BAL Tool: what else? *

Diego Perez, M.Carmen Ruiz, J.Jose Pardo, Diego Cazorla

Escuela Superior de Ingenieria Informatica
Univ. de Castilla-La Mancha
Campus Univ. s/n. 02071. Albacete, Spain.
Diego.Perez2@alu.uclm.es, {MCarmen.Ruiz, Juan.Pardo, Diego.Cazorla}@uclm.es

Time-saving methods and increased performance are currently two of the overriding concerns in performance evaluation. Not so long ago, these features were considered important in critical systems, but now their relevance has become significant in all systems.

The aim of obtaining the best running time using the minimum amount of resources goes without question and it would be extremely beneficial to have a tool (in this case a tool based on formal methods) that is able to calculate the time needed to reach a certain state (or the final state for the total time), taking into account the time actions need for their execution, the time invested in process synchronizations and the fact that the (limited amount of) resources have to be shared by all the system processes, which could cause delays when resources are not available. Bearing in mind these factors, we have developed a timed process algebra, called BTC (for Bounded True Concurrency) [RCC$^+$04], which allows us to evaluate the performance of a system as well as a tool called BAL (not an acronym) which is able to carry out this task automatically.

BTC is based on CSP [Hoa85] and extends CSP syntax in order to consider the duration of actions by means of a timed prefix operator. Likewise, the operational semantics has also been extended to consider the context (resources) in which processes are executed. BTC is able to take into account the fact that system resources must be shared by all the processes. So, if there are more processes than resources then not all of them can be simultaneously executed. A process has to wait until it allocates the resources needed to continue its execution. This means that we can find two kinds of delays in the execution of a process: delays related to the synchronization of processes, and delays related to the allocation of resources. The former is usual in a (theoretical) concurrent context, but the latter is only taken into account if we consider a limited amount of available resources.

With this formal resource-aware model the timed characteristics of the system are captured. The next step is concerned with carrying out performance evaluation: we want to be able to estimate the minimum time needed to reach a given state. By applying the rules of the operational semantics, we build a transition graph where we can abstract the information about actions and consider only the information about time (duration of actions). This graph is a weighted

* This work has been partially supported by CICYT project TIN2009-14312-C02-02, and JCCM project PEII09-0232-7745

directed graph, where weights are always positive numbers, and the problem to solve is finding the shortest path from the initial node. Usually, the number of states in the transition graph for a real system is huge so the ability of a tool to perform this task automatically becomes relevant. With this idea in mind the BAL tool has been developed, which, moreover, has been improved with some other useful features: a syntactic analyzer and a graphic interface with a user assistant.

Thus, the BAL tool can be divided into three stages. First, a syntactic analyzer checks the specification of the system to be studied. If the system specification is correct, the second stage initiates. Here the tool builds the relevant transition graph by applying the rules of the operational semantics and lastly the minimum time needed to reach the final state from the initial one is calculated and the path obtained shown. Evidently, the most delicate part in the tool development has been that relative to graph analysis where it was necessary to join different algorithms of pruning, dynamic load balancing and parallelization. This is so, since one of the main aims is for the BAL tool to be able to deal with real systems and not be limited to toy examples. A general view of BAL tool can be found in Figure 1 and its user interface in Figure 2.

