Closed-Loop Formal Verification Framework with Non-determinism, Configurable by Meta-modelling

Sandeep Patil, SayantanBhadra and ValeriyVyatkin
The University of Auckland, Level 2, Science Centre, 38 Princes Street, Auckland, New Zealand
{spat251, sbha101}@aucklanduni.ac.nz, v.vyatkin@auckland.ac.nz

***Abstract*-Formal verification of embedded control systems using closed-loop plant-controller models is getting increasingly popular. In this paper we propose a new method reducing complexity of model-checking on account of infusing non-determinism into certain parts of the plant model during formal verification process guided by a software tool. Net Condition/Event Systems (NCES) formalism is used for modular design of closed-loop models which are verified by ViVe and SESA model-checkers. Its performance is compared to modelling with finite state verified with SMV and UPPAAL and is proven to be superior**.

Keywords: Model-checking, NCES, meta-modelling, industrial automation, formal verification, simulation

1 INTRODUCTION

Formal verification [1] of embedded control systems using closed-loop plant-controller models [2, 3] is getting increasingly popular. Discrete state model-checking [4] is one of formal verification approaches, which is the process of verifying whether a set of desired specifications is satisfied over the target system (model). Closed-loop modelling allows for thorough verification of the control logic, and reduces the complexity of model checking as compared to only controller verification under an arbitrary inputs assumption. It also allows checking of specifications formulated in terms of the plant variables rather than in terms of controller inputs/outputs. In order to verify control logic under various input combinations, model developers introduce non-determinism into the model of the plant. This non-determinism, if used indiscreetly, still can bring complexity explosion to the model-checking process. Modular composition of automata models using full synchronous automata composition is also computationally challenging even before the model-checking starts. For practical systems, it results in large automata models whose model-checking potential maybe very much restricted. Moreover, it is necessary to verify the correctness of the plant model itself, before it can be trusted to use in the closed-loop verification process. Presence of non-determinism hinders this task substantially.

In this paper we propose a closed-loop modelling method by defining a new NCES formulization and describe the supporting tool-chain which has proven to be much more efficient in model-checking of control systems from industrial automation domain. The method is based upon modelling with Petri-net like formalism of Net Condition-Event Systems which eliminates the cross-product composition explosion. The central part of our solution is the use of explicit timing and selective non-determinism infused in certain parts of the plant model to prevent unnecessary state space generation. The initial plant model is timed and fully deterministic. A meta-model carries the information about the model's structure using semantic references. The model configurator tool allows the user to "infuse" non-determinism only to selected parts of the plant model to test certain controller features. This approach paves the way to systematic application of formal-verification in control systems design.

The paper is structured as follows. In Section II we review one typical approach to closed-loop modelling and verification of a simple control system by means of an existing work. We will introduce the object under control and will discuss process and results of its verification using finite state machines. In Section III we briefly introduce NCES modelling language and describe its benefits. In Section IV the model structure and formularization is introduced along with semantic meta-model that captures model's structure and purpose of its parts. In Section V the results of model-checking are illustrated in a case study. The paper is concluded with future work outline and references.

2 CLOSED-LOOP MODELLING WITH FINITE STATE MACHINES

Model-checking verification is based on exploration of the model's state space which is presented as a state-transition model, such as Kripke structure [4]. Explicit state-space development is difficult due to its large size, therefore various modular design approaches are applied, for example synchronous/asynchronous composition of finite state machines[5], input-output automata [6-8], network of timed automata [9-12] and Petri nets [13, 14].

The synchronous/asynchronous composition of finite sate machines may lead to a huge model, which is impractical to create manually, and, even if created automatically may be too complex for model-checking. The Petri nets based modelling is free of this problem since the Kripke structure is created dynamically during the model-checking.

To demonstrate our approach, we will use a pick and place object shown in Fig.1, which is composed of several mechatronic units as follows:

1) There are two horizontal cylinders and a vertical cylinder that extract and retract. The left horizontal cylinder is half the size of the right cylinder. The vertical cylinder picks up the work pieces using the suction unit attached to its end.
2) Both horizontal cylinders have two control signals (CGO: Cylinder Go Out: extending, CGI: Cylinder Go In: Retracting). The vertical cylinder has only one control

signal (VCGD: Vertical Cylinder Goes Done). When this signal is not active, the cylinder moves up (pulled by the internal spring).

