# Formal Validation of UML Statechart Diagrams Models

Stefania Gnesi[1] and Diego Latella*[2], Istvan Majzik[3], and Mieke Massink**[2]

[1] Consiglio Nazionale delle Ricerche
Istituto IEI, Pisa, IT
gnesi@iei.pi.cnr.it
[2] Consiglio Nazionale delle Ricerche
Istituto CNUCE, Pisa, IT
d.latella@cnuce.cnr.it, m.massink@cnuce.cnr.it
[3] Technical University of Budapest
Dept. of Measurement and Information Systems, Budapest, H
majzik@mit.bme.hu

## 1 Introduction

In this paper we focus on UML Statechart Diagrams (UMLSDs), which are meant for describing dynamic aspects of system behaviour. In particular, we give an overview of our work on an integrated framework and tools environment for the formal validation of UMLSDs and the automatic analysis of some quantitative attributes of theirs. Most of the work presented here is ongoing research which is taking place in CNR, Pisa in cooperation with the University of Budapest. Our overall goal is to provide an environment for formal verification via model-checking and quantitative analysis also via simulation of a behavioural subset of UMLSDs. Our position is that verification as well as quantitative assessment tools and environments must be characterised by High Quality Assurance standards both for their definition and for their implementation. This allows us to safely rely on the verification and validation results they produce. The methodology we follow for achieving this goal and assuring high standards is based on the use of (a) formal definition of the syntax and semantics of the notation subset as well as of (b) rigorous and, whenever convenient, formal proofs concerning features of the notation and correctness of the implementations. This in turn implies rigorous and, whenever applicable and tractable, formal specification of the software tools included in the environment. Consequently a central role is played by the definition of a formal operational semantics for the notation. The work described in the present paper is based on the operational semantics we proposed in [16]. Such a semantical model is defined for a restricted subset of UMLSDs, still including all the interesting conceptual issues related to concurrency in the dynamic behaviour, like sequentialisation, non-determinism and parallelism. It

* Contact author. Phone: +39 050593230. Fax: +39 050904052

** Supported by the TACIT network under the European Union TMR Programme, Contract ERB FMRX CT97 0133.

also covers state refinement and inter-level transitions. More specifically, we do not consider history, action and activity states; we restrict events to signal and call events, without parameters (actually we do not interpret events at all); time and change events, object creation and destruction events and deferred events are not considered as are branch transitions; also variables and data are not allowed so that actions are required to be just (collections of) events. We also abstract from entry and exit actions of states. The above restrictions are made essentially for simplicity since, in our opinion, most of them do not have any strong impact on the behavioural aspects of the semantics and tools at a conceptual level, but dealing with them would dramatically and uselessly complicate the notation involved. Other limitations, namely the fact that we do not deal with the object-oriented features of UMLSDs, e.g. sub-behaviours, are more serious. Basic formal semantics models and related tools, even for a restricted language, are an essential step for any further extension with the above mentioned features. The definition of a sound "basic" subset or kernel of a notation, on which to stress novel semantics concepts as well as develop mathematical theories, like behavioural preorders or equivalences, and experiments with specific tools, like model-checkers, has already proven a valuable, safe and fruitful methodology and is now quite standard practice in many fields of concurrency theory, like process-algebra. Once concepts, theories and tools have been developed for a restricted notation their extension to other, important, features of the notation can be supported by a safer background. For instance, it is our opinion that a sound formal semantics for UMLSDs is a necessary condition for extending the considered notation with the inclusion of object-oriented features like classes and subclasses. In fact the formal semantics serves as a necessary starting point for the definition of behavioural (ordering) relations which can play a role in the definition of the notion of sub-behaviour, connected to the notion of sub-classes [4]. Finally, we want our environment to be open: its user should be allowed to use different notations (UML, PROMELA, Automata, Process Algebra, Temporal Logics, etc.) since we think it is unlikely that one single notation can cover all aspects of interest of the design of a system, with current technology.

## 2 A Formal Approach to UMLSD Behavioural Semantics

All the work we are performing is based on the formal operational semantics we defined in [16], which uses a UML variant of Hierarchical Automata (HAs) [20] as abstract syntax. The semantics definition is composed by a top-level rule which makes use of the so called "core-semantics". The core-semantics is defined by a deduction system composed of only three rules. The definition is recursive, based on the hierarchical structure of HAs. We refer the interested reader to [16] for all technical details on syntax and semantics. Here we want to point out that our definition of the semantics is parametric w.r.t. the priority schema for transitions as well as the selection policy of the dispatcher. Moreover, in [14] our semantics has been proven correct w.r.t. the requirements informally stated in the UML official definition documents.

