Partial-Order application for Formal Software Verification

X. Dumas[1], F. Boniol[2], Ph. Dhaussy[3], E. Bonnafous[1]

1: CSSI, 3 rue du professeur Pierre Vellas 31300 Toulouse
2: ONERA, 2 avenue Edouard Belin 31000 Toulouse
3: ENSIETA , 2 rue François Verny 29806 Brest

Abstract: A method based on context (environment of the system) description and partial-order reduction is presented in order to improve model checking verification on large systems. The context describes the behaviour of the system environment as a set of formal use cases decomposed in scenarios. It aims at leading the system to the pertinent relevant configuration according to a desired property one want to validate. The verification is thus done by only focusing on a subset of the system by model checking verification with the OBP toolset [12, 6]. The classical combinatorial explosion of the system is thus handled by the context. Nevertheless the concurrency introduced by the different actors of the environment running in parallel could possibly generate an exponential amount of scenarios thus preventing model checking. To circumvent this problem the partial-order reduction method is introduced to factorize equivalent scenarios into a single one following the Mazurkiewicz trace semantics.

Keywords: partial-order, model checking, formal methods, CDL, SDL, OBP

1. Introduction

1.1 The context: Embedded Systems

The conception and the development of highly critical embedded systems are subjected to economical and time project constraints but must also respect drastic security standards. In avionics or aerospace those constraints are exacerbated because they must respond to robustness and reliability criteria’s to go through the certification process. For that purpose the verification and validation process are reinforced all along the development cycle of the system where the safety process is omnipresent. Although the certification is based on the testing process with very tight rules, it can't possibly ensure the system is not prone to errors because of its incompleteness.

In [1], the author defines the test as the process of finding errors meaning that no error found only means several other ones have to be done. It is a very tough task even for highly experienced engineers. In most cases, tests derive from textual specification and engineers’ judgment. As a consequence textual requirements could be interpreted in different manner from one to another. Above all the link between the requirement and the behavior of the environment is often neglected or implicitly identified relying on the knowledge of the system engineer.

Over this last decade, the improvements of high technology has increased exponentially. In the meanwhile complexity of Embedded Reactive Critical Systems has grown drastically in the same manner. To cope with this problem, formal methods have been developed to help the verification of complex systems.

1.2 Formal verification by Model Checking

Classical model checking method automatically verifies whether the system represented by a composition of automata’s fulfils a given property expressed in temporal logic formulae (LTL [2], CTL [3], etc.). This technique well defined and efficient for small systems suffers from the combinatorial explosion when dealing with huge ones. This complexity is intrinsically attached to systems composed by several processes running in parallel. As a consequence the model checking method can’t determine the process events order before the execution time and must keep track of all possible transitions interleaving leading to overwhelming storage of states.

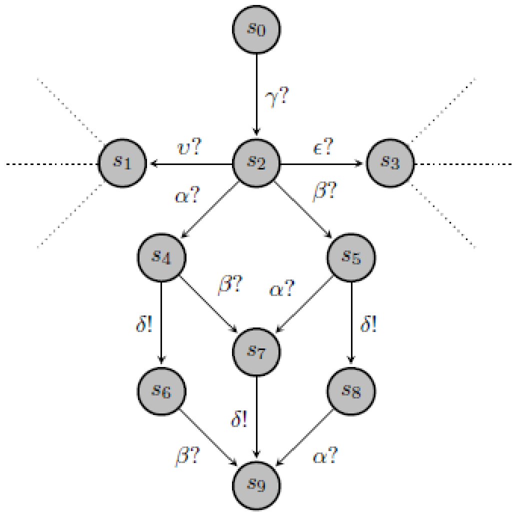

Figure 1: model checking

In the Figure 1, the model checking method will explore all states and transitions until all states have been visited (states reachable from S_1 and S_3 will be explored too). Even in case the property to be fullfiled is insensitive to the order of α? and β?, both interleaving will be memorised.

1.3 Contribution

In this paper we expose a verification method based on context description [4] combined with partial-order reduction [5]. The context represented by a set of concurrent actors describes the environment of the system. It allows restricting the system (modeled in SDL[1] to well identified operational phases such as initializations, degraded modes, reconfigurations, etc… and strongly alleviates the size of the model under verification. The reduced system only contains relevant part corresponding to the verification of a property.