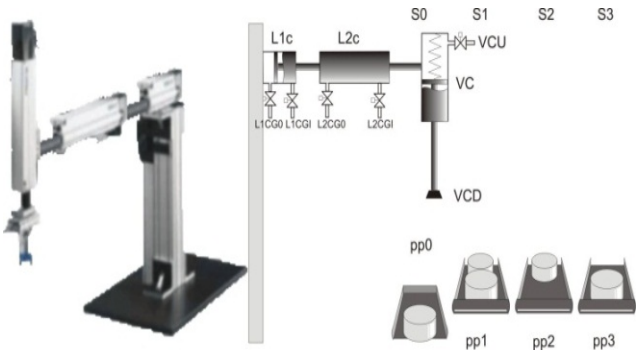


Fig.1. Reference object: pick and place manipulator.

3) S0, S1, S2 and S3 are four sensors that determine the position of the input trays and an output slider. S0 is the output slider and the rest are input tray sensors. There are also sensors in each of the three input trays (pp1, pp2 and pp3) and one in the slider (pp0) to indicate the presence of a work piece. The suction unit has a built-in sensor, VACUUM indicating the presence of the work piece.

In order to design a modular model providing for good reusability of the modelling, Machado [15, 16] proposes hierarchical model structure and considers several options for granularity. As seen in Fig.2 (level 1), one extreme is a monolithic modelling that does not allow any modularity and hence there is no reusability. However, it is too complicated to design a system this way. Level 2 has model based with functional unit and thus allows having easy links between inputs (order), movements and output (values of sensors). Level 3 is a very precise modelling approach with elementary components equivalent to sensor and actuators of the plant.

This level 3 solution was used by Machado in [15] to modularize the pick and place system and we will use same hierarchical model structure in our design of the same system, but in NCES.

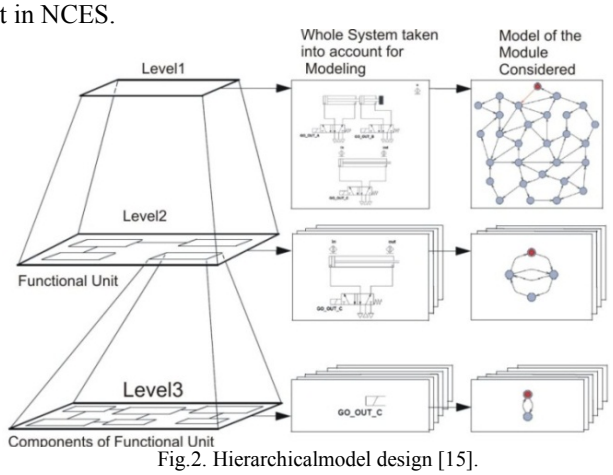
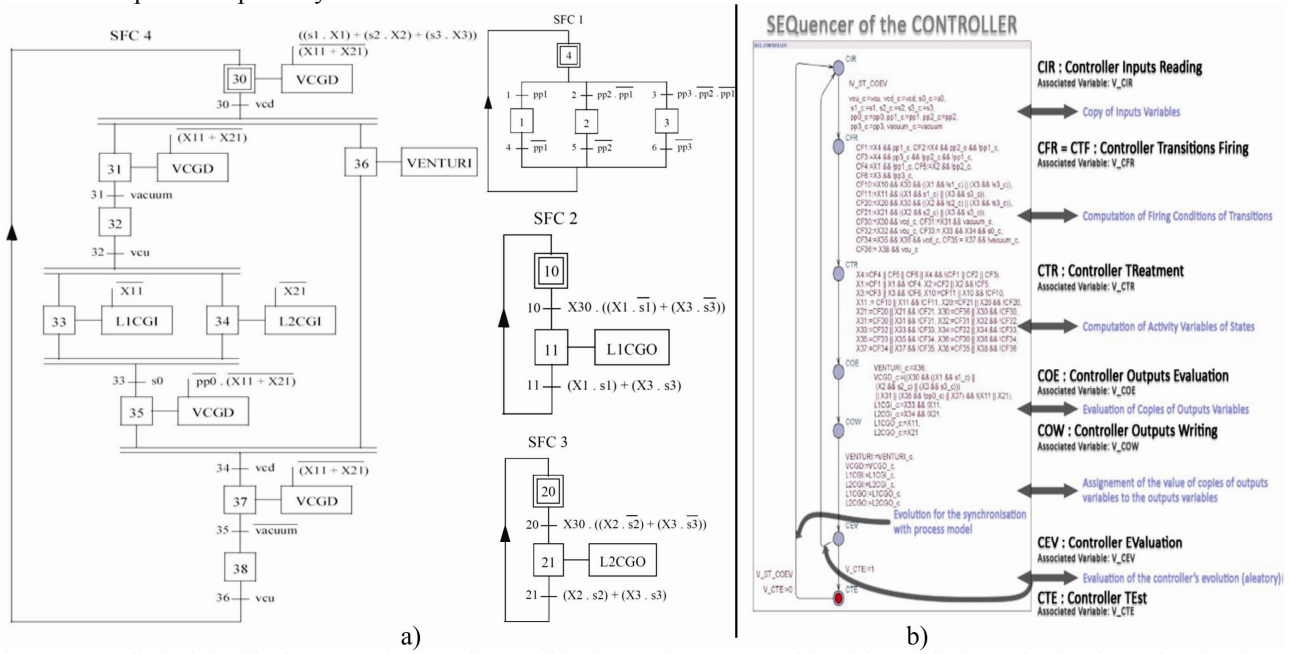