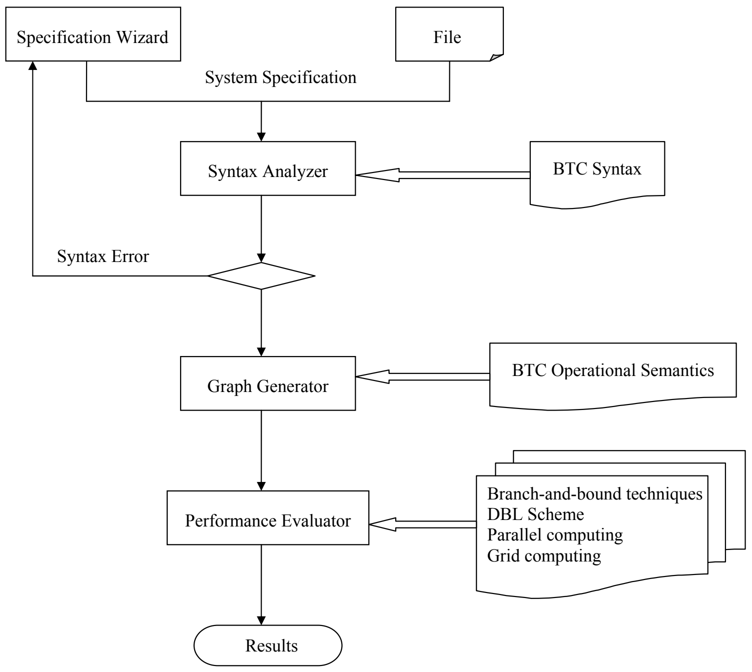


Fig. 1. Structure of BAL tool

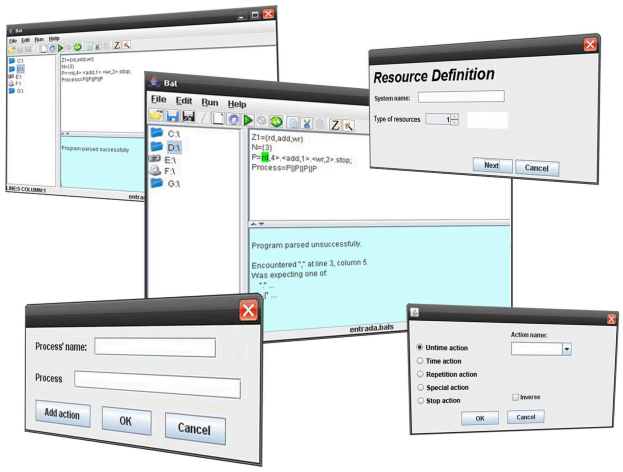
This evaluation made by BAL can be used in two different ways. On the one hand, if we have a fixed number of resources we can ascertain the time needed to evolve from the initial state to the final one (or between states), so we can check different configurations for a system before using it, thus saving an immense amount of time and money. On the other hand, if we start with some specification, we can find the appropriate number of resources of any type that we need in order to fulfil some time requirements. We can even calculate the minimum number of resources needed to obtain the best performance from a system.

Apart from the usefulness of the analysis that BAL performs, the main advantage of this tool is the fact that it is capable of dealing with real-world systems of a considerable size. This has been achieved through painstaking work linking different implementation strategies.

Currently, our work focuses on two different lines. The first is concerned with improving the tool and the second one deals with its application.

The tool uses parallel and grid computing with distributed memory, however we have decided to take advantage of improvements in hardware architecture and multicore machines architectures and have used threads which will communicate by means of shared memory.

In the user interface, improvements have also been carried out. Currently, the system under study must be specified by the BTC language. Despite having several assistants and a parser, certain knowledge in process algebras is needed. Therefore, the tool is being expanded in order to be able to accept Petri Nets or by UML modelled input systems.



```
Z1={rd,add,wr}
N={3}
P=<rd,4>.<add,1>.<wr,2>.stop;
Process=P||P||P||P
```

```
Program parsed successfully.
```

```
Program parsed unsuccessfully.

Encountered "," at line 3, column 5.
Was expecting one of:
    ":" ...
    "|" ...
```

Fig. 2. BAL Tool Graphic Interface

We are also working on issues related to the study carried out by BAL. In particular, a step-by-step simulation in which the user can even decide which path will continue will be added. Furthermore, the information provided by BAL will be improved including the number of arcs and nodes of the solution. We hope to increase both functionality and user interface when new needs arise.

So far, BAL has been used only in two scenarios. On the one hand we have focused on the use of processors. Currently, the number of available processors has increased mainly due to the use of grid computing and the emergence of multicore machines. Obviously, an increase in the number of available processors leads to an increase in the speed performance of systems with parallel processes. But this increase is not infinite, that is, There is a point in which an increase in the number of processors does not improve performance or the improvement is minimal. Then comes the question of why to use more processors than are actually needed. If it were possible to know what is the optimal number of processors to run a system, the better use of resources we will obtain. This can be done by the BAL tool. In particular we have studied a system which models the sum of matrices. The system has been modeled by varying the number of available processors (hence the shared resource) to determine when the number of processors does not improve system performance.

On the other hand, the main field where BAL proves to be useful is in the performance evaluation of flexible manufacturing system (FMS). A competitive manufacturing system is expected to be flexible enough to respond to small batches of customer demand. Due to the fact that the construction of any new production line is a large investment, current production lines must be able to be reconfigured to keep up with increased frequency of new product designs. Therefore, in the development of a FMS, particular attention must be paid to the design phase, where the performance of the system has to be accurately evaluated so that the most appropriate number and type of resources are selected from the initial design stages.

Different configurations (decreasing/increasing the number of robots, conveyor capacity and so on) have been studied to see if the system can be improved by achieving a better completion time i.e. a bigger number of completion parts per unit of time (throughput). In this context, BTC and BAL are of great utility given that they allow us to check different configurations for a FMS before being established or for future improvements.

At present, our research team is looking for new scenarios where the BAL tool can be useful.

References

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[RCC$^+$04] M. Carmen Ruiz, Diego Cazorla, Fernando Cuartero, Juan José Pardo, and Hermenegilda Macià. A bounded true concurrency process algebra for performance evaluation. In *FORTE Workshops (EPEW'04), LNCS 3236*, Toledo, Spain, October 2004. Springer.