Even though the model checking is made possible on the reduced system, the environment composed by concurrent actors could grow exponentially when expanding the formal use cases into individual scenarios. This is precisely the combinatorial explosion we want to address.

1.4 Organisation of the paper

The paper is organised as follows:

In paragraph 2 we present the CDL language (*Context Description Language*) and the OBP toolset (*Observer Based Prover*). We then expose the main objective of this paper. In paragraph 3 we briefly remind the partial-order reduction method and then we present our reduction methodology. In paragraph 4 we present the experimentations of the method made on SDL systems and in paragraph 5 we summarizes the results obtained before concluding in paragraph 6.

2. Modelling framework

2.1 CDL: an environment description language

The weak integration of formal methods in the industry is mainly the consequence of two major difficulty. First of all, the lack of industrial tools allowing formal verification on large systems generating a global state space explosion. Then there is a gap between the process of requirements designed in natural language and formal verification using properties encoded in a well defined semantics. Indeed model checking is an automatic but mathematical process where the system and the property have their own mathematical foundation so that to ensure irrefutable results. Besides this, writing requirements using temporal logics is not straightforward and needs some expertise. To cope with those two drawbacks, the CDL language is developed in order to propose an easy handling tool for model checking verification [6]. The CDL aims at modelling the behaviour of the under verification system context as a set of formal use cases which semantics relies on the semantics of traces [7]. This DSL allows specifying the context with scenarios and temporal properties using property patterns. Moreover, CDL provides the ability to link each expressed property to a limited scope of the system behavior.The idea behind context is that the requirements one want to verify are often linked to specific use cases so that it's not necessary to explore all possible scenarios on the system. Contrary to classical model checking methods where the system is explored in its entirety, the CDL language aims at reducing the system behaviour before its effective verification by interfacing with an existing model checker such as IFx, TINA or CADP [8 , 9, 10]. The context description thus contributes to reduce the complexity of the system bypassing the state explosion. To address the problem of the formalization of the requirements, following the work of Dwyer and Cheng [11], CDL incorporates temporal property patterns so that to assist the engineer in expressing system requirements directly in formal language. One of the originality of this language is that the properties called observers can be attached to a specific scope of the environment where the verification of the property is the most relevant. CDL is defined in three complementary levels. The first one describes the applications (actors) interacting with the system. The second one identifies interesting functionalities of each actor so that to dip the system into a particular situation interesting for the verification. Those functionalities are then detailed in the last level by sequences of Message Sequence Charts defined in the norm Z120 [13]. The CDL language have been integrated in the OBP [12, 6] toolset developed as an eclipse plugin in the Topcased[2] environment. OBP allows the modelling of graphical context and the connexion to different model checkers.

2.2 Scenarios explosion

The description of the environment of a system allows defining precise configurations one want to observe on the model according to a property to be fulfilled. This environment composed by several actors running in parallel could possibly lead to a huge amount of scenarios each one synchronised with the system and the property. This could prevent efficient model checking because of the size of the scenario automata generated. Basically, the combinatorial explosion of the global state space is not a problem anymore because the composition of a trace with the system is straightforward.

[1] Specification and Description Language (ITU-T Recommendation Z.100)

[2] www.topcased.org

Nevertheless the bottleneck is displaced to the combinatorial explosion produced by all the possible scenarios generated by the actors interleaving. For instance, n concurrent actors executing only one message would generate n! scenarios. This is precisely this combinatorial explosion we want to reduce. One can reasonably expect numerous equivalent scenarios only differing by events which can be interleaved in any order. This supposition is justified because embedded critical systems are designed without identifying all possible interactions between the system and the environment so that it has to be robust to handle unexpected sequences of messages. One way to remove those equivalent scenarios is the application of partial-order reduction method. For instance, if the order of all combination of the n previous messages is irrelevant, then only a single scenario among the n! will be selected.

3. Partial-order reduction for SDL

3.1 Partial-order method

It has been observed that model checking method makes redundant work by exploring all states and all transitions in the global state space. Nevertheless among all the interleaving leading to a same state, very few of them modify the truth or falsity of a property so that a lot of them could be removed from the automata before its effective construction. Hence the model checking becomes practical.