Fig.2. Hierarchicalmodel design [15].

Let's look at the controller modelling. Machado [15, 16], assumes that the controller has a cyclic behaviour, but not necessarily periodic, that is the controller executes its one iteration without interruption from SEQ_GEN, unlike plant model that is periodic. Sequential Function Chart (SFC) [17, 18] is used to define the logic of the controller (Fig.3(a)). The dynamic behaviour of the plant is the result of the combined behaviour of its own dynamic model and behaviour of the controller during its execution.

1) SFC1: The SFC determines what work piece to pick up. If all the work pieces are present, then they will be picked in the increasing order of their numbers.
2) SFC 2 and SFC 3: The SFC controls the movement of the left and right horizontal cylinders.
3) SFC 4: The SFC controls the movement of vertical cylinder and also controls when to pick up a work piece and when to drop a work piece.



a) b)

Fig.3 (a) Controller is defined by four SFCs; (b) Controller model implemented as a state machine with SFCs logic translated to the Boolean functions – transitions and assignments [19].

Controller model is implemented as a single state machine as shown in Fig.3(b)[19] with SFC's logic implemented as a number of Boolean assignments. The same model also acts as the controller sequencer (implementing the cyclic behaviour). Each iteration stores the values of the present plant variables (sensors) in its internal variable in order to calculate the next control signal to be sent to the plant.

3 PROPOSED MODELLING APPROACH

A. *Net Condition-Event Systems*

Net Condition/Event Systems (NCES) [20] can be viewed as a modular extension to Petri Nets [14]. The general idea of NCES is modelling a system as a set of modules with a particular dynamic behaviour and their interconnection via signals. An illustrative example of the graphical notation of a module is provided in Fig.4(a).

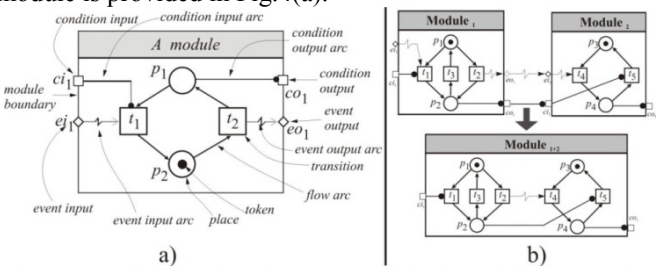


Fig.4 (a) Graphical notation of an NCES module; (b) Modular Composition of NCES modules results in a "flat" NCES.

Once designed, the modules can be re-used over and over. Each module has inputs and outputs of two types:

1) Condition inputs/outputs carrying information on marking of places in other modules.
2) Event inputs/outputs carrying information on firing transitions in other modules.

Condition and event inputs are connected with some transitions inside the module by condition and event arcs. Places of the module can be connected to the condition outputs by condition arcs, and transitions can be connected to the event outputs by event arcs. This concept provides a basis for a compositional approach to building larger models from smaller components. The "composition" is performed by "gluing" inputs of one module with outputs of another module as shown in Fig.4(b).

B. *Benefits of NCES as compared to state machines*

There are two main reasons to prefer place-transition formalisms to many others formalisms, e.g. finite automata. The first is their non-interleaving semantics (i.e. possibility of firing several transitions simultaneously), which better fits to modelling of distributed processes and of their interaction. It results in more compact reachability space, explained as follows.

