ is valid").

4 Our results

The simple process algebra recognised by EST does not support data-passing. Therefore, we annotated data in action names. We facilitated our work by introducing hierachical process description, where each process can include references to other processes.

The first part of the EURO 2000 problem is represented by four processes, one for each team. The structure of process representing the Slovenia team is shown in Figure 1. The process represents sequentiall playing of three

matches. Because of limited space we cannot present the whole process. Note that the state labels shown in Figure 1 are only comments about obtained points and not the real labels, which have to be unique. After a process executes the first part of its task, it does not stop. With regard to the number of obtained points the process continues with execution of one of 9 possible processes $SI(0), SI(1), ..., SI(9)$.

The processes representing different teams differ slightly from each other. The easiest way to explain their structure is to give the sequences of actions each process can execute. Let $\{x_1, x_2\}, \{y_1, y_2\}$ denote the pattern which is satisfied by a sequence of two actions, where the first action is x_1 or x_2 and the second action is y_1 or y_2. Then, the sequences of actions which are possible in our processes satisfy the following patterns:

1. Slovenia: $\{w_1?, t_1?, l_1?\}, \{silose!, sitie!, siwin!\}$, $\{w_2?, t_2?, l_2?\}, \{silose!, sitie!, siwin!\}, \{w_3?$, $t_3?, l_3?\}, \{silose!, sitie!, siwin!\}, \{si0pt!, si1pt!$, $si2pt!, si3pt!, si4pt!, si5pt!, si6pt!, si7pt!, si9pt!\}$
2. Yugoslavia: $\{w_1!, t_1!, l_1!\}, \{yulose!, yutie!, yuwin!\}$, $\{w_2?, t_2?, l_2?\}, \{yulose!, yutie!, yuwin!\}, \{w_3?, t_3?$, $l_3?\}, \{yulose!, yutie!, yuwin!\}, \{yu0pt!, yu1pt!$, $yu2pt!, yu3pt!, yu4pt!, yu5pt!, yu6pt!, yu7pt!, yu9pt!\}$
3. Spain: $\{w_2!, t_2!, l_2!\}, \{eslose!, estie!, eswin!\}$, $\{w_2!, t_2!, l_2!\}, \{eslose!, estie!, eswin!\}, \{w_3?, t_3?$, $l_3?\}, \{eslose!, estie!, eswin!\}, \{es0pt!, es1pt!$, $es2pt!, es3pt!, es4pt!, es5pt!, es6pt!, es7pt!, es9pt!\}$
4. Norway: $\{w_3!, t_3!, l_3!\}, \{nolose!, notie!, nowin!\}$, $\{w_3!, t_3!, l_3!\}, \{nolose!, notie!, nowin!\}, \{w_3!, t_3!$, $l_3!\}, \{nolose!, notie!, nowin!\}, \{no0pt!, no1pt!$, $no2pt!, no3pt!, no4pt!, no5pt!, no6pt!, no7pt!, no9pt!\}$

The second part of the EURO 2000 problem consists in sorting teams with regard to obtained points (Figure 2). Each process participates in comparisons which lead to the final ordering. The teams are arranged into pairs A (Slovenia and Yugoslavia) and B (Spain and Norway). First, the points of the teams in a pair are compared. The team which is better becomes the winner (state W), while the other becomes the looser (state L). Because there is no data-passing in our process algebra, we need many transitions for one comparison. For example, in Figure 2 representing compactly processes $SI(0), ..., SI(9)$ we drew a triple arrow with label $A_y?|_{y<x}$. Actually, this means that in process $SI(x)$ there is one transition $A_y?$ for each value $y, y<x$, where x is the number of points the Slovenia team obtained and y is the number of points which other team in pair A obtained. After the points are compared in pairs, the points of the winners and the points of the loosers are compared. The teams which are both times the winner or the looser take the 1st place and the 4th place in the table, respectively. The points of the other two teams which were once a winner and once a looser are compared again to determine the 2nd and 3rd place.