Partial-order takes advantage of the so called diamond property which states that whenever two adjacent transitions can be commuted without modifying the system behaviour, then only one interleaving is meaningful. In this case the two transitions are said to be independent. The partial-order theory appeared in the work of Mazurkiewicz theory of traces [14] which semantics has been the foundation of most partial-order methods.

The methodology of the algorithms implementing partial-order reduction method is based on a static analysis of the system: for each global state the smallest subset among fireable transitions is selected, sufficient to eliminate redundancies introduced by the diamond property.

Many works in this area have been achieved by several researchers. Valmari [15] first developed the stubborn set method and noticed the ignoring problem where some transitions could never been reached (fairness assumption). In the work of Doron Peled [16], another algorithm called ample set is implemented and a solution for solving the ignoring problem is presented by adding several conditions called proviso. Further reduction has been obtained by Godefroid [17] who noticed that independent transitions are not always removed applying previous algorithms. He developed the persistent set (which generalizes ample set and stubborn set method) and the sleep set method further reducing the global state space.

The above algorithms differ by the information taken from the system to compute the dependencies. Comparisons between them revealed that the more information the more reduction could be obtained but not systematically. The smallest set selected the smallest reduced graph computed is not granted. At the end of the day, there isn't a unique algorithm giving systematically the best reduction.

Those algorithms have been implemented on the fly with significant reduction results. For instance in the figure 2, assuming that α?, β? and δ! are all three independent:

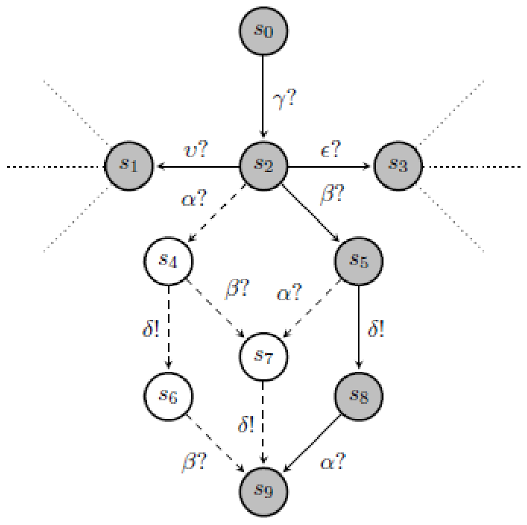

Figure 2: Partial-order

Applying the partial-order method to the first system, we obtain the reduced automata with solid lines and states filled in grey. Dashed arrows and blank states will be removed from the global state space. Moreover states reachable from S_1 and S_3 will have to be explored by applying the same method.

Introducing context behaviour before partial-order reduction method could further reduce the global state space. Let's consider the Figure 3 symbolizing the environment behaviour and the previous automata. In this case, the environment will restrict the system preventing it to explore states reachable from S_1 and S_3. Therefore after partial-order reduction only states S_0, S_2, S_5, S_8 and S_9 will be explored.

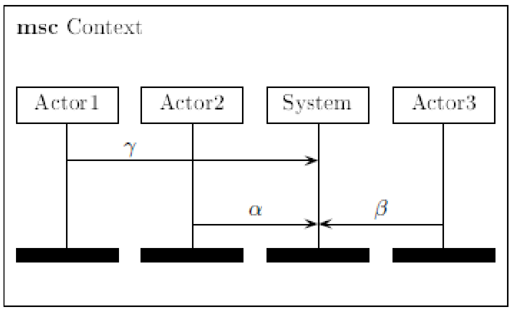

Figure 3: Context reduction

Describing the context of the system is a good way to further reduce the system behaviour. As introduced the main drawback resides in the exponential number of scenarios obtained when expanded.

The previous works apply the partial-order methodology on the fly on the global state space. In our work we apply partial-order method on the context. We have developed a new independence relation to group scenarios of the context into equivalent classes. The main difference between our independence relation and those already existing is that partial-order method can't be directly applied on the graph of scenarios (context) because we are working on two open systems: the environment and the system whereas classical methods work directly when constructing the global state space which is a closed system. In fact the independence between two actions in the context doesn't ensure their independence in the system. Summarizing our independence relation: two events from the context are independent whenever their interleaving lead to the same behaviour of the system.