Modelling of complex distributed systems with automata usually ends up in many concurrent automata models communicating via common variables, as illustrated in Fig.5(a), where two state machines A and B are combined under "asynchronous parallel operator". Thus, the overall system model is a cross-product of the component automata, and to do model analysis it is necessary to build the cross-product consisting in this case of 9 states, as one sees in Fig.5(a). Alternatively, in NCES a state of a model is determined by the marking of model places, so any global state of a distributed system is just one state of the model. This is shown in Fig.5(b), where the same model is implemented in NCES with places (p1-p6) corresponding to states of the automata A or B (in the obvious manner). In the given initial state the reachability space of the model consists of only 4 states. The same behaviour obviously will be shown by the automata model in Fig.5(a) (the outlined path A1B2→ A1B2→ A2B2→ A2B3→ A3B1→ A3B2), but to get it the whole cross-product automata needs to be built.

The other benefit of NCES is the intuitive modelling approach. Consider an NCES equivalent pick and place model conceptually presented in Fig.6. The NCES model of a moving object, such as a cylinder, is composed in a modular way following the pattern proposed in [5]. The NCES modules are connected with explicit event and data flow as compared to sharing variables in state machines. As noted in state machines model, two sequencers were used to order evolution of the automata. This is a considerable restriction of real life evolutions. In NCES, the controller output is directly connected to the plant input and vice versa, and the models make evolutions asynchronously.

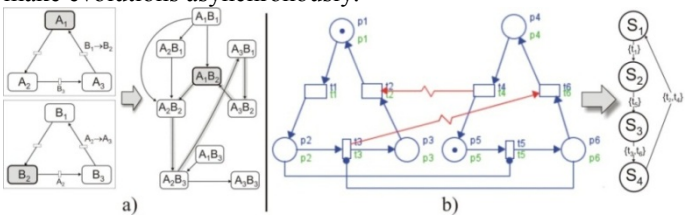


Fig.5. (a) Modelling of two communicating processes by means of concurrent state machines and their cross-product automaton; (b) The same model in NCES and its reachability graph.

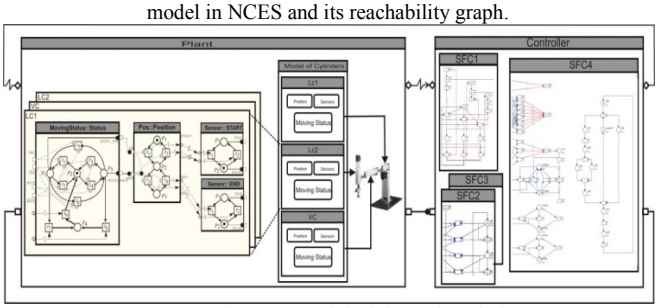


Fig.6. Conceptual NCES model of the pick and place system.

C. *Using controlled non-determinism*

We define a sub class of NCES to introduce the concept of controlled non-determinism. $_C$NCES is a tuple

$_C$NCES = {P, T, F, C_N, E_N, C_{in}, E_{in} , C_{out}, E_{out}, CI_{arc}, EI_{arc}, CO_{arc}, EO_{arc}, NDP}, where: P is the set of places, T is the set of transitions, F ⊆ (P YT) X (TYP) is the set of flow arcs, C_N⊆ P X T is the set of condition signals, E_N⊆ T X T, is the set of event signals, C_{in} is the set of condition inputs, E_{in} is the set of event inputs, C_{out} is the set of condition outputs, E_{out} is the set of event outputs, CI_{arc}⊆Cin X T is the set of condition input arcs, EI_{arc}⊆E_{in} X T the set of event input arcs, CO_{arc}⊆ P X C_{out} the set of condition output arcs, EO_{arc}⊆ T X E_{out} the set of event output arcs and NDP ⊆ P is set of places that represent non-deterministic places.

The main benefit of model-checking as compared to verification by simulation is ability to add non-determinism into the model thus checking it in more scenarios. For example, various failures in the plant will require testing the controller's behaviour for each failure and for their various combinations.

