Handling with parameterized states in Statecharts for determining performance measurements of reactive systems

N. L. Vijaykumar  
INPE/LAC - São José dos Campos, SP, Brazil  
vijay@lac.inpe.br

V. M. B. Andrade  
INPE/LAC – São José dos Campos, SP, Brazil  
valeria@lac.inpe.br

S. V. Carvalho  
INPE/LAC – São José dos Campos, SP, Brazil  
solon@lac.inpe.br

V. Abdurahiman  
ITA/IEC – São José dos Campos, SP, Brazil  
rahiman@comp.ita.br

Abstract

Specification of performance models that react to events has been in the spotlight for quite some time due to the complexities inherent in representing such systems. State-transition diagrams, queuing networks and Petri nets have been extensively used to deal with such systems in order to obtain performance measurements. Recently Statecharts, due to their appeal in their visual aspect, have been proposed to specify and deal with performance models and in this regard, a software has been developed. Statecharts are a graceful extension of state-transition diagrams with notions of hierarchy, parallelism and synchronization. The feature, known as parameterized states, was proposed originally as an extension to the Statecharts formalism. Basically, this feature may be used whenever a set of states has the same internal structure and this enhances pretty much the representation of performance models. In this paper the description of adding the use of parameterized states to the software is discussed. An example using this feature is shown.

Key words: Performance models, Specification, Statecharts, Markov chain, Parameterized states
1. INTRODUCTION

Complex systems consist of subsystems working in parallel. In order to model these systems ideas of parallelism, resource sharing, synchronization, interdependence, hierarchy and randomness (random perturbations) are to be considered.

State-transition diagrams are usually used to represent such systems. Unfortunately, these diagrams cannot cope up with the complexity of these systems. The parallelism of the actions increases exponentially the number of concurrent subsystems being considered. A detailed representation of a given system may not be able to clearly give an idea of the number of states in the model. In real applications, models may contain hundreds of thousands of states where ideas of concurrency and interdependency among the components of the system are difficult to be handled. Thus, a necessity of developing high level specification tools arises from which a mathematical model may be automatically generated.

In these last years, the complex systems involving parallelism, synchronization and interdependence of subsystems have been called as reactive systems. The main feature of a reactive system is that it is based on events, reacting from stimuli received by external and internal means. The main problem is the difficulty in describing the behavior of the reactive systems in a clear and realistic way and at the same time maintaining a rigorous formal basis that can be computationally handled. The behavior of reactive systems consists of allowed sequences of input and output events, conditions and actions [Harel, 1987].

The Statecharts formalism [Harel, 1987] and [Harel et al, 1987] allows a clear representation of hierarchy, concurrency and interdependence among the components of the system. This formalism was initially developed to specify, simulate and control real time systems. However, many other applications, such as object-oriented systems specification [Booch, 1994], have also taken advantage of this tool. In [Harel, 1987] an extension was proposed in which a set of states (state is one of the basic elements that form Statecharts) with a similar internal structure may be represented by a feature known as parameterized states. This feature comes out handy in many applications. For example, whenever queues have to be
represented, many a time it is necessary to represent buffers to hold jobs to be processed. The feature of parameterized states is very useful in such situations.

A very brief explanation of Statecharts will be given in Section 2. Section 3 gives the details of how to represent parameterized states within Statecharts. Section 4 shows an example using this feature and is followed by Section 5 with some final comments.

2. STATECHARTS

Statecharts have a visual formalism capable of, like other graphic tools (Petri nets, automata, etc.), specifying complex systems. This formalism can be seen as an extension of state-transition diagrams by adding notions of hierarchy, orthogonality and interdependence.

The basic elements that make part of this high-level specification tool are: States, Events, Conditions, Actions, Expressions, Variables, Labels and Transitions. All the resources and formalism of this powerful tool Statecharts can be seen in [Harel, 1987] and [Harel et al, 1987].

The basic idea of using a Statecharts representation is to convert it into its corresponding state-transition diagram, a Markov chain, by stimulating all the possible events [Vijaykumar, 1999]. Among these basic elements, it is necessary to give a brief explanation of the interpretation of Events as they play an important role when generating the Markov chain. Event is a very important element to observe the model of a given system. It is considered as an interference to the system in the sense that the present system behavior as a whole is changed to another behavior with its (event's) occurrence. Statecharts provide some special events such as true (condition), false (condition), entered(X) and exited(X) (these special events are abbreviated in the Statecharts formalism as tr(condition), fs(condition), en(X) and ex(X) respectively) to cope up with the internal logic of the modeled system. The first two are respectively true and false if the condition is evaluated to be true and false. The last two are related to a state X in which the event en(X) is stimulated if state X is entered whereas the event ex(X) is stimulated whenever there is an exit from state X. These four special events
are considered as immediate events which means that these are the first to be reacted automatically in any given system.