To compute those independence relations, we have developed a decision procedure in case of systems composed by a single one process.
To extend this relation to the multi process case, we needed to take into account the possible communication between internal processes which can introduce some dependency relation.

Once the independence relation computed, we applied the Mazurkiewicz trace theory to organize the scenarios into equivalent classes. Two scenarios are equivalent (belong to the same class) if one can obtain the other one by permutation of independent adjacent event. As a consequence, only one representative of each class needs to be simulated on the system.

For instance, let's suppose 3 independent actions each one occurring from a distinct actor:
A, B , C. We obtain 6 equivalent scenarios: ABC, ACB, BAC, BCA, CAB, CBA. One can check that we can obtain one another by successively permuting pair wised independent actions (A,B), (A,C) or (B,C). As a consequence all those scenarios belong to a unique class: [ABC] representing all 6 previous scenarios. Hence only one representative can be kept.

3.2 Reduction methodology

The picture depicted in the Figure 6 summarizes the different steps leading to the reduction of the context. At the very beginning we start with the context description designed by the engineer in relation with several properties he wants to check, and the SDL system.
The first step is to obtain the independence relation between events of parallel actors. This is done by a static analysis of the SDL model. Once the pair wised independent events obtained we can apply the algorithm computing the equivalent classes of scenarios using the Foata [18] normal form. It is important to denote that this first step is done by only considering messages sent from the context to the system. The main reason is that the context can't predict in which order the system will send the messages. As in the theory of game and the synthesis of controller [19], some are controllable by the context (the output) and some are not (input one). The idea is thus to separate outputs from inputs messages. This technic allows factorizing equivalent behaviour of controllable messages of the context but still keeping track of all possible reception order of the system. This one will determinate the executable scenario of the sub graph of scenarios obtained after reduction at model checking time.
For instance, let's consider the example of the following picture corresponding to the context and its entire graph of scenarios.

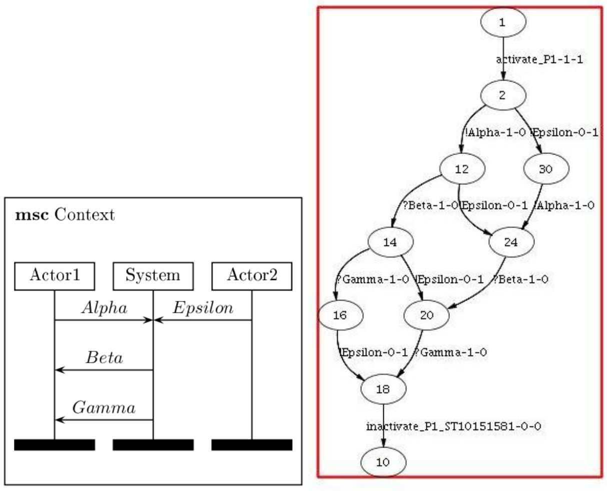

Figure 4: The context

The context is represented by two parallel actors $Actor_1$ and $Actor_2$ each one sending a controllable message (Alpha respectively Epsilon) to the system. There are also two messages only controllable by the system: Beta and Gamma. From the context point of view we can't determine in which order Beta and gamma will be sent by the system which is seen as a black box. In fact they could be sent after Alpha reception, after Epsilon reception, after both Alpha and Epsilon reception. Moreover Gamma and Beta could be separated by the sending of Epsilon message. The possible scenarios are represented in the Figure 4.

Considering the following SDL system composed by a single process, a static analysis will find out that Alpha and Epsilon are independent.

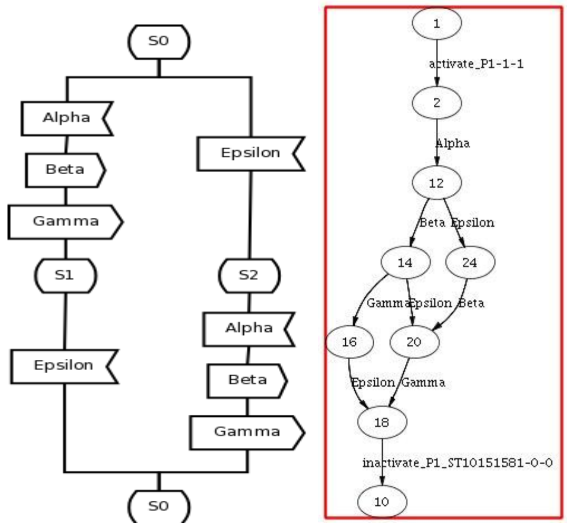

Figure 5: SDL system and reduced graph scenario

At this step we must reconstruct the possible graph scenarios by removing redundancies induced by the independent messages. In the picture 5, we can see that the initial graph context has been reduced when removing redundancies. Since we don't know anything about the order of reception of Beta and Gamma and since we don't want to execute the system to figure out the precise order, the sub graph contains the scenarios with Beta and Gamma in any possible sequence.