In our proposed modelling approach, we start with a fully deterministic plant model where prolonged actions are modelled using time delays implemented with timed arcs. Then we identify possible failures to be modelled by non-determinism. For example, in our pick and place model, the work piece may fall off or get stuck in the tray or even may get stolen (removed). Such cases of work pieces disappearing need to be modelled and tested for these abnormal scenarios. Hence non-determinism is introduced so as to make the plant behave abnormally and check if the controller is well designed to handle such an abnormality. In NCES, modelling of non-determinism is achieved via a conflict. For example, in Fig.7(a), the token can go either to place p8 or p1 from p3. Since there are no conditions associated with the transitions t3 and t5, except for p3 to have a token, the transition can occur anytime. The immediate side effect of such non-determinism is state space explosion caused by the obvious increase in the number of state transitions from a given state where place p3 is marked. One way to control state space explosion is by controlling when these non-deterministic transitions can take place.

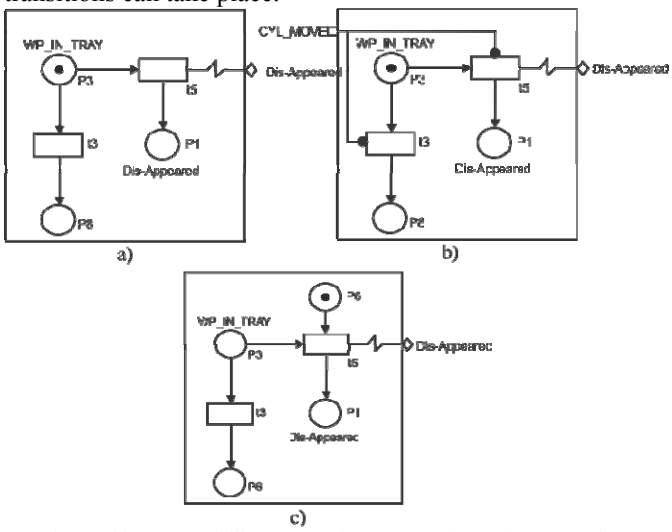


Fig.7. Different modelling approaches for non-determinism. (a) shows a conflict. (b) shows a conflict only when a condition is true (controlled non-determinism) and (c) shows how to control presence of non-determinism during modelling.

Fig.7(b), shows a modified version of the non-deterministic model, where the transitions t8 and t5 are governed by one more condition that these transitions should fire only on a condition CLY_MOVE. This way it is easier to control the state space explosion. We also propose basic model architecture in Fig.7(c) which allows us to choose whether non-determinism is present or not in a particular part of the model without changing the model structure. If a token is present in p6, then non-determinism is turned on, else it is turned off as the t5 can never fire. Fig.7 shows ease of modelling that stands out with NCES, unlike in state machines where we need to use guards and synchronization (which is difficult to understand in textual form).

4 USING META-MODELLING FOR MODEL GENERATION AND RE-CONFIGURATION

The centre of our modelling approach is the ability to introduce controlled non-determinism in selected modules of the plant, for example, only in the work-piece model of our pick and place manipulator. In order to achieve this we use a component called the Configurator in Veridemo tool chain that is currently under development.

The tool chain is used for the validation and formal verification of Programmable Logic Controller code with the aid of simulation. It encompasses many attributes and is based on active research in the field of formal verification of distributed embedded control systems notably the ones described in [3, 21-27]. However, the detailed description of the tool chain is outside of the scope of this paper. We will discuss here only one of its features, which is model generator/configurator. The tool supports automatic model generation based on the structure of the Plant. It is assumed that the Plant is assembled from intelligent mechatronic components (IMC) [24, 28, 29] and the tool's library contains a (parameterized) NCES model for each such component as shown in Fig.8.

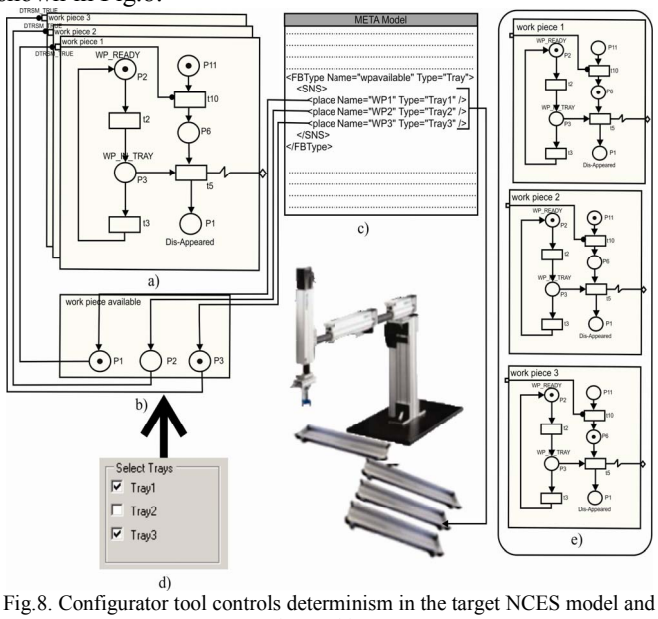


Fig.8. Configurator tool controls determinism in the target NCES model and its working.