Statecharts formalism classifies events into two categories: external and internal events. In the case of performance models, it is decided to keep these categories with the following definitions. External events are the stochastic events (where time between their activation and their occurrences follow a stochastic distribution) that have to be externally stimulated to yield new configurations. Internal events are those special (immediate) events mentioned above \(true (condition), false (condition), entered(X) and exit(X)\) where they are always checked and continuously reacted until none of them are active. Actions are also considered as internal events.

As already mentioned, a software has been developed in which a system is specified using the features of Statecharts and by making use of analytical approach steady-state probabilities are determined and they are the basis for obtaining performance measurements [Vijaykumar, 1999] and [Vijaykumar et al, 1999]. Many examples have been used to test and validate the software [Francês et al, 2000], [Vijaykumar et al 2000] and [Vijaykumar et al 2001a].

There are some important features that would enhance very much the capability of the high-level specification tool Statecharts when used to specify performance models. There are many additions that would be interesting to include in the present software in order to boost the specification features of the present implementation. One such addition already included and tested is parameterized states.

3. PARAMETERIZED STATES

Parameterized states have been suggested as a possible extension in Statecharts [Harel, 1987]. The main application of parameterized states occurs whenever a set of states has the same internal structure. When such situations arise it is interesting to make use of this feature by indexing the state in question by a parameter.
Figure 1 is a part of a detailed Statecharts specification [Harel, 1987] of a quartz multi-alarm watch. Harel considered the sub-state "1min" that belongs to the "update" macro-state of an alarm to be set. Notice that the State 1min consists of 10 sub-states. These sub-states are basic and they have no internal structure but they do have the same structure of transitions among them and a feature such as parameterized state may simplify the representation. The same example with this feature is depicted in Figure 2 [Harel, 1987].

Figure 1. Example of a state to be parameterized [Harel, 1987].

Figure 2. Example of Parameterized state [Harel, 1987].
In Figure 2 the macro-state 1min is a parameterized state and its sub-states are replaced by parameterization symbols. In order to represent parameterized states, it is proposed that a name of the parameter is added to the name of state to be parameterized followed by the range indicating the variation of the values [Harel, 1987]. Beyond this information, two more symbols have to be included in the visual syntax of Statecharts. One symbol is to be used to represent the states indexed by the parameters (overlapped states) and another symbol is to be used to represent the destination state of a transition. Both the source state and the destination state are indexed by parameters. The destination state is a diamond-shaped figure with expression in its interior to determine the state. This proposal is just an idea for a possible future extension of Statecharts and some points are not solved such as representing transitions where only one of the involved states is indexed by parameters.

The cases where parameterized states can be used are frequent in applications of performance models and therefore their graphical representation deserves a special attention for representing them. Usually, the behavior of performance models is represented using state-transition diagrams. In these diagrams, parameterized states are represented in an informal manner. As an example, a state-transition diagram of a M/M/1/N queue is shown in Figure 3 in which N is the maximum capacity of the queue, and “a” and “s” are events that denote arrival of a client and end of a service respectively. This same informal representation is shown by using Harel’s notation in Figure 4.

![State-transition diagram of a M/M/1/N queue.](image)

Figure 3. State-transition diagram of a M/M/1/N queue.

This representation is formal in the sense that the necessary information to construct the model are depicted without ambiguities. In a state-transition diagram, a state must be defined as the initial point especially when the dynamics of the behavior is to be studied. In
Statecharts, this is done by Entry by Default [Harel, 1987]. However, the representation of this feature is not mentioned in the parameterized state MM1N. With this, the main characteristic that Statecharts are very close to state-transition diagrams is lost. This is the main reason that another alternative in representing parameterized state is provided in this work.

The main interest is to keep the proposed notation very close to that of state-transition diagrams. In this representation the names of parameterized states are followed by parameters between brackets and, after a comma the limits of variation of each parameter are provided. These limits are assumed to be integer. It is also possible to provide these values using integer variables. The use of brackets coincides with the notation of providing a condition. However, as this notation is associated to states no ambiguity is introduced.

![Statechart Diagram]

Figure 4. Statecharts representation using parameterized state of a M/M/1/N queue using notation proposed in [Harel, 1987].