For practicality, this step will be done on the fly without building the entire scenario graph at first. Further composition with the system will completely determinate the only executable scenario. In fact, expanding the graph and composing each scenario with the system is not very efficient because a lot of them can't be totally synchronised with the system. For instance the scenario with the sequenced messages Epsilon followed with Beta can't be simulated on the previous system because the system will be in the state S_2 waiting for Alpha and in the meanwhile the context will be waiting for Beta. In order to avoid useless such composition, we choose to directly compose the sub graph with the system which will make his choice. Only one composition is needed instead of three if the context is expanded.

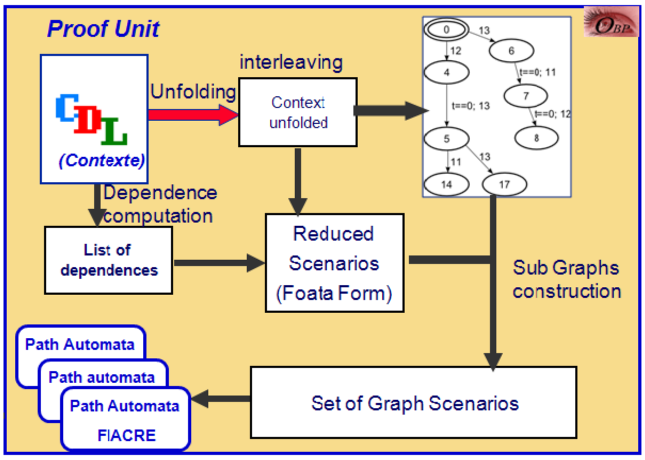

Figure 6: Reduction method

4. Experimentation: SDL models

In this paper, we focused our work on the control part of the system without taking account of the data part: affectations, parameters. We can expect that the dependence introduced with the data will not strongly decreased the reduction obtained because those data will be taken into account only on decision (comparisons) and on the parameters sent by the output messages. Decisions have already been taken into account when computing the control independence. In fact decisions generally prevent the diamond property to be fulfilled because of the introduction of non-determinism (in case condition is not executed). Moreover transmission of parameters is made when exchanging messages. Those parameters can't modify the behaviour of the system if no decisions are attached to them. We can thus reasonably expect a little loss of reduction.

We have applied this method on several examples of growing complexity designed in SDL specification language. First we applied the technique on mono process systems and then extended it with multi process systems on avionics applications.

5. Results

Model	**Processes**	**Actors**	**Initial Scn**	**Reduced Scn**
DATCAPT	**1**	**2**	**24**	**4**
CPDLC	**1**	**3**	**840**	**2**
AFN	**3**	**4**	**1238**	**301**

We can see that the reduction could be very important applying partial-order methods. In case of the CPDLC[3], we obtain a very big reduction because the process concerned the starting of the application where it has to start all the other ones. In this case, the process doesn't know in which order processes will be instantiated so that saving messages are present in all states. At the end, only two scenarios are needed corresponding to a cold start and a reset of the system. In case of the AFN[4], 3 processes have been modelled. The reduction is more limited than the CPDLC because save instances are less present and there are presence of non-determinism with the decision clauses (conditions on variables).

6. Conclusion and perspectives

We have presented in this paper a method combining context description method and partial-order application. We obtained promising results on avionic applications. This work will be further applied with the data part. Partial-order applications have been first introduced to untimed systems application. Little work has been made in case of temporised systems [20, 21]. We will extend our work to temporised SDL systems with the introduction of the timers. We can notice that the complexity of the context is mainly due to the number of concurrent actors identified. A methodology of context construction is another way to further reduce the context.