Fig.8(a) shows the three instantiations on the same work piece model that will be auto configured by the tool chain. Depending on the condition DTRSM_TRUE, which is set in the "work piece available" model as seen in Fig.8(b), the transition t10 will be fired only in those instances of the work piece model that are configured to be non-deterministic. Fig.8(c) is the meta-model (in the form of XML) that describes which is the model that supports non-determinism and number of instantiations of those models. The tool provides an interface to configure the determinism as in Fig.8(d). The configurator then instantiates a particular model type with non-determinism or without non-determinism which will configure the NCES "work piece available" model

as shown in Fig.8(b). For example, in Fig.8 work piece 1 and 3 will be non-deterministic. Fig.8(e) shows the three instantiations of the work piece model. The presence of token in p6 in the instantiation of work piece model 1 and 3 implies non-determinism and absence of token in p6 in instantiation of work piece model 2 implies determinism.

5 CASE STUDY

The base NCES model of the plant (without our controlled non-determinism) was timed and deterministic. This model was checked against a set of CTL specifications of safe and correct behaviour. One such property is, "if only work piece 2 is present, only the right cylinder should extend and the work piece should be picked up". Fig.9(a) shows a valid behaviour of the above property, left cylinder is still in retracted position and right cylinder is extended.

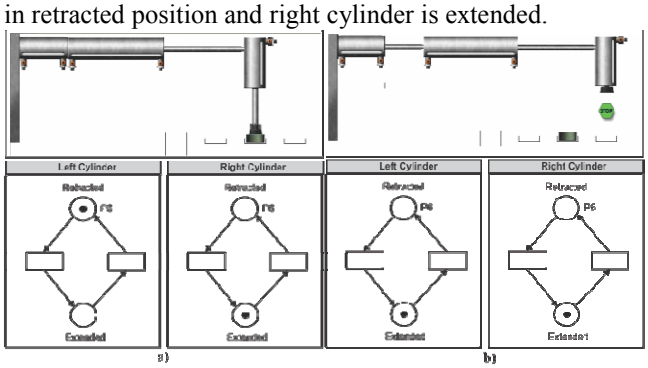


Fig.9. a)Valid behaviour in a deterministic model; b) Invalid behaviour in a non-deterministic model.

Then the model was checked after non-determinism was introduced to model one abnormal behaviour of the plant (work piece disappears) due to unpredictable external influences. The same property like in above was checked i.e. "if only work piece 2 is present, only the right cylinder should extend and the work piece should be picked up". But due to non-deterministic model where work piece disappearance was modelled, the property check failed. The failure scenario is illustrated in Fig.9 (b) which shows both left and right cylinders being extended and system in a deadlocked state. The ViVe/SESA tool also gives us a counter example trace showing how it failed. The sequence was:

1) All the three work pieces arrived.
2) Cylinder 1 starts to extend to pick up work piece 1.
3) Then work piece 1 and 3 disappear.
4) Cylinder 2 also starts extending and system enters a deadlocked state.

A. Case study results

In our case study we studied verification of Machado's finite automata model (fully non-deterministic) and both the deterministic and non-deterministic (controlled) NCES models on a system with 1) Intel dual core CPUD525@1.8 GHz, with 2GB of RAM, 2) Windows 7 operating system, 3) NuSMV 2.5.0: with option "–reorder" and "–dynamic generally" to increase the speed of verification – for finite automata, 4) ViVe 0.37b version – for NCES model.

The state space of the deterministic NCES model was less than 600 states and all the CTL properties were verified to pass. The verification in ViVe/SESA [30, 31] took less than 4 seconds.

Table 1. Number of States needed to detect a failure and time taken for a non-deterministic NCES model.

No of places where non determinism exists	No of States generated before an error was detected	Time taken to generate the reachability graph
0	532	4 seconds
1	3552	60 seconds
2	5268	90 seconds

As seen from Table 1even with non-determinism the time taken to verify the model with NCES is much lesser. Surprisingly, we also managed to spot the deadlock behaviour of the plant, hence concluding that the controller was still not yet up to the mark. Notably, this flaw was not discovered by the automata model which contained non-determinism. The finite automata model verification took 120 minutes to verify the same CTL properties. In our view, the reason for such difference is fully non-deterministic modelling of finite automata.