### 2.1 Extensions

**Multicharts** The general idea of the UML designers is to associate a distinct statechart to each class or object and then let such statecharts communicate via queues. Although we are not fully convinced of the methodological soundness of such an approach, as discussed in detail in [15], our semantics can easily be extended for coping with system descriptions consisting of more than one statechart diagram. All what is needed is to change the top-rule and to properly address messages to different HAs - every HA being equipped with its own input queue. See [10] for formal definitions and details.

**Real-time and Stochastic UML Statechart Diagrams** In [12] a stochastically timed extension of UMLSDs has been proposed. The extension is rather simple both from a notational point of view and from a semantics point of view. In particular, following the Stochastic Automata approach of D'Argenio [7, 8], we enrich UMLSDs with random (descending) clocks which can be set when states are entered and which can be used as guards for transitions: a transition can fire only when all clocks guarding it reach zero. Consequently we enrich HAs and their operational semantics accordingly. The semantics of Stochastic UMLSDs are Stochastic Automata. The operational semantics definition has been extended in order to deal with random clocks. The extension is technically simple and allows to use powerful analysis techniques. Furthermore, it is orthogonal in the sense that the automaton of the basic, untimed, operational semantics is the same as that of the stochastic semantics, once clocks and clock settings are removed. Orthogonality can be proven by derivation induction [12].

### 2.2 Verification and Assessment

**Linear Time Model-checking of UMLSDs** In [15] a linear time model-checking facility for UMLSDs has been presented. The target language is PROMELA, the specification language of the SPIN model checker. SPIN [13] is one of the most advanced analysis and verification tools available nowadays. The translation is derived from the operational semantics. It is simple, proven correct, and promising in terms of state space representation efficiency. Two implementations of the translation exist [9, 2]. Several experiments, including a specification and verification by means of model checking of the Production Cell, have been performed on the implementation described in [2]. The interested reader is referred to [18].

**Branching Time Model-checking of UMLSDs** In [11] a branching time model-checking approach to the automatic verification of formal correctness of UMLSD specifications. The approach is based on the operational semantics and the implementation of the state enumeration algorithm is under way. Our reference verification environment is JACK [3], where automata are represented in a

standard format, which facilitates the use of different tools for automatic verification. We are not aware of other proposals of branching time model-checking for UMLSDs. The benefits of branching time logics have been widely recognised in the literature and it is also well known that linear time and branching time logics are incomparable as far as the expressive power is concerned [6].

**Stochastic Analysis** We discussed above a stochastic extension of UMLSD and their semantics. By means of the rules of the operational semantics, a stochastic automaton can be generated automatically from a Stochastic UMLSD model, using essentially the same enumeration algorithm as that mentioned in the previous section. Once such a stochastic automaton is generated, analysis of quantitative features of its behaviour can be performed directly on it. To that purpose one can use discrete simulation tools available for stochastic process algebras [7, 8] when non-markovian processes are involved, which is the majority of cases due to the synchronous nature of the STEP relation. In the case the stochastic automaton is markovian then other tools like markovian model-checkers [1] or more traditional tools can be used. We want to stress here that the orthogonality theorem sets a formal link between the functional behaviour of a (enriched) UMLSD model of a system and the quantitative analysis one performs on it. Such a link contributes in increasing confidence on the results of such an analysis. In certain cases, the model for quantitative analysis needs to be more abstract than the functional one. Our approach allows us to unambiguously define what we mean by "more abstract". This can be done my means of proper behavioural relations on automata, thus preserving the formal link between functional models and quantitative ones. The issue of behavioural relations will be briefly discussed below.