When a state is indexed, in its interior each indexed sub-state by the parameters must be interpreted as a set of sub-states where each element corresponds to a value (or set of
values) of the parameter (or parameters). Each state indexed by the parameters must indicate the values of the corresponding parameters between brackets.

In order to represent the transition between parameterized states another symbol is introduced: “pseudo-states”. These are represented in the same way as a regular state but with dashed lines. Each “pseudo-state” represents a destination state and it must also be accompanied with indexed parameters besides its name has to be the same as the source state of the transition.

The destination state must also be indexed with the parameters and the necessary variations to indicate the values in relation with the parameters of the source state. Notice that the “pseudo-states” must be restricted to be within a parameterized state. Finally, the Entry by Default indicating the initial state of a parameterized state must be provided indexed with parameters between brackets and their corresponding values. Figure 5 represents the M/M/1/N queue using the alternative (to that of Harel) notation proposed in this work.

![Statecharts representation of M/M/1/N queue with parameterized state using the proposed notation.](image)

The model shown in Figure 1. is shown in Figure 6. using the notation proposed in this work. Therefore, based on this notation and where necessary, it is possible to specify a set of states with a same internal structure. Once the specification of a system as a whole is
through, it is necessary to generate the steady-state probabilities from which performance measurements can be determined. The approach used is an analytical approach.

Due to the ease of mathematical and computational description, the Markov models - especially Continuous-Time Markov Chains (CTMC) - are frequently used as a basis to model reliability and system performance. In these models, reliability and many performance measurements are probability functions of occupying the states during a certain period of time or in a long-run horizon. As the main components of a Markov Chain are nodes and arcs linking the nodes, there is a direct correspondence with the state-transition diagrams. It is possible to find in the literature many algorithms to yield steady-state probabilities, through which performance measurements can be determined, by using Markov theory [Philippe et al., 1992] and [Silva and Muntz, 1992]. The Statecharts representation is converted into a corresponding state-transition diagram by stimulating all the possible events. For more details, the readers are requested to refer to [Vijaykumar, 1999]. Once the state-transition diagram is available, by applying the proper numerical methods steady-state probabilities are determined.
4. **EXAMPLE**

Consider a file system that consists of a processor. Whenever the processor is busy, a request to use the processor is placed in a processor queue. After a request has been processed, it may leave the system with a probability of 0.6 or it may use the disc with a probability of 0.4. The disc also makes use of a queue, disc queue, for storing the requests whenever the disc is in use. After the request finishes occupying the disc service, it is returned to the processor. Figure 7 shows the specification of this example of a File System using Statecharts.

The scope of this paper is towards the representation of Parameterized states but inclusion of probabilities in Statecharts representation of performance models were adapted to the software and is discussed in [Vijaykumar et al, 2001b]. The notation of parameterized states suggested and implemented in this paper is applied to represent the Processor (P-Queue) and Disc (D-Queue) queues.

![Figure 7. A File System with Parameterized States in Statecharts.](image-url)
The input data used to produce the performance measurements are shown in Table 1 and Table 2 shows results of some performance measurements obtained for a different number of buffers both in Processor and Disc queues. It seems that the Processor remains busy by increasing the buffer size for storing the requests in the Processor queue. However, gain in the Processor usage is not very significant even after increasing the size of the buffer. Once the size of the buffer queue reaches 30, it is not advantageous to increase its size as it will not increase the usage of the Processor as shown in the Table.
5. CONCLUSIONS

Statecharts have proven to be considered as a potential candidate to represent performance models due to their visual appeal. This paper showed an alternative solution proposed by David Harel for extending Statecharts to include parameterized states. The example made it very clear that this feature fits exactly in situations where the set of states has the same internal structure. In fact, the example tackled in Section 4 attests that situations such as queues in buffers may be represented by using the feature of parameterized states. It is expected that the software available at the Laboratory of Computing and Applied Mathematics (LAC) will be very useful to representing complex reactive systems in order to obtain performance measurements.

However, at this stage it is still difficult to consider Statecharts as the best option although the models that served for testing and validating the software might show that Statecharts have been an excellent option. It is possible that, depending on the model to be represented, other tools may come out to be handy. At the moment work is in progress to develop a user-friendly interface to make specification of models in Statecharts easier.

REFERENCES


Vijaykumar, N. L.; Andrade, V.M.B.; Carvalho, S.V.; Abdurahiman, V. Introducing probabilities in Statecharts to specify reactive systems to obtain performance measurements. *Software: Practice & Experience.* 2001 (Submitted)