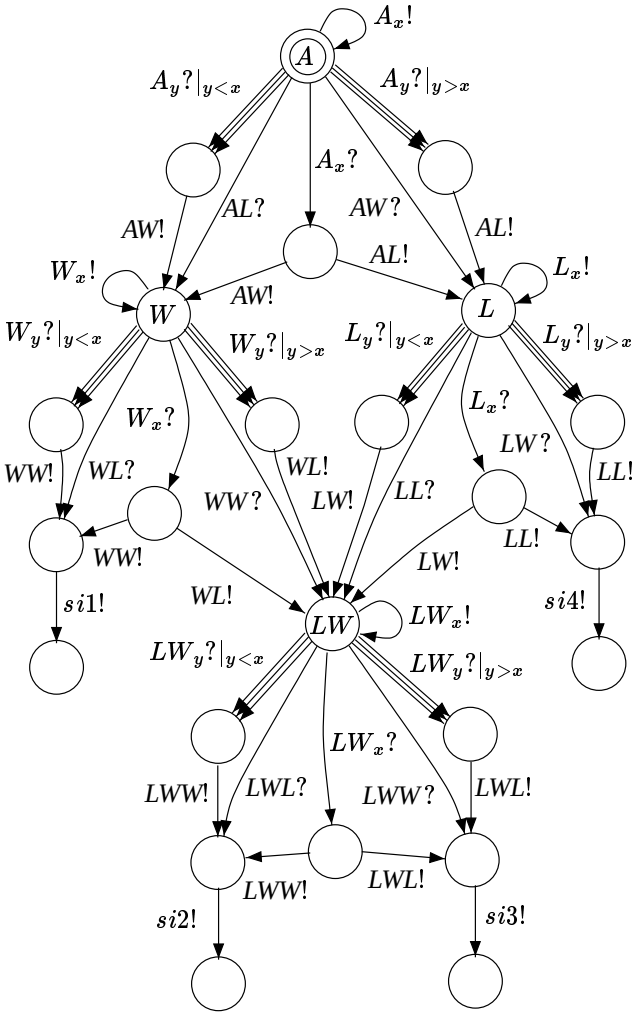

Figure 2: Process $SI(x)$

Although we used a simple process algebra without data-passing, the presented solution of the problem enabled us to find out many interesting properties of the system. We started with composing together all four processes representing the Slovenia, Yugoslavia, Spain, and Norway team. Then, the behaviour of compound system was investigated with ACTL model checking.

First, we checked the correctness of our specification by checking some properties about the number of points and the final place which the Slovenia team could obtain.

1. Certainly, the Slovenia team will play all its matches:
AF $\{si0pt!$ **OR** $si1pt!$ **OR** $si2pt!$ **OR** $si3pt!$ **OR** $si4pt!$ **OR** $si5pt!$ **OR** $si6pt!$ **OR** $si7pt!$ **OR** $si9pt!\}$;

2. The Slovenia team may obtain 0 points: **EF** $\{si0pt!\}$;

3. Certainly, the Slovenia team will be ordered in a table:
AF $\{si1!$ **OR** $si2!$ **OR** $si3!$ **OR** $si4!\}$;

4. The Slovenia team may take the first place: **EF** $\{si1!\}$;

5. If the Slovenia team is placed first, then other teams will not be placed first:
AG $[si1!]$ **NOT EF** $\{yu1!$ **OR** $es1!$ **OR** $no1!\}$;

To check whether some quadruple of points is possible in the final table, we used the following kinds of formulas:

6. Eventually, Slovenia may have 2 points, Yugoslavia 4 points, Spain 6 points, and Norway 4 points: **EF** $< si2pt! >< yu4pt! >< es6pt! >< no4pt! >$ **TRUE**;

We were able to check many different properties about how the success in matches and the number of obtained points affect the final place. Here are some of them:

7. 2 points may be enough to be placed in the quarterfinal: **EF** $\{si2pt!\}$**EF** $\{si1!$ **OR** $si2!\}$;

8. Always, 7 points are enough to be placed in the quarterfinal: **AG** $[si7pt!]$**AF** $\{si1!$ **OR** $si2!\}$;

9. If Spain obtains at least 7 points and Slovenia obtains 5 points, then Slovenia will surely be placed in the quarterfinal: **AG** $[es7pt!$ **OR** $es9pt!]$ **AG** $[si5pt!]$ **AF** $\{si1!$ **OR** $si2!\}$;

10. If Slovenia defeats Yugoslavia and draws other two matches and if Yugoslavia obtains at least 1 point, then Slovenia will surely be placed in the quarterfinal: **AG** $[siwin!]$ **AG** $[sitie!]$ **AG** $[sitie!]$**AG** $[yu1pt!$ **OR** $yu2pt!$ **OR** $yu3pt!$ **OR** $yu4pt!$ **OR** $yu5pt!$ **OR** $yu6pt!]$ **AF** $\{si1!$ **OR** $si2!\}$.