Though it is not ideal to compare the execution timings of the two approaches because of the obvious difference in modelling techniques, what we are showing here is a way to minimize the state space by controlling presence of non-determinism only in certain parts of the model.

One should also note that our NCES model did not contain any sequencing aids which restrict the behaviour of the closed-loop Plant-Controller system and take extra effort in the model design (due to synchronization). In our approach we assemble the model from modules provided by IMC's without the need to take care of synchronization. Our tool chain helped to "plant" non-determinism modelling failures in particular mechatronic parts of the plant without causing state explosion. Such ability greatly enhances the performance of the developer, making formal verification a practical tool for everyday work. In our experiment we planted non-determinism in 1 or 2 places only in order to show the working of our framework.

6 CONCLUSIONS AND FUTURE WORK

This paper has given a brief overview of modelling principles put into integrated Veridemo tool chain. These are based on configuring model of the plant with controlled non-determinism induced dynamically using a meta model. The paper shows how this controlled non-determinism in Net Condition/Event Systems helps in controlling the state space and thereby reducing the verification times. The paper also shows how by introducing controlled non-determinism only certain parts of the plant can be verified to perfection, i.e assuming that certain parts of the plant model are guaranteed to be error free then using the approach presented in this paper we can skip having non-determinism is such parts of the plant. The approach also achieves much better model-checking performance using the model-checkers ViVe and SESA as compared to the modelling with fully non-deterministic state machines and model-checking with NuSMV[32] as used in [16, 33, 34]. The use of meta modelling also reduces the time taken to model as compacted to manually creating the models drawing a line and places

and transitions manually is definitely more time consuming that a configurator do it(takes just second or two). Possible future work directions will concern, for example:

1) Developing a more realistic timed plant modelling pattern allowing for variable actions duration within a certain interval.
2) Improving the user-friendliness of the model configurator to suggest potential model-parts for "planting" non-determinism in order to study certain properties of the object.
3) Designing a meta model which contains more information about the mechatronic object so that the configurator can be extended to include the ability to configure the discretisation and variable timing on modelling these mechatronic components.
4) Further investigating the computational impact of the "planted" and incremental non-determinism methodology over range of other embedded control systems.

REFERENCES

[1] R. Drechsler, Advanced Formal Verification. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[2] H.-M. Hanisch, "Closed-Loop Modelling and Related Problems of Embedded Control Systems in Engineering," in Abstract State Machines 2004. Advances in Theory and Practice. vol. 3052, W. Zimmermann and B. Thalheim, Eds., ed: Springer Berlin / Heidelberg, 2004, pp. 6-19.

[3] V. Vyatkin, et al., "Closed-loop modelling in future automation system engineering and validation," IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, vol. 39, pp. 17-28, 2009.

[4] E. M. Clarke, et al., Model Checking. Cambridge: The MIT Press, 1999.

[5] E. W. Endsley and D. M. Tilbury, "Modular verification of modular finite state machines," in Decision and Control, 2004. CDC. 43rd IEEE Conference on, 2004, pp. 972-979 Vol.1.

[6] G. Reed and A. Roscoe, "A timed model for communicating sequential processes," in Automata, Languages and Programming. vol. 226, L. Kott, Ed., ed: Springer Berlin / Heidelberg, 1986, pp. 314-323.

[7] R. Alur and T. A. Henzinger, "A really temporal logic," J. ACM, vol. 41, pp. 181-203, 1994.

[8] T. A. Henzinger, "The theory of hybrid automata," in Logic in Computer Science, 1996. LICS '96. Proceedings., Eleventh Annual IEEE Symposium on, 1996, pp. 278-292.

[9] K. G. Larsen, et al., "Compositional and symbolic model-checking of real-time systems," in Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE, 1995, pp. 76-87.

[10] K. G. Larsen, et al., "Model-Checking for Real-Time Systems," in Proceedings of the 10th International Conference on Fundamentals of Computation Theory, Dresden, Germany, 1995, pp. 62-88.

[11] Z. Chaochen, "Duration Calculus, a Logical Approach to Real-Time Systems," in Algebraic Methodology and Software Technology. vol. 1548, A. Haeberer, Ed., ed: Springer Berlin / Heidelberg, 1999, pp. 1-7.