## 3 Behavioural Relations

In [4] a study on behavioural relations and their relationship with subtyping is presented. It is argued that failure relations (e.g. preorders), among others, play a significant role in the study of subtyping. In [10] some preliminary ideas are presented on which we are working currently for the definition of a testing theory and related behavioural relations for UMLSDs. We believe that such a theory can provide a sound framework for investigating the relation between sub-classes and their associated behaviours expressed by UMLSDs as well as for formal derivation of test cases. As for other extensions/manipulation we discussed above, also in the case of testing we only need to properly rearrange the top-level rule of the operational semantics, once again leaving the core-semantics unchanged. In the operational semantics of UMLSDs given in [16] a status is a pair containing also the current environment. In the testing framework, instead, we let statuses coincide with configurations and the current environment be modelled separately by the "experimenter". The transition relation of a HAs is now labelled by input/output pairs (e,E). The input component 'e' represents the stimulus for the STEP transition to fire while the output component 'E' is a

collection of events which the hierarchical automaton returns to the environment as (part of ) the reaction to the stimulus. We use the notion of Output Respecting Experimenters [19] in order to take care of the peculiar input/output interaction pattern of UMLSDs as opposed to synchronisation in process algebras [Hen88]. We define a proper notion of Experimental System which we use for proving the correctness of the new semantics w.r.t. that given in [16] as well as for developing the standard notions of testing theory - Computations, Successful Computations, Testing Equivalence, Testing Preorders.

## 4 Conclusions and Related Work

In this paper we presented our approach to the definition of a framework and environment for verification and analysis of UMLSDs and their timed extensions. We think that the overall design process of such an environment must be driven by formal semantics and rigorous and, whenever possible and convenient, formal derivation of tools from the semantics. Several approaches have been proposed in the literature for the definition of a formal semantics of UMLSDs, e.g. [22, 5, 16, 17], and much more has been done for classical statecharts. To the best of our knowledge, transition priorities are dealt with neither in [22], where also state refinement is not allowed, nor in [5], where model checking is addressed. Both transition priorities and state refinement constitute main issues in our work. The approach we followed is similar to that proposed in [20] for classical statecharts but it takes into consideration the peculiarities of the UMLSDs relevant for the considered subset of the notation. On the other hand, it shares the relative simplicity of the work proposed in [20]. In [17] all interesting aspects of UMLSDs semantics are covered. Unfortunately, no correctness result for the proposed semantics is provided. More emphasis is put on implementation related issues as the work constitutes a basis for a PROMELA/SPIN based model-checker for UMLSDs. In [17] a 'flat' representation of UMLSDs is used and the authors claim that such a representation is better suited for model-checking purposes than the hierarchical one used in [16]. We definitely do not share this opinion: using a hierarchical representation for UMLSDs (syntax), not only has no negative impact on tools development, but, rather, it helps very much in carrying on correctness proofs; all interesting results in our work are proven inductively and such proofs heavily exploit the hierarchical structure of our representation, which is also the basis of the structure of our semantics deduction system. In designing the UMLSDs to PROMELA translation we followed an approach similar to that of Mikk et al. [21]. Nevertheless, our work differs substantially from theirs. First of all, in [Mi+97] classical statecharts are considered instead of UMLSDs. So, the complications induced by the UML transition priorities and their reverse relation with the hierarchical structure of the statechart are not present therein, whereas they are dealt with in our work. More specifically, we use a notion of 'priority schema', on which the semantics are 'parametric', and then we instantiate it with the UML specific one. Moreover, since the UML definition of the external environment is only partially defined,

our semantics definition as well as our translation are parametric w.r.t. the environment. In the above mentioned work of Mikk et al., the environment is instead represented as a set, as required by the classical statechart semantics. Moreover, due to some simple optimisations, the code generated by our translator is considerably simpler than that of the translation proposed in [21]. Also, we do not need to use pre- and post-variables, so that the code generated by our translator does not suffer from the memory duplication problem which in the above mentioned work requires specific optimisation techniques. Finally, we are not aware of any correctness result for the work presented in [21].

## 5 Acknowledgements

The work presented in the present paper has been partially funded by the ESPRIT Project n. 27439 - HIDE. Istvan Majzik has been partially supported by the Hungarian Scientific Research Fund OTKA-F030553. Mieke Massink has been supported by the TACIT network under the European Union TMR Programme, Contract ERB FMRX CT97 0133.

## References

[1] C. Baier, J. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time markov chains. In J. Baeten and S. Mauw, editors, *Concur '99*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 1999.

[2] A. Borschet, M. Dal Cin, J. Javorszky, and C. Szasz. Specification of the HIDE environment. Technical Report HIDE/D3/TUB/1/v2, ESPRIT Project n. 27439 - High-Level Integrated Design Environment for Dependability HIDE, 1998.

[3] A. Bouali, S. Gnesi, and S. Larosa. The integration project for the jack environment. *Bulletin of the EATCS*, (54):207–223, 1994.