Results of ACTL model checking are shown in Figure 3. We performed our tests on an overclocked Pentium II 266 with 128 MB RAM and Linux operating system. The program was allowed to have at most 1250000 BDD nodes at once. The total memory consumption during the testing never exceeded 64 MB.

ACTL formula	Result	Time (s)
Formula 1	TRUE	46.4
Formula 2	TRUE	0.3
Formula 3	TRUE	45.9
Formula 4	TRUE	13.7
Formula 5	TRUE	38.7
Formula 6	TRUE	0.7
Formula 7	TRUE	26.4
Formula 8	TRUE	53.8
Formula 9	TRUE	57.5
Formula 10	TRUE	65.4

Figure 3: The results of the EURO 2000 problem

The following results are also interesting as they describe the verification in more detail. Each process representing a team consisted of 238 states and 753 transitions. To represent transition relations we needed about 1650 BDD nodes for each of them. EST spent 0.7 s for encoding all four processes. After parallel composition of all four processes we got a process with 201599 states and 560194 transitions (410444 transitions with actions other than τ). To represent its transition relation we needed 266751 BDD nodes. EST spent 44.6 s for performing parallel composition.

5 Conclusion

This paper presents an interesting study of using a process algebra for reasoning about the behaviour of concurrent systems with data. Although the process algebras with explicit data-passing seem to be the best choice for problems involving data-passing, data comparison, data sorting etc., we showed that simple process algebras can also be succesfully used for such problems as many expressivity problems can be overcome with annotating data in action names. We believe that both solutions, with and without explicit data-passing, have their own advantages. Unfortunately, for the time being we cannot compare these two approaches on the EURO 2000 problem, because we do not have results from a process algebra tool with explicit data-passing yet. This remains for us and maybe other researchers to be done in the future.

References

[1] C. Bernardeschi et al. A Formal Verification Environment for Railway Signaling System Design. *Formal Methods In System Design*, 12(2):139–161, 1998.

[2] J. F. Groote and M. A. Reniers. *Algebraic process verification*. Technical report CSR-00-05.

[3] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value passing. *Information and Computation*, 107(2):202–236, December 1993. 17th International Colloquium on Automata, Languages and Programming, Warwick University, England, 1990, LNCS 443.

[4] R. Meolic. EST Home Page. `http://www.el.feri.uni-mb.si/est/`.

[5] R. Meolic. Checking correctness of concurrent systems behavour. Master's thesis, Faculty of Electrical Engineering and Computer Science, Maribor, November 1999. In Slovene.

[6] R. Meolic, T. Kapus, and Z. Brezočnik. Specifikacija zahtev sistema z ACTL-jem. In Baldomir Zajc, editor, *Proceedings of the Eighth Electrotechnical and Computer Science Conference ERK'99 Portorož, Slovenia*, volume B, pages 23–26, Ljubljana, Slovenia, September 1999. Slovenia Section IEEE. In Slovene.

[7] R. Meolic, T. Kapus, and Z. Brezočnik. The Efficient Symbolic Tools Package. Submitted to the 8th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2000), Split, Croatia, October 2000.

[8] R. Meolic, T. Kapus, and Z. Brezočnik. Verification of concurrent systems using ACTL. In *Proceedings of the IASTED International Conference on Applied Informatics AI'2000*, pages 663–669, Innsbruck, Austria, February 2000.

[9] R. De Nicola, A. Fantechi, S. Gnesi, and G. Ristori. An action based framework for verifying logical and behavioural properties of concurrent systems. *Computer Networks and ISDN Systems*, 25:761–778, 1993.