Technical Report

Short presentation of QUASAR

S. Evangelista, C. Kaiser, J.F. Pradat-Peyre et P. Rousseau

CEDRIC CNAM Paris
292, rue St Martin 75003 Paris
{evangeli, kaiser, peyre, rousseau}@cnam.fr

February 5, 2004

1 *What is* QUASAR *?*

QUASAR, is a tool which allows analysing and validating concurrent programs. In other words, QUASAR is Quickly Undertaking the Analysis of Safety and providing Anomaly Report when necessary.

2 *What is a correct concurrent program and how can it be validated ?*

A correct concurrent program must satisfy two classes of property : safety and liveness.

Safety properties assert that nothing "bad" will ever happen during an execution (that is, the program will never enter a "bad" state). Example of safety property are the respect of mutual exclusion, the denial of placing a new item in a full buffer, the freedom from deadlock, the insurance that all dynamically allocated resources will return to the resource allocator (i.e., the number of dynamic resources remain constant).

Liveness properties assert that something "good" will eventually happen during the execution (that is, the program will eventually enter a "good" state) Example of liveness property are the absence of livelock, fairness, absence of indefinite postponement or starvation.

Owing to its advantages (such as design clarity, reactive facilities, distribution ability, scaling capabilities,...), concurrent programming is growing more

and more in importance, and appears in a wide range of applications, even for applications needing a high degree of safety. However, the application behaviour induced by concurrency is intrinsically difficult to validate.

Traditional methods, using specifications or tests, do not give full satisfaction. Specifications are often situated at a too high level of view and thus far from the final details of the source code and/or of the target platform, which can induce huge behavior modifications. Exhaustively testing is a long, tiresome process and it is often very difficult to reproduce the concurrent execution which leads to the error. Moreover testing gives only probabilistic results that depend of the coverage of the implemented tests. These experienced drawbacks, due to the indeterminism inherent to concurrency, push engineers to limit the use of concurrency, and thus unfortunately to limit also the facilities that concurrency provides.

Another approach consists in using the source code as the source formalism and to elaborate from it a model on which properties can be validated. Alike a compiler that builds a syntactic tree which allows to verify the program syntax before generating the code, this method builds a model which allows to verify the semantic of concurrency of the program before running it.

This approach bears also additional advantages. There is not necessary to transcribe the application in a different language for specification, validation or testing purposes, and this avoids transcription errors or semantic loss. The approach is also usable for analysing the full and final application program after linking all its imported components. This is very important when the checked properties, like absence of deadlock, are global properties which are not decomposable and which cannot be proven by parts.

3 *How does* QUASAR *proceed ?*

QUASAR is an automatic concurrent program analysis tool based on this promising method that uses the application source code for generating and validating a semantic model. QUASAR follows a four step process :

1. Automatic extraction of the source code which is related to the property to verify. The aim of this step is to remove those parts of the source code which are not related to the investigated property. This contributes to generate a model which is smaller than the one corresponding to the whole program, but which has the same behavior according to the investigated property. This smaller model will be easier to analyze in the next steps. This first step is called slicing.

2. Translation of the simplified program into a formal model. The target formalism used is colored Petri nets because their analysis may combine several techniques that are supported by experienced tools and that we continue to develop successfully. For translating an Ada program into a Petri net we use patterns. Each element of the Ada language has a corresponding pattern (or sub-net). These sub-nets allow a hierarchical

construction since meta-nets can be used to abstract other (list of) sub-net(s). For example, the sub-net of the loop statement contains a meta-net abstracting the statements of the loop. We produce The final net corresponding to the whole (simplified) program is produced using two basic constructors : the substitution (that substitutes a meta-net by a concrete sub-net) and the composition (that merges two different sub-nets into an unique one).

3. Analysis of the model by combining structural techniques (like Petri nets reductions) and finite state verification methods (like temporal logic formula verification). The efficiency of this combination, added to the gain obtained by the slicing step, allows to analyze complex and relatively large programs.

4. Construction of a report. When the target property is not verified, the state in which the application is faulty is displayed and a report indicates the sequence of program actions that ends by invalidating the property.

4 *What are the benefits of using* QUASAR *?*

QUASAR is able to cover most of the concurrency features and a large part of a language like Ada. It shows good performances when analysing the concurrency part of non-trivial application programs. This reflects the fact that, although intrinsically more complex, unfamiliar and prone to errors than most other aspects of programming, the part of an application program which is concerned by concurrency remains hopefully small. The slicing step of QUASAR (the first step) eliminates this large part of the application program which is not concerned with concurrency. And moreover the efficiency of the reduction techniques and state verification methods of the third step of QUASAR succeeds in restraining the combinatorial explosion of states.

For applications which specification includes a high degree of safety, the use of QUASAR permits to avoid to set up complex and cumbersome specification or test methods which must be iterative whereas each little change in the original code source can induce a huge difference in the application behavior. Using QUASAR allows to analyse applications that have already been implemented but that have never been validated, avoiding to elaborate a posteriori the specifications of the application. Furthermore, combining the use of QUASAR with specification and test methods will result in a higher level of safety as in fact these approaches provide complementary insights.

5 *What is the current implementation state ?*

The current implementation of QUASAR accepts Ada concurrent programs corresponding to a large part of the Ada language (see "Supported Language"

section). It's based on the implementation of the ASIS for GNAT Ada 95 compiler. The graphic part has been designed with GtkAda. In the current version of QUASAR we use Maria or Prod as model checker.