7. References

[1] Myers, Glenford J and Sandler, Corey: "*The art of software testing*", John Wiley & Son, 2004.

[2] Manna, Z., Pnuelli, A.: "*The temporal logic of reactive and concurrent systems*", New York, 1992.

[3] Clarke, E.M., Emerson, E.A., Sistla, A.P.: "*Automatic verification of finite-state concurrent systems using temporal logic specifications*", ACM Trans. Program. Lang. Syst., vol. 8, p. 244--263, ACM, New York USA, 1986.

[4] Dhaussy P, Auvray J, De Belloy S, Boniol F and Landel E: "*Using context descriptions and property definition patterns for software formal verification*", Workshop Modeva'08, ICST 2008, Lillehammer Norway, 2008.

[5] R. Alur and R.K.Brayton and T.A. henzinger and S.Qadeer and S.K. Rajamani: "*Partial-Order Reduction in Symbolic State Space Exploration*", p. 340--351, Springer, 1997.

[6] P. Dhaussy., PY Pillain, S. Creff, A. Raji, Y. Le Traon,B. Baudry, *Evaluating Context Descriptions and Property Definition Patterns for Software Formal Validation. In* Lecture Notes in Computer Science 5795, Springer Verlag, Andy Schuerr, Bran Selic (Eds): Model Driven Engineering Languages and Systems (Models'09), No 5795 (2009), pages 438-452.

[7] Haugen, O., Husa, K.E., Runde, R.K., Stolen, K.: "*Stairs: Towards formal design with sequence diagrams*", Software and System Modeling, 2005.

[8] B. Berthomieu, P.-O. Ribet, F. Verdanat: "*The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets*", International Journal of Production Research, vol. 42, 2004.

[9] Bozga, M., Graf, S., Mounier, L.: "*A validation environment for component-based real-time systems*", CAV 2002, LNCS, vol. 2404, p.343, Springer, Heidelberg, 2004.

[10] Fernandez, J.-C., et al.: "*CADP: A Protocol Validation and Verification Toolbox*", CAV 1996, LNCS, vol. 1102, Springer, Heidelberg, 1996.

[11] Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: "*Patterns in property specifications for finite-state verification*", Proceeding of the 21st International Conference on software Engineering, p. 411--420, IEEE Computer Society Press, Los Alamitos, 1999.

[12] Roger, J.C.: "*Exploitation de contextes et d'observateurs pour la vérification formelle de modèles*" , Phd report, Univ. Of Rennes I, France, 2006

[13] ITU-T: "*Recommendation Z.120. Message Sequence Charts*", International Telecommunication Union – Standardization Sector, Geneva, 2000.

[14] A. Mazurkiewicz: "*Trace theory. In Petri nets: Applications and Relationships to Other Models of Concurrency*", Advances in Petri Nets, Part II, Proceedings of an Advance Course, 255, Lecture Notes in computer Science, p. 279--324, Springer-Verlag, 1986.

[15] A. Valmari: "*Stubborn sets for reduced state space generation*", Proceedings of the 10th International Conference on Applications and theory of Petri Nets, p. 491--515, Springer-Verlag, London, UK, 1991.

[16] D. Peled: "*All from one, one for all: on model checking using representatives*", CAV'93: proceedings of the 5th International Conference in Computer Aided Verification, p. 377--390, Springer-Verlag, London U.K., 1993.

[17] P. Godefroid: "*Partial-order methods for the verification of concurrent systems: an approach to the state space explosion problem*", Springer-Verlag, N.Y. USA, 1996.

[18] Deikert, Volker, Métivier: "*Partial commutation and traces*", Handbook of formal languages, vol. 3: beyond words, vol. 3, p. 457--533, Springer-Verlag, New york USA, 1997.

[19] A. Arnold, A. Vincent and I. Walukiewicz: "*Games for synthesis of controllers with partial observation*", 303(1), p. 7--34, Theoretical Computer Science, 2003.

[20] Pagani F: "*Partial Orders and Verification of Real-Time systems*", FTRTFT, p. 327--346, 1996.

[21] Bengtsson J, Bengt Jonsson, Lilius J and Yi W: "*Partial Order reductions for Timed Systems*", 1998.

[3] Controller Pilot Data Link Communication

[4] ATIS Facility Notification