[4] H. Bowman and J. Derrik. A junction between state based and behavioural based specifications. In P. Ciancarini, A. Fantechi, and R. Gorrieri, editors, *IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Oriented Distributed Systems*. Kluwer Academic Publishers, 1999. ISBN 0-7923-8429-6.

[5] J. Broersen and R. Wieringa. Interpreting UML-statecharts in a modal $\mu$-calculus. Unpublished manuscript, 1997.

[6] E. Clarke and I. Draghicescu. Expressibility results for linear time and branching time logics. In J. de Bakker, C. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Tima and Partial Order in Logics and Models for Concurrency. REX School/Workshop*, volume 354 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[7] P. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, 1999.

[8] P. D'Argenio, J. Katoen, and E. Brinksma. Specification and analysis of soft real-time systems: Quantity and quality. In *Real-Time Systems Symposium*, pages 104–114. IEEE - The Institute of Electrical and Electronic Engineers, 1999.

[9] E. Giusti and D. Latella. Implementazione in SML di un traduttore da automi gerarchici a PROMELA. Technical Report CNUCE-B4-1998-018, Consiglio Nazionale delle Ricerche, Istituto CNUCE, 1998. In italian.

[10] S. Gnesi, D. Latella, G. Lenzini, C. Abbaneo, A. Amendola, and P. Marmo. Formal specification and validation of a critical system in presence of byzantine errors. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *Lecture Notes in Computer Science*, pages 535–549. Springer-Verlag, 2000.

[11] S. Gnesi, D. Latella, and M. Massink. Model checking UML statechart diagrams using JACK. In A. Williams, editor, *Fourth IEEE International High-Assurance Systems Engineering Symposium*, pages 46–55. IEEE Computer Society Press, 1999. ISBN 0-7695-0418-3.

[12] S. Gnesi, D. Latella, and M. Massink. A stochastic extension of a behavioural subset of UML statechart diagrams. In V. Winter, editor, *Fifth IEEE International High-Assurance Systems Engineering Symposium*, 2000. (To Appear).

[13] G Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[14] D. Latella, I. Majzik, and M. Massink. A simplified formal operational semantics for a subset of UML statechart diagrams. Technical Report HIDE/T1.2/PDCC/5/v1, ESPRIT Project n. 27439 - High-Level Integrated Design Environment for Dependability HIDE, 1998.

[15] D. Latella, I. Majzik, and M. Massink. Automatic verification of UML statechart diagrams using the SPIN model-checker. Technical Report CNUCE-B4-1999-008, Consiglio Nazionale delle Ricerche, Istituto CNUCE, 1999.

[16] D. Latella, I. Majzik, and M. Massink. Towards a formal operational semantics of UML statechart diagrams. In P. Ciancarini, A. Fantechi, and R. Gorrieri, editors, *IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Oriented Distributed Systems*, pages 331–347. Kluwer Academic Publishers, 1999. ISBN 0-7923-8429-6.

[17] J. Lilius and I. Paltor Porres. The semantics of UML state machines. Technical Report 273, Turku Centre for Computer Science, 1999.

[18] I. Majzik and J. Javorszky. Formal verification of UML statecharts: Case studies. Technical Report MITUB-TR-99-05, Dept. of Measurement and Information Systems - Technical University of Budapest, 1999.

[19] M. Massink. *Functional Techniques in Concurrency*. PhD thesis, University of Nijmegen, February 1996. ISBN 90-9008940-3.

[20] E. Mikk, Y. Lakhnech, and M. Siegel. Hierarchical automata as model for statecharts. In R. Shyamasundar and K. Euda, editors, *Third Asian Computing Science Conference. Advances in Computing Sience - ASIAN'97*, volume 1345 of *Lecture Notes in Computer Science*, pages 181–196. Springer-Verlag, 1997.

[21] E. Mikk, Y. Lakhnech, M. Siegel, and G. J. Holzmann. Implementing Statecharts in Promela/SPIN. Technical Report BL011272-971203-25TM, Bell Labs, Lucent Technologies, 1997.

[22] R. Wieringa and J. Broersen. A minimal transition system semantics for lightweight class and behavior diagrams. In M. Broy, D. Coleman, T. Maibaum, and B. Rumpe, editors, *Proceedings of the ICSE98 Workshop on Precise Semantics for Software Modeling techniques*, 1998.