[12] X. Nicollin and J. Sifakis, "The algebra of timed processes, ATP: theory and application," Inf. Comput., vol. 114, pp. 131-178, 1994.

[13] B. Berthomieu and M. Diaz, "Modelling and verification of time dependent systems using time Petri nets," Software Engineering, IEEE Transactions on, vol. 17, pp. 259-273, 1991.

[14] Z. Mengchu and V. Kurapati, Modelling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach. Hackensack, NJ: World Scientific Publishing Company, 1999.

[15] J. M. Machado, "Influência da Definição do Modelo de Análise de Processo na verificação formal de Sistemas de Manufactura a Eventos Discretos," PhD Thesis in cooperation between the University of Minho and École Normale Supérieure de Cachan; School of Engineering, University of Minho, Minho, 2006.

[16] J. M. Machado, et al., "Simulation and Formal Verification of Industrial System Controllers," ABCM Symposium Series in Mechatronics, vol. 3, pp. 461-470, 2008.

[17] A. Heegren, et al., "Synchronised execution of discrete event models using sequential function charts," in Decision and Control, 1999. Proceedings of the 38th IEEE Conference on, 1999, pp. 2237-2242 vol.3.

[18] R. W. Lewis, Programming industrial control systems using IEC 1131-3, Revised edition: The Institute of Electrical Engineers, 1998.

[19] V. Simoes, "Investigatopn of Automation Systems with Combination of Model Checking and Simulation," Masters Internship, University of Auckland, Auckland, 2010.

[20] H.-M. Hanisch and A. Lüder, "Modular modelling of closed-loop systems," in Proc of Colloquium on Petri Net Technologies for Modelling Communication Based Systems, ed Berlin, Germany, 2000, pp. 103-126.

[21] P. Cheng and V. Vyatkin, "Towards Formal Verification of IEC61499: modelling of Data and Algorithms in NCES," in Industrial Informatics, 2007 5th IEEE International Conference on, 2007, pp. 879-884.

[22] C. Gerber, et al. (2010) Formal modelling of IEC 61499 function blocks with integer-valued data types. Control and Cybernetics 97 - 231.

[23] C. Gerber and H.-M. Hanisch, "Does portability of IEC 61499 mean that once programmed control software runs everywhere?," in 10th IFAC Workshop on Intelligent Manufacturing Systems, Lisbon, Portugal, 2010, pp. 29-34.

[24] C. Sunder, et al., "Functional structure-based modelling of automation systems," International Journal of Manufacturing Research, vol. 1, pp. 405-420, 2006.

[25] N. A. Lynch and M. R. Tuttle, "Hierarchical correctness proofs for distributed algorithms," presented at the Proceedings of the sixth annual ACM Symposium on Principles of distributed computing, Vancouver, British Columbia, Canada, 1987.

[26] L. Szer-Ming, et al., "A component-based distributed control system for assembly automation," in Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on, 2004, pp. 33-38.

[27] V. Vyatkin and H.-M. Hanisch, "Verification of distributed control systems in intelligent manufacturing," Journal of Intelligent Manufacturing, vol. 14, pp. 123-136, 2003.

[28] V. Vyatkin, "Intelligent mechatronic components: control system engineering using an open distributed architecture," in Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference, 2003, pp. 277-284 vol.2.

[29] K. Thramboulidis, "Model-integrated mechatronics - toward a new paradigm in the development of manufacturing systems," Industrial Informatics, IEEE Transactions on, vol. 1, pp. 54-61, 2005.

[30] V. Vyatkin, "ViVe – the Visual Verifier," The University of Auckland, Auckland, 2007.

[31] V. Vyatkin, et al. (1999-2002). ViVe and SESA Model Checkers. Available: http://www.ece.auckland.ac.nz/~vyatkin/tools/modelchekers.html

[32] A. Cimatti, et al., "NuSMV 2: An OpenSource Tool for Symbolic Model Checking," in Computer Aided Verification. vol. 2404, E. Brinksma and K. Larsen, Eds., ed: Springer Berlin / Heidelberg, 2002, pp. 241-268.

[33] G. Di Guglielmo, et al., "Semi-formal functional verification by EFSM traversing via NuSMV," in High Level Design Validation and Test Workshop (HLDVT), 2010 IEEE International, 2010, pp. 58-65.

[34] M. Kadono, et al., "Using the NuSMV Model Checker for Test Generation from Statecharts," in Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim International Symposium on, 2009, pp. 37-42.